

Adaptive Solver Behavior in Mixed-Integer Programming

vorgelegt von

M. Sc.

Gregor Christian Hendel

ORCID:0000-0001-7132-5142

an der Fakultät II - Mathematik und Naturwissenschaften

der Technischen Universität Berlin

zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften

Dr.rer.nat.

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender: Prof. Dr. Tilo Schwalger

Gutachter: Prof. Dr. Thorsten Koch

Gutachter: Prof. Dr. Domenico Salvagnin

Gutachter: Prof. Dr. Sebastian Pokutta

Tag der wissenschaftlichen Aussprache: 28. September 2021

Berlin 2022

Dedicated to my dad, with whom I would have loved to share this book.

Acknowledgements

I would like to sincerely thank Thorsten Koch for giving me the opportunity to write a PhD Thesis under his supervision and his constant encouragement to “get things done”. Also, many thanks to my co-reviewers Domenico Salvagnin and Sebastian Pokutta, who have not only accepted to grade this work, but also motivated me in the process with a lot of inspiring discussions about the different aspects of this work.

I would like to thank the whole SCIP team for years of inspiring discussions in a unique work and research group, with an unprecedented dedication to the SCIP project. Ambros Gleixner has truly had an outstanding role in the group not only by taking care of the boring administrative details, but also keeping his door open day and sometimes even night to discuss paper ideas and progress. All of which he does with style. Another very important and inspiring person along my way is my former mentor Timo Berthold, who became my main teacher about academic writing at all stages of the process. Special thanks to Yuji Shinano, with whom I had the pleasure to share an office during all this time as well as many inspiring conversations and an unforgettable trip to Japan.

I would like to thank all of my co-authors for the inspiring discussions, motivational feedback, and relentless effort facing firm deadlines. Even if this meant accepting that one of my beloved sections will end in the appendix, this gave me the opportunity to learn from some of the best in our field and to grow professionally and personally. I am especially thankful to Pierre Le Bodic, whose work I have always used to admire secretly. So I immediately reached out when he was looking for a collaborator in the SCIP team. No matter the time shift, it was always a pleasure to work with him across the globe or in person during my memorable stay at Monash University.

I am very grateful to our collaboration partners from SAP. I always enjoyed our mutual workshops, as they gave us a unique opportunity to exchange new ideas and learn from their perspectives and expectations of the project.

I am indebted to Tobias Achterberg, Timo Berthold, Tristan Gally, Gerald Gamrath, Ambros Gleixner, Jacob Heintze, Pierre Le Bodic, Stephen J. Maher, and Benjamin Müller for proofreading parts of this thesis.

I would like to thank the FICO Xpress team and in particular Michael Perregaard for their warm welcome and their understanding and support during these last few months of dual responsibility between Xpress and thesis writing.

Special thanks to Isabelle, whose friendship, advise, and curiousness have been a true motivation and inspiration along the final stages.

Last but not least, I am infinitely grateful to Katharina for all her love and support during those years.¹

¹ ∞^∞

Table of Contents

Title Page	i
1 Introduction	1
2 Preliminaries: An Overview of Solution Techniques for MIP	7
2.1 Mixed-Integer Programs	7
2.2 Solving Linear Programs (LP)	12
2.2.1 The Primal and Dual LP	12
2.2.2 The Dual Simplex Algorithm	17
2.3 The Branch-and-Bound Algorithm	20
2.3.1 Fundamental Concepts	21
2.3.2 The Algorithm	25
2.3.3 Algorithmic Components	28
2.4 SCIP—Solving Constraint Integer Programs	29
2.5 Branching Rules	29
2.5.1 Strong Branching	31
2.5.2 Pseudo-Cost Branching	32
2.5.3 Reliability Branching	35
2.5.4 Hybrid Branching	35
2.5.5 Remarks	38
2.6 Primal Heuristics	38
2.7 Presolving and Propagation	41
2.7.1 Reduced Cost Strengthening	42
2.8 Machine Learning Inside B&B	43
2.8.1 A Quick Primer on Supervised Machine Learning	44
2.8.2 Regression Trees	45
2.8.3 Recent Combinations of ML and MIP	46
2.9 MIP Benchmarking	48
2.9.1 Gap	48
2.9.2 Primal and Dual Integrals	49
2.9.3 Shifted Geometric Mean	50

TABLE OF CONTENTS

2.9.4	Wilcoxon Signed Rank Test Using Shifts	51
2.9.5	IPET—An Interactive Performance Evaluation Tool	53
3	MIPLIB 2017	57
3.1	Introduction: A Motivation for a new MIPLIB	57
3.2	Related Work	60
3.3	First Steps: Collection of Instances and Data Cleanup	62
3.4	Model Groups	63
3.5	Feature Computation	63
3.5.1	Trivial Presolving	64
3.5.2	Canonical Form	66
3.5.3	Instance Features	66
3.5.4	Constraint Classification	69
3.5.5	Acquisition of Performance Data	71
3.5.6	Consistency Check of Solver Results	74
3.6	Selection Methodology	74
3.6.1	Benchmark Suitability	76
3.6.2	Diversity Preselection	77
3.6.3	Preparing Multiple Clusterings	79
3.6.4	Selection of the MIPLIB 2017 Collection	82
3.6.5	Performance Clusterings	84
3.6.6	Selecting the MIPLIB 2017 Benchmark Set	87
3.7	The Final Collection and Benchmark Sets	88
3.7.1	Representation in Feature Space	88
3.7.2	Solver Performance	90
3.7.3	The MIPLIB 2017 Web Page	93
3.8	Summary	95
4	Enhancing MIP Branching Decisions Using the Variance of Pseudo-costs	97
4.1	Reliability Branching and Fixed-Number Thresholds	98
4.2	Relative-Error and Hypothesis Reliability	100
4.2.1	Relative-Error Reliability	102
4.2.2	Hypothesis Reliability	103
4.3	Computational Results	104
4.4	Summary and Future Work	107
5	MIP Solving Phases	109
5.1	The Impact of Solving Components on the Solution Process	110
5.2	MIP Solving Phases	111
5.3	Two Phase Transition Heuristics	113

5.3.1	The Best-Estimate Transition	115
5.3.2	The Rank-1 Transition	115
5.4	Computational Results	117
5.4.1	Accuracy of the Proposed Phase Transitions	117
5.4.2	Using Phase Transitions to Control Solver Behavior	121
5.5	Summary	124
6	Adaptive Large Neighborhood Search	125
6.1	Large Neighborhood Search Heuristics for MIP	127
6.1.1	Fixing Neighborhood LNS Heuristics	128
6.1.2	LNS Heuristics Using Constraints and Auxiliary Objective Functions	130
6.2	Adaptive Large Neighborhood Search for MIP	132
6.2.1	Fixing and Unfixing Variables	133
6.2.2	Dynamic Limits	136
6.2.3	A Reward Function for Auxiliary Problems	138
6.3	Selection Strategies for Multi-Armed Bandit Problems	141
6.4	Computational Results	145
6.4.1	Auxiliary Problem Comparison	146
6.4.2	Simulation of the Selection Process	149
6.4.3	MIP Performance	154
6.5	Summary	158
7	Adaptivity Beyond Large Neighborhood Search	161
7.1	Diving Heuristics	162
7.1.1	Selection Strategy	163
7.1.2	Computational Results	164
7.2	Pricing for the Dual Simplex Algorithm	165
7.2.1	Selection Strategies for Simplex Pricing	166
7.2.2	Computational Results	168
7.3	Summary	169
8	Estimating Search Tree Size	171
8.1	Search Tree Exploration & Tree Size Estimation	172
8.1.1	Tree Profile Estimation	174
8.2	B&B Search States and Search Completion	175
8.3	Approximations of Search Completion	177
8.3.1	Gap	177
8.3.2	Sum of Subtree Gaps (SSG)	178
8.3.3	Tree Weight	179
8.3.4	Weighted Backtrack Estimator (WBE)	181
8.3.5	Leaf Frequency	182

TABLE OF CONTENTS

8.4	Estimation of Tree Size via Time Series Forecasting	183
8.4.1	General Definition of Double Exponential Smoothing (DES) . .	183
8.4.2	Double Exponential Smoothing for Tree Size Estimation	184
8.4.3	Time Series Steps and Adaptive Resolution	185
8.5	Calibration of Double Exponential Smoothing	186
8.5.1	Simulation Setup	186
8.5.2	Calibration of the Adaptive Resolution Capacity	187
8.5.3	Calibration of the DES Parameters	189
8.5.4	Example Instance <code>dancoint</code>	190
8.6	Implementation in SCIP	192
8.7	Learning the Search Completion	193
8.7.1	Data Set	193
8.7.2	Training	194
8.8	Evaluation	195
8.8.1	Comparison of Search Completion Estimation Methods	196
8.8.2	Comparison of Tree Size Estimation Methods	197
8.8.3	Comparison on Homogeneous Instance Sets	202
8.9	Summary and Outlook	204
9	Clairvoyant Restarts	207
9.1	Current Restart Strategies in MIP Solvers	208
9.2	A Clairvoyant Restart Strategy	209
9.3	Computational Results	211
9.4	Summary	218
10	Conclusion	221
	References	223
	Abstract	237
	Zusammenfassung	239
	Appendix A Appendix A	241
	Appendix B Appendix B	265
	Appendix C Appendix C	269
	Appendix D Appendix D	283
	Appendix E Appendix E	309

1

Introduction

“The function of selective discrimination with the complementary power of **adaptive** response is regarded as the root-principle of mind.”

Science, IV. 17

One of the key branches of mathematical optimization is mixed-integer programming (MIP), in which a linear objective function should be optimized subject to linear side-constraints and integrality requirements on a subset of the variables. This combination of constraints makes mixed-integer models a valuable asset for many practically relevant decision problems in business and industry. This thesis comprises a collection of adaptive algorithmic enhancements for solving mixed-integer programs.

One of the key factors for the success of the MIP paradigm is its expressiveness. As an example, take a use case from a logistics provider. This provider regularly makes decisions on building new warehouses and connecting them with customers to meet each customer’s demand. MIPs are very suitable for such business decision problems, as they allow to:

1. formulate discrete choices represented as binary or integer variables,
Should I build a warehouse here?
2. formulate cardinality or logical conditions between those variables,
How many warehouses should I construct in total?
3. link those discrete decisions with quantitative decisions.
Has customer demand been met?

The applications of MIP range from transportation, which includes railway scheduling and airline planning, to network design problems for telecommunication and energy grids to more abstract tasks such as chip verification.

The second key success factor is the availability of powerful general-purpose software to solve MIP models of practically relevant size to proven optimality. The most powerful commercial MIP solvers of today, some of which were originally released in the 1980s, have seen stunning progress over the years [Achterberg and Wunderling, 2013; Bixby, 2002]. Some models, however, are still unsolvable even by today’s sophisticated machinery. This is not at all surprising since MIP is an \mathcal{NP} -hard optimization problem in theory [Karp, 1972]. Therefore, almost all general-purpose software incorporates a form of the branch-and-bound (B&B) algorithm [Dakin, 1965; Land and Doig, 1960] as their main working horse, an algorithm with exponential worst-case behavior.

One of the main advantages of the B&B Algorithm is its completeness: When the search terminates, the best solution found is proven to be optimal. The other advantage is practicality. The algorithm may be interrupted by the user before the search is completed. In this case, the algorithm returns both the best solution found and a bound on the remaining gap to optimality.

Closing this gap further has motivated decades of research on MIP solution techniques. Numerous auxiliary algorithmic components have been introduced to aid the branch-and-bound algorithm, for example new primal heuristic algorithms. So far, a new component is often added as a standalone technique to the solver, with no interaction or communication with other components.

When I started my research position at Zuse Institute Berlin, I soon carried the label “solver intelligence” as part of the description of my (various) tasks in the group. What do we actually mean by “solver intelligence”? Quoting Legg and Hutter [2008],

“Intelligence measures an agent’s ability to achieve goals in a wide range of environments.”

For our purposes, the “agent” is the MIP solver. The “wide range of environments” are the numerous practical applications of MIP mentioned above. Intuitively, different applications may differ in the components that the solver should use predominantly for their solution. In this sense, we call an algorithm “adaptive” if it adjusts its decisions on the fly by taking into account past information collected only during the search itself without prior training. The aim of this thesis is to investigate the use of adaptive algorithmic techniques to improve MIP solvers.

What are the available techniques to adapt a MIP solver agent to a particular environment? Traditionally, MIP solvers have a huge number of user controls to enable or modulate the intensity of certain components. We use the open source MIP solver SCIP (Solving Constraint Integer Programs) for all computational experiments in this thesis. SCIP in its newest version 7.0.2 [Gamrath et al., 2020] features more than 2600

parameters. By default, these control settings are calibrated to maximize the performance of a solver on heterogeneous MIP benchmark sets. They are not specifically targeted to the application at hand. Control parameters give a user a lot of freedom, but also impose the burden to experiment with different control settings to customize the solver to the task at hand. This manual process can be cumbersome and requires a lot of knowledge, experience, and intuition. How could the MIP solver adapt automatically to a particular problem without this user effort?

When we consider adaptive or automated algorithmic decisions, we also discuss recent advances in Machine Learning (ML) and its applications inside a MIP solver. Bar some exceptions [Alvarez et al., 2016; Khalil et al., 2016], most ML approaches [Gasse et al., 2019; Khalil et al., 2017; Tang et al., 2020] studied so far that enhance MIP components use a representation of instances and/or variables in a designated feature space. This feature space is often application-specific [Xavier et al., 2019] and has to be provided by the user. After a data collection, which may be time-consuming, most ML methods [Alvarez et al., 2015; He et al., 2014] require an offline training phase of the method. In particular the training of neural networks may even require special hardware such as GPUs [Nair et al., 2020]. This cumbersome offline training phase fundamentally contradicts the power of a general-purpose MIP solver as an out-of-the-box tool.

On the contrary, the methods proposed in this thesis are often inspired by statistical methods such as time series forecasting or multi-armed bandit selection strategies. Our methods are specifically designed to be used inside a MIP solver. They require very little to no structural information about the underlying MIP problem that needs to be solved. This intended independence from structural features of a MIP often makes our methods widely applicable, which leads to substantial performance improvements of SCIP on the hardest and most heterogeneous MIP benchmark sets available.

From a methodological point of view, we often pursue the incorporation of new statistical methods in two stages. In a first stage, we collect data over a set of instances for a calibration of our methods. In a second stage, we transfer the simulation results back into the solver itself. The initial simulation stage is valuable for three reasons. First, data-driven simulations are much faster to perform outside the solver. This allows us to conduct repetitions and tests that would be impossible or too slow to do inside the solver. Second, beyond a pure calibration of the newly proposed methods, the simulation results also lead to new and interesting insights of the underlying MIP technology. For example, the simulation approach enables us to conduct one of the largest comparisons of existing Large Neighborhood Search (LNS) primal heuristics to date in Chapter 6. As a consequence, we re-enabled several LNS heuristics such as DINS [Ghosh, 2007] or Local Branching [Fischetti and Lodi, 2003] to help diversify the search inside our new heuristic framework Adaptive Large Neighborhood Search. Last, (re-)implementing an algorithm for the simulation stage provides additional testing, which may help to

identify potential bugs and pitfalls that would be hard to detect in MIP production code.

Undoubtedly, this thesis builds upon an extraordinarily rich history of more than 70 years of algorithmic development for solving MIP, starting with the introduction of the primal simplex algorithm by Dantzig [1951] and the dual simplex algorithm [Lemke, 1954, more in Chapter 2]. We argue that adaptive algorithmic behavior has been part of that rich history. Pseudo-costs [Bénichou et al., 1971] are commonly used by state-of-the-art codes to guide branching decisions inside Reliability Branching [Achterberg et al., 2004]. We make Reliability Branching adaptive by introducing new notions of reliability that take into account the sample variance of pseudo-costs. Goldilocks [Rothberg, 2007] modulate the difficulty of the subproblems solved inside LNS heuristics, but have not been used in SCIP so far, which we remedy. We propose several novel adaptive extensions to individual algorithmic components of modern branch-and-bound solvers such as branching strategies, LP pricing, and different groups of primal heuristics. Ultimately, we revise the classical branch-and-bound algorithm itself by introducing an intelligent restart strategy based on tree size estimation.

Organization of the Thesis

Before we get started, we introduce the necessary notation for our endeavor in Chapter 2, accompanied by rich and colorful illustrative examples of the well-known concepts around branch-and-bound.

In Chapter 3, we describe the contributions of the author of this thesis leading to the sixth version of the Mixed-Integer Programming Library MIPLIB 2017. For the first time, this edition was compiled using a data-driven selection process supported by the solution of a sequence of mixed-integer programming problems, which encode requirements on diversity and balance with respect to instance features and performance data. Selected from an initial pool of 5,721 instances, the new MIPLIB 2017 collection consists of 1,065 instances. A subset of 240 instances was especially selected for benchmarking solver performance.

After this methodological contribution, we turn our attention towards actual solver improvements. In Chapter 4, we generalize the established reliability branching rule [Achterberg et al., 2004]. We suggest two novel notions of reliability motivated by mathematical statistics that take into account the sample variance of the past observations on each variable individually. The first method prioritizes additional strong branching look-aheads on variables whose pseudo-costs show a large variance by measuring the relative error of a pseudo-cost confidence interval. The second method performs a specialized version of a two-sample Student’s t -test for filtering branching candidates with a high probability to perform better than the best history candidate.

Both methods have been integrated into SCIP. Our computational results on show that they are particularly valuable for accelerating the dual convergence on hard instances.

In Chapter 5, we take a closer look at the typical run of a MIP solver. Typically, the different components of a MIP solver are tuned to minimize the average overall running time to prove optimality. We argue that the solution process consists of three distinct phases, namely achieving *feasibility*, *improving* the incumbent solution, and *proving* optimality. We first show that the entire solving process can be improved by adapting the search strategy with respect to the phase-specific aims using different control tunings. Afterwards, we provide criteria to predict the transition between the individual phases and evaluate the performance impact of altering the algorithmic behavior of SCIP at the predicted phase transition points.

Chapter 6 introduces Adaptive Large Neighborhood Search (ALNS) for MIP, a novel framework for eight popular LNS heuristics. In ALNS, the decision which LNS heuristic should be executed is guided by selection strategies for the *multi-armed bandit problem*, an optimization problem under uncertainty during which suitable actions have to be chosen to maximize a reward function. To apply bandit selection strategies to our LNS setting, we propose an LNS-specific reward function. As an algorithmic enhancement to ALNS, we also propose a generic variable fixing prioritization, which ALNS employs to adjust the subproblem complexity as needed. This is particularly useful for those LNS heuristics that do not fix variables by themselves. An example is Local Branching, which only restricts the Hamming Distance between the current incumbent and an improving solution.

The proposed ALNS framework has been implemented in SCIP. An extensive computational study is conducted to compare different LNS strategies within our ALNS framework on a large set of MIP instances. The results of this simulation are used to calibrate the parameters of the bandit selection strategies. Finally, a second computational experiment shows the computational benefits of the proposed ALNS framework within SCIP.

We extend these ideas further in Chapter 7 by proposing adaptive algorithmic behavior for two other important MIP solving components, namely diving heuristics and simplex pricing strategies. For each class, we propose a selection strategy that is updated based on the observed runtime behavior, aiming to ultimately select only the best algorithms for a given MIP instance. With the goal to apply bandit selection strategies, we carefully design reward functions to rank and compare each individual diving heuristic or pricing algorithm within its respective class. Finally, we discuss the computational benefits of using the proposed adaptive algorithmic behavior within SCIP.

Some of our adaptive algorithmic decisions require an online estimation of the final size of the B&B search tree, for which we investigate different methods in Chapter 8. We review measures of progress of the B&B search, such as the well-known gap and the

tree weight [Kilby et al., 2006], and propose a new measure, which we call *leaf frequency*. We study two simple ways to transform these progress measures into B&B tree size estimates, either as a direct projection, or via double-exponential smoothing, a standard time-series forecasting technique. We then combine different progress measures and their trends into nontrivial estimates using Machine Learning techniques, which yield more precise estimates than any individual measure. The best method we have identified uses all individual measures as features of a random forest model. In a large computational study, we train and validate all methods. On average, the best method estimates B&B tree sizes within a factor of 3 on the set of unseen test instances even during the early stage of the search, and improves in accuracy as the search progresses. It also achieves a factor 2 over the entire search on each of six additional sets of homogeneous instances we have tested. Traditionally the gap used to be the de-facto progress measure of MIP search. These new estimations improve upon the accuracy of the gap by an order of magnitude.

Equipped with tree size estimation, we finally show in Chapter 9 how this information can help the search algorithmically at runtime by designing a restart strategy for MIP that decides whether to restart the search based on the current estimate of the number of remaining nodes in the tree. We refer to this type of algorithm as *clairvoyant*. Our clairvoyant restart strategy outperforms SCIP on MIPLIB 2017 by up to 20 %. Together with tree size estimation, clairvoyant restarts have been publicly available since SCIP 7.0.

Finally, we wrap up our findings with some concluding remarks in Chapter 10. All proposed techniques are available since version 7 of the branch-and-cut framework SCIP.

2

Preliminaries: An Overview of Solution Techniques for MIP

2.1 Mixed-Integer Programs

A mixed-integer program denotes the problem to minimize a linear objective function under linear inequalities and integrality restrictions for a subset of the variables. We use the term “mixed” to refer to the occurrence of two variable types, continuous and integer variables, in the problem formulation.

Definition 2.1 (Mixed-integer program). *Let $n, m \in \mathbb{N}$, $A \in \mathbb{Q}^{m,n}$ be a rational matrix, and let $c \in \mathbb{Q}^n$ and $b \in \mathbb{Q}^m$ be a cost and a right-hand side vector, respectively. Let further $\ell, u \in \mathbb{Q}_{\pm\infty}^n$ denote bound requirements for the variables, and let a subset $\mathcal{I} \subseteq \{1, \dots, n\}$ of the variable index set denote integrality restrictions. An optimization problem defined by $c, A, b, \ell, u, \mathcal{I}$ as*

$$\begin{array}{llll}
 \min_{x \in \mathbb{Q}^n} & c^\top x & & \text{(objective)} \\
 s.t. & Ax \geq b & & \text{(inequalities)} \\
 & \ell \leq x \leq u & & \text{(bound requirements)} \\
 & x_j \in \mathbb{Z} & j \in \mathcal{I} & \text{(integrality restrictions)}
 \end{array}$$

is called a mixed-integer program (MIP).

Solving a MIP consists in finding a provably optimal solution or in proving that no such solution exists. A vector $\tilde{x} \in \mathbb{Q}^n$ is called a (*feasible*) *solution* for a MIP P , if it satisfies all inequalities, bound requirements, and integrality restrictions of P . The set of all solutions of P is denoted by \mathcal{S}_P .

We use a shorthand matrix notation for the m linear inequalities, which a solution must simultaneously satisfy. We denote a single inequality (row) $i \in \{1, \dots, m\}$ using the notation $a_i^\top x \geq b_i$. We denote by A_j the j 'th column vector of A . Given a nonempty set $J \subseteq \{1, \dots, n\}$, we denote by $A_{[J]}$ the m -by- $|J|$ matrix that consists of only those columns of A that are contained in J . If $J = (j_1, \dots, j_k)$ is an ordered tuple instead of an ordinary set, the columns of $A_{[J]}$ are rearranged to respect this ordering.

Note that the exclusive use of “ \geq ” inequalities does not represent a real restriction for MIP applications modelled in the above form, since the opposite sense “ \leq ” is easily expressed by multiplying the desired row coefficients and right-hand side with -1 to match the sense “ \geq ”. An equation $a^\top x = b$ can be formulated by combining two separate rows $a^\top x \geq b$ and $-a^\top x \geq -b$.

We denote the *optimal objective value* of P by

$$Z^{\text{opt}} := \inf_{x \in \mathcal{S}_P} c^\top x,$$

which may be infinite in both directions, namely if $\mathcal{S}_P = \emptyset$ or if \mathcal{S}_P admits decreasing the objective arbitrarily. In the first case, P is called *infeasible* and $Z^{\text{opt}} = +\infty$. In the second case, in which for all $Z' \in \mathbb{Q}$ there exists a solution $\tilde{x} \in \mathcal{S}_P$ such that $c^\top \tilde{x} \leq Z'$, we call P *unbounded* with optimal objective value $Z^{\text{opt}} = -\infty$. A solution \tilde{x}^{opt} that satisfies $c^\top \tilde{x}^{\text{opt}} = Z^{\text{opt}}$ is called *optimal*. Obviously, no optimal solution exists for P if P is unbounded or infeasible. Conversely, if Z^{opt} is finite, then an optimal solution exists. Note that an optimal solution does not need to be unique in general.

Variables indexed by $j \in \mathcal{I}$ are called *integer variables*. An important subset of the integer variables are the *binary variables*. A binary variable $j \in \mathcal{I}$ is an integer variable with lower bound $\ell_j = 0$ and upper bound $u_j = 1$. We denote the set of binary variables by $\mathcal{B} \subseteq \mathcal{I}$. Conversely, integer variables that are not binary are called *general integer variables* and denoted by $\mathcal{G} := \mathcal{I} \setminus \mathcal{B}$.

The remaining variables in the set $\mathcal{C} := \{1, \dots, n\} \setminus \mathcal{I}$ are called *continuous variables*. The individual numbers of binary, general integer, integer, and continuous variables are denoted by $n^{\text{bin}} := |\mathcal{B}|$, $n^{\text{gen}} := |\mathcal{G}|$, $n^{\text{int}} := |\mathcal{I}|$, $n^{\text{con}} := |\mathcal{C}|$. Binary variables are often used to encode essential yes/no-decisions in an optimization scenario such as whether a facility should be built at a certain location. Therefore, binary variables often receive a prioritized treatment by various MIP solving components (see Section 2.3.3).

Example 1. We consider an example of a MIP for illustrating the concepts defined so far. This example will be used as a recurring example throughout this chapter. Consider

the following MIP P with $n = n^{\text{int}} = 5$ variables, i.e., $\mathcal{I} := \{1, \dots, 5\}$.

$$\begin{array}{llllllllll}
 \min & -5 & x_4 & -3 & x_5 & & -23 & x_1 & -15 & x_2 & -9 & x_3 \\
 \text{s.t.} & & & & 0 & = & 1 & -1 & x_1 & -1 & x_2 & -1 & x_3 \\
 & & x_4 & & & \geq & 0.5 & & & +4 & x_2 & +2 & x_3 \\
 & & & x_5 & \leq & 0.2 & +1.9 & x_1 & +3.7 & x_2 & +5.9 & x_3 \\
 & -2 & x_4 & +3 & x_5 & \geq & & -0.2 & x_1 & -6 & x_2 & +4.6 & x_3 \\
 & & x_4 & & & \leq & 6 & & & & & & \\
 & & & & 1 & \geq & & x_1, & & x_2, & & x_3 \\
 & & & & 0 & \leq & & x_1, & & x_2, & & x_3 \\
 & & x_4, & & x_5 & \geq & 0 & & & & & &
 \end{array} \tag{2.1}$$

We assume all lower bounds ℓ to be zero. The upper bounds u_1, u_2, u_3 are 1. Together with the lower bounds, this means that $\{x_1, x_2, x_3\}$ form the set of binary variables of P (in other words, $\mathcal{B} = \{1, \dots, 3\}$), and x_4 and x_5 are general integer. While x_4 has an explicit upper bound of 6, x_5 has an upper bound of $u_5 = \infty$, but is implicitly bounded by the rows of P , as we will see shortly.

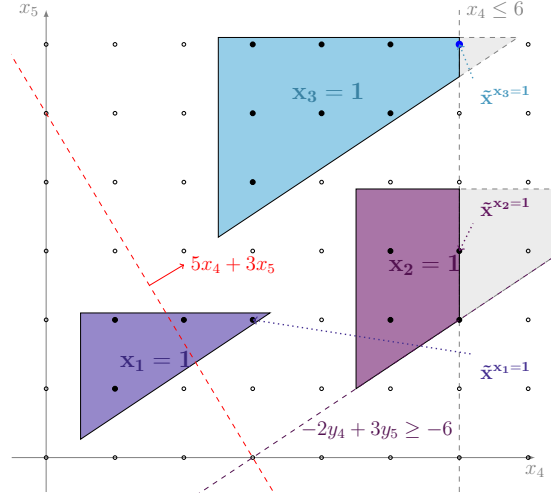
The first line contains the objective function coefficients. For the rows of P , we deviate slightly from the notation of Definition 2.1 to highlight the interaction of the variables better. Namely, the rows are presented in such a way that the effect on the two general integer variables x_4, x_5 is apparent. Therefore, we allow an equation and mix the senses of the shown inequalities. A transformation into the form of MIP of Definition 2.1 can be obtained easily by splitting the equation into two inequalities first, and multiplying all inequalities with the “wrong” sense by -1 .

The first row is an equation that requires that exactly one of x_1, x_2, x_3 be set to 1. Since x_1, x_2, x_3 are binary variables, setting one of them to 1 forces the remaining two variables to 0. A row like the first line is often called a “set partitioning” or also a “multiple-choice” constraint. Note that the explicit upper bounds on x_1, x_2 , and x_3 are not really necessary because they are implied by the multiple choice constraint, lower bounds of zero, and integrality restrictions.

Note that in this example each choice of $x_1 = 1$, $x_2 = 1$, or $x_3 = 1$ can be extended to a feasible solution. Depending on this choice, we treat x_1, x_2, x_3 as constants such that the remaining rows become inequalities on x_4 and x_5 ; the second row represents a lower bound on x_4 , the third row represents an upper bound on x_5 , and the last row constrains the feasible assignments of x_4 and x_5 further.

In Figure 2.1 we illustrate the projection of the solution set \mathcal{S}_P onto the x_4 - x_5 -plane. For each solution $\tilde{x} \in \mathcal{S}_P$, the multiple choice constraint enforces that $\tilde{x}_1 = 1$ or $\tilde{x}_2 = 1$ or $\tilde{x}_3 = 1$, and the remaining two binaries to 0. All feasible solutions correspond to grid points, because x_4 and x_5 are required to be integer. x_4 must not be larger than 6.

Figure 2.1: Illustration of the MIP Example (2.1). We show a projection of the solution set \mathcal{S}_P into the x_4 - x_5 -plane. To the right, the feasible region is bounded by $x_4 \leq 6$. The three filled regions correspond to the feasible region given by the linear inequalities on x_4 and x_5 if x_1 , x_2 , or x_3 are set to 1, respectively. Only solid circles correspond to feasible solutions in \mathcal{S}_P . Further elements shown are one example inequality $-2x_4 + 3x_5 \geq -6$, which bounds the feasible region if (and only if) $x_2 = 1$, the objective function for x_4 and x_5 , and best solutions in each region for $j = 1, 2, 3$.



Three colored regions show the feasible region as defined solely by the inequalities on x_4 and x_5 , if we drop the integrality restrictions. (The projections of) feasible solutions lie within one of these regions after choosing x_1 , x_2 , or x_3 . They are shown in the figure as filled black circles.

An example inequality is shown in the figure. Assuming the choice of $x_2 = 1$ (and hence $x_1 = x_3 = 0$), the fourth row of P simplifies to $-2x_4 + 3x_5 \geq -6$. This inequality defines a hyperplane whose boundary is shown as a dashed line. If x_1 or x_3 was selected instead of x_2 , the right-hand side of the inequality changes. We easily spot the two lines parallel to $-2x_4 + 3x_5 = -6$ bounding the two other feasible regions.

All grid points outside (the union of) those regions are infeasible for P . A projection of the objective function is also shown as a red dashed line. Note that minimizing $-5x_4 - 3x_5$ is equivalent to maximizing $5x_4 + 3x_5$. Points on the red dashed line have the same objective value. An arrow indicates the improving direction of the objective.

Note that the multiple choice constraint also has an impact on the objective: Each of x_1 , x_2 , and x_3 adds a different offset value to the objective. In order to truly find the optimal solution geometrically, we have to take into account this offset. The best projected solution for $x_1 = 1$ is the point $\tilde{x}^{x_1=1} = (3, 2)^t$, the best projected solution for $x_2 = 1$ is $\tilde{x}^{x_2=1} = (6, 3)^t$, and the best projected solution for $x_3 = 1$ is $\tilde{x}^{x_3=1} = (6, 6)^t$. By plugging everything into the original objective function for P , we calculate

$$c^t(1, 0, 0, 3, 2)^t = -44 \geq c^t(0, 1, 0, 6, 3)^t = -54 \geq c^t(0, 0, 1, 6, 6)^t = -57.$$

Therefore, the optimal solution for P is $\tilde{x}^{\text{opt}} = (0, 0, 1, 6, 6)^t$ (in the original, unprojected space) with an objective value $Z^{\text{opt}} = -57$.

We could solve the MIP of Example 1 geometrically, but what about higher dimensions? One could try to enumerate all possible solutions, at least if all variables are restricted to be integer and bounded, but at the price of having to enumerate all exponentially many possible combinations of values. Note that enumeration already falls short for the example above, in which one variable has an infinite upper bound.

It becomes clear that a more structured approach is needed for solving MIPs. MIP is known to be \mathcal{NP} -hard [Garey and Johnson, 1979]. If a MIP P has no integrality restrictions, i.e., $\mathcal{I} = \emptyset$, we call P a *linear program (LP)*. An LP \check{P} is called the *LP relaxation* of a MIP P if it is derived from P by dropping the integrality restrictions.

The close connection between P and its LP relaxation \check{P} results in two key properties that will help in developing the branch-and-bound (B&B) algorithm [Dakin, 1965; Land and Doig, 1960, explained in Section 2.3] to solve P .

The first key property of linear programs and LP relaxations is their solvability in polynomial time [Khachiyan, 1979]. Second, since the solution set $\mathcal{S}_{\check{P}}$ of \check{P} is a superset of the solution set \mathcal{S}_P of P , it holds that

$$\check{Z} \leq Z^{\text{opt}}, \quad (2.2)$$

where \check{Z} denotes the optimal objective value of \check{P} . Due to inequality (2.2), we obtain a lower bound on the optimal objective of P by solving its LP relaxation to optimality. Since \check{P} is itself a MIP according to Definition 2.1, \check{Z} is infinite if \check{P} is unbounded or infeasible. Otherwise, if \check{Z} is finite, there exists a solution $\check{x} \in \mathcal{S}_{\check{P}}$ that attains $c^t \check{x} = \check{Z}$. We never consider suboptimal LP solutions in this thesis; whenever we use an LP solution \check{x} for an LP relaxation \check{P} , we implicitly assume that \check{x} is optimal for \check{P} . If \check{x} satisfies the integrality requirements of P , i.e., if $\check{x}_j \in \mathbb{Z}$ for all $j \in \mathcal{I}$, \check{x} is also optimal for P . Conversely, an infeasible LP relaxation \check{P} proves the infeasibility of P , as well.

Note that an infeasible MIP P need not necessarily have an infeasible LP relaxation. It is possible in this case that the LP relaxation \check{P} is feasible or even unbounded.

If an LP relaxation \check{P} is unbounded, P can be infeasible or unbounded as well. The fact that P cannot have an optimal solution follows from the input data being rational. With rational data, any unbounded ray for the LP that starts from an integer feasible solution will always hit a second (and thus infinitely many) integer feasible solutions with improved objective value. This is not necessarily true in the case of irrational data.¹

The unbounded case plays a less critical role in practice, where unboundedness usually indicates a mistake in the formulation of P such as the omission of a crucial row. A MIP and its LP relaxation are always bounded if all variables have finite bounds.

For a textbook introduction to integer programming, we refer to [Wolsey, 2020].

¹For example, consider the problem $\max y : y = \pi \cdot x; x, y \in \mathbb{Z}, x, y \geq 0$. There is only one integer feasible solution but the LP relaxation (in real-valued variables) is unbounded.

2.2 Solving Linear Programs (LP)

In this section, we outline the dual simplex algorithm introduced by Lemke [1954] for solving linear programs. We review this important algorithm for several reasons. First, the dual simplex algorithm is nowadays the most common algorithm to (re-)optimize LPs during the B&B algorithm for MIPs, which we introduce in Section 2.3. Second, in Chapter 7, we propose adaptive LP pricing, which dynamically switches the pricing strategy of the dual simplex algorithm during the search. Third, there are some related notions such as reduced costs that we need later for the generic fixing scheme of ALNS in Chapter 6.

For the sake of space, we will keep the necessary notation and theory at a minimum. The reader is referred to text books by Chvátal [1983] or Vanderbei [2014] for more details. A more theoretical introduction into polyhedral theory around linear programming is given by [Schrijver, 1986]. Our notation in this section is in large parts based on a lecture on solving linear programs by Robert E. Bixby, which is available online.²

2.2.1 The Primal and Dual LP

According to Definition 2.1, an LP P' has the form

$$\begin{aligned} \min_{x \in \mathbb{Q}^{n'}} \quad & c^t x \\ \text{s.t.} \quad & A'x \geq b \\ & \ell \leq x \leq u. \end{aligned} \tag{P'}$$

with $A' \in \mathbb{Q}^{m, n'}$. As an LP, P' has no integrality restrictions.

For this section, we transform P' into an equivalent LP P with equations instead of inequalities, lower bounds of zero, and no explicit upper bounds. Without loss of generality, we may assume that $\ell = 0$. The following variable substitutions ensure this.

Let $j \in \{1, \dots, n'\}$ be a variable index of a variable x_j with lower bound ℓ_j and u_j .

- $|\ell_j| < \infty \Rightarrow$ Substitute x_j by $x_j - \ell_j$.
- $\ell_j = -\infty, u_j < \infty \Rightarrow$ Substitute x_j by $u_j - x_j$.
- $\ell_j = -\infty, u_j = +\infty \Rightarrow$ Substitute x_j by introducing two variables $x_{j+} - x_{j-}$ with lower bounds $\ell_{j+} = \ell_{j-} = 0$.

Note that all these substitutions require updates of the right-hand side b and/or matrix A' . The third substitution increases n' by one for each variable without finite bounds.³ For $j \in \{1, \dots, n'\}$ we denote by $e_j \in \mathbb{Q}^{n'}$ the j 'th unit vector. Now that

²Robert E. Bixby: Solving Linear Programs, <https://www.youtube.com/watch?v=z1xvqwQR6xU>

³Such variables are called *free* variables.

all lower bounds are 0, any remaining finite upper bound $u_j < \infty$ is added as a new inequality $-e_j^\top x \geq -u_j$ to the matrix A' and right-hand side b . Each finite upper bound thereby increases m by one.

Now that we have established lower bounds of zero and no explicit upper bounds, we transform inequalities into equations by introducing one slack variable per row. Therefore, we set $n := n' + m$ and extend A' by an m -by- m identity matrix E_m to define $A := (A', -E_m) \in \mathbb{Q}^{m,n}$. We extend $c' \in \mathbb{Q}^{n'}$ to $c \in \mathbb{Q}^n$ by setting $c_{n'+i} := 0$ for $i \in \{1, \dots, m\}$. By construction, we have $m \leq n$. Since A contains an m -by- m identity matrix, A has full rank m .

For the remainder of this section we consider an LP P in *standard form*:

$$\begin{aligned} \min_{x \in \mathbb{Q}^n} \quad & c^\top x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0. \end{aligned} \tag{P}$$

P is equivalent to P' in the sense that an optimal solution for P can be transformed into an optimal solution for P' by considering only the solution values of the first n' variables. The important difference to Definition 2.1 is the presence of equations.

P in the above form is also called *primal* LP. The search for a lower bound on the optimal objective value Z^{opt} of P gives rise to the *dual* of P :

$$\begin{aligned} \max_{\pi \in \mathbb{Q}^m, d \in \mathbb{Q}^n} \quad & \pi^\top b \\ \text{s.t.} \quad & A^\top \pi + d = c \\ & \pi \text{ free}, \quad d \geq 0 \end{aligned} \tag{P^{\text{dual}}}$$

The rationale behind P^{dual} is to find the best possible lower bound on the optimal objective value Z^{opt} of P . To this end, we search for a linear combination of the rows of A that does not exceed any of the objective coefficients c of the primal LP.

Note that π are free variables, which means that they can also take negative values. P^{dual} has one π variable for each row of P and one dual slack variable d for each column of P . The dual slack variables have a lower bound of zero, which ensures that the linear row combination $A^\top \pi$ does not exceed c .

Consequently, it is easy to verify that the following inequality holds for any primal-dual pair of solutions $\tilde{x} \in \mathcal{S}_P$, $(\tilde{\pi}, \tilde{d}) \in \mathcal{S}_{P^{\text{dual}}}$.

$$c^\top \tilde{x} = (\tilde{\pi}^\top A + \tilde{d}^\top) \tilde{x} \geq \tilde{\pi}^\top A \tilde{x} = \tilde{\pi}^\top b. \tag{2.3}$$

The first part of (2.3) holds because $(\tilde{\pi}, \tilde{d})$ is dual feasible, i.e. it satisfies $A^\top \tilde{\pi} + \tilde{d} = c$. The inequality holds because both \tilde{d} and \tilde{x} are nonnegative. The last part holds because \tilde{x} is primal feasible, i.e. $A \tilde{x} = b$. Inequality (2.3) proves the *weak duality theorem*, which

states that if both P and P^{dual} are feasible, the optimal value of P^{dual} is a lower bound on Z^{opt} . The *strong duality theorem* [see, e.g., Vanderbei, 2014] states that Inequality (2.3) becomes an equality if \tilde{x} and $(\tilde{\pi}, \tilde{d})$ are optimal for P and P^{dual} , respectively.

Definition 2.2 (Basic Solution). *Let P be an LP in standard form with m rows and n columns. Let P^{dual} be the corresponding dual LP. Let $B = (B_1, \dots, B_m)$ be an ordered tuple of m distinct column indices: $B_i \in \{1, \dots, n\}$, $B_i \neq B_j$ if $i \neq j$. Let $N := \{1, \dots, n\} \setminus B$.*

- B is called a basis if the submatrix $A_{[B]}$ is nonsingular.
- Variables $x_{[B]}$ are called basic variables and variables $x_{[N]}$ are called nonbasic variables.
- A basic solution \tilde{x} is obtained by setting $\tilde{x}_{[N]} := 0$ and $\tilde{x}_{[B]} := (A_{[B]})^{-1}b$.
- A basic solution \tilde{x} is called primal feasible if $\tilde{x} \geq 0$.
- B is called optimal if its basic solution is optimal for P .

A basis B can be used to partition the rows of P^{dual} as follows.

$$\begin{pmatrix} (A_{[B]})^t & E_{[B]} & 0 \\ (A_{[N]})^t & 0 & E_{[N]} \end{pmatrix} \begin{pmatrix} \pi \\ d_{[B]} \\ d_{[N]} \end{pmatrix} = \begin{pmatrix} c_{[B]} \\ c_{[N]} \end{pmatrix}. \quad (2.4)$$

In (2.4) we denote by $E_{[B]}$ an $m \times m$ identity matrix whose columns are ordered according to B . In this representation of P^{dual} , we see that the submatrix corresponding to the π and $d_{[N]}$ variables has full rank, because $A_{[B]}$ and $E_{[N]}$ have full rank, respectively. The corresponding dual variables π and $d_{[N]}$ are therefore called the *dual basic variables* and the $d_{[B]}$ variables are called *dual nonbasic variables*. The corresponding *dual basic solution* $(\tilde{\pi}, \tilde{d})$ is obtained by setting $\tilde{d}_{[B]} := 0$ and computing the remaining values according to (2.4):

$$\tilde{\pi} := (A_{[B]})^{-t} c_{[B]}, \quad \tilde{d}_{[N]} := c_{[N]} - (A_{[N]})^t \tilde{\pi}. \quad (2.5)$$

We use the shorthand notation A^{-t} to denote the inverse of the transposed matrix A^t . The solution values \tilde{d} of the dual slack variables are also called *reduced costs*.

We call B *dual feasible* if $\tilde{d}_{[N]} \geq 0$. A basis that is both primal feasible and dual feasible is optimal for P and P^{dual} due to the strong duality theorem. It can be shown that every LP which has an optimal solution also has an optimal basic solution. In other words, it is sufficient to consider basic solutions to solve an LP to optimality. This is the idea of the simplex algorithm, which is the subject of Section 2.2.2.

Example 2. (Example 1 continued)

We examine the LP relaxation \check{P} of our MIP P from Example (2.1). Since the integrality restrictions are dropped in \check{P} , the multiple choice row can now mix values of x_1 , x_2 , and x_3 freely as long as their sum is equal to 1.

All variables have a lower bound of zero. We express each finite upper bound on a variable u_j as a matrix inequality $-e_j^t x \geq -u_j$. We split the multiple choice constraint into two inequalities. We normalize the inequalities such that they all have the same sense \geq . We arrive at the following form of \check{P} with matrix A' and right-hand side b

$$\begin{pmatrix} 1.0 & 1.0 & 1.0 & & & & \\ -1.0 & -1.0 & -1.0 & & & & \\ & -4.0 & -2.0 & 1.0 & & & \\ 1.9 & 3.7 & 5.9 & & -1.0 & & \\ 0.2 & 6.0 & -4.6 & -2.0 & 3.0 & & \\ & & & -1.0 & & & \\ -1.0 & & & & & & \\ & -1.0 & & & & & \\ & & -1.0 & & & & \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} \geq \begin{pmatrix} 1.0 \\ -1.0 \\ 0.5 \\ -0.2 \\ 0.0 \\ -6.0 \\ -1.0 \\ -1.0 \\ -1.0 \end{pmatrix}$$

in 5 variables and 9 inequalities. The omitted entries in A' are 0.

We extend A' by a negative identity matrix and introduce slack variables, one per row. All slack variables have an objective coefficient of zero. This yields the standard form of \check{P} with 14 variables representing our primal problem:

$$\begin{pmatrix} 1.0 & 1.0 & 1.0 & & & -1.0 & & \dots & \\ -1.0 & -1.0 & -1.0 & & & & -1.0 & & \\ & -4.0 & -2.0 & 1.0 & & & & & \\ 1.9 & 3.7 & 5.9 & & -1.0 & & & & \\ 0.2 & 6.0 & -4.6 & -2.0 & 3.0 & \vdots & \ddots & \vdots & \\ & & & -1.0 & & & & & \\ -1.0 & & & & & & & & \\ & -1.0 & & & & & & & \\ & & -1.0 & & & & & & \\ & & & & & \dots & -1.0 & & \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ \vdots \\ x_{14} \end{pmatrix} = \begin{pmatrix} 1.0 \\ -1.0 \\ 0.5 \\ -0.2 \\ 0.0 \\ -6.0 \\ -1.0 \\ -1.0 \\ -1.0 \end{pmatrix}$$

The dual \check{P}^{dual} of \check{P} has 23 variables, namely 9 row multipliers π_1, \dots, π_9 and 14 dual slack variables d_1, \dots, d_{14} , one for each variable in the primal problem.

A basis consists of 9 out of the 14 primal variable indices, one per row. Not every choice of indices is valid because for a basis the corresponding submatrix $A_{[B]}$ has to be nonsingular. For example, observe that a basis must contain at least one of $\{6, 7\}$. These correspond to the slack variables of the first two rows. In a quadratic submatrix without columns A_6 and/or A_7 the two first rows are parallel such that the submatrix does not have full rank.

Consider the basis $B := (1, 2, 4, 5, 6, 8, 12, 13, 14)$. It can be verified that the corresponding submatrix $A_{[B]}$ is nonsingular. The corresponding nonbasic variables are $N := (3, 7, 9, 10, 11)$.

We compute the basic solution \tilde{x} as defined in Definition 2.2. We set the nonbasic variables $\tilde{x}_{[N]}$ to 0. To obtain the basic solution values $\tilde{x}_{[B]}$, we solve the equation system $A_{[B]}\tilde{x}_{[B]} = b$. Both the existence and uniqueness of $\tilde{x}_{[B]}$ are guaranteed because $A_{[B]}$ is nonsingular.

We obtain a basic solution with solution values

$$\tilde{x}_{[(1,2,3,4,5)]} = (0.509, 0.491, 0, 6, 2.984)$$

for the original variables, and

$$\tilde{x}_{[(6,7,\dots,14)]} = (0, 0, 3.536, 0, 0, 0, 0.491, 0.509, 1.000)$$

in the slack variables. Since all basic solution values are nonnegative, B is primal feasible. Is it also dual feasible?

Recall that B is dual feasible if all reduced costs are nonnegative. We calculate the reduced costs as part of the dual solution. We compute the dual basic solution $\tilde{\pi}, \tilde{d}$ as shown in Formula (2.5):

$$\tilde{d}_{[B]} := 0, \quad \tilde{\pi} := (A_{[B]})^{-t} c_{[B]}, \quad \tilde{d}_{[N]} := c_{[N]} - (A_{[N]})^t \tilde{\pi}.$$

We obtain as reduced costs $\tilde{d}_{[B]} = 0$ for all indices in B by definition. Furthermore, for the nonbasic indices N , we obtain

$$\tilde{d}_{[N]} = (30.070, 3.696, 0.232, 4.536, 0).$$

In summary, all reduced costs are nonnegative. Hence, B is dual feasible.

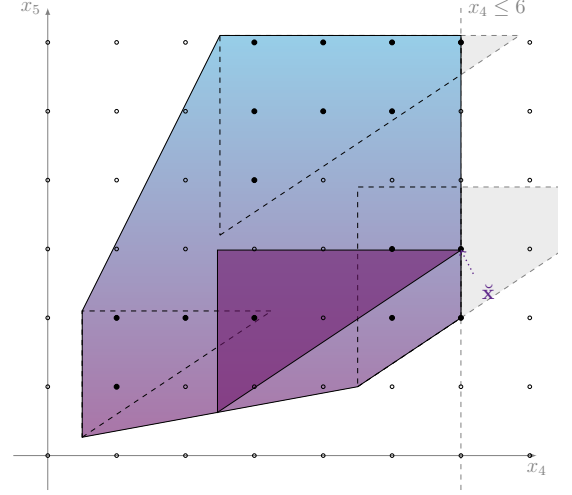
We have established that B is primal and dual feasible. Therefore, B is optimal for \check{P} due to the strong duality theorem. An optimal solution in the five original variables x_1, \dots, x_5 is given by the corresponding basic solution $\tilde{x}_{[1,2,3,4,5]}$.

We examine the LP relaxation \check{P} of our MIP P from Example (2.1) graphically in Figure 2.2. The figure shows the projection of the convex hull of all feasible solutions to the LP relaxation \check{P} in the x_4 - x_5 -plane as one single shaded region. For a better orientation, the three feasible regions depending on the choice between x_1 , x_2 , or x_3 are shown as dashed triangles.

This is indeed what happens. The optimal solution for \check{P} is $\check{x} = (0.509, 0.491, 0, 6, 2.984)^t$.

The solid triangle in the figure corresponds to the feasible region of the LP relaxation for x_1, x_2, x_3 set to their values in \check{x} . In other words, this solid triangle is a convex combination of the triangles for $x_1 = 1$ and $x_2 = 1$ corresponding exactly to the values of $\check{x}_1 = 0.509$ and $\check{x}_2 = 0.491 = 1 - \check{x}_1$.

Figure 2.2: Illustration of the LP relaxation of MIP Example (2.1) projected into the x_4 - x_5 -plane. The dashed triangles represent the boundaries of the feasible regions depending on the choice between x_1 , x_2 , or x_3 . The shaded hexagon represents the (projection of) all solutions of the LP relaxation \check{P} . It is the convex hull of the three extremal feasible regions. Note that while the LP solution $\check{x} = (0.509, 0.491, 0, 6, 2.984)^\top$ is a vertex solution of the 5-dimensional LP-feasible region, this is not the case in the projection to the x_4 - x_5 -plane.



2.2.2 The Dual Simplex Algorithm

With the notion of a primal or dual feasible basis, we are ready to state the dual simplex algorithm, see Algorithm 1. In essence, the algorithm visits a sequence of dual feasible bases with increasing cost until it becomes primal feasible. In order to pivot from one basis to the next, a carefully chosen basis index is exchanged by a non-basic index. The dual simplex algorithm guarantees that each of these basis updates preserves dual feasibility.

Algorithm 1 proceeds in iterations which are counted by k . If the initial basis is not primal feasible, the main loop starts with the pricing step in line 4. The idea of the pricing step is to select a basic variable whose current basic solution value violates the nonnegativity condition of P . This simple variant uses the index corresponding to the most negative basic solution value. In practice, this is a poor pricing rule, and much better rules exist [Forrest and Goldfarb, 1992].

Our goal is to resolve the primal infeasibility $\tilde{x}_{B_i^{(k)}}^{(k)} < 0$. We do this by removing $B_i^{(k)}$ from the basis (and therefore, we get $\tilde{x}_{B_i^{(k)}}^{(k+1)} = 0$).

But this forces us to select a new variable index $j \in N^{(k)}$ to enter the basis. Now, in order to preserve dual feasibility, we have to select j such that all reduced costs stay non-negative.

For the selected index i , we currently have $\tilde{d}_{B_i^{(k)}}^{(k)} = 0$. In order to preserve dual feasibility, we have to ensure that $\tilde{d}_{B_{i'}^{(k)}}^{(k+1)} = 0$ for all other elements $i' \neq i$ in the basis. Therefore, the overall change in the reduced costs needs to be of the form $\tilde{d}_{[B^{(k)}]}^{(k+1)} = \tilde{d}_{[B^{(k)}]}^{(k)} + \theta e_i$, the i 'th unit vector, with some $\theta \geq 0$. The values of the reduced costs of the nonbasic indices will change as $\tilde{d}_{[N^{(k)}]}^{(k+1)} = \tilde{d}_{[N^{(k)}]}^{(k)} - \theta \alpha_{[N^{(k)}]}$. The direction

Algorithm 1: The dual simplex algorithm

Input: Primal LP P , dual feasible basis $B = B^{(0)} = (B_1^{(0)}, \dots, B_m^{(0)})$.
Output: Basis B' that is either optimal for P , if it is primal feasible, or that proves infeasibility of P .

```

1  $k \leftarrow 0$ ;
2 Compute basic solution  $\tilde{x}^{(k)}$  as by Definition 2.2 and reduced costs  $\tilde{d}^{(k)}$ ;
3 while  $\tilde{x}^{(k)} \not\geq 0$  do
4   Select  $i \leftarrow \operatorname{argmin} \left\{ \tilde{x}_{B_{i'}^{(k)}}^{(k)} : i' \in \{1, \dots, m\} \right\}$ ;           /* pricing */
5   Compute  $y'$  that satisfies  $(A_{[B^{(k)}]})^t y' = e_i$ ;           /* BTRAN */
6   Set  $\alpha_{[B^{(k)}]} \leftarrow -e_i$ ,  $\alpha_{[N^{(k)}]} \leftarrow -(A_{[N^{(k)}]})^t y'$ ;
7   if  $\alpha_{[N^{(k)}]} \leq 0$  then
8     | return  $B^{(k)}$ ;           /*  $P^{\text{dual}}$  unbounded  $\Rightarrow P$  infeasible */
9   end
10  Determine  $j \leftarrow \operatorname{argmin} \left\{ \frac{\tilde{d}_{j'}^{(k)}}{\alpha_{j'}} : j' \in N^{(k)}, \alpha_{j'} > 0 \right\}$ ;           /* Ratio test */
11  Update Basis:  $B^{(k+1)} \leftarrow (B_1^{(k)}, \dots, B_{i-1}^{(k)}, j, B_{i+1}^{(k)}, \dots, B_m^{(k)})$ ;
12  Compute new basic solution  $\tilde{x}^{(k+1)}$  as by Definition 2.2;
13  Update reduced costs  $\tilde{d}^{(k+1)} \leftarrow \tilde{d}^{(k)} - \frac{\tilde{d}_j^{(k)}}{\alpha_j} \alpha$ ;
14   $k \leftarrow k + 1$ ;
15 end
16 return  $B^{(k)}$ ;

```

$\alpha_{[N^{(k)}]}$ is uniquely determined in accordance with the rule (2.5) for computing the dual basic solution.

Since we want to maintain dual feasibility, all reduced costs must stay nonnegative. Therefore, for all indices $j \in N^{(k)}$ where $\alpha_j > 0$, θ must not exceed $\frac{\tilde{d}_j^{(k)}}{\alpha_j}$. This is ensured by the ratio test in line 10.

The π variables will be updated as $\tilde{\pi}^{(k+1)} \leftarrow \tilde{\pi}^{(k)} - \theta y'$, where y' is the result of the *backward transformation* (BTRAN) operation. The change in the dual objective from iteration k to iteration $k + 1$ is computed as follows.

$$\begin{aligned} \left(\tilde{\pi}^{(k+1)}\right)^t b &= \left(\tilde{\pi}^{(k)} - \theta y'\right)^t b \\ &= \left(\tilde{\pi}^{(k)}\right)^t b - \theta e_i^t \left(A_{[B^{(k)}]}\right)^{-1} b \\ &= \left(\tilde{\pi}^{(k)}\right)^t b - \theta \tilde{x}_{B_i^{(k)}}^{(k)} \\ &\geq \left(\tilde{\pi}^{(k)}\right)^t b. \end{aligned}$$

The last inequality results from the strictly negative entry $\tilde{x}_{B_i^{(k)}}^{(k)}$, and the nonnegativity of θ . If $\theta > 0$, the inequality becomes strict. This also shows why θ needs to be bounded by at least one positive entry of $\alpha_{[N^{(k)}]}$.

Otherwise, we could raise the objective value arbitrarily large while preserving dual feasibility. In this case we have proven that P^{dual} is unbounded and P is infeasible. There are cases where the algorithm finishes in 0 iterations, namely if the initial basis $B^{(0)}$ is already primal feasible or if P^{dual} is shown to be unbounded. In the latter case, P must be infeasible because a primal feasible basis bounds the possible values of $\pi^t b$ from above due to the weak duality theorem.

Ultimately, we are interested in solving a MIP. We have taken an important step now that we are able to solve the LP relaxation \check{P} using the dual simplex algorithm.

Remarks We presented a simplified version of the dual simplex algorithm where we treat bounds on variables as additional rows in the matrix A . There exist more efficient procedures that treat variable bounds implicitly in the algorithm instead, thereby reducing the number of rows of the (transformed) primal and dual LP relaxations.

We assume in this section that we already have a dual feasible basis to start the dual simplex algorithm. It is sometimes easy to construct an initial dual feasible basis, for example if all variables have finite lower and upper bounds. In general, however, an initial dual feasible basis must be constructed first.

In this section, reduced costs are simply nonnegative, and variables have lower bounds of 0 and infinite upper bounds. In Section 6.2, we use a generalization of reduced costs by taking the lower and upper bounds into account.

Before the dual simplex algorithm was introduced by Lemke [1954] in the context of game theory, the primal simplex algorithm for solving linear programs (LPs) was

introduced by Dantzig [1951]. The primal simplex algorithm starts from a primal feasible basis. In each iteration, it selects a nonbasic variable with negative reduced costs to enter the basis while preserving primal feasibility.

An optimal basis need not be unique. Also, at intermediate steps of the dual simplex algorithm, it may happen that the step length θ is zero. We call such a pivot *degenerate*. After a degenerate pivot, care has to be taken to prevent the dual simplex algorithm from so-called cycling, which denotes the selection of the same basis $B^{(k)}$ in a later iteration $k + t$. One of the first discussions of cycling in the case of the dual simplex algorithm can be found in [Beale, 1955].

In the pricing step in line 4 of Algorithm 1, we use a “textbook approach” by selecting the most negative primal basic solution. There exist other pricing strategies for the dual simplex algorithm that require fewer iterations in practice, see also Chapter 7.

In Algorithm 1, we compute the basic solution and reduced costs from scratch in each iteration. Moreover, in the BTRAN operation in line 5, we solve a linear system, which requires a factorization of the transpose of the basis matrix $(A_{[B^{(k)}]})^t$ in each iteration. In practice, it is possible to maintain a single factorization for this purpose, which is incrementally updated in each iteration. We refer to [Forrest and Tomlin, 1972] for details about such updates.

The simplex algorithm was shown to exhibit exponential worst case behavior [Klee and Minty, 1970], and new algorithms for solving LPs in polynomial time were introduced by Khachiyan [1979] and later by Karmarkar [1984]. The latter method, known as interior-point or barrier algorithm, is run in modern MIP solvers during the so-called concurrent phase against the primal and dual simplex algorithms in parallel. During the branch-and-bound search, the dual simplex algorithm is by far the most commonly used algorithm for solving LP relaxations. The reason is that branching on a fractional variable of an optimal LP solution preserves the dual feasibility of the basis, whereas primal feasibility is lost. The dual simplex algorithm can therefore be warmstarted from the previous basis. It can often restore primal feasibility within a few simplex iterations. The barrier algorithm lacks this superior warmstarting property of the dual simplex algorithm [Potra and Wright, 2000].

2.3 The Branch-and-Bound Algorithm

The branch-and-bound algorithm is commonly attributed to the works of Land and Doig [1960] and Dakin [1965]. For more information on the historical development of the method, please see [Cook, 2012].

The idea behind branch-and-bound is as follows: a MIP P is recursively split into smaller subproblems, thereby creating a *search tree* and implicitly enumerating all potential assignments of the integer variables. The root node of the search tree represents the original input MIP P . Each descendant node v represents a MIP P_v that was derived

from P by a sequence of additional restrictions imposed via branching. Due to the close relationship between a node v and the corresponding MIP P_v , we will use these two terms interchangeably, i.e., we use the simplified notation v for the corresponding MIP P_v , as well. As potential branching disjunctions, we only consider restrictions of the lower and upper bounds of integer variables in this thesis.

From now on, we will always assume that the LP relaxation of P is infeasible or has a finite optimal objective value. This is always the case of all variable bounds are finite. Moreover, boundedness is not a strong assumption in the sense that real-world MIP instances that originate from industrial applications are usually bounded. Before we dive into the algorithm, we highlight its two fundamental concepts: branching and bounding.

2.3.1 Fundamental Concepts

The integer variables that take a noninteger value in an LP solution \check{x} have a special role.

Definition 2.3 (Fractionals). *Let \check{P} denote the LP relaxation of a MIP P with integrality restrictions \mathcal{I} , and let \check{x} denote an optimal solution of \check{P} . The set*

$$\mathcal{F} := \{j \in \mathcal{I} : \check{x}_j \notin \mathbb{Z}\} \quad (2.6)$$

is called fractionals, and every variable $j \in \mathcal{F}$ is called fractional (variable). For each fractional variable $j \in \mathcal{F}$ we define its up-fractionality and down-fractionality as

$$f_j^+ := \lceil \check{x}_j \rceil - \check{x}_j \quad \text{and} \quad f_j^- := \check{x}_j - \lfloor \check{x}_j \rfloor,$$

respectively.

Fractional variables are the most popular candidates for problem subdivision during the branch-and-bound algorithm.

Example 3. (Example 1 continued) Consider again the MIP P from Example (2.1), whose LP relaxation is visualized in Figure 2.2. The optimal solution for the LP Relaxation \check{P} is $\check{x} = (0.509, 0.491, 0, 6, 2.984)^\top$, which is quite far from the (in this case unique) optimal solution of P , $\tilde{x}^{\text{opt}} = (0, 0, 1, 6, 6)^\top$. Interestingly, x_3 is set to 0 in \check{x} , in other words, it is not even partially selected, in contrast to x_1 and x_2 . The set of fractionals arising from \check{x} is $\mathcal{F} = \{1, 2, 5\}$.

The term *branching* denotes a problem subdivision of a given MIP P . It works as follows. We first solve the LP relaxation \check{P} . If the LP solution value $\check{Z} = \infty$, i.e., if \check{P} is infeasible, this also proves the infeasibility of P and the solution process terminates. Otherwise, since we assume that \check{P} is bounded, we obtain a finite \check{Z} together with an LP solution \check{x} that attains \check{Z} .

We immediately inspect the fractional variables \mathcal{F} in \tilde{x} . If $\mathcal{F} = \emptyset$, \tilde{x} fulfills all integrality requirements of P . Hence, we have $\tilde{x} \in \mathcal{S}_P$, and since $c^t \tilde{x} = \check{Z} \leq Z^{\text{opt}}$ due to inequality (2.2), \tilde{x} is a proven optimal solution for P .

Most likely, however, \tilde{x} violates some integrality requirements of P . In this case the set of fractionals $\mathcal{F} \neq \emptyset$ is nonempty. Now, branching consists of splitting the problem into two or more subproblems. Usually, this is done in such a way that \tilde{x} is no longer feasible for the LP relaxations of each of the subproblems.

In practice, one often resorts to so-called "branching on variables". This special case of branching consists of selecting any fractional variable $j \in \mathcal{F}$, and creating two new MIP problems P' , P'' with refined solution sets by restricting the bounds of j as follows:

$$\mathcal{S}_{P'} := \mathcal{S}_P \cap \{x_j \leq \lfloor \tilde{x}_j \rfloor\} \quad \text{and} \quad \mathcal{S}_{P''} := \mathcal{S}_P \cap \{x_j \geq \lceil \tilde{x}_j \rceil\}. \quad (2.7)$$

Since in every solution $\tilde{x} \in \mathcal{S}_P$, j has an integer solution value $\tilde{x}_j \in \mathbb{Z}$, it holds that either $\tilde{x}_j \leq \lfloor \tilde{x}_j \rfloor$, in which case $\tilde{x} \in \mathcal{S}_{P'}$, or $\tilde{x}_j \geq \lceil \tilde{x}_j \rceil$, in which case $\tilde{x} \in \mathcal{S}_{P''}$. From this observation, we can safely state that branching disjunction (2.7) preserves each feasible solution for P in (exactly) one of the two new solution sets, i.e.

$$\mathcal{S}_{P'} \cup \mathcal{S}_{P''} = \mathcal{S}_P. \quad (2.8)$$

One of the benefits of the branching disjunction (2.7) lies in the tightening of the LP relaxations \check{P}' , \check{P}'' . Concretely, it holds that $\check{Z}_{P'} \geq \check{Z}_P$ and that $\check{Z}_{P''} \geq \check{Z}_P$ because of the additional restriction on the bound of j imposed by a branching disjunction (2.7). Note that \tilde{x} is not feasible anymore for either \check{P}' or \check{P}'' .

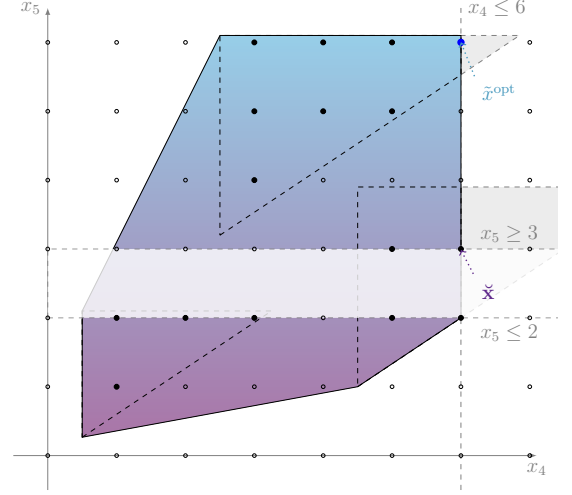
Combining the above inequalities and Equation (2.8), we obtain the following inequality on the lower bounds.

$$\check{Z}_P \leq \min\{\check{Z}_{P'}, \check{Z}_{P''}\} \leq \min\{Z_{P'}^{\text{opt}}, Z_{P''}^{\text{opt}}\} = Z_P^{\text{opt}} \quad (2.9)$$

In a nutshell, the branching disjunction (2.7) preserves all solutions for P and possibly tightens the lower bound on Z_P^{opt} at the price of introducing two MIPs that must be solved in order to solve P .

As most readers might expect at this point, a single branching operation is usually not enough to solve most MIPs. For a notable example class of MIPs where a single branching operation suffices, we refer to Le Bodic and Nemhauser [2015]. The general case, however, requires performing branching (2.7) multiple times in a recursive manner, thereby creating a search tree. The subproblems P' and P'' created by branching become the *children* of P in the search tree, and P is accordingly called the *parent* of P' and P'' . After the first branching, the algorithm will continue by selecting one of the two newly created P' or P'' , solving its LP relaxation, and eventually branching again on a fractional variable from the obtained LP relaxation solution.

Figure 2.3: Illustration of a branching disjunction of MIP Example (2.1) projected into the x_4 - x_5 -plane. The solution of the LP relaxation was $\check{x} = (0.509, 0.491, 0, 6, 2.984)^t$, with three fractional variables $\mathcal{F} = \{1, 2, 5\}$. After branching on the variable x_5 , we obtain two subproblems, one with the additional inequality $x_5 \leq 2$ and one with the additional inequality $x_5 \geq 3$. No integer solution is discarded by this split. In particular the optimal solution $\tilde{x}^{\text{opt}} = (0, 0, 1, 6, 6)^t$ is contained in the $x_5 \geq 3$ subproblem.



The root node of the search tree, v_0 , represents P itself. For our purposes, it will be convenient to partition the nodes of the search tree into the following three sets. Upon creation in iteration k of the branch-and-bound algorithm 2, each node is first assigned to the set of *open nodes* V_k^{open} . V_k^{open} represents the list of unsolved subproblems from which the algorithm selects in each iteration.⁴

When v gets selected for processing in a subsequent iteration $k' \geq k$, v is removed from $V_{k'}^{\text{open}}$. Its LP relaxation is solved. Depending on the result, v either enters the set of *inner nodes* $V_{k'}^{\text{inner}}$, if a branching was performed on P_v . Otherwise, v enters the set of *final leaf nodes* $V_{k'}^{\text{leaf}}$, namely if its LP relaxation \check{P}_v is either infeasible, or if the optimal LP objective value exceeds the primal bound, or if solving \check{P}_v yields an integer feasible solution $\check{x}^v \in \mathcal{S}_P$.

The nodes of the search tree at iteration k are given by the union of the three above sets: $V_k := V_k^{\text{inner}} \cup V_k^{\text{leaf}} \cup V_k^{\text{open}}$. We use two intuitive operators: $\text{children}(v)$ denotes the set of children of an (inner) search tree node $v \in V_k^{\text{inner}}$ created by branching; $\text{parent}(v)$ denotes the immediate parent node of $v \in V_k \setminus \{v_0\}$ whose branching led to the creation of v .

Besides branching, the second fundamental concept of the branch-and-bound algorithm is the availability of lower and upper bounds for the optimal solution value Z^{opt} . The first such lower bound is obtained after solving the LP relaxation of the root node (in iteration 1 of the branch-and-bound algorithm 2). Due to inequality 2.9, a tighter bound for Z^{opt} becomes available after solving the LP relaxations of the children, grand children etc. of P .

⁴The unprocessed subproblems are also called “node frontier” [Koch et al., 2013].

Definition 2.4 (Node/Global Dual Bound). Let P be a MIP and let k be an iteration of the branch-and-bound algorithm 2. Let the nodes of the search tree be $V_k = V_k^{inner} \cup V_k^{open} \cup V_k^{leaf}$. For an inner node $v \in V_k^{inner}$, we denote by \check{Z}_{P_v} the optimal solution of its LP relaxation. For a node $v \in V_k$, we define its node dual bound as

$$z_k^*(v) := \begin{cases} -\infty, & \text{if } k = 0, \\ \infty, & \text{if } v \in V_k^{leaf}, \\ \check{Z}_{P_{parent(v)}}, & \text{if } v \in V_k^{open}, \\ \min \{z_k^*(v') : v' \in children(v)\}, & \text{if } v \in V_k^{inner}. \end{cases}$$

The node dual bound of the root node v_0 is called (global) dual bound and denoted by

$$Z_k^* := z_k^*(v_0).$$

Due to inequality (2.9), the node dual bound $z_k^*(v)$ of each node $v \in V_k$ is nondecreasing in k , in particular the global dual bound Z_k^* . During the course of the branch-and-bound algorithm, when nodes are solved and branched, the newly created children first inherit the (necessarily finite) LP objective value of their parent until their own LP relaxation is solved in a subsequent iteration. Conversely, the smaller of the children's dual bounds of an inner node $v \in V_k^{inner}$ represents a lower bound on the best solution of the MIP corresponding to v that has not been discovered until iteration k .

If the node LP relaxation solution \check{x}^v is feasible for P , no further branching is needed on v . Therefore, v is added to the final leaf nodes V_{k+1}^{leaf} . From iteration $k+1$ onwards, it has a node dual bound of $z_{k+1}^*(v) = \infty$ due to Definition 2.4.

The algorithm maintains a set of feasible solutions $S_k \subset \mathcal{S}_P$. We call S_k *solution pool*. The solution pool is initially empty, i.e. $S_0 = \emptyset$. Each integer feasible LP relaxation solution \check{x}^v is added to the solution pool.

We call the solution in the solution pool with the smallest objective value the *incumbent*, which we denote by

$$\tilde{x}^{\text{best}} := \operatorname{argmin} \{c^\top \tilde{x} : \tilde{x} \in S_k\}. \quad (2.10)$$

We call the objective value of the current incumbent *primal bound*. We denote the primal bound by

$$Z_k := c^\top \tilde{x} : \tilde{x} \in S_k. \quad (2.11)$$

It follows that at the beginning of the search, $Z_k = +\infty$.

In general, the primal bound represents an upper bound on Z^{opt} . It is not necessarily optimal for P because v may be a node deep in the search tree with many additional bound restrictions on integer variables imposed through a branching disjunction (2.7).

Each feasible solution found during the course of the branch-and-bound algorithm potentially provides an improved primal bound. Besides LP relaxation solutions, also primal heuristics (see Section 2.6) are used to decrease the primal bound faster (see also line 13 of Algorithm 2).

Even a suboptimal primal bound Z_k can be used to filter out open nodes based on their node dual bound. If there exists a node $v' \in V_k^{\text{open}}$ with node dual bound $z_k^*(v') \geq Z_k$, it is unnecessary to continue searching under v' for improving solutions. Therefore, v' can be added to the final leaf nodes V_{k+1}^{leaf} . Eliminating such open subproblems based on node dual bounds is called *bounding* or also *pruning*.

Due to bounding, a branch-and-bound search can often be completed in reasonable time even on large MIPs with thousands or even millions of integer variables, where a complete enumeration of the solution space would be hopeless.

2.3.2 The Algorithm

We are now ready to formulate a version of the branch-and-bound algorithm in our notation. Algorithm 2 works essentially as described in the previous section. In iteration k , the algorithm selects an open subproblem v from the nodes V_k^{open} , solves its LP relaxation \tilde{P}_v , and either branches on v or prunes v depending on the outcome. Our Algorithm 2 differs from existing descriptions of the B&B algorithm in that it records a sequence of so-called search states. Search states are a useful concept for the discussion of the tree size estimation techniques introduced in Chapter 8.

Definition 2.5 (Search State). *Let P be a MIP that is solved by the B&B algorithm. For an iteration $k = 0, \dots, U$ of the B&B algorithm, the search state T_k consists of*

$$T_k := (V_k^{\text{inner}}, V_k^{\text{leaf}}, V_k^{\text{open}}, Z_k, z_k^*, S_k),$$

where V_k^{inner} are the inner nodes at iteration k ; V_k^{leaf} are final leaves at iteration k , i.e., nodes that were infeasible, or whose LP relaxation was integer feasible, or nodes that were pruned; V_k^{open} are open nodes at the end of iteration k , which are still to be processed; Z_k is the primal bound at iteration k ; $z_k^* : V_k \rightarrow \mathbb{Q}_{\pm\infty}$ are the node dual bounds for each node; and $S_k \subset \mathcal{S}_P$ denotes the set of already found integer feasible solutions.

A search state summarizes all the information available at (the end of) iteration k of the branch-and-bound algorithm. A *search state sequence* consists of all the $k + 1$ search states from the initial search state until the current iteration.

As input, Algorithm 2 receives a search state sequence and a desired limit K on the number of (additional) iterations that should be performed. This unusual input specification allows us to launch Algorithm 2 as a subroutine, and to interrupt it for a decision whether to continue or perhaps restart the search, based on an inspection of the current search state (sequence).

Algorithm 2: branch-and-bound algorithm for MIP

Input: Search state sequence $T = (T_0, \dots, T_k)$, additional iteration limit $K \geq 1$

Output: Search state sequence of length (at most) $k + K + 1$

```

1  $k_0 \leftarrow k;$  // save current length of sequence
2 while  $V_k^{\text{open}} \neq \emptyset$  and  $k < k_0 + K$  do
3    $k \leftarrow k + 1;$ 
4   select  $v \in V_{k-1}^{\text{open}}$  and set  $V_k^{\text{open}} \leftarrow V_{k-1}^{\text{open}} \setminus \{v\};$ 
5   apply preprocessing to  $v$ ;
6    $\check{Z}_{P_v} \leftarrow$  LP objective value of  $\check{P}_v$ ; //  $\infty$  if infeasible
7    $Z_k \leftarrow Z_{k-1}$  and  $S_k \leftarrow S_{k-1};$ 
8   if  $\check{Z}_{P_v} < Z_{k-1}$  then
9     if  $\check{x}^v \in S_P$  then
10        $Z_k \leftarrow \check{Z}_{P_v}$  and  $S_k \leftarrow S_k \cup \check{x}^v;$  // feasible LP relaxation solution
11     end
12   end
13   use primal heuristics to (potentially) extend  $S_k$  and improve  $Z_k$ ;
14   if  $\check{Z}_{P_v} = \infty$  or  $\check{Z}_{P_v} \geq Z_k$  then
15      $V_k^{\text{inner}} \leftarrow V_{k-1}^{\text{inner}}, V_k^{\text{leaf}} \leftarrow V_{k-1}^{\text{leaf}} \cup \{v\};$  // add  $v$  to terminal nodes
16   else branch on  $v$ :
17      $V_k^{\text{leaf}} \leftarrow V_{k-1}^{\text{leaf}},$ 
18      $V_k^{\text{inner}} \leftarrow V_{k-1}^{\text{inner}} \cup \{v\};$ 
19     select fractional  $j \in \mathcal{F};$  //  $\mathcal{F} \neq \emptyset$  in this path of the algorithm
20     create two children  $v'$  and  $v''$  by creating a branching disjunction (2.7) for
        $j$ :
       
$$S_{v'} \leftarrow S_v \cap \{x_j \leq \lfloor \check{x}_j^v \rfloor\} \quad \text{and} \quad S_{v''} \leftarrow S_v \cap \{x_j \geq \lceil \check{x}_j^v \rceil\}$$

        $V_k^{\text{open}} \leftarrow V_k^{\text{open}} \cup \{v', v''\};$ 
21   end
22    $T \leftarrow (T_i = (V_i^{\text{inner}}, V_i^{\text{leaf}}, V_i^{\text{open}}, Z_i, z_i^*, S_i) : i = 0, \dots, k);$ 
23 end
24 return  $T;$ 

```

We start the B&B search algorithm from the *initial search state* defined as

$$T_0 := \left(V_0^{\text{inner}} = \emptyset, V_0^{\text{leaf}} = \emptyset, V_0^{\text{open}} = \{v_0\}, Z_0 = \infty, z_0^* = -\infty, S_0 = \emptyset \right).$$

where v_0 denotes a (yet) isolated node that represents the original MIP problem without any branching restrictions. With the intention of completing the B&B search without interruption, we assume an iteration limit of $K = \infty$.

At the beginning of every iteration $k = 1, 2, \dots$, a node v is selected from the currently open nodes. In line 5, node preprocessing is applied to tighten the variable bounds of the MIP at node v , see also Section 2.3.3. The first iteration $k = 1$ of Algorithm 2 selects v_0 as the only open node and solves the LP relaxation \check{P} in line 6.

The primal bound Z_k is updated to \check{Z}_{P_v} , if $\check{Z}_{P_v} < Z_k$ and the solution \check{x}^v is feasible for P , otherwise $Z_k = Z_{k-1}$. Recall that if the optimal solution to the root LP relaxation already satisfies all integrality requirements, it is necessarily optimal for P itself. Even in this extreme case that the solution process only requires a single node, i.e., the root node v_0 , there will be two search states T_0 and T_1 . The use of primal heuristics in line 13 may also provide an improved or even optimal primal bound.

In all cases in which $Z_1 > \check{Z}_{P_{v_0}}$, at least one branching operation takes place. Every time that \check{Z}_{P_v} is still smaller than the current primal bound Z_k , a branching operation in line 16 creates two children for the currently active node v and adds them to V_k^{open} . It should be noted that the algorithm can only reach line 16 and the following lines if the set of fractionals \mathcal{F} is nonempty.

Namely, if the LP relaxation is infeasible, we have $\check{Z}_{P_v} = \infty$, and v is added to the set of leaves. In case that \check{x}^v satisfies all integrality restrictions, the primal bound Z_k has already been updated when \check{x}^v was added to the solution pool in line 10. Therefore, $\check{Z}_{P_v} \geq Z_k$ if $\mathcal{F} = \emptyset$.

Every node $v \in V_k^{\text{leaf}} \cup V_k^{\text{inner}}$ is denoted *solved*. Note that the number of solved nodes increases by 1 with every iteration, such that $k = |V_k^{\text{leaf}} \cup V_k^{\text{inner}}|$, i.e., the number of solved nodes is always equal to k . An important detail of Algorithm 2 is that pruning is performed explicitly so that each pruned node is counted. This ensures that each search state T_k represents a binary search tree, i.e., a tree in which each node is either a terminal node or an inner node with exactly two children.

It is not guaranteed that the branch-and-bound algorithm terminates. In fact, there exist examples where the procedure continues the search ad infinitum. The algorithm is finite if all integer variables have finite lower and upper bounds.

If the search terminates, let the final search state be denoted by T_U . We denote by $U^{\text{bb}} := |V_U|$ the total number of solved nodes. For every iteration $k = 0, \dots, U$ it holds that $Z^{\text{opt}} \leq Z_k$. At search state U , $Z^{\text{opt}} = Z_k$.

2.3.3 Algorithmic Components

Algorithm 2 provides a basic branch-and-bound algorithm. For the sake of notational brevity and clarity, we omit a lot of the details regarding particular choices of the algorithm such as which node to explore next, or which fractional variable to branch on. Such details are not essential for the algorithm to work correctly. However, in particular a careful branching decision is fundamental for a branch-and-bound implementation to work in practice, where simply choosing any fractional candidate is a hopeless rule.

In modern MIP solvers such as SCIP, the basic branch-and-bound method as presented in Algorithm 2 is enhanced by various auxiliary algorithms with the purpose of improving the primal or dual convergence of the branch-and-bound method. We call such auxiliary algorithms *solving components*. Among the most important types of solving components are

1. *Branching rules*: a scoring mechanism that ranks the fractionals for the decision on which of them a branching disjunction (2.7) should be imposed to split the current subproblem’s feasible region.
2. *Node selection*: A node selection rule determines the choice of the next open node from the search tree. Classical node selection rules include depth-first, breadth-first, best-bound and best-estimate [Bénichou et al., 1971].
3. *Primal heuristics*: Primal heuristics are auxiliary algorithms aimed at providing feasible solutions and improving the primal bound during search.
4. *Presolving*: Presolving (Also preprocessing) transforms the given problem instance into an equivalent instance that is (hopefully) easier to solve. Presolving removes redundant constraints or variables and strengthens the LP relaxation by exploiting integrality information. For more details on presolving, see [Achterberg et al., 2019] about presolving in general, and [Gamrath et al., 2015b] for SCIP-related information.
5. *Cutting plane separation*: Cutting planes separate the current LP relaxation solution from the convex hull of the solutions of the MIP. They are added to the LP relaxation to tighten it. For an overview of computationally useful cutting plane techniques, see [Cornuéjols, 2008; Marchand et al., 2002].

We now introduce SCIP, the MIP solver used throughout this thesis. Afterwards, we will take a closer look at branching rules in Section 2.5, primal heuristics in Section 2.6, and presolving in Section 2.7 as a foundation for the methods introduced in the subsequent chapters.

2.4 SCIP–Solving Constraint Integer Programs

All computational experiments for this thesis have been conducted with the academic MIP solver SCIP, which is short for *Solving Constraint Integer Programs*. SCIP is a state-of-the-art code for solving mixed-integer programs written in the programming language C [Kernighan and Ritchie, 1988], with interfaces to other languages such as Java and Python. SCIP has been introduced by Achterberg [2007a, 2009] as a framework for solving so-called Constraint Integer Programs (CIPs), a generalization of mixed-integer programs. After its release in 2007, SCIP has been extended to incorporate solving capabilities for mixed-integer quadratic [Berthold et al., 2010] and also mixed-integer nonlinear optimization problems (MINLPs), see [Vigerske, 2013].

SCIP does not solve LPs by itself; for solving LP relaxations, SCIP uses an external LP solver such as **SoPlex** [Soplex]. By the time of this writing, the newest version of SCIP is version 7.0 [Gamrath et al., 2020], which incorporates all the (polished and documented) code used for the computational experiments in this thesis. Since version 3.2, each major release of a new SCIP version is accompanied by a technical report (the “release report”) with detailed descriptions of new features and also conclusive performance results. The full source code of the SCIP software can be obtained as part of the SCIP optimization suite (that contains **SoPlex** as well) from www.scipopt.org.

SCIP has a plugin-based design to allow for user-written extensions of the so-called “core”, which implements a branch-and-bound algorithm as described in Section 2.3. Among others, SCIP provides particular plugin classes for all *solving components* introduced in Section 2.3.3 to enrich the basic B&B Algorithm 2, for example branching rules and primal heuristics. Each plugin added to SCIP registers a handful of callback functions that the SCIP core executes at the appropriate stages of the search. A branching rule plugin, for example, provides a callback function that implements the branching on a suitable fractional candidate in line 16 of Algorithm 2. When the core algorithm reaches the point at which a branching decision must be made, it calls the registered branching rules in decreasing order of a (user-specified) priority until one successfully splits the problem. If no branching rule has been registered, SCIP will simply branch on the first unfixed integer variable. SCIP already features a handful of branching rule plugins implementing certain popular choices such as strong branching, a pseudo-cost based branching rule [Bénichou et al., 1971], and hybrid branching [Achterberg and Berthold, 2009], which will be explained in the next section.

2.5 Branching Rules

Research on branching rules for MIP has been a focus of interest since the advent of the branch-and-bound procedure in the 1960’s [Dakin, 1965; Land and Doig, 1960]. *Pseudo-costs*, which measure the average objective gain for every integer variable, and

their use for branching first appeared in [Bénichou et al., 1971]. The use of degradation bounds for the pseudo-cost initialization was suggested in [Gauthier and Ribière, 1977]. Equipped with more computational power, strong branching was first applied in the context of the Traveling Salesman Problem [Applegate et al., 1998], whereas its first use for general MIP solving is attributed to the commercial MIP solver CPLEX.⁵ An important computational study of these and different node selection techniques has been conducted by Linderoth and Savelsbergh [1999]. Liberatore [2000] shows that the problem of choosing an optimal branching variable is \mathcal{NP} -hard in the context satisfiability (SAT) problems. This result also holds for MIP because MIP is a generalization of SAT.

As already briefly explained in Section 2.3.3, a *branching rule* is a scoring mechanism to guide the selection of a branching variable at each inner node of the search tree in line 16 of Algorithm 2. The decision on which fractional variable to branch is crucial for the success of the branch-and-bound search.

With the explicit intention to “fool MIP solvers”, Le Bodic and Nemhauser [2015] introduce a certain family of graphs for which they wish to determine the chromatic number.⁶ Even if the graphs grow in size, Le Bodic and Nemhauser [2015] prove that the corresponding MIP formulation of the studied graph family always admits a constant size branch-and-bound search tree of size $U^{\text{bb}} = 3$ in the best case in which a particular candidate variable is immediately selected at the root node. Inferior branching choices, however, may result in trees of exponential size. Le Bodic and Nemhauser [2015] demonstrate for several MIP solvers that these often fail to find the good branching decision on larger instances.

Recall that at a node v of the search tree with fractional LP solution \check{x}^v , branching consists of a problem split, the so-called branching disjunction (2.7),

$$\mathcal{S}_{P'_v} := \mathcal{S}_{P_v} \cap \{x_j \leq \lfloor \check{x}_j^v \rfloor\} \quad \text{and} \quad \mathcal{S}_{P''_v} := \mathcal{S}_{P_v} \cap \{x_j \geq \lceil \check{x}_j^v \rceil\}$$

after a suitable choice of a fractional $j \in \mathcal{F}$. Branching disjunctions on variables can be performed with little computational overhead by restricting the lower and upper bounds of j in the children.

Concretely, we obtain P'_v by setting $u'_j \leftarrow \lfloor \check{x}_j^v \rfloor$ and P''_v by setting $\ell''_j \leftarrow \lceil \check{x}_j^v \rceil$. In branching terminology, P'_v is also called the *down-child* of P_v because the domain of variable j is restricted to a lower interval than in P''_v , which is also called the *up-child*. If x_j is binary, i.e. $j \in \mathcal{B}$, branching on j always results in fixing x_j to 0 in the down-child and to 1 in the up child.

How to choose a good $j \in \mathcal{F}$? Ultimately, we would like to choose a fractional for branching that minimizes the size of the search tree under the current node. Recall that due to inequality (2.9), the LP relaxation value of the children will be no smaller than

⁵<https://www.ibm.com/analytics/cplex-optimizer>

⁶In graph theory, the *chromatic number* of a graph denotes the smallest possible number of colors in a vertex coloring such that each pair of adjacent vertices receives two different colors.

the LP relaxation value at the current node. Typically, branching rules aim at raising the difference between the parent and both child LP relaxations, which we call *gain*, in both subproblems as much as possible.

Definition 2.6. *Let P be a MIP and let $v \neq v_0$ be a node in the search tree. The gain of v is defined as the LP objective difference between \check{P}_v and its parent*

$$\Delta(P_v) = \check{Z}_{P_v} - \check{Z}_{P_{\text{parent}(v)}} \quad (2.12)$$

Due to the observations in Section 2.3, the gain $\Delta(P_v)$ is always nonnegative.

2.5.1 Strong Branching

Strong branching scores the fractional candidates by explicitly creating and solving the resulting child node LP relaxations for all possible branching disjunctions. The first use of such a technique in the context of branching is due to [Gauthier and Ribière, 1977]. The term strong branching was coined by [Applegate et al., 1998] in the context of solving traveling salesman problems.

Definition 2.7 (Strong Branching Score). *Let \mathcal{F} be the remaining fractionals at a node v of the $B\&B$ search tree, let $j \in \mathcal{F}$ be a candidate for branching, and let $\varepsilon > 0$. Let P_j^- denote the (potential) down-child after branching on j , and P_j^+ the potential up-child. We compute the strong branching score of j as*

$$\vartheta^{\text{str}}(j) := \max \left\{ \Delta(P_j^-), \varepsilon \right\} \cdot \max \left\{ \Delta(P_j^+), \varepsilon \right\} \quad (2.13)$$

The value of ε guarantees that the strong branching score is never zero even if one of the gains is zero. In SCIP, the value of ε equals 10^{-6} by default.

The strong branching rule selects the candidate that maximizes the strong branching score:

$$j \leftarrow \operatorname{argmax}_{j' \in \mathcal{F}} \vartheta^{\text{str}}(j'). \quad (2.14)$$

Since the gains in the children are unknown by the time a candidate needs to be selected, the strong branching rule determines $\vartheta^{\text{str}}(j)$ by virtually solving $2 \cdot |\mathcal{F}|$ child node relaxations and evaluating their gains (2.12). Although strong branching is guaranteed to select the best local candidate regarding the objective gain, the exhaustive solving of auxiliary LP relaxations makes the computational cost of this procedure often too expensive to be used throughout the search. However, it is well suited as an initialization method for pseudo-costs.

Figure 2.4 illustrates strong branching at the root node of our example MIP 1. The LP solution has three fractional candidates $\mathcal{F} = \{1, 2, 5\}$. Strong branching creates six child node relaxations in total. In the figure, we show the projections of each of these feasible regions in color. We blend out the part that is no longer feasible.

For example, in the child node relaxation P_1^+ that corresponds to the branching restriction $x_1 \geq 1$, we find ourselves in the triangle in the lower left. In the down branch $x_1 \leq 0$, however, all convex mixtures of x_2 and x_3 are still possible. The root node LP relaxation has an LP objective value of $\check{Z} = -58.023$. The strong branching LPs have gains of $\Delta(P_1^-) = 0.723$, $\Delta(P_1^+) = 12.473$, $\Delta(P_2^-) = 0.231$, $\Delta(P_2^+) = 1.323$, $\Delta(P_5^-) = 7.023$, and $\Delta(P_5^+) = 0.023$.

The strong branching score (2.13) of each fractional is simply the product of the two gains, and x_1 gets selected for branching.

Remarks The use of the product in the strong branch score (2.13) to combine the gains of the two individual directions was first proposed by Achterberg et al. [2004] as a means to favor candidates that yield a balanced improvement in both directions. Achterberg, Koch, and Martin [2004] showed that the product was empirically superior to previous methods that used a convex combination of the up and down gains. Recently, Le Bodic and Nemhauser [2017] proposed a novel function as a replacement of the product function. To this end, Le Bodic and Nemhauser [2017] derived an explicit combinatorial formula for the size of the resulting subtrees under the assumption that the gains stay constant. This method has also been recently made available in SCIP.

Gamrath [2013] improved the strong branching procedure of SCIP by applying domain propagation techniques at each sub-node in addition to solving the LP relaxation. Fischetti and Monaci [2012] observed unnecessary strong branching effort in the presence of *chimerical* variables, i.e. fractional variables with little or no effect on the objective of the LP solutions. They exploit this fact to safely ignore such candidates for the strong branching procedure. Strong branching as presented here performs a lookahead of one level into the prospective search tree. Gamrath [2020] has recently discussed a generalization of this concept called “lookahead branching” that strong branches deeper than one level on some candidate variables.

Cloud branching [Berthold and Salvagnin, 2013] has been proposed as a novel approach for better dealing with the frequent degeneracy of LP solutions. It computes several optimal LP solutions and considers variable fractionalities as intervals rather than points.

2.5.2 Pseudo-Cost Branching

The *pseudo-costs* [Bénichou et al., 1971] of a fractional variable $j \in \mathcal{F}$ are the most widely used estimation of the impact of a branching decision on the children objective gains $\Delta(P_j^-)$, $\Delta(P_j^+)$. The idea of the estimation is to use the average of the already observed gains after past branching decisions for each integer variable individually.

Concretely, let v be a node with LP relaxation \check{P}_v . Assume that P_v emerged from its parent node $\text{parent}(v)$ by branching down on a variable $j \in \mathcal{F}$ that was fractional in the

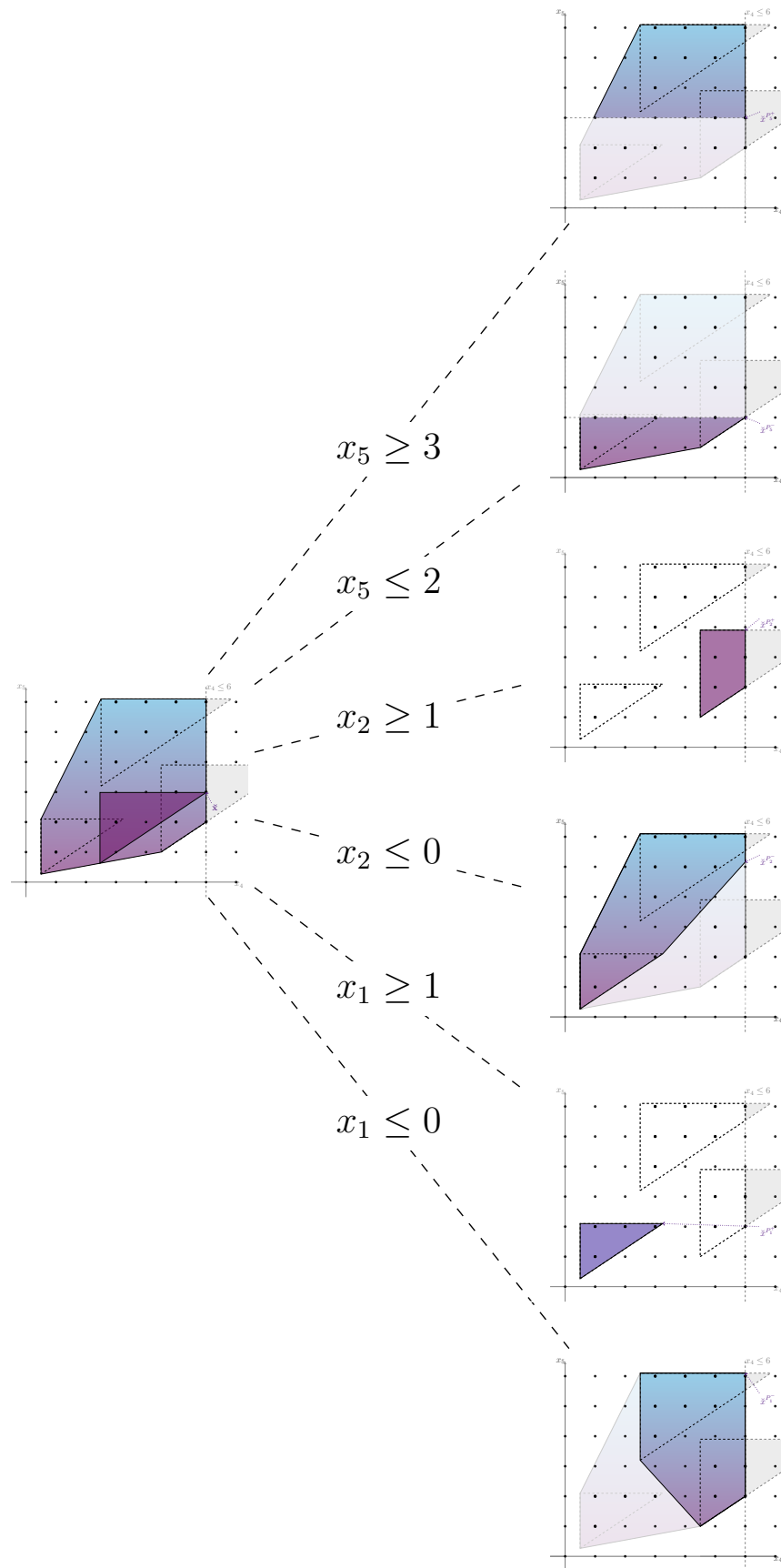


Figure 2.4: We illustrate strong branching on our example.

LP relaxation solution $\check{x}^{\text{parent}(v)}$. We call the normalized gain between the LP relaxation objectives of P_v and $\text{parent}(v)$,

$$\varsigma^-(P_v) := \frac{\Delta(P_v)}{\check{x}_j^{\text{parent}(v)} - \lfloor \check{x}_j^{\text{parent}(v)} \rfloor}$$

the *unit gain* of j at P_v . This unit gain is well-defined because $\check{x}_j^{\text{parent}(v)} - \lfloor \check{x}_j^{\text{parent}(v)} \rfloor > 0$ since j was fractional at $\text{parent}(v)$. Like the gain, the unit gain is always nonnegative.

Let $\mathcal{D}_j^- := \{P_1, \dots, P_{\eta_j^-}\}$ denote all feasible LP relaxations that resulted from a down branch on j and have been solved already. We average all observed unit gains via

$$\Psi_j^- := \sum_{P_i \in \mathcal{D}_j^-} \frac{\varsigma^-(P_i)}{\eta_j^-}. \quad (2.15)$$

By definition of an empty sum, $\Psi_j^- = 0$ if $\eta_j^- = 0$. Similarly, we define Ψ_j^+ over all feasible LP relaxations that have been solved after branching up on j . If η_j^- or η_j^+ is 0, we call j *uninitialized* in this direction.

Definition 2.8 (Pseudo-costs [Bénichou et al., 1971]). *Let $j \in \mathcal{I}$ be an integer variable of a MIP P and let Ψ_j^+ and Ψ_j^- denote the current average unit gains after branching up and down on j , respectively. We define the pseudo-cost function of j as*

$$\begin{aligned} \Psi_j : \mathbb{Q} &\rightarrow \mathbb{Q}_{\geq 0} \\ z &\mapsto \begin{cases} \Psi_j^+ \cdot z, & \text{if } z \geq 0 \\ -\Psi_j^- \cdot z, & \text{if } z < 0. \end{cases} \end{aligned} \quad (2.16)$$

At a node v for which $\check{x}_j^v \notin \mathbb{Z}$, the pseudo-costs for branching up and down on j are computed as $\Psi_j(f_j^+)$ and $\Psi_j(-f_j^-)$, respectively.

The definition of the pseudo-cost function ensures that pseudo-costs are always nonnegative. When the branching rule is invoked, pseudo-costs represent an estimation of the gains of the potential children P_j^- and P_j^+ of a fractional $j \in \mathcal{F}$. For the purpose of branching, it would be sufficient to restrict the domain of Ψ_j to the interval $[-1, 1]$ instead of \mathbb{Q} . However, in Chapter 6, we use this function to compute a score based on inputs for Ψ_j outside of $[-1, 1]$ in general.

For branching, the *pseudo-cost score function* combines the pseudo-cost functions

$$\vartheta^{\text{ps}}(j) := \max \left\{ \Psi_j(-f_j^-), \varepsilon \right\} \cdot \max \left\{ \Psi_j(f_j^+), \varepsilon \right\} \quad (2.17)$$

that estimate the objective gains in the (potential) children after branching on $j \in \mathcal{F}$ using the product score, following the same logic like the strong branching score (2.13). As before, the role of ε is to avoid a pseudo-cost score of zero if one of the two directions has a pseudo-cost estimation of zero. In contrast to the strong branching score, the pseudo-cost

score function is cheap to evaluate at every node. Practically, an implementation only needs to store the four values $\Psi_j^-, \eta_j^-, \Psi_j^+, \eta_j^+$, from which pseudo-costs can be efficiently computed and updated, but not the lists \mathcal{D}_j^- and \mathcal{D}_j^+ .

2.5.3 Reliability Branching

The pseudo-cost branching rule is an effective replacement of the strong branching rule at later stages of the search but lacks information at the beginning. For that reason, several combinations of pseudo-cost branching and strong branching have been developed: One possibility is to use a single strong branching initialization on uninitialized variables and pseudo-costs for every initialized candidate. Another possibility consists of strong branching at the topmost d levels of the tree and pseudo-cost branching at deeper levels. In SCIP, all uninitialized fractional variables use the average of all unit gains, i.e., from initialized variables and solved B&B node LP relaxations, for their pseudo-cost score. This idea is due to [Eckstein, 1994].

The state-of-the-art branching scheme, which is applied by most modern MIP solvers albeit the concrete implementation might vary, is *reliability branching* [Achterberg et al., 2004], which we discuss in more detail in Chapter 4. The basic idea of reliability branching is a distinction between candidate variables with “reliable” and “unreliable” pseudo-cost scores. Achterberg et al. [2004] suggests to base the reliability of a pseudo-cost score $\vartheta^{\text{ps}}(j)$ (2.17) on the number of past branching observations, denoted by η_j^-, η_j^+ . Too few observations render $\vartheta^{\text{ps}}(j)$ unreliable, and a strong branching score is computed instead.

2.5.4 Hybrid Branching

Pseudo-costs are not the only piece of information that the solver might learn about the integer variables that it branches on during the search. As shown in line 5 of Algorithm 2, node preprocessing is applied before the LP relaxation in iteration k is solved. Starting from the bound requirement imposed by branching, node preprocessing may often infer additional domain tightenings for other variables. This information can be stored as inference history, which measures the impact of a branching decision $j \in \mathcal{F}$ upon the domains of other variables.

Similarly to pseudo-costs, the average number of domain reductions inferred from branching on j is recorded and updated during the search to form the *inference score* $\vartheta^{\text{inf}}(j)$ as the product of the average domain reductions after up and down branches on j . An analogous score has also been proposed and successfully applied in the area of Constraint Programming (CP) [Refalo, 2004].

A subtlety in the Definition 2.8 of pseudo-costs is that they are only recorded from feasible child LP relaxations, i.e., nodes with a finite LP objective. If a node is cut off due to an infeasible LP relaxation, no pseudo-costs are recorded because the gain would be infinite in this case. Instead, the number of cases where branching up/down on

variable $j \in \mathcal{I}$ resulted in an infeasible LP relaxation is recorded as separate branching history.

The corresponding *cutoff score* $\vartheta^{\text{cut}}(j)$, which combines the two average numbers of infeasible nodes resulting from branching up and/or down on j , can be used for selecting a good branching variable, as well. By selecting a candidate with a high cutoff score for branching, chances are that the resulting child nodes are infeasible or that, at least, the subtree under the current node stays small.

In team sports like Basketball, the value of each player is determined not only by how often he or she scores in a game, but also how often he or she assisted in another player's score by passing the ball. Similarly, in MIP branching, an infeasible LP relaxation is often the result of the effort of a team of branching decisions rather than only the final one that is rewarded by the cutoff score.

The technique of finding a subset of branching decisions that represent the “reason” of an infeasibility is called *conflict analysis*. Conflict analysis originates from the area of Satisfiability (SAT) solving and has been successfully transferred into a MIP solving component of SCIP by Achterberg [2007b], with recent extensions by Witzig et al. [2017].

The theory behind conflict analysis requires too much notation overhead to be rolled out here for the sake of explaining a branching score. Briefly, a conflict clause in MIP is a disjunction of bound restrictions, which can be linearized if all involved variables are binary. Conflict clauses can be derived from infeasible subproblems using the so-called “conflict graph”.

A second form of conflict analysis consists of constructing so-called “Farkas proof”. A Farkas proof is a globally valid linear inequality that is violated within the local variable bounds. Farkas proofs are derived from an infeasible node LP relaxation, see [Witzig et al., 2017].

A conflict clause learned in one part of the search tree may help to detect the infeasibility of open nodes of the search tree, if their local bound changes violate the conflict clause. Ideally, a conflict clause contains only few variables such that it is active high in the search tree.

Conflict scores for branching in SCIP borrow an idea originally suggested in the area of SAT solving called variable state independent decaying sum [Moskewicz et al., 2001]. Variables are preferred if they are part of many recent conflicts. Every time a new conflict clause is added, the counters of the involved variable-direction combinations are incremented by 1. Periodically, the counters are multiplied with a small scalar less than 1 to geometrically decay the contribution of older conflict clauses on the score. We denote this *conflict score* by $\vartheta^{\text{conf}}(j)$, which multiplies the two counters, one per branching direction of j , as usual.

The combination of all the above scores and pseudo-costs into a single score was introduced as *Hybrid branching* by Achterberg and Berthold [2009]. Technically, Achterberg and Berthold [2009] observe that the four branching scores $\vartheta^{\text{ps}}(j)$, $\vartheta^{\text{inf}}(j)$,

$\vartheta^{\text{cut}}(j)$, and $\vartheta^{\text{conf}}(j)$ generally act on very different scales and need to be normalized first. Therefore, they propose to normalize each score in two steps as follows. First, they divide each score by the corresponding average score across all variables. For pseudo-costs, a fractionality of 0.5 is used to compute this average score. We denote the four average scores by $\overline{\vartheta^{\text{ps}}}$, $\overline{\vartheta^{\text{inf}}}$, $\overline{\vartheta^{\text{cut}}}$, and $\overline{\vartheta^{\text{conf}}}$. Second, the resulting scores are normalized using the function

$$s : \mathbb{R}_{\geq 0} \rightarrow [0, 1), \quad s(x) = \frac{x}{x + 1}.$$

Hybrid branching combines these ingredients into the following hybrid score:

$$\vartheta^{\text{hybrid}}(j) := w^{\text{ps}} s\left(\frac{\vartheta^{\text{ps}}(j)}{\overline{\vartheta^{\text{ps}}}}\right) + w^{\text{inf}} s\left(\frac{\vartheta^{\text{inf}}(j)}{\overline{\vartheta^{\text{inf}}}}\right) + w^{\text{cut}} s\left(\frac{\vartheta^{\text{cut}}(j)}{\overline{\vartheta^{\text{cut}}}}\right) + w^{\text{conf}} s\left(\frac{\vartheta^{\text{conf}}(j)}{\overline{\vartheta^{\text{conf}}}}\right). \quad (2.18)$$

Here, the four weights w^{ps}, \dots denote positive weights for the individual scores. By default, SCIP version 7.0 uses weights of $w^{\text{ps}} = 1.0$, $w^{\text{conf}} = 10^{-2}$, $w^{\text{inf}} = w^{\text{cut}} = 10^{-4}$ at the beginning of the search.

A fifth ingredient to the above hybrid score (2.18) of SCIP measures the average length of the conflicts of a variable, an idea due to [Kılınç Karzan et al., 2009]. By default, this score is disabled in SCIP.

While Achterberg and Berthold [2009] proposed the method with static weights as above throughout the search, several adaptive enhancements have been proposed. The first dynamic weight adjustment was implemented as a byproduct of the implementation of phase-specific settings for Chapter 5 based on the pruning behavior of the search. If many more nodes are detected to be infeasible compared to nodes that exceeded the primal bound (assuming a finite primal bound), driving the search towards infeasibility seems a superior strategy instead of raising the dual bound. Therefore, in the above situation, the weight of pseudo-costs is dynamically weakened in favor of the other three feasibility-based scores.

Concretely, let k denote an iteration of the branch-and-bound Algorithm and let U_k^{infeas} and U_k^{prune} denote the number of leaf nodes that were infeasible and the number of leaf nodes that were pruned because the LP relaxation objective exceeded the primal bound, respectively. We multiply the three weights w^{conf} , w^{cut} , and w^{inf} each with $\frac{U_k^{\text{infeas}} + 1}{U_k^{\text{prune}} + 1}$, and we multiply w^{ps} with the reciprocal of this fraction. Due to this dynamic factor, a large number of infeasible leaves U_k^{infeas} that exceeds the number of pruned nodes U_k^{prune} by a factor of 10 will make the conflict score the dominant score of Hybrid Branching (2.18) at this stage of the search. The second adaptive behavior presented by Berthold et al. [2019a] adjusts the weights at each branch-and-bound node depending on the level of dual degeneracy. At highly degenerate nodes, the weight of pseudo-costs is reduced in favor of the other scores. Nowadays, SCIP by default uses both dynamic adjustments in Hybrid Branching.

2.5.5 Remarks

Note that in this thesis, we only consider variable-based branching. This concept is generalizable by incorporating branching on general disjunctions, which was introduced by Ryan and Foster [1981] in the context of problems with set-partitioning (multiple choice) constraints.

For branching methods on variables that use other techniques than history information, see, e.g., [Gilpin and Sandholm, 2011; Kılınç Karzan et al., 2009]. The branching strategies presented in [Pryor and Chinneck, 2011] aim at quickly finding feasible solutions. To this end, solution densities are approximated by means of normal distributions. Fischetti and Monaci [2011] proposed a method for restricting the set of branching candidates by calculating so-called *backdoor sets* in advance.

For more information on branching with a focus on recent advances of branching methods in SCIP, we refer to [Gamrath, 2020].

2.6 Primal Heuristics

The B&B Algorithm finds a sequence of incumbent solutions for a MIP P by storing integer feasible LP relaxation solutions, thereby shrinking the primal bound until it is optimal for P . With each improved primal bound, chances are that open nodes of the search tree can be pruned without necessitating further exploration. However, integer feasible LP relaxation solutions are usually located at deeper nodes of the search tree and may be hard to find. In addition, as long as the primal bound is large or even infinite, the B&B search may get stuck in a part of the search tree that could be cut off right away with a better (or any) solution and corresponding primal bound at hand.

Therefore, modern MIP solvers employ primal heuristics tightly integrated with the main B&B search. Berthold [2014b] defines primal heuristics as “algorithms that try to find feasible solutions of good quality within a reasonably short amount of time”, but without a guarantee to succeed. Primal heuristics are often designed with a specific application or problem structure in mind. In contrast, the primal heuristics discussed in the following and in subsequent chapters make little assumptions about the structure of the MIP problem, except that some require the presence of binary variables or a nonzero objective function. Over the years, many different primal heuristic algorithms have been proposed, which are highly diverse in the computational effort they require. A survey article on primal heuristic algorithms for MIP can be found in [Fischetti and Lodi, 2010]. For a general overview of the primal heuristics implemented in SCIP, we refer the reader to [Berthold, 2014b] and the references therein.

Following the classification by Achterberg, Berthold, and Hendel [2012], primal heuristics for MIP can be grouped based on the techniques they apply into rounding, propagation, diving and large neighborhood search heuristics. The first group searches

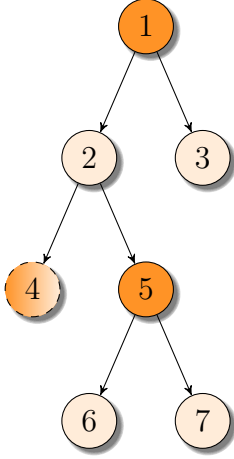


Figure 2.5: Illustration how SCIP executes heuristics during branch-and-bound. The nodes labeled 1 – 5 are the branch-and-bound search tree, representing the order in which the search nodes are explored. In SCIP, each primal heuristic is executed according to its frequency parameter $f \geq 0$ that determines the depth levels of the search tree at which the heuristic is called. Only heuristics with frequency $f = 1$ are called at every branch-and-bound node. An example of a heuristic H with frequency $f = 2$ is illustrated here. We highlight those nodes at which H is called. Node 4 is special because it may have been pruned before or after H has been called. It may even be that H found a solution, e.g., by rounding the LP solution at 4, and 4 could be pruned as an immediate result.

for improving solutions in the set of feasible roundings among the fractionals after each LP solution.⁷ As propagation heuristics, we denote constructive algorithms that greedily perform a sequence of local assignments and apply domain propagation in between to increase the chances to terminate with a feasible solution.

An example of a propagation heuristic called *shift-and-propagate* (*S&P*) has been introduced by Berthold and Hendel [2014]. Let P be a MIP, for which S&P is called during the B&B algorithm.⁸ Let ℓ' and u' denote the current local lower and upper bounds that may include fixings of selected variables and subsequent domain reductions on other variables. Throughout the algorithm, shift-and-propagate maintains an assignment $\tilde{x}^{\text{s\&p}}$ of the variables that satisfies the bound requirements $\ell \leq \ell' \leq \tilde{x}^{\text{s\&p}} \leq u' \leq u$ and integrality restrictions $\tilde{x}_j^{\text{s\&p}} \in \mathbb{Z}$ for $j \in \mathcal{I}$ of P , but potentially violates the rows of P . As soon as $A\tilde{x}^{\text{s\&p}} \geq b$, S&P returns the feasible solution $\tilde{x}^{\text{s\&p}} \in \mathcal{S}_P$.

If, however, there still exists a row that is violated by $\tilde{x}^{\text{s\&p}}$, S&P selects an (unfixed) integer variable $j \in \mathcal{I}$ that appears in a maximal number of violated rows, and computes a (finite) value $\ell'_j \leq y_j \leq u'_j$ that maximizes the number of additionally satisfied rows if we reassign (shift) $\tilde{x}_j^{\text{s\&p}}$ to y_j . It is possible that $\tilde{x}_j^{\text{s\&p}} = y_j$. Then, S&P fixes the domain of j temporarily to y_j by setting $\ell'_j \leftarrow y_j$ and $u'_j \leftarrow y_j$. Since j was unfixed before, this fixing restricts the search domain and therefore the size of the remaining search space even if $\tilde{x}_j^{\text{s\&p}} = y_j$ might stay unchanged in this iteration of the algorithm. The domain change of j is then propagated to find further domain reductions, and undone (backtracked) if domain propagation concludes that the remaining subproblem within the local bounds became infeasible. In this case, j is discarded from the set of variables to select from, and S&P continues the search with the remaining, unfixed

⁷See the neighborhood definition (6.3) of the RENS heuristic [Berthold, 2014a] for a definition of the set of feasible roundings.

⁸We should note at this point that line 13 of the B&B algorithm is not the only place during the processing of a node at which SCIP calls primal heuristics. S&P, as an example, is actually called before the LP relaxation is solved, whereas diving heuristics are typically called after branching.

integer variables. If no feasibility was detected, S&P assigns $\tilde{x}_j^{\text{s\&p}} \leftarrow y_j$ and continues in the reduced search domain.

Propagation heuristics such as S&P are considered computationally inexpensive, because they do not solve intermediate LP relaxations to drive the assignment. S&P does not require a feasible solution as starting point and is called before the initial root node LP relaxation is solved.

A higher computational effort is usually required by diving and objective diving heuristics. Both heuristic classes solve sequences of modified LP relaxations to drive the search forward. According to Fischetti and Lodi [2010], diving heuristics belong to the “folklore” of heuristic strategies, like rounding. Diving heuristics in SCIP [Achterberg, 2007a] work similarly to a branch-and-bound tree search with depth-first node selection and limited backtracking; if a diving heuristic encounters an infeasible node, it backtracks to try the alternative direction, until it finishes in a node where both children are infeasible, or due to a depth or simplex iteration limit. One specialty is that diving in SCIP is conducted in an *auxiliary path* of the search tree. Starting from a search tree node, the auxiliary path is only maintained during the dive and is discarded afterwards. Hence, diving strategies which might be considered ineffective for a global tree search can be used within a diving heuristic as a temporary replacement of the main branching strategy to search for improving solutions, without a negative effect on the main B&B search tree. We postpone a more detailed discussion of the diving strategies in SCIP until Chapter 7.

An effort similar to diving is required by the feasibility-pump [Fischetti, Glover, and Lodi, 2005] procedure, which has seen numerous improvements [Achterberg and Berthold, 2007; Dey et al., 2018; Fischetti and Salvagnin, 2009, to name only a few] over the years. The basic idea behind feasibility pump as originally presented by Fischetti et al. [2005] is to solve a sequence of LPs with a modified objective function. Feasibility pump iterates between a rounding step and a so-called projection step: In its first iteration, feasibility pump rounds the LP relaxation solution \tilde{x} of a MIP P to obtain $\tilde{x}^{\text{fp}} := [\tilde{x}]$. Of course, the rounding operation $[\cdot]$ is restricted to the integer variables. Clearly, while \tilde{x} potentially violates integrality restrictions, \tilde{x}^{fp} potentially violates linear constraints. If $A\tilde{x}^{\text{fp}} \geq b$, feasibility pump returns \tilde{x}^{fp} . Otherwise, during the projection step, feasibility pump solves an LP relaxation with a new objective function, trying to minimize a distance function to \tilde{x}^{fp} . As distance function, Fischetti et al. [2005] suggest the binary distance, which we define in Section 6.1.2. The binary distance has the advantage that it can be expressed as a linear objective function without additional variables. Early improvements to the projection step consisted in new distance functions; Bertacco et al. [2007] proposed to include the integer variables of the problem at the expense of additional variables and constraints, whereas [Achterberg and Berthold, 2007] mix the actual objective function of P into the projection step. A recent survey article [Berthold, Lodi, and Salvagnin,

2019b] is dedicated to these and many more enhancements proposed for feasibility pump over the years.

At the most expensive end of the scale are *large neighborhood search (LNS)* heuristics that solve auxiliary MIPs, such as *Relaxation Induced Neighborhood Search (RINS)* [Danna et al., 2005]. We give an overview of LNS techniques in Section 6.1.

Since [Berthold, 2014b], there have been several new developments in SCIP covering most of the aforementioned primal heuristic categories. Two of these developments are presented in detail in Chapters 6 and 7. Besides, Witzig and Gleixner [2020] proposed two new diving heuristics that aim at both finding feasible solutions and at deriving useful conflict clauses. Another recent development in SCIP are specialized primal heuristics that exploit user decompositions, if provided. For an overview of these primal heuristics, see [Gamrath et al., 2020].

Gamrath et al. [2015a, 2019] proposed three new propagation heuristics to construct initial feasible solution. Similarly to S&P, all three heuristics conduct greedy variable assignments and use propagation to deduce necessary domain reductions on other variables. In contrast to S&P, these *structure-based* heuristics by Gamrath et al. [2015a, 2019] select variables and values based on auxiliary structures such as the “conflict graph” [Atamtürk et al., 2000] on the binary variables. The conflict graph, which is called “clique table” in SCIP because of the ambiguity with the conflict graph used for conflict analysis, collects constraints of the form

$$\sum_{j \in \mathcal{B}^+} x_j + \sum_{j \in \mathcal{B}^-} (1 - x_j) \leq 1,$$

where $\mathcal{B}^+ \cup \mathcal{B}^- \subseteq \mathcal{B}$. Constraints of the above form are often called cliques. From the multiple choice constraint from Example (2.1), a clique can be readily derived. However, it is also possible to infer cliques that are implied by general linear constraints.

For their clique-driven heuristic, Gamrath et al. [2015a, 2019, Algorithm 2] propose to first select a clique C from the clique table with maximum number of unfixed variables, and to assign the binary variable j contained in C with minimum cost to 1 or 0 depending on whether $j \in \mathcal{B}^+$ or $j \in \mathcal{B}^-$, respectively. In the subsequent propagation step, this fixing will always force that all other variables from C be fixed to either 0 or 1, depending on whether they are in $\mathcal{B}^+ \setminus \{j\}$ or $\mathcal{B}^- \setminus \{j\}$. This clique-driven fixing scheme aims at quickly fixing as many binary variables as possible. Of course, the clique table must cover most binary variables of a MIP for this procedure to make good selections.

2.7 Presolving and Propagation

In Section 2.3.3 we mentioned presolving as one of the commonly used components inside a MIP solver. Recall that presolving denotes valid transformations that preserve the optimal objective value of an input MIP.

Some of the most common operations applied during presolving are

- the removal of fixed variables (variables with equal lower and upper bounds),
- the removal of redundant rows, and
- the tightening of variable lower and upper bounds based on row activities [Brearley et al., 1975] (see Section 2.7.1 below for a special case of activity-based bound tightening).

It may happen during presolving that a variable domain becomes empty. This proves the infeasibility of the original input MIP.

Since the tightening of variable lower and upper bounds can result in new fixed variables, presolving is usually applied in rounds. Presolving is iterated until no more valid transformations can be found, the problem is proven infeasible, or working limits are reached.

Presolving techniques are applicable at each node of the B&B search tree. A node is created by adding a bound restriction on an integer variable through branching. This bound restriction can help to reduce the domains of other variables.

Typically, the full set of presolving transformations is applied only at the root node of a MIP before the initial LP relaxation is solved. All applied transformations to the root node of the search are valid thereafter also in descendant nodes. At descendant nodes, only a reduced set of transformations is applied, typically activity-based bound tightening starting from the new bound on the branching variable.

The application of valid problem transformations at non-root nodes is called *propagation*. In our version of the B&B algorithm 2, we execute “node preprocessing” in line 5. By node preprocessing, we denote

- presolving before the root node of the search, and
- propagation at all other nodes.

There are many more valid problem transformations than listed above. For more details on presolving, see [Achterberg et al., 2019] about presolving in general, and [Gamrath et al., 2015b] for SCIP-related information.

2.7.1 Reduced Cost Strengthening

In fact, propagation need not only be applied at the very beginning of a node. An example of a valid transformation that involves LP information is the so-called *reduced cost strengthening*. Reduced cost strengthening uses the dual solution values to deduce tighter bounds on variables that are nonbasic in the optimal LP basis. One of the earliest works that describe reduced cost fixing in the context of the traveling salesman problem is [Dantzig et al., 1954].

Reduced cost strengthening is a special type of activity-based bound tightening. We explain this reduction here because reduced costs also play a role in Chapter 6, where we use reduced costs to design a variable fixing score.

Let P be a MIP with lower bounds $\ell = 0$ for all variables, infinite upper bounds, and LP relaxation \check{P} . We consider an optimal basic LP solution $\check{x} = (\check{x}_{[B]}, \check{x}_{[N]})$ for \check{P} with corresponding dual basic solution $(\tilde{\pi}, \tilde{d})$. Note that all reduced costs \tilde{d} are nonnegative because B is dual feasible.

Let \check{Z} be the objective value of \check{x} . Note that $\check{Z} = \tilde{\pi}^t b$. Let Z' be a given upper bound on Z^{opt} , the optimal objective of P . The upper bound Z' can be considered as an inequality $c^t x \leq Z'$ on the objective function of P .

Let $j \in N$ be a variable whose column is nonbasic with positive reduced costs $\tilde{d}_j > 0$. We derive an upper bound on x_j using its reduced cost d_j as follows.

$$\begin{aligned}
 c^t x &= (\tilde{\pi}^t A + \tilde{d}^t) x \leq Z' \\
 \Leftrightarrow \quad &\tilde{\pi}^t b + \tilde{d}^t x \leq Z' \\
 \Leftrightarrow \quad &\check{Z} + \tilde{d}^t x \leq Z' \\
 \Leftrightarrow \quad &\sum_{j'=1}^n \underbrace{\tilde{d}_{j'} x_{j'}}_{\geq 0} \leq Z' - \check{Z} \\
 \Rightarrow \quad &\tilde{d}_j x_j \leq Z' - \check{Z} \\
 \Leftrightarrow \quad &x_j \leq \frac{Z' - \check{Z}}{\tilde{d}_j}.
 \end{aligned}$$

If the right term in the last inequality is smaller than the upper bound u_j , we can tighten the domain of x_j . If, in addition, $j \in \mathcal{I}$, the right term in the last inequality may be rounded down.

2.8 Machine Learning Inside B&B

So far, we have reviewed classical techniques for solving MIP. We have pointed out that branching rules such as the pseudo-cost score (2.17) represent an estimation of the potential gains in the prospective children. Recently, a lot of research has been conducted to enhance B&B solving components by sophisticated techniques inspired by Machine Learning (ML), in particular to derive new branching rules. In this section, we briefly introduce ML at the examples of linear regression and of a regression tree [Breiman et al., 1983], which will be useful for Chapter 9. Then, we review the recent literature at the cross-section between ML and MIP.

2.8.1 A Quick Primer on Supervised Machine Learning

A very practical setup usually looks as follows. We have collected n independent data points (samples) in d dimensions, which we represent as a feature matrix $X \in \mathbb{R}^{n,d}$. The i 'th row of X , $x_i \in \mathbb{R}^{1,d}$, contains the features of the i 'th such sample. In addition, each data point also has a continuous label $y_i \in \mathbb{R}$. We assume that features and labels are connected via an underlying functional relationship $f : \mathbb{R}^n \rightarrow \mathbb{R}$, such that for each record $i = 1 \dots n$ the label $y_i = f(x_i) + \epsilon_i$. In words, y_i is the result of f applied to the sample (the i 'th row of X) x_i and some random noise ϵ_i that may result from properties of the object corresponding to x_i unaccounted for by our d -dimensional feature vector, or from errors in measuring y_i .⁹

We wish to approximate (aka “learn”) f as well as possible. Since we have features and labels as training data, this is called a *supervised* learning problem. Since the labels are a continuous quantity, this learning problem is called more specifically a *regression* problem.

Typically, for learning f , we restrict ourselves to a family of parametrized model functions $F_\Theta := \{f_\theta : \theta \in \Theta\}$. Here, each element θ from the parameter space Θ describes a function $f_\theta : \mathbb{R}^n \rightarrow \mathbb{R}$. For example, if $d = 1$, we are facing a 1-dimensional regression problem, and we may wish to find a linear relationship $y_i \approx ax_i + b$ between the (single) input feature and output label of our training data. In this case, the parameter space $\Theta = \mathbb{R}^2$ and each $\theta = (a, b) \in \Theta$ describes one feasible function in this model. Due to the (assumed) linear relationship between the inputs and the output, this regression problem is the famous *linear regression*, sometimes also called simple linear regression in case of 1-dimensional input.

In a regression task, we commonly search for a model that minimizes the *residual sum of squares*

$$\theta^* = \operatorname{argmin}_{\theta \in \Theta} \left\{ \sum_{i=1}^n (y_i - f_\theta(x_i))^2 \right\}.$$

In the 1-dimensional linear regression example, there is an analytic solution for θ^* that can be computed in linear time in the number of data points n :

$$\theta^* = \left(a^* = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}, b^* = \bar{y} - a^*\bar{x} \right). \quad (2.19)$$

Here, \bar{y} and \bar{x} denote the mean values of the labels and inputs. Statistically adept readers immediately notice the appearance of the variance in x and the covariance between x and y . A closed formula similar to (2.19) also exists for linear regression with multidimensional input.

⁹For the more general case in which errors can occur in both X and Y , we refer to [York et al., 2004].

At this point, we familiarized ourselves with regression tasks in general, and are able to address them by linear regression, for which there exists an optimal closed formula to compute the optimal parameters θ^* .¹⁰ However, this is of course not the end of the story, because we made the implicit assumption that the relationship between the input features and output label is linear.

2.8.2 Regression Trees

Obviously, linearity is a strong assumption about the “true” nature of the function f , and may be too much of a restriction of the function space. As an alternative, Breiman et al. [1983] introduced a regression technique in which the set of data points is recursively partitioned with the goal of reducing the residual sum of squares. Similarly to branch-and-bound, the recursive partitioning of the regression task eventually leads to a tree hierarchy, which is appropriately named *regression tree*.

A regression tree is a quadruple $G = (V, E, g, h)$, where V and E are the nodes and edges of a binary tree. In addition, we equip each node with two functions $g : V \rightarrow \{1, \dots, d\}$ and $h : V \rightarrow \mathbb{R}$. We use g and h to describe the split at an inner node of G , and a prediction at a leaf.

A prediction by a regression tree works as follows. Let $z \in \mathbb{R}^d$ be a new data point in feature space, for which we wish to compute its label $f(z)$. The prediction follows the unique path from the root of G to reach the leaf node that describes the average label of all training data in the vicinity around z .

Concretely, we begin at $v := \text{root}(G)$. If v is a leaf, we return $h(v)$ as prediction for $f(z)$. Otherwise, if v is an inner node, we compare $z_{g(v)}$ with $h(v)$. If $z_{g(v)} \leq h(v)$, we continue in the left subtree of v , otherwise we continue in its right subtree.

The classic CART (Classification and Regression Tree) algorithm is a greedy algorithm that constructs a regression tree by recursively partitioning the training data. For a node v , let $I(v) \subseteq \{1, \dots, n\}$ denote the indices of the training data that respect all splits from the root node to v . We use bracket notation to denote the labels that correspond to the records in $I(v)$ as $y_{I(v)} \in \mathbb{R}^{|I(v)|}$.

The algorithm declares v a leaf node if the prediction cannot be improved further (all remaining data points $I(v)$ have the same label $y_{I(v)}$), in which case $h(v) := \bar{y}_{I(v)}$. Otherwise, it selects the feature dimension $d^* \in \{1, \dots, d\}$ and split value $s^* \in \mathbb{R}$ that achieve the highest variance reduction by partitioning the training data at v further. In this case two children are created for v . The left child v' will contain the training data indices $I(v') = I(v) \cap \{i : x_{i,d^*} \leq s^*\}$, and the right child v'' will contain the complement $I(v'') = I(v) \cap \{i : x_{i,d^*} > s^*\}$. Additionally, we set $g(v) := d^*$, $h(v) := s^*$.

Practically, the CART algorithm terminates earlier, for example at nodes v where $|I(v)|$ falls below a given limit to allow for a further split. Advantages of regression

¹⁰The closed formula (2.19) works at least for all data sets with nonzero variation in the input features.

trees are, for example, that they can cope with largely nonlinear functional relationships between the inputs and the output. In addition, a regression tree provides a simple explanation (the data splits along a path) for its predictions on unknown data points.

However, regression trees as such are not that frequently used in practice. One of their main disadvantages is that they have a tendency to overfit to the training data but lack generalization abilities to unseen data. The discretized nature of the predictions may not have the desired properties of an output, which we will see in Chapter 8 at the example of B&B tree size prediction.

More often, regression trees are used as so-called weak learners within ensemble methods such as random forests [Breiman, 2001] or Adaboost [Friedman, 2001], which compensate the discussed weaknesses of single trees by training many regression trees, each of which on randomly selected subsets of the training data and with only a subset of the feature dimensions available to each split.

While in this chapter, we focus solely on a regression task with continuous input, regression trees are straightforwardly generalizable to other types of input data such as categorical input dimensions like the gender of a patient. Similarly, we have only addressed regression on a continuous label y . If y represented a categorical class label instead of a continuous quantity, we would be facing a so-called *classification* task, for which there exist other ML methods, one of which are classification trees. Classification trees work similarly to regression trees, but differ in the way they predict a class label: Instead of returning the mean label, which does not make sense in the case of categorical labels, they predict the majority label among the samples represented by a leaf node.

2.8.3 Recent Combinations of ML and MIP

Several ML methods have recently been proposed in the context of solving mixed-integer programs. Much work has been conducted in specific application areas, for example to solve unit commitment instances repeatedly [Xavier et al., 2019]. In the following, we mainly focus on literature that addresses general purpose MIP solving.

A classic playground for ML methods is strong branching, whose score (see Definition 2.7) generally produces good branching decisions leading to small B&B trees, but which is very expensive to compute. Alvarez et al. [2015] were among the first to approximate strong branching via Machine Learning. To this end, they use static and dynamic properties that describe the search state and the variables as features. The observed strong branching scores are used as labels. On this data, they train a variant of random forest regression offline, which they apply at runtime to predict strong branching scores and select branching variables. In a follow-up work, Alvarez et al. [2016] present an online algorithm that records strong branching scores and features during the search, and uses gradient descent to modify the weights of a linear regression type strong branching score prediction.

Directly predicting the strong branching score itself has some disadvantages: It is the product of the two objective gains and may vary by orders of magnitude between different MIP instances. Khalil et al. [2016] observe that instead of predicting actual strong branching scores, it is sufficient to predict how strong branching will *rank* two candidates relative to each other. Khalil et al. [2016] record some features and strong branching scores during the first 500 nodes of the search tree, after which they interrupt the search to train a ranking Support Vector Machine [Joachims, 2002] on the recorded data once. Afterwards, they continue the search using the trained ranking method for selecting the branching variable.

All of those methods have in common that they use hand-made features that aggregate the MIP instance at hand. In contrast, recent work often uses a loss-free encoding of MIP instances using graph convolutional neural networks, used for example by [Gasse et al., 2019]. Similarly, in a very recent work, Nair et al. [2020] propose two deep neural network architectures based on graph convolutional neural networks to address branching and finding feasible solutions. They make use of modern GPU processors to accelerate the collection of strong branching data, which enables them to train their neural branching with a lot more strong branching records than was previously possible.

Learning approaches from the recent literature are not limited to branching. Khalil et al. [2017] propose to use logistic regression to predict the success rate of a diving heuristic. Fischetti et al. [2019] propose to train classifiers to predict at specific points in time whether a solution process will terminate before the time limit. Tang et al. [2020] suggest training cutting plane separation, more specifically Gomory cuts, via reinforcement learning. Interestingly, they show that their trained models are able to generalize to unseen instances that are larger (in terms of variables and constraints) than the instances they use for training.

Another aspect that can be learned about MIP solvers are good parameters for the instances at hand. The study of finding good parameters of search algorithms is called *algorithm configuration* and often colloquially referred to as *tuning*. Commercial MIP solvers are usually equipped with internal tuning tools, which try a variety of promising parameter combinations. Algorithm configuration is often approached via methods for black-box function optimization to minimize the number of costly MIP solver evaluations. Hutter et al. [2011] presented an algorithm called *SMAC*, which stands for *sequential model-based algorithm configuration*. SMAC approximates the parameter landscape by a multivariate Gaussian distribution to quickly identify regions of the parameter space with high expected improvement. A competing method for algorithm configuration is *iterated F-race* [López-Ibáñez et al., 2016], in which a few candidate configurations are randomly sampled and then compared in a racing-like fashion. One of the main ideas behind F-race is to quickly discard inferior configurations based on a statistical test. Discarded configurations are withdrawn from the remaining race to reserve computing resources for better configurations. The best surviving configuration

is then used to update the sampling distributions before new configurations are sampled for the next race. Recently, Li et al. [2018] presented *Hyperband*, a well-performing algorithm configuration method based on random search.

Most of the aforementioned work mainly addresses MIP solver performance. Berthold and Hendel [2021] are the first ones to explicitly improve numerical stability of the FICO Xpress solver by learning the appropriate scaling method for each instance. The trained scaling improves not only the numerical stability substantially, but also improves the performance of the primal and dual simplex routines as a side effect.

2.9 MIP Benchmarking

In each of the subsequent chapters, we discuss computational results of our proposed methods. The central goal of integrating new solver components for MIP has always been to reduce the overall runtime of the solver on a representative set of MIP benchmark instances. However, besides runtime, there also exist several other performance criteria, each of which has certain advantages depending on the experimental setup. In this section, we give a brief overview of other performance criteria such as the well-known gap and the more recently proposed primal and dual integrals [Berthold, 2013]. The compilation of a representative MIP benchmark set is subject of Chapter 3.

2.9.1 Gap

Recall that we always assume that the objective sense is minimization. We use a standard gap definition.

Definition 2.9 (gap). *Let $Z \in \mathbb{Q}_{\pm\infty}$ and $Z^* \in \mathbb{Q}_{\pm\infty}$ denote a primal and dual bound of a MIP P , respectively. The (primal dual) gap is the relative difference between Z and Z^* , computed as*

$$\gamma(Z, Z^*) := \begin{cases} 1, & \text{if } Z^* < \infty \text{ and } Z = \infty, \\ 0, & \text{if } Z \leq Z^*, \\ \min \left\{ 1, \frac{Z - Z^*}{\max\{|Z|, |Z^*|\}} \right\}, & \text{otherwise.} \end{cases} \quad (2.20)$$

Before the first solution is found, the gap is equal to 1. The use of “min” in the third case of the definition of $\gamma(Z, Z^*)$ is necessary to ensure that the gap is bounded from above by 1 even if $Z^* < 0 < Z < \infty$. Definition 2.9 is different from the gap definition that SCIP uses for reporting the gap [see, e.g., Gamrath, 2020].

We sometimes refer to the gap between the current primal and dual bounds more precisely as *primal-dual gap* to distinguish it from the following primal gap and dual

gap, which take into account the optimal solution value Z^{opt} of a MIP P . The primal gap is defined as

$$\gamma^{\text{primal}}(Z) := \gamma(Z, Z^{\text{opt}})$$

and the dual gap is defined as

$$\gamma^{\text{dual}}(Z^*) := \gamma(Z^{\text{opt}}, Z^*).$$

At a search state T_k of the B&B algorithm, a primal gap $\gamma^{\text{primal}}(Z_k)$ of zero means that the incumbent is an optimal solution, although this might not be proven so far because the dual bound $Z_k^* < Z^{\text{opt}}$ is less than the optimal objective, or, in other words, the dual gap $\gamma^{\text{dual}}(Z_k^*) > 0$.

Note that all three gaps are well-defined also in case that P is infeasible. Since the optimal objective value Z^{opt} is unknown before termination of the branch-and-bound algorithm, both the primal and the dual gap are artificial measures used for benchmarking that can only be computed when the search is complete.

2.9.2 Primal and Dual Integrals

The (primal-dual) gap is an established measure to compare two solvers at termination. An important drawback is that the gap is always recorded at an instant, for example at the precise moment when a solver hit an allowed time limit. Reporting the gap at termination may not reflect the progress during the search as a whole.

A performance measure to capture the progress during the entire search, the *primal integral*, has been proposed by Berthold [2013]. Consider the primal and dual bounds $Z(t)$ and $Z^*(t)$ as functions of the elapsed time $t \in \mathbb{R}_{\geq 0}$ since the B&B algorithm started.

For a point in time $T \geq 0$, the *primal integral* is the integral of the primal gap of $Z(t)$ from the beginning of the search until T :

$$\Gamma^{\text{primal}}(T) := \int_{t=0}^T \gamma^{\text{primal}}(Z(t)) dt. \quad (2.21)$$

Berthold [2013] suggests the primal integral as a performance measure that captures the benefits of primal heuristics particularly well. If a search finished at a time $T' < T$, the primal gap function is zero by definition for the remaining time $[T', T]$. For comparing two different B&B search algorithms in a computational experiment, it is therefore fair to use the overall time limit as the point in time T in above integral calculation because $\Gamma^{\text{primal}}(T') = \Gamma^{\text{primal}}(T)$.

Analogously to the primal integral, we compute the *dual integral*

$$\Gamma^{\text{dual}}(T) := \int_{t=0}^T \gamma^{\text{dual}}(Z^*(t)) dt \quad (2.22)$$

as the integral of the dual gap over time since the beginning of the search until T , where T is usually the time limit of a computational experiment. The distinction between the primal and the dual integrals enables us to analyze solving components regarding their influence on the primal and dual progress separately.

We define the primal and dual integrals as integrals of a continuous function. Observe that changes to the primal and dual bound during the B&B search occur as discrete events. Hence, the corresponding functions $Z(t)$ and $Z^*(t)$ are piecewise constant functions of t . Therefore, the integrals can be computed by recording each change to the primal/dual bound and the time of this event.

The primal and dual integrals allow to analyze the effects of a MIP solving component on the primal and dual convergence of the B&B algorithm individually. For example, while primal heuristics are expected to mainly improve the primal convergence, other solving components such as branching merely work on the dual side. We refer to Section 5.1 for an evaluation of the impact of solving components on the primal and dual convergence of SCIP.

For completeness, it is possible to also define a *primal dual integral* as

$$\Gamma(T) := \int_{t=0}^T \gamma(Z(t), Z^*(t)) dt$$

based on the primal-dual gap. We do not use this integral for the computational experiments in this thesis. In practice, however, the primal dual integral is the only integral that a solver can actually evaluate at runtime.

The solvers SCIP and Xpress¹¹ report the measured primal dual integral in their final statistics. More recently, SCIP has been extended to also report *primal reference* and *dual reference* integrals, if a user provided a so-called *reference value*. The reference value replaces the optimal objective value Z^{opt} when computing the primal and dual gaps. If this reference value is indeed the optimal objective value, the primal reference and dual reference integrals coincide with the above definitions considering the optimal objective.

2.9.3 Shifted Geometric Mean

In computational experiments we are often interested in a compressed overall performance indicator that summarizes a measure such as runtime or primal integral across a set of MIP instances. The exponential nature of the B&B algorithm poses some challenges upon this aggregation process. When we compare runtimes X_A and X_B observed for two versions of the B&B algorithm for an instance, it is often more meaningful to compare the relative runtime $\frac{X_A}{X_B}$ instead of an absolute difference $|X_A - X_B|$. This is

¹¹<http://www.fico.com/en/Products/DMTools/xpress-overview/Pages/Xpress-Optimizer.aspx>

even more true when considering sets of heterogeneous benchmark instances such as MIPLIB [Achterberg et al., 2006; Bixby et al., 1998; Gleixner et al., 2021; Koch et al., 2011; MIPLIB, more in Chapter 3], where the solving time of different MIP instances may vary by several orders of magnitude: seconds, minutes, or even hours.

As a common practice in evaluating computational results, we often aggregate sets of measurements such as the observed runtime, required B&B nodes, or the primal/dual integrals obtained over sets of instances using the so-called *shifted geometric mean* [Achterberg, 2007a]. For a shift $\tau \geq 0$, the shifted geometric mean of a nonnegative vector $X \in \mathbb{R}_{\geq 0}^d$ is

$$\text{sgm}(X) := -\tau + \prod_{i=1}^d (X_i + \tau)^{\frac{1}{d}}. \quad (2.23)$$

The shifted geometric mean weakens the influence of extreme cases at both ends of the measurement scale: As a geometric mean, it is less affected by single large measurements. The shift value τ , on the other hand, is used to weaken the influence of extreme measurements at the lower end of the scale, for example time measurements of less than a second. Especially when we compare solving times in a computing environment, there could be measurement discrepancies caused by the environment that we cannot control.

For example, if we record 0.1 seconds as runtime of a setting A and 0.2 seconds as runtime of setting B , the ratio between A and B is 0.5, i.e., A is reported as 50% faster than B . However, we are not sure that the improvement of A compared to B was purely algorithmic, or rather caused by random noise from the environment. Moreover, MIP instances that take less than a second to solve are usually considered easy. Using a shift value of $\tau = 1$ sec., the ratio becomes $\frac{X_A + \tau}{X_B + \tau} \sim 0.92$, a much less pronounced improvement of A over B .

2.9.4 Wilcoxon Signed Rank Test Using Shifts

Recall that MIP benchmark results such as the total solving time can vary between instances by orders of magnitude, which is why we are more interested in relative improvements or deteriorations than in absolute improvements. It can happen that a single instance transcends from easy to unsolvable in a computational experiment comparing two settings. Such an outlier instance can dominate the shifted geometric mean even though the shifted geometric mean is less sensitive to such single observations than the arithmetic mean.

The question arises if an improvement or degradation in shifted geometric mean is also statistically significant. For some of our computational studies, we use a modified version of the standard Wilcoxon Signed Rank Test to address this question.

The Wilcoxon Signed Rank Test is a nonparametric alternative to the paired t-test [Falk et al., 2003] if the underlying distribution cannot be assumed as normal. This

test can be used to falsify the hypothesis that two vectors of samples X and Y come from the same distribution.

While the standard version considers differences $X - Y$, we are more interested in ratios X/Y . In addition, in our modified version of this test, we also take the shift value τ into account. Here, we present the main idea behind this modified test. For further information, we refer to [Hendel, 2014, Chapter 2, and the references therein].

Let $n \in \mathbb{N}$, and (X_1, \dots, X_n) and (Y_1, \dots, Y_n) be two independent and nonnegative observations that come in pairs (X_i, Y_i) for $1 \leq i \leq n$. An example of such paired observations are benchmark results of two MIP solvers X and Y on a set of n MIP instances, such that X_i and Y_i are the observed realizations of one of the discussed performance metrics such as the solving time on the i -th instance.

Let $1 \leq i \leq n$, and let $\tau > 0$ be a shift value like for the shifted geometric mean (2.23). We define the *logarithmic shifted quotient* as

$$Q_i := \log \left(\frac{X_i + \tau}{Y_i + \tau} \right). \quad (2.24)$$

Note that Q_i is well-defined because $(X_i + \tau)/(Y_i + \tau) > 0$ since X_i and Y_i are nonnegative. A logarithmic shifted quotient $Q_i < 0$ is negative if and only if $X_i < Y_i$. The use of the logarithm is an order-preserving transformation of the shifted quotients. It has the additional property that an improvement by $1 + \epsilon$, i.e., $1 > \frac{X_i + \tau}{Y_i + \tau} = \frac{1}{1 + \epsilon} > 0$ and a deterioration by ϵ , i.e., $\frac{X_j + \tau}{Y_j + \tau} = 1 + \epsilon$, yield the same absolutes for Q_i and Q_j :

$$|Q_i| = \left| \log \left(\frac{X_i + \tau}{Y_i + \tau} \right) \right| = \left| \log \left(\frac{1}{1 + \epsilon} \right) \right| = |\log 1 - \log(1 + \epsilon)| = \log(1 + \epsilon) = |Q_j|.$$

We test against the hypothesis H_0 that the underlying distributions of X and Y are equal. Under H_0 , the distribution of Q_i should be centered around the origin. Without loss of generality, we assume that $|Q_1| < |Q_2| < \dots < |Q_n|$, such that each index i also represents the rank of the i -th sample and that $|Q_1| \neq 0$. In practice, the samples are reduced first by filtering out all occurrences of $Q_i = 0$, and ties between ranks are solved by assigning the average rank to each of the samples in question.

We compute the *Wilcoxon sum statistics* W^+ , W^- as

$$W^+ := \sum_{i=1:Q_i>0}^n i \quad \text{and} \quad W^- := \sum_{i=1:Q_i<0}^n i.$$

The intuition behind the test is that if X and Y come from the same distribution, the two sum statistics should be approximately equal. Notably, W^+ and W^- always sum up to $n \cdot (n+1)/2$, the sum of all ranks. Under the hypothesis H_0 , W^+ and W^- are identically distributed around a mean $\mu = n \cdot (n+1)/4$ with variance $\sigma^2 := n(n+1)(2n+1)/24$ [Falk et al., 2003], and can be approximated by means of a normal distribution if n is sufficiently large. Let $W_{\min} := \min\{W^-, W^+\}$ denote the minimum of W^- and W^+ , and let $\alpha \in (0, 1)$

be a fixed error rate which we use as threshold for rejecting H_0 . Let $z_{(1-\alpha/2)}$ denote the $\alpha/2$ -quantile of the standard normal distribution, i.e.

$$\mathbb{P}\left(\frac{z - \mu}{\sigma} \geq z_{(1-\alpha/2)}\right) = \frac{\alpha}{2}.$$

We reject H_0 if the condition $W_{\min} \leq \mu - z_{(1-\alpha/2)}\sigma$ is satisfied, see [Hendel, 2014] for a deduction and further reference.

2.9.5 IPET—An Interactive Performance Evaluation Tool

The scientific assessment of computational results in the field of mixed-integer programming requires special, nonstandard measures and tools, some of which we have discussed in this section. We conclude this section with a brief overview of the interactive performance evaluation tools (IPET) [Hendel], which has been written by the author of this thesis in the programming language Python. IPET is a Python module for facilitating the extraction of scientific tables from raw solver output.

IPET specifically supports aggregation via shifted geometric mean, the modified Wilcoxon test, and the computation of primal and dual integrals. It has been first presented in [Hendel, 2014] as a graphical user interface for this purpose. Since then, the graphical user interface has been discontinued. Instead, the core functionality of IPET has been rewritten and extended to suit the needs as a backend tool for the continuous integration and performance testing by the SCIP group.

At its heart, IPET now consists of three main parts, namely

1. parsing capabilities for raw solver output,
2. tools for validating the correctness of results,
3. an evaluation engine.

IPET comes with parsing capabilities for raw solver output of 11 different commercial and noncommercial MIP solvers, including SCIP, CBC¹², FICO Xpress¹³, and all of the solvers participating in the curation of MIPLIB 2017, which we present in Chapter 3.

For IPET, the “Solver” class basically corresponds to a list of regular expressions that define how IPET recognizes the MIP solver that produced the log file and a few key statistics: the solving time, number of branch-and-bound nodes, primal and dual bound at termination, and so on. Support for new solvers can be very easily integrated into IPET itself by defining a corresponding class.

Moreover, IPET automatically loads additional solvers written by the user and stored in a special directory under the users home directory, which makes it possible to

¹²<https://projects.coin-or.org/Cbc>

¹³<http://www.fico.com/en/Products/DMTools/xpress-overview/Pages/Xpress-Optimizer.aspx>

extend IPET’s parsing capabilities to proprietary solver output. IPET also supports the addition of custom readers for further data parsing from the solver output, such as new output added during the development of a new solving component.

The parsed data is automatically validated against external information about the feasibility and optimal objective values of the tested instances, if provided. Examples of such external validation information can be found on the web page of MIPLIB 2017, Section *Test Scripts and Solution Information*.¹⁴ The validation of the correctness and consistency of computational results is essential in the development of new solving components for MIP. Otherwise, it might happen that for example a coding mistake in a presolving technique wrongly renders a feasible MIP instance infeasible, thereby even saving the time to conduct the tree search. If such a mistake slips into the solver unnoticed, this erroneous result may be accidentally reported as a substantial speed up. With the intention to identify numerically spurious instances, IPET validation has been used to identify inconsistent results from the more than 40,000 individual data records collected during the MIPLIB 2017 selection process. See also Section 3.5.5 for details.

Finally, at the heart of IPET lies its evaluation functionality. An IPET evaluation consists of a series of rules for processing and displaying the parsed data in tables. Rules consist of data columns that should be displayed, and filter groups to select records to display and summarize. A typical column, for examples, defines that the parsed solving time should be aggregated by a shifted geometric mean with a shift value of 1 sec, and that missing data should be filled by the time limit used, as well as further formatting options like a name for displaying the column.

Column rules need not correspond to existing records in the data, instead they can also be computed from existing columns. As an example, take the computation of the ratio between the runtime of each setting and each instance, and the smallest runtime across all evaluated runs on each instance. IPET keeps track of the inter-column dependencies as a topological sorting.

Filter groups allow selecting subsets of the data matching different criteria. A typical example of a filter group are the so-called *brackets*, which contain all instances for which at least one setting needed more than 10/100/1000 seconds. See Table 7.1 for an example table using bracket notation.

The rules can be stored in XML format. Once an evaluation has been established in XML format, it can be readily used for new data as well. One of the advantages of such an evaluation specification over experiment-specific evaluation code is that it transparently documents how exactly the computational results in a manuscript were obtained. The possibility to reuse evaluations on new or modified data even saves time that is usually spent on writing such manual evaluation code.

¹⁴<http://miplib.zib.de/download.html>

IPET is publicly available on Github.¹⁵ All experimental results in this thesis have been evaluated using IPET.

¹⁵<https://github.com/GregorCH/ipet>



MIPLIB 2017

Measuring performance on benchmark test instances has lain at the heart of computational research since the early days of mathematical optimization. Hoffman et al. [1953] first reported on a computational experiment comparing implementations of three algorithms for linear optimization. Their observation that “[many] conjectures about the relative merits of the three methods by various criteria could only be verified by actual trial” seems to hold to an even greater extent today. The variety of and complex interaction between different techniques regularly calls for empirical evaluation and motivates the collection and curation of relevant test instances.

Brought into existence by Bixby, Boyd, and Indovina [1992], the goal of the MIPLIB project has been to provide the research community with a curated set of challenging, real-world instances from academic and industrial applications that are suitable for testing new algorithms and quantifying performance. It has previously been updated four times [Achterberg et al., 2006; Bixby et al., 1998; Koch et al., 2011; MIPLIB] in order to reflect the increasing diversity and complexity of the MIPs arising in practice and the improved performance of available MIP solvers. In this chapter, we describe its sixth version, MIPLIB 2017, together with a new selection methodology developed during this process. This chapter is a shortened version of the journal article *MIPLIB 2017: Data-Driven Compilation of the 6th Mixed-Integer Programming Library* [Gleixner et al., 2021].

3.1 Introduction: A Motivation for a new MIPLIB

The exceptional algorithmic progress in solving real-world MIP instances over the last decades is recorded in various articles [Bixby, 2002; Koch et al., 2011; Laundry et al., 2009; Lodi, 2009]. Specifically, this can be observed by examining results on the previous version, MIPLIB 2010, both in terms of solvability and speed. By the end of 2018, the

number of unsolved instances was reduced by nearly half. Of the 134 instances for which no solution with provable optimality guarantee was initially known, only 70 remain open. Comparable progress in the overall speed of solvers can be observed in the results of benchmark testing with different versions of available solvers. Since its release in April 2011, the subset of instances of MIPLIB 2010 that form the so-called “benchmark set”, consisting of 87 problem instances, has been the accepted standard for evaluating solvers. Using this benchmark set, Hans Mittelmann has been evaluating a number of MIP solvers, including CPLEX¹, GUROBI², and XPRESS³. When MIPLIB 2010 was released, the version numbers of these three commercial solvers were CPLEX 12.2.0.2, GUROBI 4.5.1, and XPRESS 7.2. Aggregating the benchmark results of these three solvers at that time, we can construct results corresponding to a so-called “virtual best” solver and a so-called “virtual worst” solver. These are hypothetical solvers that, for each instance, produce runtimes that are equal to the best and the worst of the three, respectively. Doing this analysis yields shifted geometric mean runtimes of 36.3 and 113.0 seconds for the virtual best and virtual worst solver, respectively.⁴ In December 2018, the solver versions were CPLEX 12.8.0, GUROBI 8.1.0, and XPRESS 8.5.1. On the same hardware (with a newer operating system) the shifted geometric means of the runtimes had decreased to 13.5 seconds for the virtual best, and 31.3 seconds for the virtual worst solver. This corresponds to speed-up factor of 2.70 and 3.62, respectively, which amounts to roughly 16 % per year, just from improvements in the algorithms.

It was because of this development that the MIPLIB 2010 benchmark set was no longer considered to sufficiently reflect the frontier of new challenges in the field and the process of constructing a new MIPLIB began. In November 2016, a public call for contributions was launched and a group of 21 interested researchers, including representatives of the development teams of nine MIP solvers formed a committee in order to steer the process of compiling an updated library.⁵ As with MIPLIB 2010, the overall goal was the compilation of two sets of instances. The MIPLIB 2017 *benchmark set* was to be suitable, to the extent possible, for performing a meaningful and fair comparison of the average performance of MIP solvers (and different versions of the same solver) across a wide range of instances with different properties, in a reasonable amount of computing time. The larger MIPLIB 2017 *collection* was to provide additional instances for a broader coverage without restrictions on the total runtime of the test set, including unsolved instances (as a challenge for future research) and instances not

¹<https://www.ibm.com/analytics/cplex-optimizer>

²<http://www.gurobi.com/products/gurobi-optimizer/gurobi-overview>

³<http://www.fico.com/en/Products/DMTools/xpress-overview/Pages/Xpress-Optimizer.aspx>

⁴The means were computed with a shift of 1 second. The computations used 12 threads. The corresponding log files can be found at [Mittelmann].

⁵The members of the MIPLIB 2017 committee were Tobias Achterberg, Michael Bastubbe, Timo Berthold, Philipp Christophel, Mary Felenon, Koichi Fujii, Gerald Gamrath, Ambros Gleixner, Gregor Hendel, Kati Jarck, Thorsten Koch, Jeff Linderoth, Marco Lübbecke, Hans Mittelmann, Derya Ozyurt, Imre Pólik, Ted Ralphs, Domenico Salvagnin, Yuji Shinano, Franz Wesselmann, and Michael Winkler.

suitable for benchmarking due to problematic numerical properties, special constraint types (such as indicators), or exceptionally large memory requirements.

It should be emphasized that the constructed benchmark set is designed for the purpose of comparing the overall performance of general purpose solvers on a wide-ranging set of instances. Average performance on this set is not a suitable criterion to decide which MIP solver to use in a particular application scenario. For such decisions, it is important to consider what specific class(es) of instances are relevant, as well as what criteria beyond the raw speed and the ability to solve a wide range of problems are of interest. This is also underlined by the fact that each of the eight solvers that were used to collect performance data (see Section 3.5.5) proved to be the fastest solver on at least one instance.

Compiling a representative and meaningful instance library is a nontrivial endeavor. Compared to previous editions of MIPLIB, the increased number of submissions, the goals of compiling a significantly larger collection of instances and including a larger number of representatives of solvers posed new challenges to the selection process. In addition, MIPLIB 2017 is the first edition to provide supplementary data regarding the instances, such as the matrix structure and decomposability, as well as the underlying models from which the instances originated, where available. In order to produce a well-balanced library in a fair and transparent manner, we designed a new, heavily data-driven process. The steps applied between the initial submissions and the final MIPLIB 2017 are outlined in Figure 3.1. Driven by a diverse set of instance features, our methodology used multiple clusterings to populate a MIP model that was then solved to generate suitable candidates for the final library to be presented to the MIPLIB committee.

We consider this process of selecting instances from a large pool of submissions to be the main new feature of MIPLIB 2017. By contrast, the instances constituting previous versions of MIPLIB were manually selected by the members of the committee, depending heavily on their expertise in benchmarking to avoid common pitfalls like overrepresentation of certain problem classes. As one byproduct of this data-driven approach, we are now able to identify similar instances, which leads to sometimes surprising insights into connections between different, seemingly unrelated instances in the library. In addition to the raw feature data, we provide, for each instance, the five most similar instances in the collection on a newly designed web page (see Section 3.7.3).

This selection process inherently requires many manual decisions and heuristic choices. Some examples are the choice of features, the number of clusters to use when clustering instances according to this feature data, the definition of which instances to consider as computationally easy or hard, and our formalization of diversity and balance with respect to the feature data. All of these decisions have a high impact on the final result. This work tries to both adhere to and extend the established standards for MIP benchmarking, which we summarized in Section 2.9. Hence, in the remainder of this

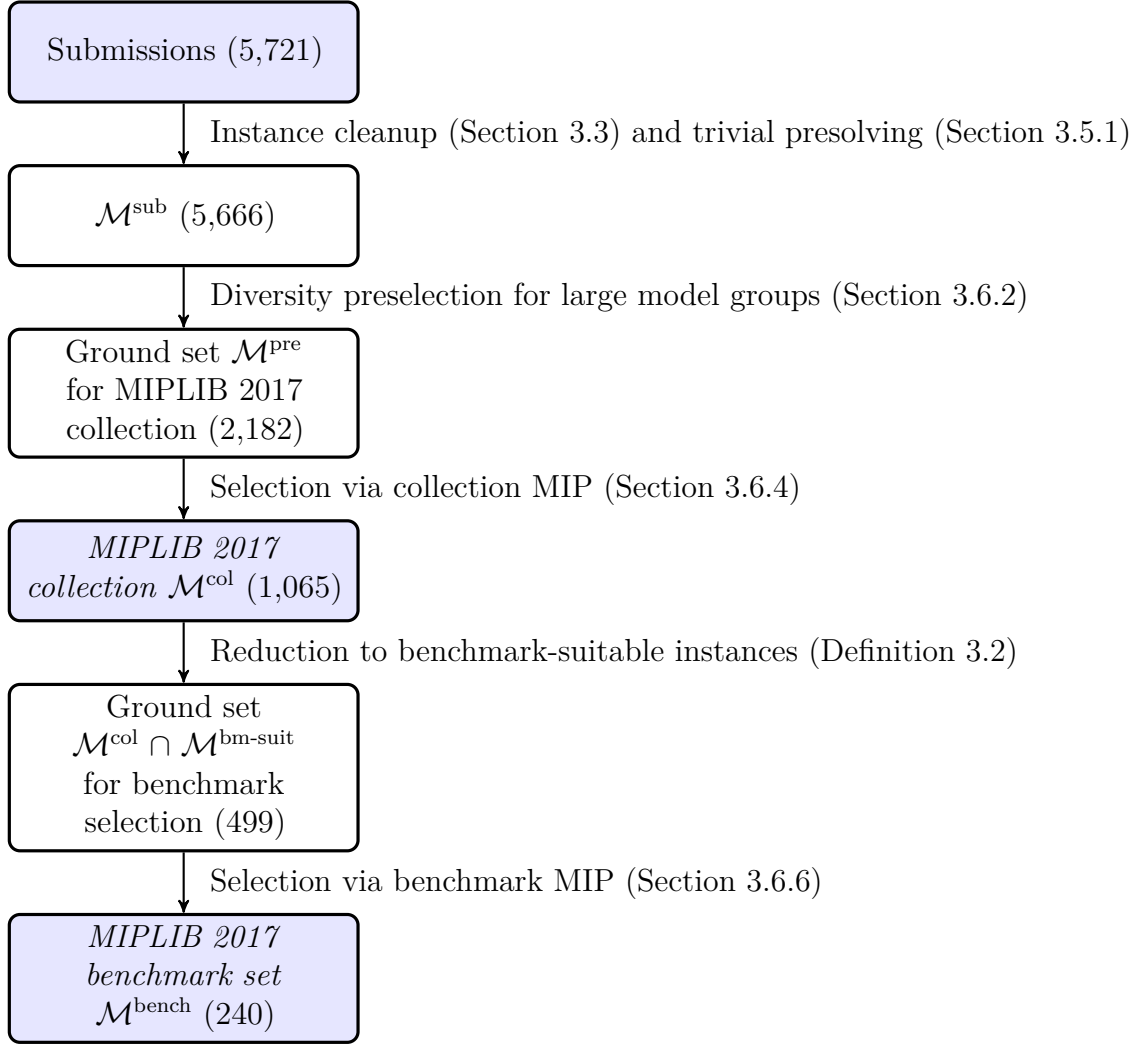


Figure 3.1: Outline of the steps involved in the selection of the MIPLIB 2017 collection and benchmark set. Number of instances remaining are given in parentheses.

chapter we try to describe the collection and selection of MIPLIB 2017 at a sufficient level of technical detail in order to make this process transparent to the reader.

The chapter is organized as follows. Section 3.3 briefly summarizes the submission and data cleanup phases of the MIPLIB 2017 project. In Section 3.4, we explain how we categorized all instances from the submission pool into so-called model groups. Section 3.5 details the collection of feature data for each instance that forms the basis for establishing similarity between instances and balance of a given selection. In Section 3.6, we describe how this data was used as input in order to compute good candidates for the library by solving a MIP model. Section 3.7 summarizes the obtained library of MIP instances. We wrap up this chapter with some concluding remarks in Section 3.8.

3.2 Related Work

With the evolution of computational research, standards and guidelines for conducting computational experiments were proposed and updated. Next to performance measures,

software availability and machine specifications, these guidelines also discuss the choice of test problems and the availability of standard test sets.

In 1978, Crowder et al. [Crowder et al., 1978] presented an early list of best practices, emphasizing the necessity of reproducibility of computational experiments. In this course, they also discussed the value of test problems being documented and the importance of using real-world problems. At the same time, they pointed out that sharing test problems is expensive and time-consuming—an issue that we have fortunately overcome.

The work of Jackson et al. [Jackson et al., 1991] was motivated by a controversy around the first published computational results for interior point LP solvers. Besides others, the ad-hoc committee stated: “We recommend that standard test problem sets and standard test problem generators be used wherever possible.” Further, they encouraged researchers that “whenever new problems are used in a computational evaluation, these problems be . . . submit[ed] to a standard collection like netlib.”

In his seminal paper from 1993, John Hooker characterized an *empirical science of algorithms* [Hooker, 1994]. He discussed the importance of identifying typical representatives of a problem class to conduct a study on, at the same time mentioning that any choice is open to the criticism of being unrepresentative. There are several resolutions for this issue; a well-known, long-standing and publicly available standard test set is certainly one of them. Another resolution that Hooker pointed out is to make “the issue of problem choice . . . one of experimental design”. This means making the question of how performance depends on test set characteristics part of the experiment itself. The approach taken to set up MIPLIB 2017—a data-driven instance selection that parameterizes the creation of a test set—can be seen as an extension of this idea.

There are various publications that formalize the problem of compiling a test set. McGeoch [McGeoch, 1996, 2001] developed an abstract model of algorithms and the paradigm of simulation research, partially in response to Hooker’s paper.

A complementary line of research that is not touched on in this chapter is the creation of new test instances to fill “gaps” in a test set, by learning from the instance parameter space, see [Smith-Miles and Bowly, 2015]. Smith-Miles et al. [Smith-Miles et al., 2014] use such an approach to work out the strengths and weaknesses of optimization algorithms on an enlarged instance set that they extrapolated from a standard test set. They detect so-called *pockets* where algorithm performance significantly differs from average performance. This take on instance diversity complements our approach of trying to achieve a best possible coverage of the feature space subject to a fixed set of candidate instances.

Work on standard test sets for benchmarking naturally connects to work on algorithm selection [Rice, 1976]. The work of Bischl et al. [Bischl et al., 2016] brings both fields together by publishing a benchmark library for algorithm selection. They introduce a data format to formally define features, scenarios, working limits and performance metrics. Since 2016, this library contains a MIP section, based on MIPLIB 2010.

Finally, related work of course includes various other libraries for mathematical optimization problems, including, but not limited to MINLPLib [Bussieck et al., 2003], QPLIB [Furini et al., 2019], Netlib [Browne et al., 1995], the COR@L collection [Linderoth and Ralphs, 2005], and OR-Library [Beasley, 1990]. The latter is one of the oldest online collections of optimization problems, existing for 30 years now and still being regularly updated.

3.3 First Steps: Collection of Instances and Data Cleanup

Following a public call for contributions, we received more than 120 submissions, where a single submission sometimes contained more than 100 MIP instances. One particularly large contribution comprises a curated set 785 new instances from the NEOS server. The NEOS Server is a free internet-based service for solving numerical optimization problems hosted by the Wisconsin Institute for Discovery at the University of Wisconsin in Madison, with remote solving services provided by various sites, such as Arizona State University. Details on the selection of the new NEOS instances can be found in [Gleixner et al., 2021]. An even larger contribution consisted in 988 MIP models of instances that were part of the MiniZinc Challenges from 2012 to 2016.⁶

In total, 3,670 instances were newly submitted to MIPLIB 2017. We arrived at a total of 5,721 instances by adding 2,051 instances that were submitted for inclusion in MIPLIB 2010, keeping their original submission information intact. Those 2,051 comprised most of the submissions to MIPLIB 2010 except for a few duplicates already present in other MIPLIB 2017 submissions.

During a subsequent data cleanup phase, all instances were converted into MPS [IBM, 1969; Nazareth, 1987] format. Further data cleanup steps consisted in

- the conversion of maximization into minimization,
- the conversion of lazy constraints into normal constraints,
- a renaming of files to match the MIPLIB file name conventions, and
- a renaming of NEOS files using names of rivers as suffices to facilitate communication.

Again, for more details, we refer the interested reader to [Gleixner et al., 2021]. Not all 5721 instances were considered as candidates. After removing all instances for which the integer variables could be entirely fixed via trivial presolving explained in Section 3.5.1, leaving the solver with either an empty problem or an LP to solve, the remaining set of candidates consisted of 5666 nontrivial instances.

⁶<https://www.minizinc.org/challenge.html>

3.4 Model Groups

In most cases, the instances in a single submission are closely related in the sense that they originate from the same or a very similar MIP model. Some submissions, however, contain many unrelated instances. Therefore, we introduced *model groups* to keep track of this form of meta information that may not be directly inferable from the submission ID or the numerical instance features described in Section 3.5. A model group represents a set of instances that is based on the same model or a very similar formulation but with different data. This grouping allowed us to avoid overrepresentation of a particular application or model class in the final library by limiting the number of instances with known similar model background during the selection process.

Each instance was assigned to one model group as follows. Initially, a submission of homogeneous instances was assigned to its own model group. If a submission contained multiple sets of instances, each implementing a different model for the same problem and data set, an individual group was created for each of the different model types. This procedure was applied to both the submissions to MIPLIB 2017 and the submissions to MIPLIB 2010. Publicly available instances with known application were grouped by hand by the authors. Examples for such cases are the MiniZinc instances submitted to MIPLIB 2017 and the instances from older MIPLIB versions and public instance sets submitted to MIPLIB 2010.

Instances from the NEOS server, however, are anonymous and lack meta data from the submitters that could be used to infer model groups. Nevertheless, users often submit multiple, similar instances. Hence, we used feature data described in the next section in order to infer synthetic model groups in an automated way. In order to group the NEOS instances, a k -means clustering was computed with respect to the entire instance feature space (see Section 3.5). The parameter $k = 110$ was chosen manually to achieve a clustering with very similar instances in each NEOS model group. This clustering was applied to all 1,176 NEOS instances both from new submissions and previously available sources.

Table 3.1 summarizes the number of resulting model groups and the corresponding group sizes for the different sources of instances. These numbers are given with respect to the 5,666 instances in \mathcal{M}^{sub} (see Figure 3.1). The largest model group *cmflsp* comprises 360 instances of a capacitated multi-family lot-sizing problem.

3.5 Feature Computation

The ultimate goal of the selection process was to select a representative sample of all available instances with respect to problem structure and computational difficulty, while avoiding overrepresentation of models that are too similar. In the spirit of data-driven decision-making and in light of related work in the fields of algorithm selection and

Type	Groups	Group Size \in			
		$\{1\}$	$\{2, \dots, 5\}$	$\{6, \dots, 10\}$	$\{11, \dots, 360\}$
MiniZinc	80	0	16	38	26
NEOS	110	8	24	34	44
MIPLIB 2017 submissions	130	81	15	10	24
MIPLIB 2010 submissions	241	172	30	16	23

Table 3.1: Model groups and counts for the different instance sources. Each group is counted only in the first applicable row.

machine learning, we based this process both on performance data and on an extensive set of instance features. The first step in the selection process was simply to determine the features of interest. In the terminology of [Smith-Miles et al., 2014], this means we defined a *feature space* of measurable characteristics and computed a *feature vector* associated to each element of the *problem space* of candidate instances.

Although this may seem straightforward, it is important to note that the feature vector corresponding to an instance can be affected by seemingly irrelevant properties of its representation in MPS format. For instance, some raw MPS instances contained modeling language artifacts or artificial redundancies. For this reason, the instance features were computed only after applying some straightforward simplification steps, which we refer to as *trivial presolving*. We first describe this presolving process before describing what features of the instances were used and how their values were determined for each instance.

3.5.1 Trivial Presolving

As is traditional for MIPLIB, the submitted instances are being made available in their original, unmodified form (except minor corrections to MPS formatting that were necessary in some cases). This means that the distributed instances were not presolved or simplified in any way for the purposes of *distribution*. For the purposes of *extracting features* of the instances, however, so-called “trivial” presolving was applied, as described below. It may seem strange that the version of each instance made publicly available in the final collection may actually be slightly different from the version considered during the selection process, but there are good reasons for this approach that we elaborate on in this section.

The justification for distributing the instances in their original submitted form is simply that this allows the most complete and realistic testing of the ability of each solver to deal with real-world instances, including all of the idiosyncratic artifacts that may arise in the modeling process. In particular, algorithms for presolving instances are actively developed and have a high impact on the performance of a solver (see [Achterberg et al., 2019; Gamrath et al., 2015b]). They not only strengthen the original formulation, but also simplify and remove unnecessary artifacts. These procedures are computational

in nature and their efficiency and effectiveness also needs testing. In some cases, there may be choices made in the presolving that can only be made with foreknowledge of the properties of the solution algorithm itself. For all these reasons, it was considered highly desirable in promoting test conditions that are as reflective of real-world conditions as possible to avoid modifying the distributed versions of the instances.

On the other hand, because MIP solvers do universally apply certain well-known simplification procedures to an instance before the branch-and-bound search, the unmodified descriptive data of the original instance may not properly reflect the “true” features of that instance for the purposes of clustering instances according to similarity, as we did during the selection process. The features considered that may be affected by presolving include not only obvious properties, such as instance size, but less obvious ones, such as the type of constraints and variables present in the model. A model may, for example, have all variables integer except for one continuous variable whose value is fixed to 1 and whose purpose is to model an objective offset. It would be unreasonable to consider such an instance to be an instance with both integer and continuous variables. At the other extreme, we may have an instance in which all binary and integer variables are implicitly fixed, leaving a purely continuous problem after presolving.

While it seems necessary to do some presolving before computing instance features, the full presolving done by solvers is itself a difficult computational balancing act and each solver does it differently. Too much presolving before feature computation would result in a presolved instance with features no more representative of the “true” ones than the completely unpresolved instance. As a compromise, all instance features introduced in Sections 3.5.3–3.5.4 were collected after applying a reduced set of the most obvious presolving techniques to the instance, but no more sophisticated techniques.

For this *trivial presolving*, we used SCIP 5.0, but disabled most presolving techniques, applying only simple ones, such as the removal of redundant constraints and fixed variables, activity-based bound tightening, and coefficient tightening. In contrast to standard SCIP presolving, which stops if the problem size could be reduced by only a small percentage during the last presolving round, we applied the simple presolving steps until a fixed point was reached. The complete set of SCIP parameters used to do the presolving is provided on the MIPLIB web page (see Section 3.7.3) as part of the feature extractor download.

For 55 of the 5,721 submitted instances, trivial presolving turns the instance into a pure LP or is even able to solve the instance by fixing all variables. These instances were not considered for inclusion and also serve to emphasize the importance of this preprocessing step. Overall, trivial presolving reduced the number of variables on 3,782 instances (66 % of the submission pool), sometimes by as much as 93 % (instance `a2864-99b1p`). For 445 instances (8 %), more than 50 % of the variables were fixed. On average, trivial presolving reduced the number of variables by 15 %.

3.5.2 Canonical Form

Let P be a trivially presolved MIP with m rows. For computing the instance features of P , we represent the rows of P by both a lower and an upper bound $b^{\text{left}}, b^{\text{right}} \in \mathbb{Q}_{\pm\infty}^m$, on the row activities to obtain the following representation of P ,

$$\min \left\{ c^t x : b^{\text{left}} \leq Ax \leq b^{\text{right}}, \ell \leq x \leq u, x \in \mathbb{Q}^n, x_j \in \mathbb{Z} \text{ for all } j \in \mathcal{I} \right\}. \quad (3.1)$$

The main difference between Representation (3.1) and our usual representation of MIP as by Definition 2.1 is that for the latter, two rows are required to represent an equality constraint. In contrast, in the notation of (3.1), an equation $a^t x = b$ can be represented as a single row using $b_i^{\text{left}} = b_i^{\text{right}} = b$.

It is important to note that the canonical form (3.1) is not uniquely determined for each instance. There remain certain degrees of freedom to formulate equivalent instances by scaling continuous columns, scaling the objective function $c^t x$, or scaling a constraint $b_i^{\text{left}} \leq a_i^t x \leq b_i^{\text{right}}$. This may cause problems with the computation of some features. For example, some features of interest involve comparison of row coefficients, but this comparison is difficult if coefficients of different rows of the constraint matrix differ by several orders of magnitude. We address these issues by normalizing the objective coefficients c and every constraint $b_i^{\text{left}} \leq a_i^t x \leq b_i^{\text{right}}$ by their maximum absolute coefficient $\|c\|_\infty$ and $\|a_i\|_\infty$, respectively, so that all objective and matrix coefficients lie in the interval $[-1, 1]$ before computing the feature matrix F (the downloadable instances are not altered).

It is based on this final presolved canonical representation that we define the $Q = 105$ features we consider. This results in a *feature matrix* $F \in \mathbb{R}^{N \times Q}$, where N is the total number of instances submitted. Table 3.2 lists these features, divided into $K = 11$ *feature groups*. Feature groups were used for the selection process, during which instance clusters were computed for each feature group individually. Every feature group was chosen to represent a particular aspect of an instance in the form specified by Equation (3.1). The computation of features in most of the groups only requires information that can be extracted directly from the input of the (trivially presolved) problem. Two exceptions are the constraint classification and decomposition groups, which need to identify structures in the model.

3.5.3 Instance Features

Here, we describe the first nine feature groups in Table 3.2. We use the shorthand *vector statistics* to refer to five values summarizing the entries of a vector $v \in \mathbb{R}_{\pm\infty}^d$.

Definition 3.1 (Vector Statistics). *Let $v \in \mathbb{R}_{\pm\infty}^d$, and let $d' = |\{j : |v_j| < \infty\}|$ be the number of finite entries of v , which can be smaller than d in the case of, e.g., bound*

Group	D	Description	Scaling
SIZE	3	size m, n of matrix, nonzero entries $ \{A \neq 0\} $	$\log_{10}(x)^2$
VARIABLE TYPES	3	Proportion of binary, general integer, and continuous variables $\frac{n^{\text{bin}}}{n}, \frac{n^{\text{gen}}}{n}, \frac{n^{\text{con}}}{n}$	
OBJECTIVE NONZERO DENSITY	5	Nonzero density of objective function $\frac{ \{c \neq 0\} }{n}$ both total and by variable type (bin., gen., cont.), 0-1 indicator for feasibility problems without objective	
OBJECTIVE COEFFICIENTS	6	vector stats. and dynamism of c	c normalized by $\ c\ _\infty$
VARIABLE BOUNDS	12	Finite densities $\frac{ \{\ell < \infty\} }{n}, \frac{ \{u < \infty\} }{n}$ of bounds, vector stats. of upper bounds u and bound ranges $u - \ell$.	vector stats. scaled by $\text{siglog}(x)$
MATRIX NONZEROS	6	vector stats. of nonzero entries $ \{a_i \neq 0\} $ by row in A , nonzeros per column $\frac{ \{A \neq 0\} }{n}$	$\log_{10}(x)$ for nonzeros per column
MATRIX COEFFICIENTS	19	vector statistics of the four m -dimensional vectors describing the min, mean, max, and std of the nonzero coefficients $\{a_i \neq 0\}$ in each row	every a_i normalized by $\ a_i\ _\infty$
ROW DYNAMISM	5	vector stats. of row dynamism $\frac{\ a_i\ _\infty}{\min_j \{ a_{ij} \neq 0 \}}$	$\log_{10}(x)$
SIDES	19	vector stats. of left- and right-hand sides $b^{\text{left}}, b^{\text{right}}$ and concatenated $(b^{\text{left}} b^{\text{right}})$, nonzero and finite densities of $b^{\text{left}}, b^{\text{right}}$	every a_i normalized by $\ a_i\ _\infty$
CONSTRAINT CLASSIFICATION	17	Proportion of classes of special linear constraints: singleton, precedence, knapsack, mixed binary (see Section 3.5.4)	
DECOMPOSITION	10	Features describing a decomposition found by GCG with maximum area score. Area Score, Number of Blocks, vector stats. of block sizes relative to the rows and columns. Not available for all instances.	

Table 3.2: Description of instance features used. Set notation is abbreviated, e.g., $\{A \neq 0\}$ denotes $\{(i, j) \in \{1, \dots, m\} \times \{1, \dots, n\} : a_{i,j} \neq 0\}$.

vectors, and let v' be the restriction of v to its finite entries. We assume without loss of generality that v' is sorted, $v'_1 \leq v'_2 \leq \dots \leq v'_{d'}$. The vector statistics of v' (and v) are

- $\min : v \mapsto v'_1$,
- $\max : v \mapsto v'_{d'}$,
- $\text{mean} : v \mapsto \frac{1}{d'} \sum_{j=1}^{d'} v'_j$,
- $\text{median} : v \mapsto \left(v'_{\lfloor \frac{d'+1}{2} \rfloor} + v'_{\lceil \frac{d'+1}{2} \rceil} \right) / 2$, and
- $\text{std} : v \mapsto \sqrt{\frac{1}{d'} \sum_{j=1}^{d'} (v'_j - \text{mean}(v'))^2}$.

Note that infinite entries can only occur for the variable bound vectors ℓ and u and the left- and right-hand side vectors b^{left} , b^{right} . For a vector v that contains only infinite entries, i.e., for which $d' = 0$, the above vector summaries are not well-defined. If $d' = 0$, the corresponding statistics were set to 0 in the data. Note that even if the original formulation has infinite bounds on variables, trivial presolving may often infer finite bounds for those variables.

The *dynamism* of a vector with finite entries is the ratio of the largest and smallest absolute entries, i.e., $\|v\|_\infty / \min\{|v_j| : v_j \neq 0\}$. The dynamism is always at least 1. If the dynamism of any single constraint exceeds 10^6 , this is an indication of a numerically difficult formulation. Note that the dynamism is invariant to the normalization procedure. Combining the dynamism of each constraint yields an m -dimensional vector, which can be summarized using vector statistics.

The feature group `MATRIX COEFFICIENTS` summarizes the nonzero coefficients of the matrix A as follows. First, each row a_i , $1 \leq i \leq m$, of A is normalized by its largest absolute coefficient, such that all coefficients are in the range $[-1, 1]$. The nonzero entries of a_i are then summarized by four of the five vector statistics explained above, namely the min, max, mean, and std. Going through all rows, we obtain four m -dimensional vectors describing the min, max, mean, and std per row. Each of these vectors is then summarized via vector statistics, which yields a total of 20 statistics that summarize the coefficients of A . Examples are the mean minimum coefficient over all m rows, or the standard deviation of all m maximum coefficients, etc. The feature group comprises 19 out of these 20 coefficient statistics. We dropped the maximum over all m maximum coefficients because it was equal to 1 for every instance in our data set.

For the feature group `SIDES`, the m -dimensional left and right-hand side vectors b^{left} and b^{right} are summarized individually via vector statistics of all their finite elements. Besides, we compute vector statistics for the finite elements of the concatenated $2m$ -dimensional vector $(|b^{\text{left}}|, |b^{\text{right}}|)$ that combines the absolute left and right-hand sides of all rows. Note that the row normalization by the maximum absolute coefficient also affects the row left and right-hand sides.

The last group in Table 3.2 consists of 10 features describing a decomposition found by GCG [Gamrath and Lübbecke, 2010]. A decomposition of a MIP is a partition of the rows and columns of A into blocks such that all the nonzero coefficients of a column (row) lie in the corresponding row (column) block, except for a special row and column border that may contain the rest. A suitable decomposition is a prerequisite for specialized solution techniques for large MIPs such as Dantzig-Wolfe reformulation [Dantzig and Wolfe, 1960] and Benders’ decomposition. A trivial decomposition always exists, in which all rows/columns are labeled as “border”. However, not every MIP admits a nontrivial decomposition into several, independent blocks. The features describe the number and quality of the decomposition as well as vector statistics of the relative sizes of the blocks. We refer to [Gleixner et al., 2021] for details.

For features such as the row or objective dynamism, which may differ by orders of magnitude between instances, we used a logarithmic scaling. While logarithmic scaling is fine for vectors with positive entries, it is not applicable to vectors with potentially negative entries such as the variable upper bounds u . In those cases, we apply a customized scaling

$$\text{siglog} : \mathbb{R} \rightarrow \mathbb{R}, x \mapsto \text{sig}(x) \log_{10}(|x| + 1)$$

to every entry of the corresponding column in the feature matrix F . The map siglog preserves the sign of each entry.

The collection of the instance features was performed with a small C++ application called the *feature extractor*, which extends SCIP by the necessary functionality needed to report features after trivial presolving and optionally accepts a settings file to modify the default presolving explained in Section 3.5.1. The feature extractor is a modified version of a code used already by [Georges et al., 2018] and available for download on the MIPLIB 2017 web page (see Section 3.7.3).⁷

3.5.4 Constraint Classification

Table 3.3 lists the constraint classification types used for the feature group CONSTRAINT CLASSIFICATION. A total of 17 types of linear constraints that often occur as a subset of the constraints of MIP instances were identified. The table is sorted from most specific to most general. If a constraint belongs to multiple types, the classification always assigns the most specific, i.e., topmost, type that applies. Note that even empty, free, and singleton constraints are listed. While these types are removed during trivial presolving, they may well be present in the original formulation.

⁷The actual computations reported in the following were carried out with five additional, redundant matrix features. Only during the preparation of the manuscript, they were identified to be identical to other features.

Type	Linear constraints of the form...
EMPTY	$a = 0$ (no variables)
FREE	$b^{\text{left}} = -\infty, b^{\text{right}} = \infty$ (no finite side)
SINGLETON	$a_j = 0$, for all $j \neq j^*$ (single variable)
AGGREGATION	$a_1x_1 + a_2x_2 = b$
PRECEDENCE	$ax_1 - ax_2 \leq b$ where <i>both</i> x_1 and x_2 are binary/general/continuous
VARIABLE BOUND	$a_1x_1 + a_2x_2 \leq b, x_1 \in \{0, 1\}$
SET PARTITIONING	$\sum x_j = 1, x_j \in \{0, 1\}$ for all j
SET PACKING	$\sum x_j \leq 1, x_j \in \{0, 1\}$ for all j
SET COVERING	$\sum x_j \geq 1, x_j \in \{0, 1\}$ for all j
CARDINALITY	$\sum x_j = k, x_j \in \{0, 1\}$ for all $j, k \geq 2$
INVARIANT KNAPSACK	$\sum x_j \leq b, x_j \in \{0, 1\}$ for all $j, b \in \mathbb{N}, b \geq 2$
EQUATION KNAPSACK	$\sum a_jx_j = b, x_j \in \{0, 1\}$ for all $j, b \in \mathbb{N}, b \geq 2$
BINPACKING	$\sum a_jx_j + a_{j'}x_{j'} \leq a_{j'}, x_{j'}, x_j \in \{0, 1\}$ for all $j, a_{j'} \in \mathbb{N}, a_{j'} \geq 2$
KNAPSACK	$\sum a_jx_j \leq b, x_j \in \{0, 1\}$ for all $j, b \in \mathbb{N}, b \geq 2$
INTEGER KNAPSACK	$\sum a_jx_j \leq b, x_j \in \mathbb{Z}$ for all $j, b \in \mathbb{N}$
MIXED BINARY	$\sum_{j \in \mathcal{B}'} a_jx_j + \sum_{j \in \mathcal{C}'} a_jx_j \{\leq, =\} b, \mathcal{B}' \subseteq \mathcal{B}, \mathcal{C}' \subseteq \mathcal{C}$
GENERAL LINEAR	$b^{\text{left}} \leq a^{\text{t}}x \leq b^{\text{right}}$ (no special structure)

Table 3.3: Classification of a linear row $b^{\text{left}} \leq a^{\text{t}}x \leq b^{\text{right}}$, sorted from most specific to most general. A constraint is always assigned the first (topmost) type that applies. Note that we omit the row index i to simplify notation.

There are several types of constraints supported by the MPS format [IBM, 1969; Nazareth, 1987] that are not strictly linear as required by Equation (3.1). A well-known extension are *indicator* constraints, which are conditional, linear constraints that only need to be satisfied if a corresponding binary variable, the so-called *indicator variable*, is set to 1. It is possible to linearize such a constraint by employing a sufficiently large coefficient M for the indicator variable, in which case the reformulation is called a *big-M formulation*. In many practical applications, big- M formulations require a very large value of M , which is why they often lead to numerically difficult models. Directly expressing such constraints as indicator constraint allows the solver to circumvent these numerical difficulties.

Indicator constraints were allowed into the MIPLIB 2017 collection, but (as we describe later) were not allowed in the benchmark set. The only feature used that involves indicator constraints was their fraction among all constraints. This feature is also part of the feature group CONSTRAINT CLASSIFICATION in Table 3.2. Other features regarding, e.g., the linear part of the indicator constraint, are not collected. In total, 437 of the submitted instances contain indicator constraints. Many of them

appear in instances of the MiniZinc submission, which were additionally submitted as big- M formulations.

There are other special types of constraints allowed by MPS, such *special-ordered sets* (SOSs), *semicontinuous variables*, and piecewise linear constraints that not all solvers support. However, none of the submitted instances used such constraints.

3.5.5 Acquisition of Performance Data

The selection of the complete MIPLIB 2017 collection (see Sections 3.6.2–3.6.4) was mainly driven by the feature data described above. The selection of the benchmark set (see Sections 3.6.1, 3.6.5, and 3.6.6), however, took into account information about the computational and numerical difficulty of the instances. In order to quantify these empirically, we collected performance data using current solver software.

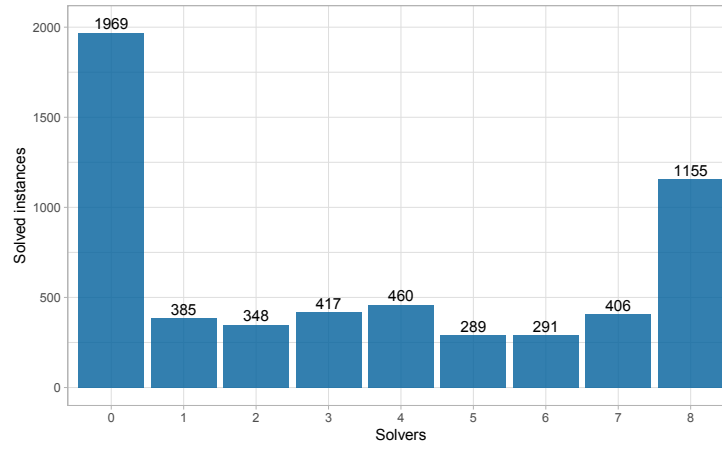
Every submitted instance was processed with each of the eight solvers listed in Table 3.4 to collect performance data. The experiments were performed on two Linux clusters. The first one consisted of 32 nodes, each equipped with two 3.20 GHz 4-core Intel Xeon X5672 CPUs and 48 GB RAM, the second consisted of 16 nodes, each equipped with two 2.50 GHz 10-core Intel Xeon E5-2670 v2 CPUs and 64 GB RAM. Two such jobs were run in parallel on the same cluster node, each job using a time limit of 4 hours and 4 threads, except for SCIP and MATLAB, which are both single-threaded.⁸ In total, this performance evaluation required almost 40 CPU years.

Figure 3.2a shows for every possible cardinality $k = 0, 1, \dots, 8$ the number of instances solved by exactly k solvers. 1,969 of the 5,721 instances (34 %) were not solved by any solver within 4 hours (an instance was considered solved if there were no inconsistencies between solvers and the solution was verified to be feasible (see Section 3.5.6). There were 1,155 instances (20 %) that could be solved by all eight solvers.

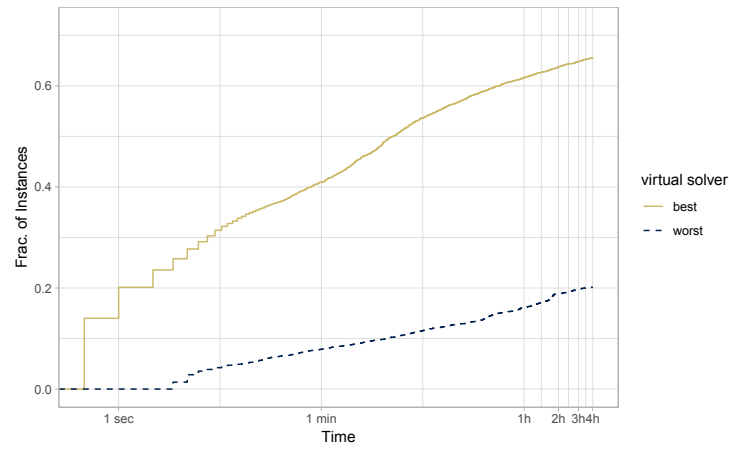
To summarize the results of the experiments, we report here the performance measures for *virtual solver*, as described in the introduction. For each instance, a virtual solver is a summary of all tested solvers by means of an aggregation function such as min, max, and median, resulting in the best, worst, and median virtual solvers, respectively. The term “virtual” is used to distinguish the presentation from the best (fastest) or worst (slowest) actual solver over the complete set of instances. The performance measures collected are the time to optimality and the number of branch and bound nodes processed.

Figure 3.2b compares the fraction of instances solved by the virtual best and worst solvers. A large discrepancy between the curves can be observed. The virtual best solver finished on about 20 %, 40 %, and 60 % of the submissions within 1 sec., 1 minute, and 1 hour, respectively. The virtual worst solver required more than a second for any instance, and solved only 20.2 % of the instances within the time limit of 4 hours. The virtual best solver solved more instances in 2 seconds than the virtual worst solver was able to

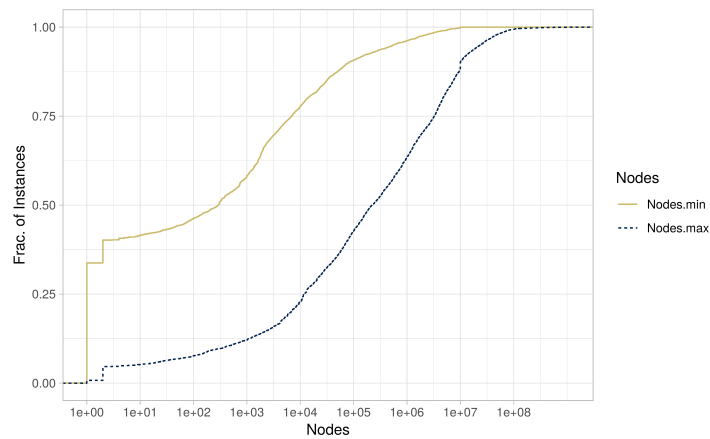
⁸MATLAB was run using the command `intlinprog`, which is part of the Optimization Toolbox (TM).



(a) Number of instances solved by a specific number of solvers.



(b) Fraction of instances solved by virtual best and worst solver for different time limits.



(c) Explored branch-and-bound nodes for *solved* instances.

Figure 3.2: Aggregated results of the performance evaluation on the entire submission.

Solver	Version	Threads
CBC ⁹	2.9.8	4
IBM CPLEX ¹⁰	12.7.1	4
Gurobi ¹¹	7.5.1	4
MATLAB ¹²	R2017b	1
MOSEK ¹³	8.1.0.30	4
SAS/OR ¹⁴	14.2	4
SCIP ¹⁵	4.0.0	1
FICO Xpress ¹⁶	8.2	4

Table 3.4: List of solvers used for performance evaluation.

solve in 4 hours. Note that all eight tested solvers contributed to the performance of the virtual best solver, i.e., each solver was the fastest on at least one instance.

Figure 3.2c summarizes data about the number of branch-and-bound nodes processed. As expected, the number of branch-and-bound nodes varies significantly between instances, but also between solvers on individual instances. In Figure 3.2c, the minimum and maximum number of explored nodes are shown. In this figure, we consider only runs that completed successfully. Note that there are differences in how solvers report the required number of branch-and-bound nodes. Concretely, the solution of an instance during presolving may be reported as 0 or 1 nodes depending on the solver used. We therefore normalize the node results so they are always at least 1, i.e., we consider presolving as part of the root node solution process. This is also justified because we only consider instances that could not be solved completely by trivial presolving. At the left end of the scale are the instances that could be solved within 1 node. This group amounts to 1,507 instances, which corresponds to 40 % of the instances that could be solved at all and 26 % overall. For more than 50 % of the solved instances, the solution process required less than 1,000 nodes. The maximum number of explored nodes is considerably larger. Less than 25 % of the considered instances were solved within 1,000 nodes by all solvers that finished within the time limit. Note that for all 385 records (see Figure 3.2a) for which only one solver finished within the time limit, the minimum and maximum number of explored nodes coincide.

⁹<https://projects.coin-or.org/Cbc>

¹⁰<https://www.ibm.com/analytics/cplex-optimizer>

¹¹<http://www.gurobi.com/products/gurobi-optimizer/gurobi-overview>

¹²<https://de.mathworks.com/products/optimization.html>

¹³<https://www.mosek.com/>

¹⁴https://www.sas.com/en_us/software/or.html

¹⁵<http://www.scipopt.org/>

¹⁶<http://www.fico.com/en/Products/DMTools/xpress-overview/Pages/Xpress-Optimizer.aspx>

3.5.6 Consistency Check of Solver Results

In order to identify numerically challenging instances and incorrect answers returned by the solvers, the results of the performance runs were independently verified in two different ways.

As for MIPLIB 2010, a solution checker tool has been used that verifies the feasibility of each returned solution against the input MIP in rational arithmetic using the arbitrary precision library GMP [Granlund and the GMP development team, 2016]. The refinements of the solution checker for MIPLIB 2017 consist of a more fine-grained computation of the activity of a constraint, and the integration of indicator constraints into the checker. For details, we refer to [Gleixner et al., 2021].

Following the feasibility check via the solution checker tool, which can be done independently for each solver and solved instance, the results were compared between solvers to identify discrepancies in the optimal values reported or inconsistencies in primal and dual bounds. We used the publicly available tool IPET [Hendel] to parse the solver log files and validate the results. We considered results inconsistent when solver A reported a verified, feasible solution \tilde{x} with objective value $Z = c^t \tilde{x}$, while solver B timed out reporting a dual (lower) bound that was higher than Z . This included the special case that an instance had been reported infeasible by solver B. For example, on the instance `bc1`, seven of eight solvers agreed on an optimal value of 3.338 after exploring search trees with 3k–20k nodes. The eighth solver, however, reported a solution of value 3.418 as optimal after 720 nodes. The eighth solver cut off the optimal solution. Note that while such a behavior can be caused by a bug in the solver, it is also possible that different optimal values can be “correctly” obtained when different tolerances are used. Since all MIP solvers rely on floating-point arithmetic and use feasibility tolerances, the definition of “the optimal objective value” for a problem instance is ambiguous. In particular, for numerically challenging problems, a solver might return a different optimal objective value as a result of applying slightly stricter tolerances within the algorithm. Instances exhibiting such ambiguity are not suitable for benchmarking, since handling this numerical ambiguity can be done in different ways, requiring different amounts of computational effort. This leads to difficulties in comparison. Therefore, we disregarded all instances with such inconsistencies during the selection of the benchmark set (see Section 3.6.6), unless the inconsistency was obviously caused by a bug in one solver; 328 instances (5 %) were removed for this reason.

3.6 Selection Methodology

Due to the vast number of collected instances and the stark overrepresentation of some problem classes and instance types, it was crucial to reduce the submitted instances to a carefully chosen selection that provides both researchers and practitioners with a

meaningful basis for experimental comparison. MIPLIB 2017 provides two main instance sets, namely the *benchmark set* and the *collection*. In the following, we discuss the actual selection process and the obtained result.

We approached this task in reverse order by first selecting the larger MIPLIB 2017 collection from the submitted instances, and then choosing the MIPLIB 2017 benchmark set as a subset of the collection. An overarching goal for both the collection and benchmark sets was to provide good coverage of the feature space of all submissions while maintaining balance. Note that we thus explicitly avoid allowing the distribution of instance properties observed in the set of submitted instances to affect the distribution in the final collection, since it is to be expected that the set of submitted instances would be highly unbalanced in its instance feature representation, if for no other reason than that some submissions contained many more related instances than others. A second overarching goal was to choose a collection as large as possible, in order to obtain a rich and diverse test set, but without sacrificing balance. As explained above, the benchmark selection step required additional restrictions. With respect to the benchmark, the goal was to choose a large set of instances, but with a bias towards instances that are currently hard for all solvers and keeping in mind that it should be possible to perform benchmarking in a “reasonable” amount of time.

It seems quite natural to formulate the selection task as an optimization problem. In fact, we approach the generation of MIPLIB 2017 with a sequence of optimization problems: a set of diversity preselection MIPs, the collection MIP, and finally, the benchmark MIP. After an initial cleanup, large model groups (see Section 3.4) are cut down to a handful of diverse instances by the application of the diversity preselection model described in Section 3.6.2. The main purpose of the first selection procedure was to avoid overrepresentation of instance types from large and very homogeneous submissions that do not add to the diversity of the instance library. Sections 3.6.3 and 3.6.5 introduce the clustering procedures to partition the instances based on instance features and performance data, respectively. Sections 3.6.4 and 3.6.6 describe the mixed-integer programming models used to compute the MIPLIB 2017 collection and benchmark sets. Although the selection of the benchmark set is only the final step of the process, the initial reduction steps must be aware of which instances are judged to be suitable for the benchmark set. Otherwise, too many benchmark-suitable instances might be excluded initially for selecting a good benchmark set later. Hence, we start in Section 3.6.1 by giving our definition of benchmark suitability.

Note that before the selection steps outlined here, the submission pool of 5,721 instances was already reduced to 5,666 instances by the removal of LPs (no discrete variables) and instances that are empty after a trivial presolving (see Section 3.5.1). Furthermore, we removed pairs of duplicate instances identified during the selection process.

3.6.1 Benchmark Suitability

The following definition characterizes the requirements for an instance to be in the benchmark set of MIPLIB 2017. It is important to point out that because we allow infeasible instances, successful “solution” of an instance for the purposes of this definition means that the solver either produced a (claimed) optimal solution or a (claimed) proof of infeasibility.

Definition 3.2 (Benchmark-suitable instance). *Denote by \mathcal{M}^{sub} the 5666 instances from the submission pool. We call an instance $i \in \mathcal{M}^{sub}$ benchmark-suitable if*

1. *it can be solved by at least one considered solver within 4 hours;*
2. *it requires at least 10 seconds with 50 % of the solvers;*
3. *it has a constraint and objective dynamism of at most 10^6 (see Section 3.5.3);*
4. *the absolute value of each matrix coefficient is smaller than 10^{10} ;*
5. *the results of all solvers on i are consistent (see Section 3.5.6);*
6. *it has no indicator constraints (see Section 3.5.4);*
7. *it has a finite optimum if it is feasible;*
8. *the solution (objective) value of i is smaller than 10^{10} ;*
9. *it has at most 10^6 nonzero entries.*

The subset of benchmark-suitable instances from the ground set \mathcal{M}^{sub} is denoted by $\mathcal{M}^{bm-suit}$.

2 eliminates instances from the benchmark selection that are too easy. Conversely, 1 ensures that benchmark instances can be solved by at least one solver, as already done for MIPLIB 2010. This avoids the situation of MIPLIB 2003, for which four instances still remain unsolved 15 years after the release of the test set. The criteria 3, 4, 5, 8 ensure that the benchmark set does not contain numerically difficult instances for which results may be ambiguous. Furthermore, benchmark instances should not contain special constructs that are not supported by all solvers. As noted in Section 3.5.4, the only special constraint type in the submissions are constraints of indicator type, which are excluded from the benchmark set via 6. 7 excludes feasible instances that do not have a finite optimal value from the benchmark set for two reasons. First, a feasible, rational MIP has a finite optimal value if and only if its LP relaxation has a finite optimal value, rendering detection of this property more a continuous than a discrete problem. Second, there is currently no clear consensus on the expected behavior and output of MIP solvers

Crit.	Exclusion reason	Instances
1	Too hard: min. solver time > 4 hours	1,958
2	Too easy: median solver performance ≤ 10 seconds	741
3	Objective or constraint dynamism too large	552
4	Absolute matrix coefficients $> 10^{10}$	525
6	Presence of indicator constraints	437
5	Instances excluded for inconsistent results	334
7	Unbounded instances	87
8	Best known solution exceeds 10^{10}	80
9	Too many ($> 10^6$) nonzeros	40

Table 3.5: Number of instances considered not benchmark-suitable ($\mathcal{M}^{\text{sub}} \setminus \mathcal{M}^{\text{bm-suit}}$), as described in Section 3.6. The column **Crit.** refers to the corresponding criterion in Definition 3.2. The ground set is the set of 5,666 instances available before preselection.

in the case of a feasible MIP without a finite optimum. Note that in contrast, infeasible instances are deliberately not excluded. Finally, 9 reduces the hardware requirements to perform tests with the benchmark set.

Table 3.5 lists for each criterion the number of excluded instances. Note that an instance may be excluded for several reasons. In total, 3,407 instances were labeled as not benchmark-suitable, the majority of them because no solver solved them within the time limit of four hours.

The larger MIPLIB 2017 *collection* covers a broader range of MIP instances. It includes at least one instance from each submitter, a constraint that cannot be enforced for the benchmark set due to runtime considerations. It may contain instances that are considered too easy or too hard for the benchmark set. It may contain instances with more dubious numerics suited for testing the robustness of solvers in general and techniques that explicitly aim at increasing numerical robustness. It may contain unbounded instances and instances with indicator constraints. It may contain up to five instances from each model group.

3.6.2 Diversity Preselection

As with previous editions of MIPLIB, the number of instances varies significantly between different submissions and, more importantly, also between the model groups described in Section 3.4. While some model groups contain a single MIP instance that represents an optimization problem on a specific data set, other model groups contain hundreds of instances using the same underlying model for different data. Hence, for larger model groups, we preselect a diverse subset of instances as follows.

Let $\mathcal{M} = \mathcal{M}^{\text{sub}}$ denote the index set of submitted instances and let $\mathcal{M}^{\text{bm-suit}} \subseteq \mathcal{M}^{\text{sub}}$ be the subset of benchmark-suitable instances according to Definition 3.2. The choice of a subset of instances can be naturally encoded using a vector of binary variables x_i equal to one if and only if instance $i \in \mathcal{M}$ is selected. For two instances $i, j \in \mathcal{M}$, $d_{i,j}$ denotes

the Euclidean distance of their feature vectors. Then for a given model group $G \subset \mathcal{M}$ of instances and a target cardinality $\kappa_G \in \mathbb{N}$, we wish to choose κ_G instances maximally diverse in the sense that the minimum distance between two selected instances becomes maximal. Moreover, if the model group contains benchmark-suitable instances, at least one of these should be included in the preselection. Such a preselection can be performed by solving the mixed binary optimization problem

$$\max \quad z \quad (3.2a)$$

$$\text{s.t.} \quad z \leq (d_{i,j} - \bar{d})x_i x_j + \bar{d} \quad \text{for all } i, j \in G, i \neq j \quad (3.2b)$$

$$\sum_{i \in G} x_i = \kappa_G \quad (3.2c)$$

$$\sum_{i \in G \cap \mathcal{M}^{\text{bm-suit}}} x_i \geq 1 \quad \text{if } G \cap \mathcal{M}^{\text{bm-suit}} \neq \emptyset \quad (3.2d)$$

$$x \in \{0, 1\}^G, z \in [0, \bar{d}] \quad (3.2e)$$

The value $\bar{d} := \max\{d_{i,j} : i, j \in G\}$ acts as big- M in Constraint (3.2b). In order to solve this optimization problem with a MIP solver, the bilinear product $x_i x_j$ in (3.2b) must be linearized by replacing it with an auxiliary binary variable $w_{i,j}$ under the additional constraints $w_{i,j} \leq x_i$, $w_{i,j} \leq x_j$, and $w_{i,j} \geq x_i + x_j - 1$ for all pairs $i \neq j \in G$.

This preselection was performed for each model group with six or more instances. We use $\kappa_G = 5$ for groups with $6 \leq |G| \leq 10$ and $\kappa_G = 10$ for larger groups. The number of variables and constraints of the preselection model depends on the size of the corresponding model group. For the largest model group “cmflsp”, which comprises 360 instances of a capacitated multi-family lot-sizing problem, the diversity preselection instance has 129,242 rows, 64,981 columns, and 323,485 nonzeros. Except for this largest model group, which required approx. 800 seconds, all preselection problems could be solved to optimality within a time limit of 500 seconds using Gurobi 7.5.1.

As an example, Figure 3.3 depicts the results of the preselection procedure for the model group “drayage”, which consists of 165 instances in total. The plot shows the instances from this group three times. The x and y -coordinates are computed using *t-distributed stochastic neighbor embedding (t-SNE)* [van der Maaten and Hinton, 2008], a technique to embed points of the high-dimensional feature space in 2D based on their distances. The leftmost plot highlights the optimal solution of the corresponding diversity preselection instance. Visually, the selected solution for this group is scattered evenly across the feature space of this group. The middle and right plot show the instances from this group that are selected for the collection and benchmark set, respectively, for which stricter cardinality restrictions on model groups apply. Despite those cardinality restrictions, the corresponding selections appear evenly spread across the ground set. A look at the performance results, which are taken into account for the selection of the benchmark set, reveals that the two selected instances also vary substantially regarding solution time. The easier of the two instances, namely **drayage-100-23** could

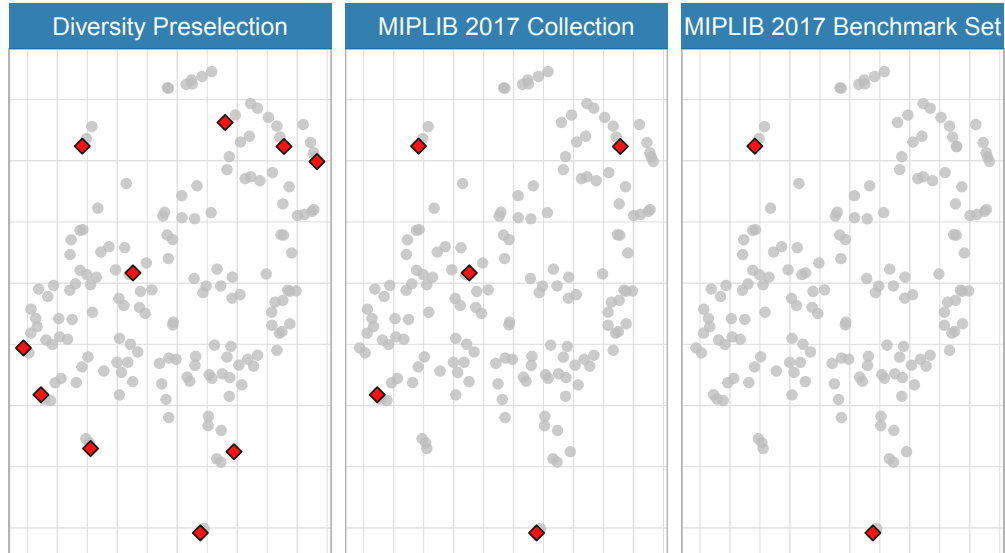


Figure 3.3: The results of diversity preselection for the model group “drayage”, which contains a total of 165 instances, in a t-SNE plot. A red diamond indicates that the instance has been selected for the corresponding set.

be solved by seven solvers with a median running time of 40 seconds. The harder instance *drayage*-25-23 on the other hand could only be solved by three solvers and hence has a median running time of four hours. The three other instances that are part of the collection lie in between. In total, diversity preselection reduces the instance set from 5,666 to 2,182 instances.

The preselected instances form a reduced index set \mathcal{M}^{pre} , which serves as input for the selection of the MIPLIB 2017 collection. Although the following selection procedure could have been applied to the entire set of submissions, we noticed several benefits of preselection empirically. It improves the results of the k -means clustering heuristic in the next Section 3.6.3, reduces the size and difficulty of the selection MIPs to be solved, and finally leads to a larger collection and benchmark set.

3.6.3 Preparing Multiple Clusterings

One major challenge in selecting a test set is how to navigate the trade-off of good coverage of all observed instance properties against a balanced selection that avoids overrepresentation. The first goal is best achieved by simply selecting all suitable instances, while balance explicitly asks for removing instances from overrepresented problem classes.

A straightforward method would be to compute one clustering according to the entire feature matrix and pick instances uniformly from each cluster. When applied in a high-dimensional feature space, as in our setting, this naïve approach suffers from several problems well-known in data analysis, such as the curse of dimensionality [Beyer et al., 1999] and the relative scaling of numerical features. The first term refers to the

fact that with increasing dimensionality of the feature space, the difference of distances of one point to its nearest and to its farthest neighbor becomes smaller in relative terms. Hence, similar instances cannot be identified reliably. Conversely, depending on scaling, the distance with respect to one crucial feature may be dominated by less useful features, such that different instances cannot be distinguished reliably. Arguably, the same problem holds true everywhere where we use Euclidean distances in the selection process, for example during the diversity preselection in the previous section. An important difference between this section and the preselection is that for preselection, we only considered one model group (with many instances) at a time so that we already knew that all instances were similar to each other. In this and the following sections, however, this model group association is no longer used for the clustering, as we now incorporate also instances with no known model group association.

We therefore counteract the problems of a high-dimensional feature space by using multiple clusterings of the entire preselected instance set \mathcal{M}^{pre} according to disjoint groups of features. Subsequently, we select instances such that they are balanced with respect to each of these clusterings. This selection process is more complex and cannot be achieved by simply picking uniformly from each cluster of each clustering. Instead, we formulate a mixed-integer programming problem with approximate balancing constraints for each of the multiple clusterings.

Formally, for a given index set \mathcal{M} of instances we have K different clusterings, i.e.,

$$\mathcal{M} = C_{k,1} \cup \dots \cup C_{k,L_k} \quad (3.3)$$

for $k \in \mathcal{K} = \{1, \dots, K\}$, with disjoint $C_{k,1}, \dots, C_{k,L_k}$ being a partition of the index set \mathcal{M} for every k . The number of clusters L_k is allowed to vary, since different subsets of features may require a different number of clusters to achieve a high-quality clustering. We denote the index set of all clusters by $\mathcal{C} := \{(k, \ell) : k = 1, \dots, K, \ell = 1, \dots, L_k\}$. Furthermore, the cluster sizes contain outliers, which need special treatment in order to avoid limiting the size of the resulting test set too much. Hence, we partition the set of clusters into small, medium (regular-sized), and large clusters and denote the respective index sets by $\mathcal{C}^{\text{small}}$, $\mathcal{C}^{\text{medium}}$, and $\mathcal{C}^{\text{large}} \subseteq \mathcal{C}$, as follows. A cluster $C_{k,\ell}$ is denoted small if its size is less than half the average size of $C_{k,1}, \dots, C_{k,L_k}$. On the other hand, a cluster is treated as large if it is displayed as an outlier in a typical boxplot. Concretely, $C_{k,\ell}$ is considered large if its size exceeds the 75 % quantile among $C_{k,1}, \dots, C_{k,L_k}$ by more than 1.5 interquartiles.

For the selection of the MIPLIB 2017 collection, we use one clustering for each of the $K = 11$ groups of instance features listed in Table 3.2. The clusterings of all preselected instances (2,182 instances) are computed using a k -means heuristic [Hartigan and Wong, 1979], which yields a first family of clusterings denoted by $\mathcal{K}_1 = \{1, \dots, 11\}$.

Feature group	Quality [%]	L_k	Small	Large	$\delta_{k,p}$	
					Min.	Max.
VARIABLE BOUNDS	91.28	23	7	1	6.57	22.03
MATRIX COEFFICIENTS	87.73	41	2	1	5.33	13.95
MATRIX NONZEROS	89.10	33	3	0	5.86	16.70
DECOMPOSITION	99.99	9	1	1	11.54	15.53
ROW DYNAMISM	91.70	17	3	3	9.85	18.12
CONSTRAINT CLASSIFICATION	87.77	35	6	1	6.63	12.92
OBJECTIVE NONZERO DENSITY	93.17	9	0	1	9.95	14.01
OBJECTIVE COEFFICIENTS	96.51	43	2	3	4.39	23.05
SIDES	98.33	35	10	0	1.00	18.77
SIZE	87.46	9	2	0	10.95	15.01
VARIABLE TYPES	95.26	7	0	1	8.89	13.54

Table 3.6: Clustering statistics for the collection MIP in Section 3.6.4.

Table 3.6 gives insight into the result of this clustering process. It shows the total number of clusters (value of L_k) for each feature group clustering. The quality column describes the percentage of the total feature group distance between clusters. More formally, for a clustering $k \in \{1, \dots, K\}$ of instances, the *quality* of this clustering is

$$\frac{\sum_{i \neq j \in \mathcal{M}} d_{i,j} - \sum_{\ell=1}^{L_k} \sum_{i \neq j \in C_{k,\ell}} d_{i,j}}{\sum_{i \neq j \in \mathcal{M}} d_{i,j}} \cdot 100 \, \%.$$

The value range of the above fraction is the interval $[0, 1]$ such that the quality lies between 0 and 100 %. A clustering has a high quality if long distances between instances are between different clusters. Note that for the distance computation for the quality measure, only features contained in the corresponding feature group were considered. The table shows that the quality of the clustering was at least 87 % and often significantly above 90 %, yielding an average quality of 92 %. The individual value of L_k has been manually selected for every feature group as the minimum integer L that admits a clustering with at least 90 % quality over the set \mathcal{M}^{sub} (5,666 instances) unless the targeted quality was not achievable using a reasonable amount of clusters. In addition to the number of clusters L_k , the table presents the number of small and large clusters, which are constrained less strictly than the medium clusters, see Constraints (3.5a) and (3.5b) below.

The last two columns report the minimum and maximum *total dissimilarity* per feature group. For each cluster $(k, \ell) \in \mathcal{C}$, its total dissimilarity $\delta_{k,\ell} \geq 1$ is defined as the shifted geometric mean of the pairwise Euclidean distances $\{d_{i,j} : i < j \in C_{k,\ell}\}$ between its instances, using a shift of 1. Here, distances are computed with respect to the entire feature space, in contrast to the above cluster quality computation. Because of the shift by 1, the smallest possible value of $\delta_{k,\ell}$ is 1, which only occurs for clusters

containing exactly one element, or for clusters that contain only instances which are indistinguishable in the feature space. All cluster parameters from Table 3.6 enter the selection constraints described in the next section.

For the selection of the benchmark set, we additionally use performance data to hand-craft three clusterings for each of the eight participating solvers, which yields additional clusterings $\mathcal{K}_2 = \{12, \dots, 35\}$, see Section 3.6.5 below.

3.6.4 Selection of the MIPLIB 2017 Collection

In the following, we describe linear formulations to enforce the requirements specified by the committee. At this stage, the instance set was limited to the instances $\mathcal{M} = \mathcal{M}^{\text{pre}}$ left after the diversity preselection procedure. The set of clusterings $\mathcal{K} = \mathcal{K}_1$ was the one determined using the instance feature groups from Table 3.2.

To express balance, consider one clustering $\mathcal{M} = C_{k,1} \cup \dots \cup C_{k,L_k}$. Naïvely, we would like to pick the same number of instances from each cluster, i.e., $\sum_{i \in C_{k,\ell}} x_i \approx y_k$ for an auxiliary variable $y_k \geq 0$. However, enforcing this for all clusterings is highly restrictive. Furthermore, while the instances in each of the clusters $C_{k,1}, \dots, C_{k,L_k}$ should be homogeneous with respect to the features that were used to compute clustering k , they may be heterogeneous with respect to the entire feature vector. This interaction between different clusterings must be taken into account.

To achieve this, we consider the total dissimilarity of the clusters that was introduced above. Arguably, from clusters with higher total dissimilarity, more instances should be picked, i.e., $\sum_{i \in C_{k,\ell}} x_i \approx \delta_{k,\ell} y_k$. Introducing a tolerance parameter ϵ , $0 < \epsilon < 1$, we arrive at the balance constraints

$$(1 - \epsilon)\delta_{k,\ell}y_k \leq \sum_{i \in C_{k,\ell}} x_i \leq (1 + \epsilon)\delta_{k,\ell}y_k. \quad (3.4)$$

Concretely, we used $\epsilon = 0.5$ for the selection of the collection and the benchmark set. In practice, we discard the left inequality for small clusters and the right inequality for large clusters and use

$$\sum_{i \in C_{k,\ell}} x_i \geq (1 - \epsilon)\delta_{k,\ell}y_k \quad \text{for all } (k, \ell) \in \mathcal{C} \setminus \mathcal{C}^{\text{small}}, \quad (3.5a)$$

$$\sum_{i \in C_{k,\ell}} x_i \leq (1 + \epsilon)\delta_{k,\ell}y_k \quad \text{for all } (k, \ell) \in \mathcal{C} \setminus \mathcal{C}^{\text{large}}. \quad (3.5b)$$

Additionally, if two instances have identical feature vectors, then at most one of them should be chosen, i.e.,

$$x_i + x_j \leq 1 \quad \text{for all } i, j \in \mathcal{M} \times \mathcal{M} \text{ with } i < j, d_{i,j} = 0. \quad (3.5c)$$

At most five instances should be selected from each model group. If the model group contains benchmark-suitable instances, at least one of those should be included into the MIPLIB 2017 collection. Let $\mathcal{M} = G_1 \cup \dots \cup G_P$ denote the partition of instances into different model groups, then this condition reads

$$\sum_{i \in G_p} x_i \leq 5 \quad \text{for all } p = 1, \dots, P, \quad (3.5d)$$

$$\sum_{i \in G_p \cap \mathcal{M}^{\text{bm-suit}}} x_i \geq 1 \quad \text{for all } p = 1, \dots, P \text{ with } G_p \cap \mathcal{M}^{\text{bm-suit}} \neq \emptyset. \quad (3.5e)$$

Furthermore, from each submitter at least one instance should be selected, i.e.,

$$\sum_{i \in \mathcal{S}_s} x_i \geq 1 \quad \text{for all } s = 1, \dots, S, \quad (3.5f)$$

where $\mathcal{M} = \mathcal{S}_1 \cup \dots \cup \mathcal{S}_S$ denotes the partition of instances with respect to S submitters.

Finally, we imposed relative limits on a small number of specific subsets of instances, $\mathcal{R}_r \subset \mathcal{M}$, by requiring

$$\sum_{i \in \mathcal{R}_r} x_i \leq \rho_r \sum_{i \in \mathcal{M}} x_i \quad \text{for all } r \in \{\text{MiniZinc, NEOS, small, medium, BP}\}. \quad (3.5g)$$

The concrete values for ρ_r , for both the collection MIP and the benchmark MIP described in Section 3.6.6, are given in Table 3.7. The numbers in parentheses show the size of respective ground sets \mathcal{M}^{pre} and $\mathcal{M}^{\text{col}} \cap \mathcal{M}^{\text{bm-suit}}$, from which instances were selected. The number of instances from the NEOS server was limited by the committee because of the lack of reliable information on their application and model background. Purely binary problems (BP) were limited because they often represent academic applications such as combinatorial puzzles, but less often occur in industrial, “real-world” instances. The limit ensures that enough actual mixed-integer instances are selected for the collection and benchmark sets. For the groups in Table 3.7 that refer to instance features, the features are always evaluated after trivial presolving.

Subject to those constraints, our objective was to include as many instances as possible, preferring benchmark-suitable instances. Hence, we formulate the collection MIP as the mixed binary optimization problem

$$\max \left\{ \sum_i \beta_i x_i : (3.5a) - (3.5g), x \in \{0, 1\}^{\mathcal{M}^{\text{pre}}}, y \in \mathbb{R}_{\geq 0}^{\mathcal{K}} \right\}, \quad (3.6)$$

with objective coefficients β to prefer benchmark-suitable instances,

$$\beta_i = \begin{cases} 2, & \text{if } i \in \mathcal{M}^{\text{bm-suit}}, \\ 1, & \text{otherwise.} \end{cases}$$

k	Sets	Coll. MIP (2182)		Bench. MIP (499)	
		Size	ρ_k	Size	ρ_k
1	MiniZinc instances	484	0.05	14	0.05
2	NEOS instances	696	0.33	182	1.00
3	small ($n \leq 2,000$)	691	0.20	124	0.20
4	medium ($n \leq 10,000$)	1,309	0.50	290	0.50
5	BP ($n = n^{\text{bin}}$)	422	0.20	109	0.20

Table 3.7: Instance sets for which relative limits on the selection apply, with limits shown for the collection and benchmark MIPs individually. The Size columns show the size of this set within the respective ground set the selection is based on. The parameter ρ_k is a relative limit on the allowed instances from this set in a solution as specified by Constraint (3.5g).

We solved the collection MIP over the ground set \mathcal{M}^{pre} of 2,182 instances (after diversity preselection). Despite our efforts to remove obvious duplicates, there remained 48 pairs of instances in \mathcal{M}^{pre} with a feature distance of zero. From each such pair, at most one instance was selected for \mathcal{M}^{col} because of Constraint (3.5c). We obtained the MIPLIB 2017 collection \mathcal{M}^{col} , which comprises 1,065 instances, 499 of which are benchmark-suitable.

The choice of the balance parameter ϵ used in Constraints (3.5a) and (3.5b) clearly plays a central role in the formulation of the collection MIP. In order to analyze the sensitivity of the result to this parameter, we solved the collection MIP for different values of ϵ . Table 3.8 reports on the sizes of the resulting collection sets. The smallest tested value of 0.1 makes the collection MIP infeasible, while the second-smallest value of 0.2 did not return with a feasible selection after 15 minutes. For larger values of ϵ , the collection MIP is feasible and can be solved reasonably quickly. The number of selected and benchmark suitable instances increases along with the balance parameter and allows us to control the number of selected and benchmark-suitable instances.

3.6.5 Performance Clusterings

In addition to instance features that depend exclusively on instance data, computational difficulty is an important aspect to consider for the benchmark set. We assessed the computational difficulty of every instance empirically by considering the performance data of the eight tested solvers (see Section 3.5.5). To quantify performance, we consider the observed running time $\tau_i(w) > 0$ for each solver $w \in \mathcal{W} := \{1, \dots, W\}$ and instance $i \in \mathcal{M}$. If w could not solve the instance i , $\tau_i(w)$ was set to the time limit of four hours. We denote by $\mathcal{M}_w \subseteq \mathcal{M}$ the set of instances that were solved by w within the time limit.

For each of the participating solvers, we created three different clusterings of the instances to capture different aspects of performance. The base set \mathcal{M} for these clusterings

ϵ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
selected instances	–	–	740	946	1065	1110	1125	1140	1140
benchmark-suitable	–	–	355	465	499	546	560	561	564

Table 3.8: Influence of the parameter ϵ used in Constraints (3.5a) and (3.5b) on the number of instances selected for the collection set. The second row reports how many of the selected instances are benchmark-suitable.

are the 499 benchmark-suitable instances within the MIPLIB 2017 collection, i.e., $\mathcal{M} = \mathcal{M}^{\text{col}} \cap \mathcal{M}^{\text{bm-suit}}$. The overall goal was to avoid a biased selection of instances, i.e., to avoid that the absolute and relative performance of a solver on the benchmark set appears different than on \mathcal{M}^{col} . Each of the three clusterings avoids a different bias.

Absolute performance clustering. The first clustering uses an absolute performance ranking. For each solver w , we sorted the instances in \mathcal{M}_w according to increasing running time $\tau_i(w)$. For a fixed number of clusters B , which we set to $B = 11$, we grouped the instances in \mathcal{M}_w into B equally-sized clusters w.r.t. increasing rank, i.e., we assigned the instances solved fastest to the first cluster, the next fastest set of instances to the second cluster, \dots , and the slowest instances to the last cluster, so that each cluster contained approximately $|\mathcal{M}_w|/B$ instances.

The instances in $\mathcal{M} \setminus \mathcal{M}_w$ that could not be solved by solver w seem indistinguishable with respect to performance. However, it could be that the solution process was terminated only seconds prior to concluding the proof of optimality, or that the solution process would have continued unfinished for days or even months. We took this into account by inspecting the performance of the other solvers and formed two more clusters from those instances: instances that could be solved by exactly one solver and instances that could be solved by more than one solver. The case that instances could be solved by no other solver does not appear since such instances are not benchmark-suitable (Definition 3.2, Criterion 1).

Relative performance clustering. In contrast to the absolute performance clustering, the instances were ranked based on relative solver performance for a second clustering as follows. To this end, we defined the *relative performance* of solver w on instance i with respect to the other solvers as

$$t_{w,i}^{\text{rel}} := \frac{\tau_i(w) + \tau}{\min_{w' \neq w} \tau_i(w') + \tau}, \quad (3.7)$$

where $\tau \in \mathbb{R}_{\geq 0}$ is a nonnegative shift as in the computation of shifted geometric means. Relative performance locates the individual solver performance relative to all other solvers on an instance, regardless of the absolute scale. The motivation is that solvers and solver improvements are traditionally measured by the shifted geometric mean

instead of the arithmetic mean. For the instances that could be solved by this solver, we used this ranking to define B equally-sized clusters in the same fashion as with the absolute performance ranking. The timeout instances were again divided into two further clusters of instances that could be solved by exactly one and by more than one other solver, respectively.

Binned absolute performance clustering. The third clustering uses absolute solving time directly, partitioning possible solving times into $B' = 7$ intervals

$$[T_0 = 0, T_1), [T_1, T_2), \dots, [T_{B'-1}, T_{B'}), \quad (3.8)$$

whose breakpoints are equal for all solvers. The concrete bin width used grows exponentially as follows.

$$T_j = 10^{-3.5+0.5j} \cdot 14400, \quad j = 1, \dots, 7.$$

Hence, the rightmost bin T_7 has the time limit of four hours as right breakpoint. Then for each solver $w \in \mathcal{W}$ we formed B' clusters $\{i \in \mathcal{M}_w : \tau_i(w) \in [T_{j-1}, T_j)\}$, $j = 1, \dots, B'$. Empty clusters were discarded. This is different from the absolute and relative performance clusterings in that it partitions the instances solved by a solver into clusters that differ in size. The instances in $\mathcal{M} \setminus \mathcal{M}_w$, which could not be solved by solver w , were again treated as two further clusters as above.

All in all, this led to 24 clusterings, $\mathcal{K}_2 = \{12, \dots, 35\}$. The ranking-based clusterings yield approximately equal cluster sizes over \mathcal{M}_w , but these cluster sizes may differ to the ones on $\mathcal{M} \setminus \mathcal{M}_w$. The binned absolute performance clustering does not control cluster size and may yield very unequally sized clusters. In the following benchmark MIP we picked from the performance clusters according to their size, i.e., we use $\delta_{k,\ell} = |C_{k,\ell}|$ for all $k \in \mathcal{K}_2$ in Constraint (3.5a) and (3.5b).

In the case of SCIP, as an example of an absolute performance clustering, each of the 11 parts contained between 20 and 22 instances, and the remaining 263 instances were split into 58 instances which could only be solved by one solver, and 205 instances solved by at least two solvers. Although the absolute and relative performance clusters were, by design, almost equal in size, we observed quite different partitions of the ground set. An example is the fastest relative performance cluster for SCIP, which shares 8 of its 21 instances with the fastest absolute cluster. The remaining 13 instances, for which SCIP was particularly fast compared to its competitors, were spread across 8 of 10 possible absolute clusters. This shows that, to some extent, the two suggested clusterings exhibit an almost orthogonal instance partition.

3.6.6 Selecting the MIPLIB 2017 Benchmark Set

The benchmark set was selected from the ground set of benchmark-suitable instances in the MIPLIB 2017 collection, $\mathcal{M} = \mathcal{M}^{\text{col}} \cap \mathcal{M}^{\text{bm-suit}}$. The balance constraints (3.5a) and (3.5b) are now defined using instance feature and performance clusterings, $\mathcal{K} = \mathcal{K}_1 \cup \mathcal{K}_2$. The rationale is that the current performance of solvers in the collection should be reflected by the performance on the benchmark set in order to avoid any unintentional bias towards a solver during the selection of instances. The relative limit constraints (3.5g) are kept, but the restriction on instances from the same model group is reduced to one instance from NEOS groups and two instances, otherwise. The stricter limit on NEOS instances is imposed because little information is available for these anonymously submitted instances and the chance for duplicate instances in the same model group is deemed higher.

In addition, executing one benchmark run on this test set should be possible within a reasonable time frame on modern hardware, possibly a compute cluster. We specified a total time limit τ^{lim} of 32 days for running the benchmark set with a hypothetical solver with median running times capped to a time limit of two hours, i.e., $\bar{t}_i := \min\{\text{median}\{\tau_i(w) : w \in \mathcal{W}\}, 7200\}$. The resulting *benchmark MIP* reads

$$\max \sum_i \beta_i x_i \tag{3.9a}$$

$$\text{s. t. } (3.5a), (3.5b), (3.5g),$$

$$\sum_{i \in G_p} x_i \leq \begin{cases} 1 & \text{for all } p = 1, \dots, P \text{ from NEOS,} \\ 2 & \text{otherwise,} \end{cases} \tag{3.9b}$$

$$\sum_{i \in \mathcal{M}} \bar{t}_i x_i \leq \tau^{\text{lim}}, \tag{3.9c}$$

$$x \in \{0, 1\}^{\mathcal{M}}, \tag{3.9d}$$

$$y \in \mathbb{R}_{\geq 0}^{\mathcal{K}}. \tag{3.9e}$$

This approach has a potential drawback. Representing current solver performance may overly favor instances that are tractable by current solver technology. This is opposed to one main goal of the MIPLIB project, which is to provide a test bed that drives solver development forward. Hence, we used the objective coefficient

$$\beta_i := 1 + \frac{1}{20} \sqrt{\min_{w \in \mathcal{W}} \tau_i(w)} \tag{3.10}$$

for instance $i \in \mathcal{M}$. This favors instances that are challenging even for the virtual best solver. The concrete choice of the square root was empirically motivated.

Using $\mathcal{M}^{\text{col}} \cap \mathcal{M}^{\text{bm-suit}}$ containing 499 instances as ground set, we solved the benchmark MIP that respects all feature group and performance clusterings. The solution to the benchmark MIP contained 240 instances and now constitutes the MIPLIB 2017 benchmark set $\mathcal{M}^{\text{bench}}$. We note that the imposed running time constraint (3.9c) was not tight on our performance data. A solver with median running times and a time limit of two hours would take about 14 days to process all benchmark instances sequentially.

We also note that for several reasons, the goal of representing computational difficulty in the reduced benchmark set cannot be achieved perfectly and is not even well-defined: The performance data is only a snapshot of current algorithms. It was gathered using a time limit, performance variability and parallel scalability were not captured, and correctness was only enforced approximately with respect to the tolerance parameter ϵ . Last but not least, the different performance clusterings may even contradict each other. However, we hope that it helps to avoid unintentionally and unconsciously introducing a bias both towards instances of a particular difficulty and towards any of the solvers.

3.7 The Final Collection and Benchmark Sets

The MIPLIB 2017 collection \mathcal{M}^{col} has been compiled with a focus on a balanced and diverse representation/coverage of the feature space. The benchmark set $\mathcal{M}^{\text{bench}}$ incorporates similar requirements also for the performance data. This section discusses the feature and performance aspects of the compiled sets. We also assess the descriptive power of the feature space by (re-)detecting known model group associations.

3.7.1 Representation in Feature Space

A frequent question during the discussions about the MIPLIB 2017 compilation process was whether the choice of features and their scaling is able to distinguish instances in a useful way. Ideally, two instances based on the same model for the same application, but with different data, should be close to each other in the feature space, regardless of variations of, e.g., the size of the matrix. For MIPLIB 2017, we have two sources to assess similarity between instances, namely their distances in feature space and the model groups G from Section 3.4. In this paragraph, we evaluate the descriptive power of the feature space by comparing similarity in feature space and model group association of instances.

Let $\mathcal{M} = \mathcal{M}^{\text{col}}$ denote the instances in the MIPLIB 2017 collection ($|\mathcal{M}| = 1065$). The complete graph $K_{\mathcal{M}} = (\mathcal{M}, E)$ on the vertex set \mathcal{M} has $|E| = \binom{|\mathcal{M}|}{2} = 566580$ edges. Now consider two subsets of the edges. Let E_G denote edges between instances from the same model group. Furthermore, let for every $i \in \mathcal{M}$, $S_i \subset \mathcal{M} \setminus \{i\}$ denote the set of five most similar instances to i in the collection w.r.t. the distance in the scaled

feature space. With the sets S_i , we define E_S to be the set of *similarity edges* as

$$E_S := \{(i, j) \in E : i \in S_j \text{ or } j \in S_i\}.$$

Note that $i \in S_j$ does not necessarily imply the opposite containment $j \in S_i$, but holds in many cases. The actual cardinalities of the two sets are $|E_G| = 1327$ and $|E_S| = 3747$ and hence comprise less than 1 % of the total edge set. Indeed, computing the probability for an edge e that was selected uniformly at random from E to be a similarity edge is

$$\mathbb{P}(e \in E_S) = \frac{|E_S|}{|E|} \approx 0.007$$

Now what is the probability that a group edge is also a similarity edge? This question can be answered by computing the conditional probability

$$\mathbb{P}(e \in E_S | e \in E_G) = \frac{\mathbb{P}(e \in E_S \cap E_G)}{\mathbb{P}(e \in E_G)} = \frac{|E_S \cap E_G|}{|E_G|} = \frac{974}{1327} \approx 0.734.$$

The majority of group edges is contained in the similarity set, and the probability for a group edge to be a similarity edge is more than 100 times higher than for a randomly selected edge.

Recall from Section 3.4 that the model groups have been partially derived from the feature data. For submissions from the NEOS server, which have an unknown origin, clustering has been used to group similar NEOS instances into pseudogroups. If we omit all NEOS instances from the above computations (by considering the complete graph $K_{\mathcal{M} \setminus \mathcal{M}^{\text{NEOS}}}$ with 714 vertices), the probability for an edge to be a similarity edge is about the same, $\mathbb{P}(e \in E_S) \approx 0.008$, whereas the conditional probability of a group edge to be a similarity edge is ≈ 0.815 and hence even higher than for $K_{\mathcal{M}}$.

From this observation, we conclude that the feature space has been designed sufficiently well for the clustering approach. In fact, the used feature space recovers the model group data better than we expected. Even for an instance that does not belong to a dedicated model group or lacks bibliographical information, the similarity to other model groups can yield interesting hints at the type and application of this instance. Therefore, the web page of MIPLIB 2017 (see also Section 3.7.3) allows to browse the five most similar instances for every instance of the MIPLIB 2017 collection.

Figure 3.4 uses t-SNE [van der Maaten and Hinton, 2008] to give a spatial impression of the locations of the MIPLIB 2017 benchmark set, the benchmark-suitable instances, and the collection, relative to each other in feature space. The distance computation is based on the feature vectors after they have been scaled over the entire set of submissions. Note that there is a subset relation for those sets, i.e.,

$$\text{Benchmark Set} \subsetneq \text{Benchmark-suitable} \subsetneq \text{Collection},$$

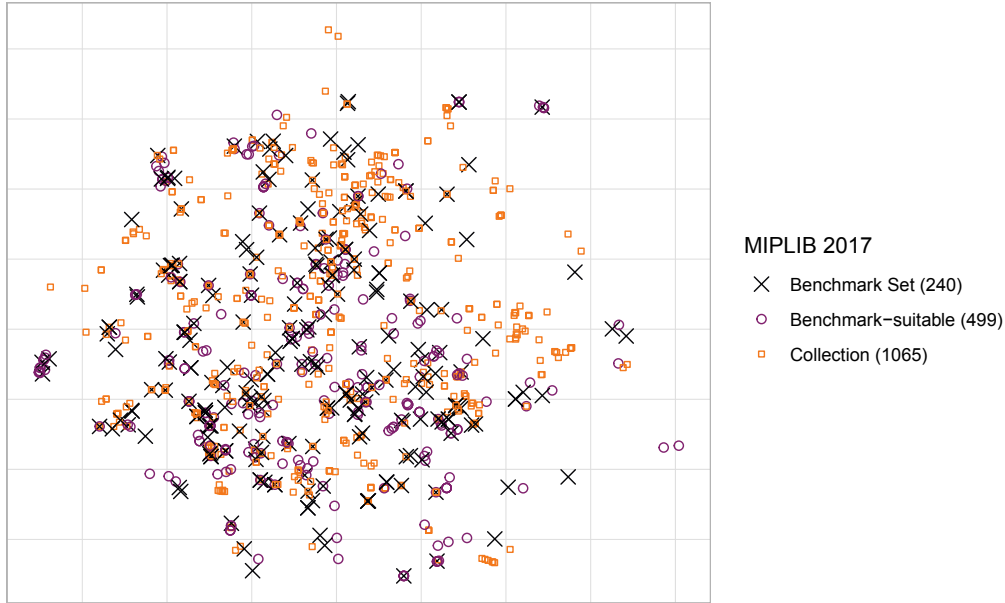


Figure 3.4: Two-dimensional embedding of the distances in the feature space using t-SNE.

such that the plot only shows the innermost set membership for every instance. Figure 3.4 shows that benchmark-suitable instances cover the majority of the feature space that is comprised by the collection except for a few regions.

3.7.2 Solver Performance

One of the goals of MIPLIB has always been to provide a representative set to measure and compare solver performance. In this section, we analyze the solver performance on the collection, and in particular on the new benchmark set of 240 instances. For this analysis, we use the computational results obtained during 4-hour runs conducted for the selection process. Note, however, that in this chapter, solver performance is not reported directly for several reasons. One reason is that due to hardware restrictions, not all runs could be performed exclusively on the same hardware, and should hence not be reported in a way that could be confused for an actual benchmark. Again, the individual results are aggregated into the virtual best solver, i.e., a solver that always finishes as fast as the fastest actual solver for each individual problem instance.

In Figure 3.5, we compare the performance of this virtual best solver on the benchmark sets of MIPLIB 2010 and 2017. One of the motivations for the creation of MIPLIB 2017 was the demand for a harder benchmark set. As a consequence of Definition 3.2, the virtual best solver solves all instances within 4 hours (or 14400 seconds) as this is one of the criteria for benchmark suitability. The plot shows that the majority of the old benchmark set can be solved by the virtual best solver in less than 100 seconds, and that there is no instance left where the virtual best solver requires one hour or more. By contrast, the benchmark set of MIPLIB 2017 is much more demanding, as a significant

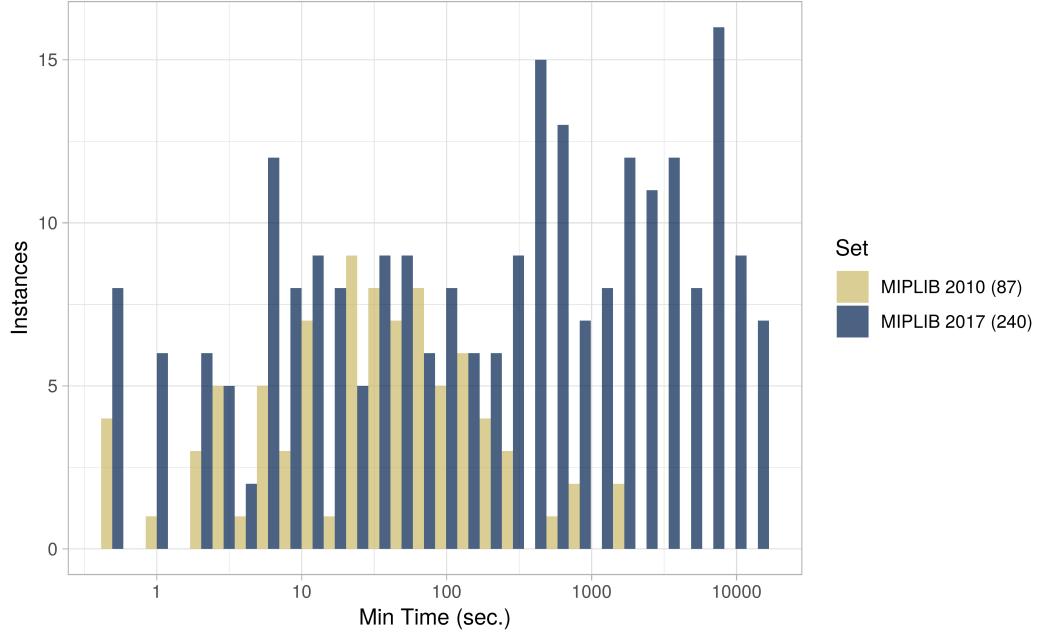


Figure 3.5: Minimum solving time in seconds of the benchmark sets of MIPLIB 2010 and MIPLIB 2017.

	virtual median		virtual best	
	2010	2017	2010	2017
> 1 h	17.2 %	66.7 %	0 %	19.6 %
> 2 h	12.6 %	61.7 %	0 %	8.8 %
> 3 h	6.9 %	51.2 %	0 %	4.2 %
> 4 h	5.7 %	48.3 %	0 %	0.0 %

Table 3.9: Percentage of benchmark instances that could not be solved within 1–4 hours by the virtual best and median solvers. The percentages are relative to the individual benchmark sets (2010: 87 instances, 2017: 240 instances).

portion of instances lies at the right end of the scale. Due to its increased size, the bars for the MIPLIB 2017 benchmark set lie almost consistently above the ones for the previous set. The MIPLIB 2017 benchmark set covers much more of the relevant performance scale than its predecessor covers nowadays. Note that the MIPLIB 2010 benchmark set appearing easy is an impressive result of 7 years of continuous solver improvements.

Table 3.9 shows the percentage of instances of the respective benchmark set (2010 or 2017) for which the virtual best solver takes longer than a certain time threshold, which we vary between 1 and 4 hours. As mentioned before, all instances of the 2010 benchmark set could be solved within one hour by the virtual best solver. The table shows that among the new benchmark set, almost 20 % of the instances cannot be solved within one hour by the virtual best solver. Along with the virtual best solver, Table 3.9 also shows the performance of the virtual median solver, based on the median

solution time over the eight involved solvers. Since the number of involved solvers is even, the median is computed by averaging the timing result of the two solvers ranking 4th and 5th for each instance individually. Therefore, the virtual median solver is faster than half of the solvers. On the MIPLIB 2010 benchmark set, 17.2% of the instances are not solved within one hour by the virtual median solver. In contrast to the virtual best solver, the virtual median solver still times out after 4 hours on 5.7% (5 of 87) instances even on the 2010 benchmark set. On the MIPLIB 2017 benchmark set, the virtual median solver needs more than one hour on two thirds of the instances, and still needs more than four hours of solving time on almost 50% of the instances.

In Figure 3.6, the fraction of instances solved by the virtual median solver as a function of time is shown. The figure shows the corresponding curves for the MIPLIB 2017 collection of 1,065 instances, the set $\mathcal{M}^{\text{col}} \cap \mathcal{M}^{\text{bm-suit}}$ of 499 benchmark-suitable instances, and the MIPLIB 2017 benchmark set (240 instances). Recall that an instance is only a candidate for the benchmark set if it takes the virtual median solver at least 10 seconds to solve it, which is also visible from the plot. As minimum for any time measurement, 0.5 seconds were used, which is visible in the curve of the MIPLIB 2017 collection. The objective function for the benchmark set was designed to prefer harder instances. The effect of this design choice is visible in the figure, in which the percentage of solved instances of the benchmark set consistently lies below the curve for the 499 benchmark-suitable instances. The slope of the curve for the collection is approximately linear for about 90% of the visible area. Due to the logarithmic scaling of the horizontal (time) axis, this suggests that to a certain extent, the solving behavior of the virtual median solver can be approximated by fitting a logarithmic function. Note that the clear change in behavior of the curves, which are otherwise almost logarithmic, towards the right end of the scale is an artifact from the median computation, which weighs in as soon as the 4th solver could still solve the instance, but the 5th solver couldn't. Their timings can even be very different. On the instance `blp-ic98`, the four best performing solvers finish within 1,700 seconds, while the fifth solver times out after four hours. The median solver therefore has a performance of 8,050 seconds. All individual curves of the actual solvers tested have a similar, almost logarithmic shape without the median artifact.

Statistically speaking, the curves in Figure 3.6 describe empirical cumulative density functions (CDF) of the random variable that represents median solving time for an instance. The Kolmogorov-Smirnov (KS) test is a statistical approach to measuring the similarity between a pair of CDF F_1, F_2 . To this end, the KS test measures the maximum vertical distance D between F_1 and F_2 .¹⁷ With increasing D , the likelihood decreases that F_1 and F_2 represent samples from the same distribution. In order to further quantify the hardness of the benchmark set, a KS test has been performed for every solver, comparing its individual CDF pair on the benchmark set and the set of

¹⁷This distance D is the supremum norm $\sup_{x \in \mathbb{R}} |F_1(x) - F_2(x)|$.

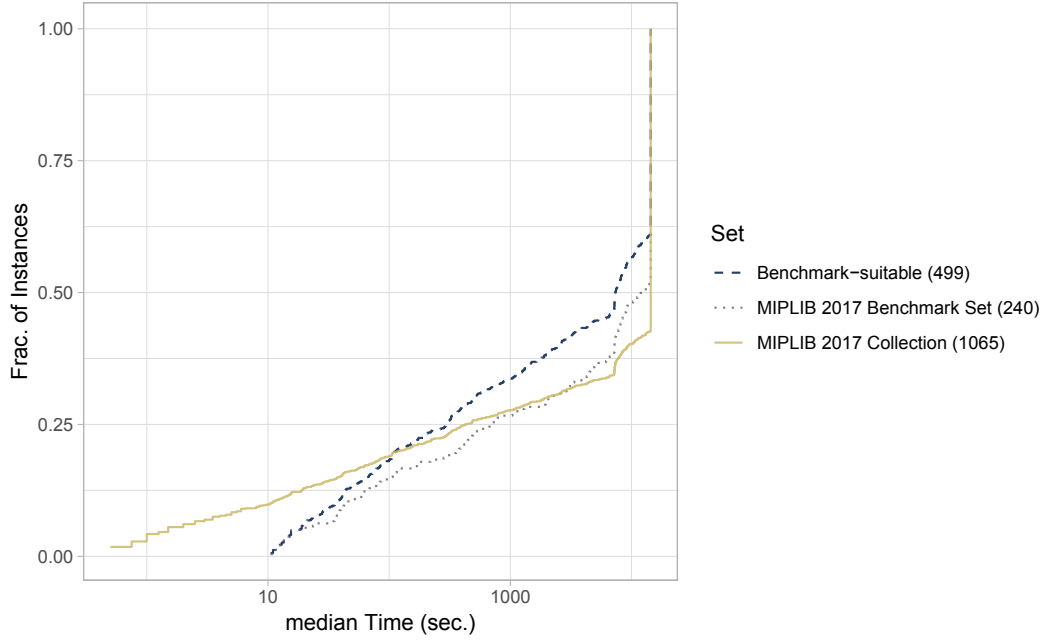


Figure 3.6: Virtual median solver performance on subsets of the MIPLIB 2017 collection.

benchmark-suitable instances. As alternative serves the hypothesis that the CDF of the benchmark set lies below the CDF of the larger set.

For the performance of the virtual median solver as depicted in Figure 3.6, the distance D is approximately 0.10, which results in a p -value of 0.04. A much smaller p -value of $6.513 \cdot 10^{-5}$ is obtained for the virtual best solver at a distance of $D = 0.17$. For four of the actual solvers, the maximum D is larger than 0.1, and the accompanying p -value is smaller than 1%. This is significant evidence that for those solvers, the benchmark set has been selected as a particularly hard selection among all suitable instances. For the other four solvers, the value of D is smaller, which in turn results in larger p -values (each greater than 0.1). Note that even here, the curves of the CDF on the benchmark set tends to undercut the CDF of the set of suitable instances, but this discrepancy is not large enough to render the KS test significant. Hence, for those four solvers, the performance curve is more or less representative of the CDF over all suitable instances. The results show that the selection methodology has achieved both its conflicting goals, hardness and representability, with respect to the solver performance, equally well.

3.7.3 The MIPLIB 2017 Web Page

For the release of MIPLIB 2017, the web page `miplib.zib.de` has been written from scratch and received a modern design. The main page shows the current status of instances regarding the categories easy, hard, and open. The two main tables list the instances of the MIPLIB 2017 collection and benchmark set together with some key properties, their model group, and their optimal or best known objective value. All

tables use tags on the instances to highlight certain properties that may be interesting for researchers, such as pure feasibility instances with no objective function, instances for which good decompositions are known, instances with critical numerics, the presence of indicator constraints, etc. It is possible to search and filter for instances by name, status, model group, or tag, or to sort the table by column.

The individual instance pages offer a short description of this instance and bibliographical information. Also, more information on the constraint mix for this instance before and after presolving is displayed, as well as decomposition information, if available. Finally, the optimal or best known solutions for every instance are displayed, as well as the five most similar instances as explained in Section 3.7.1. The web page also offers additional downloadable content, including

- the MIPLIB 2017 collection and benchmark sets,
- lists of instances with certain tags,
- available solutions,
- optimal/best known objective values for all instances,
- the collection and benchmark MIPs,
- the feature extractor, and
- bash scripts to run and validate solver performance experiments.

Some tags may change over time, such as the instances that fall into the categories easy, medium, and hard. Also, best known objective values are naturally changing or eventually proven to be optimal. Therefore, we provide versioned files for accurate referencing. The versioned files are periodically updated. All downloadable solutions are checked for feasibility with the solution checker from Section 3.5.6. Also, their exact solution values after fixing all integer variables are computed using SoPlex with iterative refinement. The actual collection and benchmark MIPs are also available for download. Obviously, these instances cannot be part of the actual collection and benchmark sets, respectively, since their presence would alter the feature space and hence their own formulation. Contributions in terms of updated bibliographic information or corrections to the instance descriptions are very welcome. In particular, we are constantly accepting and checking improving solutions to the open instances of the MIPLIB 2017 collection. In contrast to previous MIPLIBs, not only new optimal, but any improving solution will be considered for the periodic update of the page data. Improving solutions should be sent to the maintainers of the page, together with a description of how they have been obtained. Note that every submitted solution must adhere to the format accepted by the MIPLIB solution checker (Section 3.5.6), which is also available on the web page.

3.8 Summary

The sixth version of MIPLIB has substantially increased in size compared to its predecessors. The distinction between a dedicated benchmark set and the entire collection, which was introduced with MIPLIB 2010, has been preserved. These sets now contain 240 and 1,065 instances, respectively. The process of how to reduce the initial submission pool of over 5000 instances to a balanced selection of this size, however, has been completely redesigned. Beyond the new MIPLIB 2017 itself, the development of this fully data-driven and optimization-based methodology is the main contribution of this chapter.

We propose two related MIP models that have successfully provided decision support for the selection process to the MIPLIB committee. One key ingredient of this approach is the definition of a feature space covering more than a hundred different dimensions for characterizing a MIP instance. In order to ensure a balanced selection with respect to these features and, for the benchmark set, with respect to performance data, we advocate the use of multiple instance clusterings over partitions of the feature vector. A comparison with manually assigned model groups available from meta data of the submissions shows the high descriptive power of the used feature space. By approaching the selection problem as a MIP that encodes a balanced, simultaneous selection from each such clustering, the collection and benchmark MIPs provide the flexibility to incorporate both feature coverage and performance requirements as well as other restrictions from the committee. Besides improving the final outcome, this formalization of the selection criteria has served to increase the transparency of the selection process.

While the selection methodology proposed here is not intended as a general blueprint for construction of test sets, we hope that parts of the process of constructing the MIPLIB instance sets may apply to the curation of other test sets in the future. Certainly, many variations and adjustments of the approach are possible. Not only the chosen constants or heuristic clustering methods can be adapted, but also the role of objective function and constraints may be redefined. Furthermore, the interplay between the main selection models and the diversity preselection offers potential for variation. For example, a different approach may directly select and fix a number of instances with maximum diversity from each large model group for the collection and afterwards complete the collection with instances from smaller groups and instances with no known model group association. In light of these degrees of freedom and many ad hoc decisions that had to be made in advance, the final result is clearly only one of many possible and justified outcomes. However, we believe that the collection and benchmark sets presented in this chapter are a profound attempt to provide the research community with test sets that represent the versatility of MIP.

One of the main characteristics of the benchmark set is that it provides a snapshot of current solver performance. Our hope is that the performance of solvers on this

benchmark set will serve as a sort of bellwether for progress in the field of mixed-integer optimization as a whole in the coming years. As future work, we propose to assess the performance representability of the benchmark set from hindsight, i.e., by comparing speed-ups for entire model groups as well as for the selected instances. Such data will finally allow better comparisons of different (pre-)selection models such as, e.g., the one presented here that favors diversity to a different one that selects nearest neighbors and maximizes representability.

Another benefit of our MIP-based selection process is the fact that the MIP models can be used to approach questions beyond the initial creation of the test set. One example is the following case, in which it occurred that benchmark instances needed to be replaced as new computational data became available. Despite all the care that was taken to exclude numerically critical instances from the benchmark set, problematic numerical properties of the two instances **neos-5075914-elvire** and **neos-3754224-navua** remained undetected during the selection process.¹⁸ A variant of the benchmark selection MIP was used to compute a minimal update of the benchmark set that exchanges the discussed instances while preserving the balance requirements. An accordingly updated version of the benchmark set was published in June 2019.

Finally, by the time of this writing, the challenges of MIPLIB 2017 collection have already attracted a wide audience. In fact, we have received many new solutions to previously open instances. While some of those optimality or infeasibility proofs have been obtained by the use of massively parallel codes such as the Ubiquity Generator framework [Shinano, 2018; Shinano et al., 2016], other instances inspired the development of customized cutting plane approaches, or even a rigid mathematical proof of infeasibility without any code in the case of the instance **fhnw-sq3**. In total, 70 originally open instances have already been solved.¹⁹ We are looking forward to further contributions and many more years (and versions) of MIPLIB to come.

¹⁸Both instances are at the border between feasibility and infeasibility, but at the time of collecting solution data no inconsistencies could be observed. For the first instance, two solvers agree on the optimal solution value although the instance should mathematically be infeasible. The second instance has only been declared infeasible by one solver during the selection process; we received a solution that is feasible within tolerances half a year after the original publication of the benchmark set.

¹⁹Compare the downloadable files **open-v1.test** and **open-v14.test**

4

Enhancing MIP Branching Decisions Using the Variance of Pseudo-costs

The selection of a good branching variable is crucial for small search trees in mixed-integer programming. Most modern solvers employ a strategy guided by history information, mainly the pseudo-costs of integer variables, which are used to estimate the objective gain. At the beginning of the search. Such information is usually collected via an expensive look-ahead strategy called strong branching until variables are considered reliable.

The current state-of-the-art branching rule for balancing between strong branching and estimation, *reliability branching* [Achterberg et al., 2004] uses a fixed number of branching decisions after which the variable information is considered *reliable*. This approach has the disadvantage that it uses the same fixed reliability threshold for all variables. In practice, however, it appears natural that variables that are structurally different inside a MIP model also have different reliability requirements. Another disadvantage of a fixed parameter is that it might not scale well with increasing problem size.

The aim of the present chapter is to introduce different notions of reliability by exploiting more statistical information during the branching process. Using the sample variance of past observations, we formulate two criteria for switching between strong branching and estimation that take into account each variable history individually. We perform computational experiments on standard MIP test sets to evaluate the impact of our approach.

The work for this chapter was developed independently of the work by Kadioglu, O’Mahony, Refalo, and Sellmann [2011], who study the use of variance of past

branching observations to derive risk-aware and risk-averse strategies for impact-based search [Refalo, 2004] in the area of constraint programming (CP). The approach presented here uses variations of past branching information for the decision if strong branching should be continued on a variable or not. It extends the idea of reliability branching by taking into account each variable individually. In the present chapter, we further concentrate only on pseudo-costs and do not consider other branching history information such as inference or conflict scores, see Section 2.5.4. We do not collect any information prior to the actual search as in [Fischetti and Monaci, 2011; Kılınç Karzan et al., 2009].

This chapter is an extended version of the proceedings paper *Enhancing MIP Branching Decisions by Using the Sample Variance of Pseudo Costs* [Hendel, 2015] in the notation of this thesis. It extends the original paper by additional illustrations and more details about the Wellford formula for computing variance.

The chapter is organized as follows: First, Section 4.1 introduces the necessary notation and presents the reliability branching rule by Achterberg et al. [2004] in more detail. Afterwards, we introduce new notions of reliability in Section 4.2, and present computational results with SCIP in Section 4.3. We finish with concluding remarks in Section 4.4. Appendix B contains the detailed outcomes for each instance of our computational experiments.

4.1 Reliability Branching and Fixed-Number Thresholds

Throughout this chapter, we will give definitions and explanations only for the down-branch. The according formula and argumentation for the other direction can be derived analogously. We give a definition of reliability branching that is more general than the original definition by Achterberg et al. [Achterberg et al., 2004] in that it only assumes a subdivision of the fractional variables into reliable and unreliable candidates as input. This generalization allows for plugging in our novel notions of reliability later.

Definition 4.1 (Reliability branching). *Let P be a MIP with non-empty set of fractional variables \mathcal{F} . Given a subdivision $\mathcal{F} = \mathcal{F}^{rel} \cup \mathcal{F}^{url}$ of the fractional variables into reliable and unreliable branching candidates, we define the reliability branching score function of $j \in \mathcal{F}$ as*

$$\vartheta^{rel}(j) := \begin{cases} \vartheta^{str}(j), & \text{if } j \in \mathcal{F}^{url}, \\ \vartheta^{ps}(j), & \text{if } j \in \mathcal{F}^{rel}. \end{cases} \quad (4.1)$$

Reliability branching performs strong branching on the set of unreliable candidates \mathcal{F}^{url} to determine their exact gains (2.12), but uses pseudo-cost estimates for all other

branching candidates. It is characterized by its *notion of (un-)reliability*, i.e. a rule how to split the branching candidates into a reliable and an unreliable set.

We refer to this classical notion of reliability by Achterberg et al. [2004], as *fixed-number threshold reliability*:

Definition 4.2 (Fixed-number threshold reliability). *Given a reliability parameter $\eta > 0$, fixed-number threshold (fnt)-reliability splits the fractionals according to*

$$\mathcal{F}_{fnt}^{ur}(\eta) := \{j \in \mathcal{F} : \min\{\eta_j^-, \eta_j^+\} < \eta\}.$$

We call a variable $j \in \mathcal{F} \setminus \mathcal{F}_{fnt}^{ur}(\eta)$ (fnt)-reliable.

Using the term “fixed-number”, we emphasize that (fnt)-reliability of a variable solely depends on the number of previous branching observations. Achterberg et al. [2004] suggested to use 8 as threshold. By the time of this writing, SCIP uses 5, based on experimentations to yield a good average performance on a variety of MIP instances.

As briefly explained by Achterberg [2007a], in practice in SCIP, the threshold number is dynamically adjusted at every node depending on the proportion of simplex iterations spent on solving strong branching and the total number of iterations spent on solving regular LP relaxations. If strong branching requires too many resources, the reliability parameter is decreased. Assume we are in iteration k of the B&B Algorithm 2. We denote by κ_k^{bb} the number of simplex iterations spent on solving node LP relaxations (line 6 of Algorithm 2), and by κ_k^{sb} the number of simplex iterations spent on computing strong branching scores (2.13) until (and including) iteration k . The reliability threshold is computed before branching is executed and therefore based on κ_{k-1}^{sb} . By default, the maximum number of allowed strong branching simplex iterations is set to

$$\kappa_k^{\text{sb-max}} := \frac{1}{2}\kappa_k^{\text{bb}} + 100000. \quad (4.2)$$

All three simplex iteration counts are used for determining the value of

$$\alpha_k := \max \left\{ \min \left\{ 1, \frac{\kappa_k^{\text{sb-max}} - \kappa_{k-1}^{\text{sb}}}{\kappa_{k-1}^{\text{sb}} + 1} \right\}, \frac{\frac{1}{2}\kappa_k^{\text{bb}} - \kappa_{k-1}^{\text{sb}}}{\kappa_{k-1}^{\text{sb}} + 1} \right\}, \quad (4.3)$$

which intuitively represents the relative distance between the consumed and the available strong branching iterations. After staring at the computation of α_k for a moment, we realize that $\alpha_k < 0$ if and only if $\kappa_{k-1}^{\text{sb}} > \kappa_k^{\text{sb-max}}$, i.e., if strong branching has consumed all of its current iteration budget. It is possible that $\alpha_k > 1$ if strong branching spent only little of its available budget so far, concretely less than 25 % of the node LP iterations κ_k^{bb} . Let $\eta^{\min} \leq \eta^{\max}$ denote a lower and upper reliability threshold. By default, $\eta^{\min} = 1$ and $\eta^{\max} = 5$ in SCIP version 3.2, which we used in this chapter, and also in the most recent release.

Then, the reliability η_k at iteration k is computed before the branching decision must be made as

$$\eta_k := \begin{cases} 0, & \text{if } \kappa_{k-1}^{\text{sb}} > \kappa_k^{\text{sb-max}} \Leftrightarrow \alpha_k < 0 \\ \eta^{\min} + \alpha_k (\eta^{\max} - \eta^{\min}), & \text{otherwise.} \end{cases} \quad (4.4)$$

such that (fnt)-reliability constructs the set $\mathcal{F}_{\text{fnt}}^{\text{url}}(\eta_k)$ in iteration k of the B&B algorithm. Despite this dynamic adjustment of η_k to keep the strong branching effort reasonable, a single threshold is used for all variables.

In the next sections, we introduce novel notions of reliability that take into account the individual variance of the past branching observations.

4.2 Relative-Error and Hypothesis Reliability

The drawback of (fnt)-reliability is that a fixed threshold is used to measure the reliability of all variables of the problem equally well. Intuitively, it seems desirable to have a more individual look at the available pseudo-cost information of every variable and to continue strong branching on those candidates whose pseudo-costs fail to converge. In the following, we extend the statistical model for pseudo-costs by including the sample variance, which allows for the construction of confidence intervals and testing of hypotheses. For textbooks that cover the statistical topics in more detail, see, e.g. [Roussas, 2014].

Definition 4.3. *Let X_1, \dots, X_n be independent, identically distributed samples. The corrected sample variance about the sample mean \bar{X} is given by*

$$s_n^2 = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})^2. \quad (4.5)$$

The square root of the corrected sample variance, $s := \sqrt{s_n^2}$, is called standard deviation.

The well-known formula (4.5) of the corrected sample variance is not very convenient for incremental updates when new samples are observed, which is necessary for computing the sample variance of pseudo-costs during the B&B search.

Lemma 4.4. *The corrected sample variance can be equivalently written as*

$$s_n^2 = \frac{1}{n-1} \sum_{i=1}^n X_i^2 - \frac{1}{n(n-1)} \left(\sum_{i=1}^n X_i \right)^2. \quad (4.6)$$

Moreover, let $\bar{X}_k := \frac{1}{k}(X_1 + \dots + X_k)$ the partial mean up to $k = 1, \dots, n$ ($\bar{X}_0 = 0$), and let

$$M_0 := 0 \quad \text{and} \quad M_n := M_{n-1} + (X_n - \bar{X}_{n-1})(X_n - \bar{X}_n).$$

Then, for all $n \in \mathbb{N}$

$$(n-1)s_n^2 = M_n. \quad (4.7)$$

Proof. We infer Equation (4.6) from Definition 4.3 of the corrected sample variance:

$$\begin{aligned} (n-1) \cdot s_n^2 &= \sum_{i=1}^n (X_i - \bar{X})^2 \\ &= \sum_{i=1}^n (X_i^2 - 2X_i\bar{X} + \bar{X}^2) \\ &= \sum_{i=1}^n X_i^2 - 2\bar{X} \sum_{i=1}^n X_i + n\bar{X}^2 \\ &= \sum_{i=1}^n X_i^2 - n\bar{X}^2 \\ &= \sum_{i=1}^n X_i^2 - \frac{1}{n} \left(\sum_{i=1}^n X_i \right)^2. \end{aligned}$$

The second formula (4.7) can be shown using Equation (4.6) via induction on n .

The corrected sample variance is an unbiased estimate of the variance of the underlying distribution of the X_i . Note that in principle, the right term of Equation (4.6) allows for constant-time updates of s_n^2 every time a new sample is observed. However, subtraction of two large terms like in Equation 4.6 performed in floating point arithmetic may suffer from cancellation. We therefore prefer the second formula from Lemma 4.4, which allows for incremental updates of the variance without the cancellation. Such incremental updates according to Formula (4.7) are also known as *Wellford's algorithm*.

With increasing n , we can expect \bar{X} to approach the mean of the distribution from the law of large numbers. Under the assumption that the samples X_1, \dots, X_n are drawn from a normal distribution with unknown mean μ and variance σ^2 , the random variable

$$T := \frac{\bar{X} - \mu}{s/\sqrt{n}}$$

computed from the empirical mean and standard deviation of X_1, \dots, X_n is distributed along a Student's t -distribution with $n-1$ degrees of freedom. This relation can be used to construct a *confidence interval* I , which contains the true value of μ with a probability of $1 - \alpha$ for any *error rate* $0 < \alpha < 1$:

$$I = \left[\bar{X} - t_{\alpha, n-1} \frac{s}{\sqrt{n}}, \bar{X} + t_{\alpha, n-1} \frac{s}{\sqrt{n}} \right],$$

denoting by $t_{\alpha, n-1} > 0$ the α -percentile of the distribution of T . The distance of the endpoints of I relative to its center $\bar{X} \neq 0$,

$$\epsilon^{\text{rel}} = t_{\alpha, n-1} \cdot \frac{s}{\sqrt{n}|\bar{X}|}, \quad (4.8)$$

is called the *relative error* of the estimation.

4.2.1 Relative-Error Reliability

Applied to pseudo-costs, we determine the relative error for the pseudo-costs associated with each variable. Therefore, we treat an observed unit gain $\varsigma^-(P)$ at a node P resulting from a down-branch on a variable $j \in \mathcal{I}$ as independent sample from a normally distributed random variable with unknown *mean* μ_j^- and *variance* $(\sigma_j^-)^2$. It should be noted here that a normal distribution model for unit gains is of limited accuracy because unit gains are always nonnegative. The collected average unit gains Ψ_j^- (cf. Equation (2.15)) represent an estimate for μ_j^- . By using the *corrected sample variance*, we obtain an estimate for the variance, as well.

Whenever a new unit gain for variable $j \in \mathcal{F}$ in the down-branching direction at a node P was observed, we increase the counter η_j^- by 1 and update the average unit gain Ψ_j^- . In addition, we keep track of the sum of squared unit gains. This enables us to calculate the sample variance $(s_j^-)^2$ whenever $\eta_j^- \geq 2$. Whenever a branching decision must be made, we calculate the relative error ϵ_j^- associated with the current average unit gain Ψ_j^- as

$$\epsilon_j^- := \begin{cases} t_{\alpha, \eta_j^- - 1} \cdot \frac{s_j^-}{\sqrt{\eta_j^- \Psi_j^-}}, & \text{if } \Psi_j^- > 0 \\ 0, & \text{else.} \end{cases} \quad (4.9)$$

Recall that an average unit gain Ψ_j^- is necessarily nonnegative. Hence, we can omit the absolute in the denominator of (4.8). Furthermore, if Ψ_j^- is equal to zero, this also holds for the sample variance $(s_j^-)^2$. We therefore set the relative error to zero in this case.

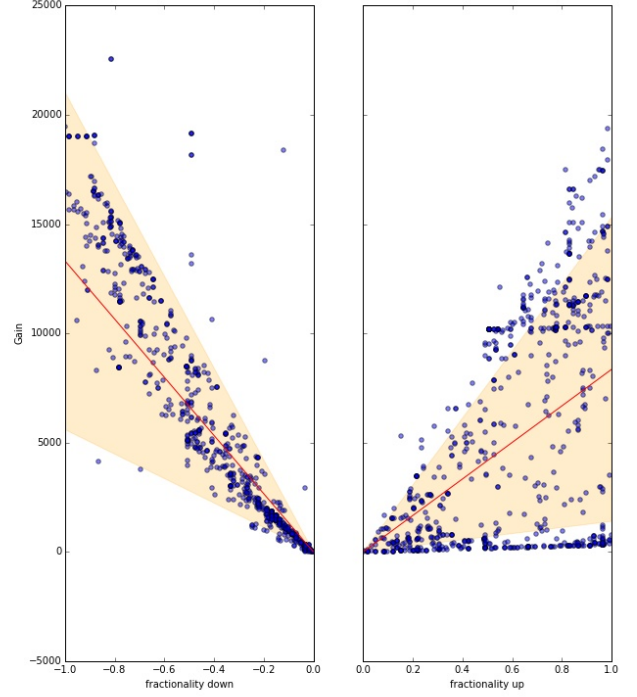
Definition 4.5 (Relative-error reliability). For $\eta^{rer} > 0$, relative-error (rer)-reliability splits the fractionals according to

$$\mathcal{F}_{rer}^{url}(\eta^{rer}) := \{j \in \mathcal{F} : \max\{\epsilon_j^+, \epsilon_j^-\} \geq \eta^{rer}\}. \quad (4.10)$$

We call a variable $j \in \mathcal{F} \setminus \mathcal{F}_{rer}^{url}(\eta^{rer})$ (rer)-reliable.

The rationale of (rer)-reliability is to continue strong branching on the subset of variables with highly varying objective gains, whereas variables with constant gains are early considered (rer)-reliable. In order to obtain relative errors for the branching directions, we need at least two observations in each direction. Note that a variable, which has already been (rer)-reliable, can become (rer)-unreliable again when the relative error rises again above the threshold after new information becomes available. It should be noted that in general there is no containment relation between the variable subsets considered by reliability branching with fixed number thresholds and our approach, i.e. neither is a strict subset of the other.

Figure 4.1: We illustrate the notions of pseudo-costs and their variance on an actual variable. Each point corresponds to one (strong) branching observation, separated into up and down branches. For a better visual experience, the left plot uses negative down fractionality as horizontal coordinate. The plot shows a clear relationship between fractionality and gain. The red lines have slopes $-\Psi^-$ and Ψ^+ . A point on a line therefore corresponds to the current pseudo-cost estimation at a given down or up fractionality. Additionally, we highlight the region around the pseudo-costs that fall within one standard deviation s^-, s^+ . The border of this region is calculated as $f^+(\cdot\Psi^+ \pm s^+)$ for branching up, and analogously for branching down.



4.2.2 Hypothesis Reliability

The disadvantage of (rer)-reliability is that it is likely to spend much strong branching effort on variables with overall low objective gains, but high relative error. In order to overcome this, it is possible to restrict the variables that are selected for strong branching evaluation to only candidates with a probability to be actually better than the best candidate j^{ps} according to pseudo-cost branching. Roughly speaking, we want to ensure that there is little probability that $f_j^- \mu_j^- \geq f_{j^{\text{ps}}}^- \mu_{j^{\text{ps}}}^-$ for $j^{\text{ps}} \neq j \in \mathcal{F}$.

Therefore, we test against the hypothesis that a fractional $j \in \mathcal{F}$ has an objective gain at least as high as j^{ps} , i.e., $f_j^- \mu_j^- \geq f_{j^{\text{ps}}}^- \mu_{j^{\text{ps}}}^-$. For two variables $i, j \in \mathcal{F}$ with fractionalities f_i^- and f_j^- , we use the *pooled variance*

$$S_{i,j}^- := \frac{(\eta_i^- - 1)(f_i^-)^2 (s_i^-)^2 + (\eta_j^- - 1)(f_j^-)^2 (s_j^-)^2}{\eta_i^- + \eta_j^- - 2}$$

to calculate a *2-sample t-value* for i and j ,

$$T_{i,j}^- := \sqrt{\frac{\eta_i^- \eta_j^-}{\eta_i^- + \eta_j^-}} \frac{f_i^- \Psi_i^- - f_j^- \Psi_j^-}{\sqrt{S_{i,j}^-}}.$$

Under the hypothesis, $T_{j^{\text{ps}},j}^-$ follows a Student-t distribution with $\eta_{j^{\text{ps}}}^- + \eta_j^- - 2$ degrees of freedom. If, for a given threshold $0 < \alpha < 1$, $T_{j^{\text{ps}},j}^-$ exceeds $t_{\alpha, \eta_{j^{\text{ps}}}^- + \eta_j^- - 2}^-$, we

can reject the hypothesis with an error probability of at most $\alpha/2$. The division by two is justified because the hypothesis is one-sided. Conversely, if the hypothesis cannot be safely rejected, it is safer to perform strong branching on the two candidates.

The second novel notion of reliability in the present chapter rules out fractional variables with little probability to be better than the best pseudo-score candidate:

Definition 4.6 (Hypothesis reliability). *Let $j^{\text{ps}} \in \mathcal{F}$ be the best pseudo-cost fractional candidate for branching, and let $0 < \alpha < 1$ be a rejection probability. The unreliable fractional set for hypothesis reliability is*

$$\mathcal{F}_{\text{hyp}}^{\text{url}}(\alpha) := \left\{ j \in \mathcal{F} : T_{j^{\text{ps}},j}^- < t_{\alpha,j^{\text{ps}},j}^- \text{ and } T_{j^{\text{ps}},j}^+ < t_{\alpha,j^{\text{ps}},j}^+ \right\}. \quad (4.11)$$

Variables $j \in \mathcal{F} \setminus \mathcal{F}_{\text{hyp}}^{\text{url}}(\alpha)$ are called (hyp)-reliable.

For practical reasons, we also include variables j with $\min\{\eta_j^-, \eta_j^+\} \leq 1$. It should be noted that the best pseudo-cost candidate j^{ps} is never (hyp)-reliable because $T_{j^{\text{ps}},j^{\text{ps}}}^- = T_{j^{\text{ps}},j^{\text{ps}}}^+ = 0$. If no other candidate than j^{ps} is (hyp)-unreliable, this means that no other fractional variable has an estimated objective gain nearly as good as j^{ps} . In this case, we immediately branch on j^{ps} without strong branching. In the experiments in the following section, we tested an error probability of $\alpha = 0.05$, i.e. the error probability for ruling out a better candidate based on the current branching history is $\alpha/2 = 2.5\%$.

Note also that the variant of a t -test that we use for (hyp)-reliability is, in theory, only applicable when the two variables can be assumed to have equal variances. An alternative of the t -test that is more robust towards unequal variances is *Welch's t -test* [Welch, 1947]. There are also some statistical tests available for the hypothesis of equal variances, such as the *F-test of equality of variances* [see, e.g., Box, 1953].

4.3 Computational Results

We implemented the new reliability notions from Section 4.2 into the existing reliability branching rule of a development version of SCIP version 3.1.0.2, which we compiled with a gcc compiler version 4.8.2. As underlying LP-solver, we used SoPlex version 2.0. We used SCIP with default settings except for the following changes: For using a pure objective-based branching score function as in Section 4.1, tie-breakers such as, e.g., inference scores were deactivated by setting their corresponding weight to 0. Furthermore, we set the known optimal solution values – in case they exist – minus a small threshold 10^{-9} as objective cutoffs, so that only a proof for the optimality/infeasibility of a problem needed to be found. We also disabled all primal heuristics and activated depth-first search node selection as an attempt to minimize performance variability [Danna, 2008; Koch et al., 2011] that might be caused by other solving components than the tested branching rules. Finally, the child node selection was changed to use solely pseudo-costs, where SCIP with default settings uses a hybrid approach together with inference scores.

The test bed for our comparison of the different approaches consists of a subset of instances from the three publicly available libraries MIPLIB 3.0 [Bixby et al., 1998], MIPLIB 2003 [Achterberg et al., 2006], and MIPLIB 2010 [Koch et al., 2011], from which we omitted four instances for which an optimal objective value is not known by the time of this writing. Since we are mainly interested in reducing the search tree size, we further dropped all 29 instances that could be solved before or during the processing of the root node. Our final test bed thus contains 135 MIP instances.

The computations were performed on a cluster of 32 computers, each of which runs with a 64bit Intel Xeon X5672 CPUs at 3.20 GHz with 12 MB cache and 48 GB main memory. The operating system was Ubuntu 14.4. Hyperthreading and Turboboost were disabled. We ran only one job per computer in order to minimize the random noise in the measured running time that might be caused by cache-misses if multiple processes share common resources. Finally, all experiments were run with a time limit of 2h and a 40 GB memory limit.

The newly proposed notions of reliability from Section 4.2 are represented by four different settings: **(hyp)** renders candidates (hyp)-unreliable according to the rule (4.6), whereas **(rer)-0.01**, **(rer)-0.05**, and **(rer)-0.1** use (rer)-reliability regarding relative errors in pseudo-cost confidence intervals at three different threshold levels 1 %, 5 %, and 10 %. Note that these levels represent different levels of the relative error threshold η^{rer} , whereas the confidence level $1 - \alpha$ is kept fixed at 95 % for both (hyp) and (rer)-reliability. We compare them to (fnt)-reliability at a fixed threshold of 5, denoted by **(fnt)-5**, which constitutes the default of SCIP at the time of this writing. In contrast to the default settings of SCIP, we disabled the dynamic adjustment of this threshold during the search based on the simplex iterations consumed by strong branching, which is achieved by setting $\eta^{\min} = \eta^{\max} = 5$ for the computation (4.4) of η_k . Note that this setting does not affect the temporary disablement of strong branching whenever the strong branching iterations exceed the maximum allowed simplex iteration budget, which may also disable the novel notions of (rer)- and (hyp)-reliability.

In this section, we only present compressed results of our experiments. For an instance-wise outcome, please refer to Tables B.1 and B.2 in the Appendix. The first three tables show the aggregated results regarding the solving **time** and the number of explored search tree **nodes** for all instances and for only those which could be solved within the time limit by all settings. We consider node results incomparable between settings where the solution status differs and thus only show time results for all instances. We report shifted geometric means with a shift of 10 seconds and 100 nodes, respectively. The columns **time**_% and **nodes**_% show the percentage deviation from the result for the reference setting **(fnt)-5**; values below 100 represent an improvement in this respect.

In Table 4.1, we compare the results over all instances from the test bed. 98 instances could be solved by all settings within the time limit of 2h, for which the reference run was fastest regarding the solving time, but also required the most branch-and-bound

settings	98 instances solved by all				135 total	
	time	time%	nodes	nodes%	time	time%
(fnt)-5	81.9	100.0	3402.3	100.0	290.3	100.0
(rer)-0.01	87.0	106.2	2722.3	80.0	303.1	104.4
(rer)-0.05	84.7	103.4	2716.5	79.8	298.1	102.7
(rer)-0.1	85.6	104.5	2902.9	85.3	300.3	103.4
(hyp)	82.7	101.0	2774.0	81.5	289.4	99.7

Table 4.1: Computational results for all 135 instances.

nodes on average. For our novel notion of (rer)-reliability, an evaluation of the different thresholds comes with a surprise: there is no significant difference between the levels 1 % and 5 % error tolerance. The highest node reduction of 20.2 % was obtained with the setting (rer)-0.05, closely followed by (rer)-0.01, which could not reduce the overall solving nodes further in the shifted geometric mean. Regarding the running time, (rer)-0.05 was the fastest to finish the tests among the three (rer)-settings, yet we observe a slight slow down of 3.4 % for the group of instances solved by all settings, and 2.7 % over all 135 instances compared to the reference run. By using (hyp)-reliability, we obtained a node reduction of 18.5 % and an almost performance neutral result for the running time.

The discrepancy between a reduction of the tree size at the cost of more solving time per node is the result of a more aggressive use of strong branching by the novel notions of reliability. The notion of (hyp)-reliability hereby appears to be more effective than relative-error reliability for guiding strong branching effort because it focuses on resolving cases among the top pseudo-cost score branching candidates where the estimation alone may lead to inferior branching decisions.

Table 4.2 contains only instances for which at least one of the settings needed more than 1000 nodes before termination. With (hyp)-reliability, we could improve the performance of SCIP w.r.t. the reference run by 28.8 % nodes and also obtain a slight time reduction in total, whereas the time on instances in the group containing only solved instances is competitive with the reference run (fnt)-5. Among the (rer)-settings, (rer)-0.05 is fastest regarding the solving time, but is still 4.1 % slower on average than the reference setting. All three settings (rer)-0.05, (rer)-0.01, and (hyp) show a similar decrease in the number of nodes, but at different computational efforts spent per node.

For the sake of completeness, we also present the remaining instances, for which no solver took more than 1000 branch-and-bound nodes before termination, in Table 4.3. Out of the 47 instances in this group, there is only one, namely stp-3d, that could not be solved by any of the settings. All novel notions of reliability reduce the search tree size, although the effect is less striking than on the instances that required larger search

settings	52 instances solved by all				88 total	
	time	time%	nodes	nodes%	time	time%
(fnt)-5	212.6	100.0	49741.5	100.0	897.9	100.0
(rer)-0.01	232.0	109.1	35322.9	71.0	947.4	105.5
(rer)-0.05	221.3	104.1	35104.9	70.6	924.0	102.9
(rer)-0.1	224.9	105.8	38227.1	76.9	932.5	103.9
(hyp)	212.9	100.1	35427.6	71.2	885.9	98.7

Table 4.2: Large Trees: $U^{\text{bb}} > 1000$ with at least one setting.

trees. Note that the node reduction obtained with (rer)-0.01 and (rer)-0.05 is now considerably better than the reduction obtained with (hyp)-reliability.

For those 37 instances for which optimality could not be proven within the time limit by at least one of our settings, we computed integrals of the dual gap as a function of time. This measure, which was suggested in [Achterberg et al., 2012; Berthold, 2013], attempts to compare the convergence of the dual gap towards zero. Table 4.4 shows the shifted geometric mean dual integral for all settings using a shift of 1000. All novel notions of reliability decrease the dual integral of the reference run, where the decrease is best with (hyp) yielding a reduction of 14.6 %. A similar result is obtained with (rer)-0.01, which outperforms other thresholds for (rer)-reliability in this respect.

4.4 Summary and Future Work

We introduced two novel notions of reliability: the (rer)-reliability based on pseudo-cost confidence intervals, and (hyp)-reliability implementing a variant of Student’s t -test. Our experimental results in SCIP show that these methods are promising for effectively reducing the size of branch-and-bound-trees compared to the current state-of-the-art fixed-number threshold, especially for large trees. Note that node reductions and the resulting reductions of the required memory play an important role, e.g., in the context of solver-parallelization [Koch et al., 2012] to reduce load-coordination overhead. Together

settings	46 instances solved by all				47 total	
	time	time%	nodes	nodes%	time	time%
(fnt)-5	23.8	100.0	74.1	100.0	27.8	100.0
(rer)-0.01	24.5	103.0	61.7	83.3	28.6	102.8
(rer)-0.05	24.5	103.0	62.1	83.8	28.6	102.8
(rer)-0.1	24.6	103.4	68.8	92.9	28.7	103.2
(hyp)	24.3	102.4	67.5	91.1	28.5	102.3

Table 4.3: Small trees: All solvers needed $U^{\text{bb}} \leq 1000$ nodes.

settings	integral*	integral* _%
(fnt)-5	73867.8	100.0
(rer)-0.01	66392.1	89.9
(rer)-0.05	73454.5	99.4
(rer)-0.1	72959.7	98.8
(hyp)	63101.8	85.4

Table 4.4: Shifted geom. mean dual integral for 37 time limit instances.

with the presented smaller dual integrals for instances which hit the time limit, we consider the reliability notions useful for proving optimality faster on harder instances.

Our implementation only considers pseudo-cost information, but could be readily applied to different history information such as, e.g., the inference history of a variable, as well.

In the computational study presented, we collected very little history information before using the statistical methods. Combining them with traditional fixed-number threshold reliability might increase the power of the hypothesis and relative error thresholds significantly.

5

MIP Solving Phases

A typical situation for branch-and-bound is that an optimal solution is found long before the proof of optimality is given and that a first feasible solution is found long before the optimal solution. The idea of this chapter is to partition the solution process into three distinct solving phases, which we name after the specific goal which should be achieved during this phase: First, the solver tries to find a feasible solution during the *feasibility phase*. During the subsequent *improvement phase*, a sequence of solutions with improving objective is generated until the incumbent solution is eventually optimal. During the remaining *proof phase*, the search aims at proving optimality. Two questions must be answered to obtain an adaptive solver that dynamically tunes its behavior w.r.t. different solving phases: How should the solver detect the transition between the improvement and the proof phase, and how should it react on the phase transitions?

We mainly focus on the first question and present heuristic criteria for deciding whether a given incumbent solution is already optimal. Such criteria cannot be expected to be exact because the decision problem of proving whether a given solution is optimal is still \mathcal{NP} -complete in general (it is equivalent to deciding feasibility for a MIP amended by an objective cutoff constraint). Concerning the second question, we build upon the results by Hendel [2014] for choosing adequate parameter tunings for each phase. Our computational results evaluate the impact of altering the solver settings at the predicted phase transition points.

It has been suggested to use different node selection and branching rules as long as no feasible solution has been found during a MIP solve, see [Linderoth and Savelsbergh, 1999]. In the notation that we introduce in Section 5.2, this could be seen as a 2-phase approach, switching branching parameters at the first phase transition.

Clearly, every estimate for the final search branch-and-bound tree size (cf. Chapter 8 and the references therein) can be used to extrapolate the remaining solving time until termination, and vice versa. Hence, previous prediction methods for the end of the

solution process estimate the end of the third phase in the terminology of Section 5.2. Estimated solution times or search tree sizes can be used for ranking different algorithms for the same problem, e.g., by running several algorithms in parallel for a limited amount of time (a so-called *racing ramp-up* [Shinano et al., 2012]) and afterwards continuing the search with the algorithm that yielded the smallest tree size estimate. All methods are based on partial information of a search tree of the problem and designed to be early available.

The transition heuristics that we propose in Section 5.3 are different in that they attempt to recognize the point in time when the incumbent solution is optimal, which can happen long before the end of the solution process. Another difference is that transition heuristics are used to adapt the solver behavior directly during the search.

This chapter is an edited copy of the article *From feasibility to improvement to proof: three phases of solving mixed-integer programs* [Berthold et al., 2018] in the notation of this thesis. It is organized as follows: In Section 5.1, we present the influence of MIP solving components discussed in Section 2.3.3 on the primal and dual progress of the solution process separately. We formalize the solving phases described above in Section 5.2 and discuss computational aspects of the different phases. The main contribution of this chapter are heuristic criteria for deciding when the solver should stop searching for better solutions and concentrate on proving optimality. We present two such heuristic criteria that take into account global information of the search progress in Section 5.3. Finally, we present two computational studies to evaluate the accuracy of the heuristic criteria and their benefits when used inside an adaptive solver in Section 5.4.

5.1 The Impact of Solving Components on the Solution Process

The integration and execution of MIP solver components (cf. Section 2.3.3) inside of a complete solver such as SCIP influences the overall solver performance in various ways. As we will confirm in this section, some components mainly affect the primal bound, while others mainly contribute to the dual bound development. As a consequence, special settings for individual solving phases should put different emphases on each of the components.

In order to categorize the components' influence on the primal and dual convergence individually, Hendel [2014] conducted an experiment where the solving components of SCIP were deactivated one at a time or were replaced by a simple default mechanism (e.g., branching on random variables) on 168 MIP instances from MIPLIB 3.0 [Bixby et al., 1998], MIPLIB 2003 [Achterberg et al., 2006], and MIPLIB 2010 [Koch et al., 2011]. The result of this experiment is visualized in Figure 5.1, which shows the percentage degradation in the primal and dual bound development compared with default solver

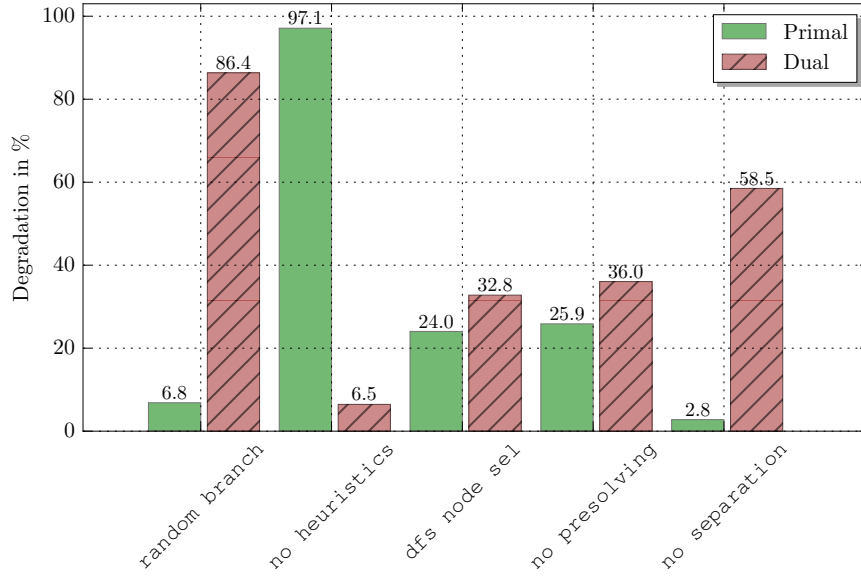


Figure 5.1: Percentage degradation compared to the SCIP performance with default settings regarding the shifted geometric mean primal and dual integral.

settings. Here, primal and dual integrals [Berthold, 2013, see also Section 2.9] are used to measure the component’s influence on the primal and dual convergence. The results for each instance are aggregated using a shifted geometric mean (2.23). Each presented degradation is the percentage of the relative increase in the respective shifted geometric mean compared to the default settings of SCIP.

Not surprisingly, primal heuristics mainly affect the primal bound. Its average degrades by a factor of almost two when primal heuristics are deactivated, while the dual integral becomes only 6.5% worse on average. Branching and cutting plane separation mainly contribute to the development of the dual bound. The impact of node selection and presolving is mixed, both bounds deteriorate substantially if presolving is deactivated or a simple depth-first search strategy is used for node selection. Notably, Figure 5.1 shows that all components have a positive impact on both the dual and the primal integral.

5.2 MIP Solving Phases

The main idea addressed in this chapter is a partition of the branch-and-bound solving process into a set of phases, which we formalize as follows

Definition 5.1. *Let P be a feasible MIP with optimal objective value $Z^{opt} \in \mathbb{Q}$, and let $\gamma^{primal}()$ and $\gamma^{dual}()$ be the primal and dual gap functions, respectively. Denoting the point in time when the first solution is found by t_1^* , we define the three solving phases*

\mathcal{P}_1 , \mathcal{P}_2 , and \mathcal{P}_3 as the following time intervals:

$$\begin{aligned}\mathcal{P}_1 &:= [0, t_1^*[, \\ \mathcal{P}_2 &:= \{t \geq t_1^* : \gamma^{\text{primal}}(Z(t)) > 0\}, \text{ and} \\ \mathcal{P}_3 &:= \{t \geq t_1^* : \gamma^{\text{primal}}(Z(t)) = 0, \gamma^{\text{dual}}(Z^*(t)) > 0\}.\end{aligned}$$

Clearly, the phases are disjoint. Furthermore, if T is the total time spent by a solver for solving P to optimality, the phases are a tripartition of the interval $[0, T]$. Each of the phases emphasizes a different goal of the solving process, so that it seems natural to pursue these goals with different parameter settings, which are tailored to achieve the phase objective as fast as possible.

The objective during the feasibility phase \mathcal{P}_1 consists of finding a first feasible solution; the quality of this solution only plays a minor role. Feasible solutions are either provided by a node's LP relaxation solution or by primal heuristics. The first feasible solution plays an important role for the solving process: First, it indicates the feasibility of the model to the user. Second, the bounding procedure of the branch-and-bound algorithm and some node presolving routines depend on a primal bound. Furthermore, several primal heuristics require a feasible solution as starting point to search for improvements.

After an initial feasible solution was found, the search for an optimal solution is conducted during the improvement phase \mathcal{P}_2 . During the improvement phase, a sequence of IP-feasible solutions with decreasing objective value is produced until the solver eventually finds an optimal solution. For users of MIP solving software, this is often the most important phase. In many practical applications, MIP models are not required to be solved to proven optimality, reaching a small optimality gap is considered sufficient [Berthold, 2013]. The reasons for this are threefold: there might be strict running time limitations, the models often are too large to complete the search and the input data itself might only be based on estimates.

The remaining time, which is called proof phase \mathcal{P}_3 , is spent on proving the optimality of the incumbent solution. Such a proof requires the full exploration of the remaining search tree until there are no more open nodes with dual bound lower than the optimal primal objective value.

Figure 5.2 illustrates a typical primal and dual gap function for a MIP solving process. We draw the negative of the dual gap function to make the convergence of the primal and dual bound more intuitive. The feasibility phase is finished at t_1^* with the first feasible solution. After three more solutions, an optimal solution is found at t_2^* , such that the remainder of the solution process is dedicated to the proof of optimality.

It is possible for both the improvement phase and the proof phase to be empty. If the first feasible solution is also an optimal one, then $\mathcal{P}_2 = \emptyset$; similarly, if the best possible dual bound is found before an incumbent with this optimal objective value is found, then $\mathcal{P}_3 = \emptyset$. In the special case that the MIP is a pure feasibility problem ($c = 0$), both

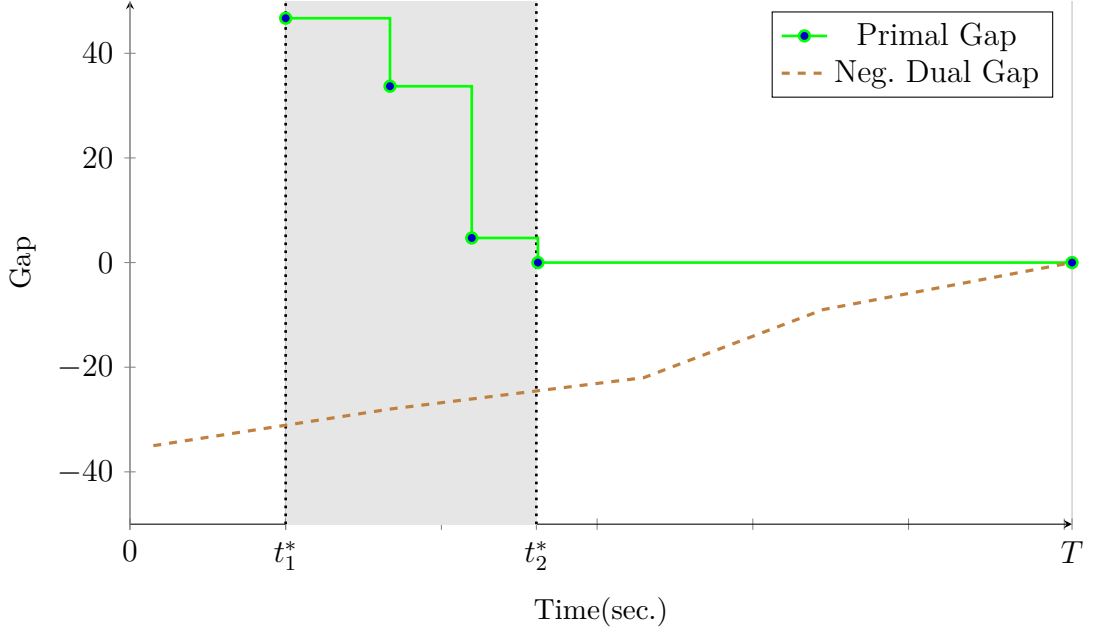


Figure 5.2: Solving phases of a MIP solution process with phase transitions t_1^* and t_2^*

the improvement phase and proof phase are empty. For the test set that we use for our computational experiments in Section 5.4, only one of 161 instances is a pure feasibility problem. Further, 11 instances have an empty improvement phase and 34 instances have an empty or almost empty proof phase, their union being 40 instances. Hence, it applies for a majority of 75 % of our test cases that the solving process is indeed partitioned into three nonempty phases by Definition 5.1.

We refer to the two points in time that mark the boundaries between the phases as *phase transitions*. More precisely, we call t_1^* the *first phase transition*, and we define the *second phase transition* as

$$t_2^* := \sup \mathcal{P}_1 \cup \mathcal{P}_2.$$

The recognition of the second phase transition t_2^* after the improvement phase requires knowledge about the optimality of the current incumbent prior to the termination of the solving process. If $\mathcal{P}_3 \neq \emptyset$, the decision problem of proving that there exists no solution better than \tilde{x}^{opt} for our input MIP P remains to be solved. This problem is co- \mathcal{NP} -complete in general (because it is equivalent to proving infeasibility of a MIP).

In the next section, we address the problem of how to heuristically estimate the second phase transition during the solving process.

5.3 Two Phase Transition Heuristics

If $\mathcal{P}_3 \neq \emptyset$, the detection of t_2^* requires knowledge of the optimal solution value. In this section, we present two heuristic criteria to estimate the second phase transition point

t_2^* without knowledge of the optimal solution value. These *phase transition heuristics* will be used to switch to settings for the proof phase when the criteria indicate that the current incumbent is optimal. The phase transition heuristics based on a property called *node estimation* [Bénichou et al., 1971] of all nodes in the node frontier. The node estimation constitutes an estimate of the objective value of the best attainable solution of a node.

More formally, let P be a (subproblem associated to a) node of the search tree. With a slight abuse of notation, we use P to denote both a MIP subproblem as well as the node of the search tree representing this subproblem.

Recall Definition 2.16 of pseudo-costs to estimate the branching impact of a fractional integer variable $j \in \mathcal{F}$ on the lower bound. Apart from their use in the selection of the best candidate for branching, pseudo-costs can also be applied to estimate the best solution attainable from a node P .

Definition 5.2 (Node estimation [Bénichou et al., 1971]). *The node estimation for a node P with optimal LP relaxation solution \check{x} is defined as*

$$\hat{Z}_P := c^t \check{x} + \sum_{j \in \mathcal{F}(P)} \min \{ \Psi_j(f_j^+), \Psi_j(-f_j^-) \},$$

where $c^t \check{x}$ is the node LP bound.

The rationale behind Definition 5.2 is to consider the smaller of the up and down pseudo-costs of each fractional variable $j \in \mathcal{F}(P)$ as necessary cost of making j integer.

In the following, we will be mainly interested in node estimations of open nodes, for which no LP relaxation has been solved so far. In order to determine an estimate of an *open node* Q , we simply subtract the contribution of the branching variable and direction that led to the creation of Q . Let Q be the child of another node P after branching upwards on $j \in \mathcal{F}(P)$. An initial estimate of Q can be calculated via

$$\hat{Z}_Q = \hat{Z}_P - \min \{ \Psi_j(f_j^+), \Psi_j(-f_j^-) \} + \Psi_j(f_j^+(P)),$$

thereby extending Definition 5.2 to open search nodes.

The node estimation does not account for a possible interplay between variables. This observation makes \hat{Z}_P likely to overestimate the actual integer objective value Z_P^{opt} of the best attainable solution from the subtree rooted at P . It is, on the other hand, also possible to underestimate Z_P^{opt} . Another important aspect concerns a possible degeneracy of the LP relaxation: Whenever there exist different optima to the node LP relaxation, they might lead to different estimates. Dependencies on a concrete LP solution is a common source for performance variability in MIP solvers, see, e.g., [Berthold et al., 2014].

5.3.1 The Best-Estimate Transition

The minimum node estimation among the set of open nodes $V^{\text{open}}(t)$ at time $t \geq 0$,

$$\hat{Z}^{\text{best}}(t) = \min\{\hat{Z}_Q : Q \in V^{\text{open}}(t)\}$$

is called the *best-estimate* [Bénichou et al., 1971]. In SCIP, the best-estimate is used as primary criterion for the default node selection and is one possible estimate of the optimal objective value of a given MIP, for other possible estimates, we refer to [Wojtaszek and Chinneck, 2010]. As our first phase transition heuristic, we propose to switch to the proof phase when the best-estimate exceeds the incumbent objective:

$$t_2^{\text{estim}} := \min \left\{ t \geq t_1^* : Z(t) \leq \hat{Z}^{\text{best}}(t) \right\}, \quad (5.1)$$

where $Z(t)$ is the primal bound at time t . Note that by requiring $t \geq t_1^*$, we make sure that there is indeed an incumbent solution.

5.3.2 The Rank-1 Transition

With an increasing number of explored branch-and-bound nodes, it intuitively becomes less and less likely to encounter a solution better than the current incumbent. Yet, every unprocessed node $Q \in V^{\text{open}}(t)$ has the potential to contain a better solution in the subtree underneath.

Definition 5.3. *Let S be the search tree after termination, and define d_Q as the depth and Z_Q^{opt} as the (integer) optimal objective value for every node $Q \in S$ (or ∞ if there is no feasible solution for Q). We define the rank of Q as*

$$\text{rg}(Q) := \left| \{Q' \in S : d_{Q'} = d_Q, Z_{Q'}^{\text{opt}} < Z_Q^{\text{opt}}\} \right| + 1.$$

The rank $\text{rg}(Q)$ represents the minimum position of node Q in any list that contains all nodes at depth d_Q in nondecreasing order of their optimal solution.¹ The root node P_0 trivially has a rank of 1, because it is the only node at depth 0. Indeed, if S were known in advance, the rank is defined in such a way that an optimal solution can be found by following a path of nodes of rank 1, starting at the root node. If the solving process has not found an optimal solution yet, there exists a node of rank 1 among the open nodes $V^{\text{open}}(t)$. Note, however, that there may even be nodes of rank 1 present in the node frontier although the current incumbent is already optimal.

The second phase transition heuristic is based on Definition 5.3. As for the best-estimate transition, we use the node estimation (cf. Definition 5.2) to circumvent the

¹Such groups among open subproblems based on properties such as the depth are also called *contours*, see [Morrison et al., 2016] and the references therein.

absence of true knowledge about best solutions in the unexplored subtrees. We impose a partial order relation \prec on the nodes of the search tree S :

$$Q' \prec Q \iff Q' \text{ was processed before } Q \text{ with } d_{Q'} = d_Q, \quad Q' \neq Q \in S.$$

With this partial order relation, we define the set of rank-1 nodes

$$\mathcal{Q}^{\text{rank-1}}(t) := \{Q \in V^{\text{open}}(t) : \hat{Z}_Q \leq \inf\{\hat{Z}_{Q'} : Q' \in S, Q' \prec Q\}\} \quad (5.2)$$

as the set of all open nodes with a node estimation at least as good as the best evaluated node at the same depth. Note that $\mathcal{Q}^{\text{rank-1}}(t)$ may become empty much earlier than $V^{\text{open}}(t)$, i.e., prior to the termination of the search, as soon as sufficiently many nodes with small node estimation have been processed, one at every depth of the current tree.

Using the following *rank-1 transition*, we assume that the current incumbent is optimal when $\mathcal{Q}^{\text{rank-1}}(t)$ becomes empty:

$$t_2^{\text{rank-1}} := \min\{t \geq t_1^* : \mathcal{Q}^{\text{rank-1}}(t) = \emptyset\}. \quad (5.3)$$

If there is an open node Q at a depth d_Q deeper than any of the processed nodes so far, it holds that $Q \in \mathcal{Q}^{\text{rank-1}}(t)$ since

$$\hat{Z}_Q \leq \inf\{\hat{Z}_{Q'} : Q' \in S, Q' \prec Q\} = \inf \emptyset = \infty.$$

The main difference between the best-estimate and the rank-1 transitions is that the rank-1 transition does not directly compare incumbent solution objectives and node estimations. On the one hand, it is an intuitive restriction to only compare nodes of the same depth because node estimations can be assumed to gain precision with increasing depth. Furthermore, it has a computational benefit because our update procedure only needs to compare newly inserted open nodes with other open nodes at the same depth.

For every depth d , we keep track of the minimum node estimate at this particular depth so far, including feasible nodes, i.e. subproblems with feasible LP relaxation solutions. Every time a node is branched on, its two children are inserted in an array V_d^{open} of open nodes at their depth d . V_d^{open} is sorted in nondecreasing order of the node estimations of the nodes. In order to keep the set $\mathcal{Q}^{\text{rank-1}}$ updated, we store for every depth the best-estimate over all nodes already processed, which we update every time a node $Q \in \mathcal{Q}^{\text{rank-1}}$ was selected to be explored next.

5.4 Computational Results

In this section, we first evaluate the potential of the two proposed transitions used with default settings. Second, we show how the use of the proposed transitions together with phase-specific solver settings affects the solution process.

5.4.1 Accuracy of the Proposed Phase Transitions

In this section, we analyze the accuracy of the proposed phase transition heuristics from Section 5.3. We based our implementation on SCIP 3.1.0 together with SoPlex 2.0 as LP-solver. All computations were performed on a cluster of 32 computers. Each computer runs with a 64bit Intel Xeon X5672 CPUs at 3.20 GHz with 12 MB cache and 48 GB main memory. The operating system was Ubuntu 14.4. A gcc compiler was used in version 4.8.2. Hyperthreading and Turboboost were disabled. We ran only one (single-threaded) job per computer in order to minimize the random noise in the measured running time that might be caused by cache-misses if multiple processes share common resources.

As test library, we use a combined library of MIPLIB 3 [Bixby et al., 1998], MIPLIB 2003 [Achterberg et al., 2006], and MIPLIB 2010 [Koch et al., 2011] after the removal of three infeasible instances. In addition, we excluded the four instances for which, by the time of this writing, the optimal objective value was unknown, so that it is not possible to determine the actual phase transition t_2^* . On the remaining 161 instances we ran SCIP with default settings and a time limit of 2h. We record the solving time in seconds after which a transition criterion was reached for the first time. Before we start checking the transition criteria, we require the search to explore at least 50 branch-and-bound nodes for the node frontier to be meaningfully initialized.

It is noteworthy that the node estimations in SCIP are not updated dynamically together with the pseudo-costs due to running-time considerations, i.e., all nodes keep their initial estimation during the entire time they are in the node queue, although more recent pseudo-cost information on the variables might be available.

The goal of this experiment is to compare the proposed transition points with the actual second phase transition t_2^* . It may happen that $t_2^* > 2h$, i.e., an optimal solution is not found within the time limit, or a transition criterion is not met. Therefore, we first consider instances for which the solver finished the improvement phase within the time limit and at least one of the transition criteria was met.

We present Figure 5.3 to compare the true second phase transition t_2^* and the phase transition t_2^{crit} that we recorded for the phase transition $\text{crit} \in \{\text{rank-1}, \text{estim}\}$. We compare the relative difference between the two points in time by means of their quotient $(t_2^{\text{crit}} + \tau)/(t_2^* + \tau)$ using a shift of $\tau = 10$ seconds, where t_2^{crit} is either $t_2^{\text{rank-1}}$ or t_2^{estim} . As for the shifted geometric mean (2.23), the use of the shift value τ compensates for very

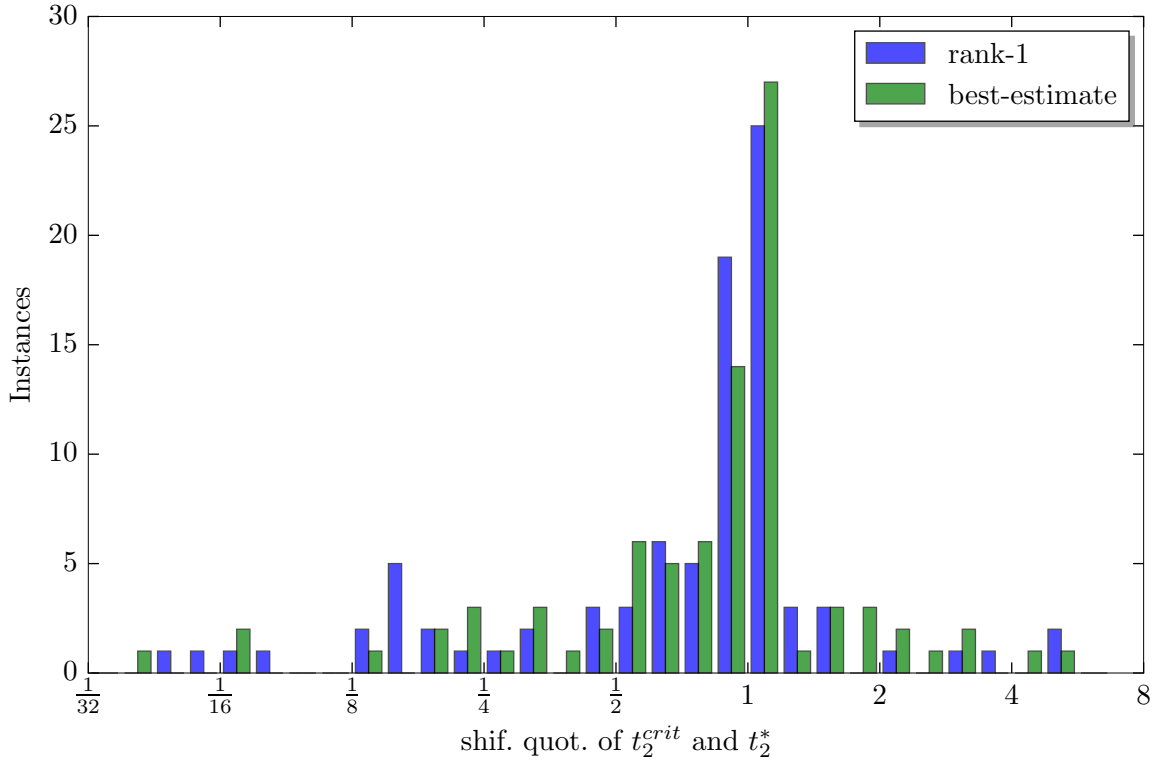


Figure 5.3: Relative deviation of the proposed transition criteria from the actual second phase transition t_2^* on instances where both values are smaller than 7200 seconds in our test.

large or small quotients caused by numbers that are close to zero and hence also shifts our attention to harder instances. Throughout this chapter, we use a shift of 10 seconds or 100 branch-and-bound nodes. A shifted quotient larger than one for an instance means that a phase transition heuristic correctly classifies an optimal incumbent solution. A quotient smaller than one, however, is encountered for instances where the transition criterion was met during the improvement phase. A bin width of 0.25 on the logarithm of the shifted quotients is used for this histogram. The two bins around one therefore denote the time span shortly before or after t_2^* . We do not show instances that could be solved during the root node. Out of the remaining 147 instances, SCIP finds optimal solutions for 117. Among those, the rank-1 and the best-estimate criterion are reached for 91 and 93 instances, respectively. Note that both the rank-1 and the best-estimate transitions are trivially met whenever there is no open node left in the tree, i.e., when the search is completed.

We see in Figure 5.3 that the bars for both transitions are centered around one, the rank-1 transition with 44 instances and the best-estimate transition with 41 instances. Both distributions slightly tend to take values smaller than 1, i.e., they tend to underestimate the second phase transition. Except for very few outliers, both distributions show a shifted quotient between $\frac{1}{8}$ and 8. In 75 out of 91 cases, the rank-1

best-estimate transition				rank-1 transition			
	optimal	not optimal	Σ		optimal	not optimal	Σ
reached	5	9	14	reached	4	15	19
not reached	1	21	22	not reached	2	15	17
Σ	6	30	36	Σ	6	30	36

Table 5.1: Contingency tables that group the 36 time limit instances into four categories according to whether the transition criterion was reached, and whether the incumbent at termination was optimal.

transition approximates the actual phase transition by a factor of 5, the best-estimate transition in 81 out of 93 cases.

Next, we compare the primal-dual gap and the primal gap at the time of transition. Recall that the primal gap requires knowledge of the optimal objective value for each instance. We consider all instances for which the transition point in time was reached within the time limit. If a transition criterion is met during the proof phase, the primal gap is zero by definition. The average primal-dual gap at the rank-1 transition point is 19.26, while the primal-dual gap at the best-estimate transition point averages to 16.39. Furthermore, the average primal gap regarding the optimal objective value is 7.01 and 5.71 for the rank-1 and the best-estimate transitions, respectively. This is in line with the previous observation that the best-estimate transition seems to occur later during the search than the rank-1 transition on our test set. It is interesting to note that the rank-1 transition occurs more often on instances where the primal gap is already small but is not yet proven by the primal-dual gap. More precisely, the transition point in time occurs on 46 out of 91 instances of our test bed when the primal gap is already smaller than 1 %, but the primal-dual gap is still larger than 1 %. The best-estimate transition achieves this on 38 out of 93 instances.

We now focus on instances where the solution process did not finish within our 2h time limit. For the instance `stp-3d`, no incumbent solution is found within two hours, so that neither transition criterion is met by definition. Among the remaining 36 instances that hit the time limit with an incumbent, there are 6 for which the incumbent is already optimal. We present in Table 5.1 two contingency tables grouped into two categories: whether the transition criterion was reached within the time limit or not and whether the incumbent solution at termination was already optimal. The entries on the diagonal represent instances for which the transition gives a correct classification, whereas the anti-diagonal gives the number of false positives and false negatives, respectively. Note that we measure only if a criterion was met at some point during the search, not if it was met precisely at termination. For the best-estimate transition, we see that for 5 out of 14 instances for which the transition criterion was reached, SCIP indeed finds an optimal solution. However, when the transition criterion is not reached within the time limit, this is in line with a suboptimal incumbent at termination in 21 out of 22 cases.

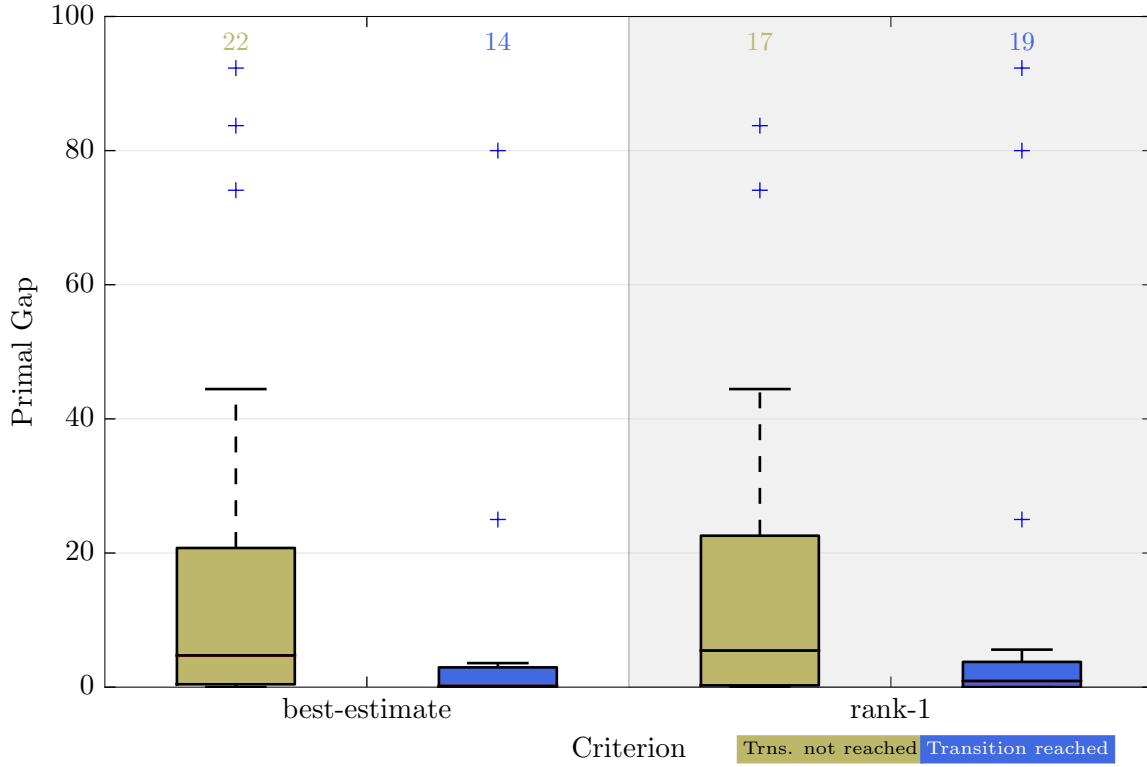


Figure 5.4: The primal gap at termination for 36 instances that could not be solved within the time limit.

Based on the best-estimate transition, we can classify 26 out of the 36 instances correctly. The table for the rank-1 transition does not show a similar result. Here, instances with suboptimal incumbent and with optimal incumbent are spread almost evenly across the two groups for this criterion. Overall, the best-estimate transition appears to be more conservative than the rank-1 transition. This matches our observations for the set of instances that could be solved within the time limit.

As a second comparison, we quantify the incumbent quality at termination by the primal gap. Figure 5.4 shows the primal gap at termination for both transition criteria. We show two box plots per transition for the groups of instances for which the criterion was not reached (left box) and for which it was reached (right box) within the time limit. The number of instances in each group is shown at the top of the box plot. In contrast to the primal gap statistics presented before, we focus on the primal gap at termination, not on the primal gap at the time of the transition. For instances for which the respective transition criterion was not reached, the median primal gap is 4.7 for the best-estimate and 5.4 for the rank-1 transition. For both transition criteria, there is a clear tendency of the right group towards zero, for which the median values are at 0.17 for best-estimate and at 0.92 for rank-1.

5.4.2 Using Phase Transitions to Control Solver Behavior

In a separate experimental study [Hendel, 2014], promising settings for each of the three solving phases were determined individually. In this section, we combine those phase settings and the phase transition heuristics to a phase-based MIP solver.

For the feasibility phase, we use a two-level node-selection as follows: We apply UCT [Sabharwal et al., 2012] for the first 31 nodes of the tree and afterwards a depth-first search that restarts periodically from the node with the best estimation. UCT, which has been presented for MIP in [Sabharwal et al., 2012], is a node selection heuristic that balances between exploration and exploitation using a hybrid score between node estimation and the number of times the node has already been visited during the search. Every selection step follows a path of nodes with best current score, starting at the root node. The counting of node visits leads to changes in the path selection during the search. Furthermore, we altered the branching rule to branch exclusively on inference information [Achterberg and Berthold, 2009]. This setting constituted the fastest setting in a feasibility phase experiment that also found feasible solutions for all instances within the time limit of 1 hour. During the improvement phase, we employ a setting that uses the default settings of SCIP except for an altered node-selection rule (UCT) inside Large Neighborhood Search heuristics.

After a transition criterion for the second phase transition is reached, we continue the node selection of open nodes with a pure depth-first search, which is the fastest method to traverse the remaining tree if the cutoff bound is optimal (which we assume in this phase). We also disable all primal heuristics for the remainder of the search, because no further solutions are necessary during the proof phase. Instead, we activate the separation of local cutting planes for the remainder of the search, as can be achieved with an aggressive emphasis on cutting planes in SCIP. With this setting, separation is performed at lower bound defining nodes at every 10'th level of the tree. For a detailed description of the methods involved and a comprehensive list of the used parameters, we refer to [Hendel, 2014].

We compare four different versions of our phase-based solver: By **default**, we denote the default settings of SCIP used throughout all three phases. The version **estim** uses the phase transition heuristic best-estimate and **rank-1** the rank-1 transition. Both switch between the settings described above at the transition points t_1^* and t_2^{estim} or $t_2^{\text{rank-1}}$. The version **oracle** uses the exact phase transition point t_2^* , by receiving the optimal solution value as input. **oracle** is used to estimate the potential improvement of a phase-based solver and serves as a reference for the actual improvements of **estim** and **rank-1** that use heuristic predictions. For all runs, we set a time limit of two hours. Detailed results for each tested instance can be found in Table A.1 in the appendix.

The first column of Table 5.2 shows the number of instances that were solved to optimality. **default** could solve 124 instances within the time limit. All other versions

5. MIP Solving Phases

settings	solved	123 instances solved by all			52 hard ($\max t > 200$)		
		time	time _Q	p	time	time _Q	p
default	124	90.7	1.000	–	799.0	1.000	–
estim	126	85.0	0.938	0.529	695.2	0.870	0.338
oracle	127	84.5	0.932*	0.004	665.9	0.833*	< 0.001
rank-1	125	83.3	0.918*	0.002	685.0	0.857*	0.022

Table 5.2: Shifted geometric mean results for **time** for the 123 instances solved by all versions and on the subset of 52 hard instances. Improvements by at least 5 % are printed in **bold**. Significant improvements are indicated in **bold and blue***

solved between 1 and 3 instances more in total. The best version in this respect is **oracle**, which solved 127 instances. In the following, we focus on the subset of 123 instances that were solved by all versions. We restrict ourselves to the subset of solved instances to better compare results regarding both solving time and nodes.

For those 123 instances and a subset of 52 hard instances, the table shows the shifted geometric mean (2.23) solving time, both absolute and relative compared to **default**. An instance is considered hard if at least one solver needed more than 200 sec. We use a shift of $\tau = 10$ seconds. The table also shows p -values in column **p** obtained from a modified two-sided Wilcoxon signed rank test that uses logarithmic shifted quotients and filtering to quantify if a version differed significantly from the default, see Section 2.9 for more details on this test methodology. We use a significance level of 0.05 for our experiments.

On the set of all instances, we observe improvements in the shifted geometric mean solving time for every version compared to **default**. The **oracle** version is 6.8 % faster than **default** in the shifted geometric mean. An even higher improvement is measured for the **rank-1** version, with which we achieve the highest speed-up over **default** of 8.2 %. These two improvements are accompanied by small p -values of 0.004 and 0.002, whereas the improvement shown for **estim** is not significant according to the Wilcoxon test.

On the hard instances, however, we observe significant improvements of up to 16.7 % with **oracle**, and still 14.3 % with **rank-1**, whereas **estim** improves the shifted geometric mean time by 13.0 % but the corresponding p -value of 0.338 does not identify this improvement as significant. The reason for this lies in the fact that the **estim** version extremely improves the time on a few instances, namely **acc-tight5** and **lectsched-4-obj**, for which **estim** achieves speed-up factors of 20 and 3, respectively, as well as six more instances with speed-ups of at least 2. However, disregarding these extreme speed-ups, the **estim** version shows more slow-downs compared to **oracle** and **rank-1**, both of which yield fewer extreme but more moderate speed-ups. The instance **acc-tight5** is a pure feasibility instance in the sense that the dual bound is already provided by the initial LP relaxation and a feasible solution of this objective needs to be found. Thus, the performance on this instance is greatly affected by our modifications to the settings

settings	123 instances solved by all			52 hard (max $t > 200$)		
	nodes	nodes _Q	p	nodes	nodes _Q	p
default	2565.5	1.000	–	17179.8	1.000	–
estim	2454.6	0.957	0.209	15119.3	0.880	0.059
oracle	2377.3	0.927*	< 0.001	13935.6	0.811*	< 0.001
rank-1	2512.2	0.979	0.221	16577.4	0.965	0.347

Table 5.3: Shifted geometric mean results for the number of branch-and-bound nodes **nodes**. Results are restricted to 123 instances for which all versions could finish the solve within two hours, and the subset of 52 hard instances, respectively. Improvements by at least 5 % are printed in **bold**. Significant improvements are indicated in **bold and blue***.

of SCIP during the feasibility phase. The other phase transition based settings yield the same speed-up for acc-tight5. Yet, an improvement of 14 % in the shifted geometric mean with **rank-1** is accompanied by a p -value of less than 3 %, which indicates that the rank-1 criterion is the better criterion w.r.t. the solving time.

Table 5.3 shows the shifted geometric mean results for all versions regarding the number of branch-and-bound nodes until optimality was proven. As in the previous table, we restrict the instances for the node comparison to the subset that could be solved to optimality by all settings within the time limit. The table further shows the results for the 52 hard instances of this set of instances. For the calculation of the mean, a shift of 100 nodes is used. The setting **oracle** significantly improves the overall shifted geometric mean of **default** by 7.3 % and by 18.9 % on the hard instances. For the criteria **estim** and **rank-1**, the obtained node reductions amount to 4.3 % and 2.1 %, respectively. In this case, however, the **p**-column does not indicate either of the improvements as significant. The split into easy and hard instances attributes the observed reductions mainly to the hard instances, for which **estim** shows an improvement of 12 % compared to **default**. Table 5.3 indicates that the significant solving time reduction of the **rank-1** transition does not stem from a systematic reduction of the overall search tree size. Instead, the **rank-1** transition mainly achieves a decreased solving time per node, which should be mainly attributed to a better coordination of the different solver components compared to the **default** settings.

Our experiments in the previous section revealed that **estim** transitions occur later during the search than **rank-1**-transitions. This makes **estim** a more conservative transition criterion. While it achieves a good performance w.r.t. overall running time, **rank-1** performs even better, in particular when taking the statistic significance of the results into account. In the previous section, we saw that **rank-1** has a tendency to underestimate the point of phase transition. We interpret our results such that switching settings shortly before the second phase transition, i.e. when the solver is about to find the optimal solution, is sufficient, if not preferable, to improve performance. The fact that only one of the two transition heuristics achieves a significant time speed-up shows that the performance gain cannot be attributed only to the changes to the feasibility

phase, during which all three versions that employ phase-based settings have exactly the same behavior.

Using the rank-1 phase transition, we obtain a solving time improvement that is similar to the improvement obtained with `oracle`. Comparing the results for an oracle-based phase transition and the phase transition criteria that we introduced, we conclude that the rank-1 transition is sufficient in practice to achieve a solving-time performance similar to what can be obtained in principle if we could determine the phase transition exactly.

5.5 Summary

In this chapter, we discussed the partition of a MIP solving process into three phases: *feasibility*, *improvement*, and *proof*. Each of them benefits from different algorithmic components. We introduced and empirically analyzed two criteria to predict the transition between the improvement and the proof phase: the best-estimate transition and the rank-1 transition.

We have shown that a phase-based version of the MIP solver SCIP, using the rank-1 transition, improves SCIP's mean running time by 8% on general MIP instances, and 14% on hard instances, while simultaneously reducing the number of branch-and-bound nodes. Hence, our computational experiments provide evidence that those easy-to-evaluate criteria correlate sufficiently well with the actual, hard-to-detect phase transition to make use of this approach in practice.

6

Adaptive Large Neighborhood Search

Large Neighborhood Search (LNS) heuristics are among the most powerful but also most expensive primal heuristics for MIP. Their computational effort makes it impractical to apply all of them frequently within the solver. Ideally, a solver adaptively concentrates its limited computational budget by learning which LNS heuristics work best for the MIP problem at hand. Following this line of thought, we propose *adaptive large neighborhood search (ALNS)* for MIP.

This chapter is an edited copy of the revised version of the journal article *Adaptive Large Neighborhood Search for Mixed Integer Programming* [Hendel, 2018] which is currently under review. We address in particular the question of how to select from the set of available auxiliary problems, which are introduced in Section 6.1.

In Section 6.2, we propose a suitable reward function for LNS heuristics to learn to discriminate between the auxiliary problems during the search. We also propose a generic variable fixing scheme that can be used to extend the set of fixed variables within a selected neighborhood to reach a target fixing rate. This has a particular impact on LNS heuristics that do not fix variables by themselves and may hence be too expensive on larger problems, such as, e.g., Local Branching [Fischetti and Lodi, 2003] (cf. Section 6.1.2).

The framework is obliged to trade off between exploration and exploitation, because only one auxiliary problem is selected and evaluated at a specific call. Such a selection scenario is also referred to as *multi-armed bandit problem*, in which a player tries to maximize their reward by playing one available action at a time and observing the particular reward of this action only.

We review three selection algorithms for the multi-armed bandit problem in Section 6.3. Two computational experiments are presented in Section 6.4, a first one to tune the selection process of the ALNS heuristic, and a second experiment to show that ALNS improves the MIP performance of SCIP on a large set of publicly available benchmark instances from the collections of MIPLIB [Achterberg et al., 2006; Bixby et al., 1998; Koch et al., 2011] and Cor@l [Coral].

Related Work

The notion of an Adaptive Large Neighborhood Search has already been coined in the literature, particularly in the context of Constraint Programming, where ALNS is usually tailored to a particular application. The authors of [Laborie and Godard, 2007] were the first to describe an adaptive LNS technique for single-mode scheduling problems, which selects from a finite set of so-called search operators, which are a CP analogue to the auxiliary problems for MIP (see Section 6.1). Building upon their method, ALNS has also been applied for different types of Vehicle Routing Problems, see [Pisinger and Ropke, 2007] for an overview. Throughout the remainder of this chapter, we will shortly write “ALNS” to denote our proposed “ALNS for MIP”.

A different, MIP specific approach to learn how to run heuristics has been recently proposed in [Khalil et al., 2017]. Their work uses logistic regression to predict the probability of success for different diving heuristics. The prediction is based on state information about the current node and the overall search. Their approach is fundamentally different from our proposed method in that it learns one regression for each individual diving heuristic, but does not attempt to prioritize between them.

The selection strategies presented here are truly online learners. They do not use any information that was collected offline before the search. The only feedback that the selection method receives is the reward of the selected action. Attempts with popular ML technology may consider additional features of the problem instance or search statistics to train a more informed selection method.

Focusing on the feature-independent setup of the bandit selection strategies has several advantages over an ML approach that uses features: First, the number of LNS executions in a MIP solver is typically small and may not suffice to collect enough training data for a feature based ML approach, whereas our simulation in Section 6.4.2 shows that the bandit strategies already learn useful information within a “typical” number of calls of an LNS heuristic. An alternative may be to collect the data only once to then train the ML approach offline and apply the learned model during the search, but this violates the “online” scope of this paper.

Finally, an ML approach may lack explainability why it prefers certain auxiliary problems in certain situations. In contrast, especially the α -UCB bandit strategy provides

a simple explanation, namely the UCB-score, why it prefers one auxiliary problem over another.

The first use of bandit related ideas inside MIP solvers concerns the integration of a node selection rule [Sabharwal et al., 2012] into CPLEX. This node selection approach balances exploration and exploitation of the search tree. It is inspired by a successful method for game search trees, which is related to the Upper Confidence Bounds (UCB) selection algorithm [Bubeck and Cesa-Bianchi, 2012] explained in Section 6.3.

6.1 Large Neighborhood Search Heuristics for MIP

The class of *Large Neighborhood Search* (LNS) heuristics lies at the most expensive end of the scale of general purpose primal heuristics because they solve an *auxiliary problem* with branch-and-bound techniques. To this end, LNS heuristics typically restrict the search space of an input MIP instance to a particular neighborhood of interest. The resulting *auxiliary problem* (cf. Definition 6.1) is again a MIP, which is then partially solved by a branch-and-bound algorithm under reasonable working limits, and eventual solutions are kept for the main search process. Many different LNS techniques have been proposed in recent years [Berthold, 2014a; Danna et al., 2005; Fischetti and Lodi, 2003; Fischetti and Monaci, 2014; Ghosh, 2007; Rothberg, 2007]. Briefly, an LNS heuristic solves an auxiliary MIP under strict working limits, which is derived from the original MIP by fixing variables, adding constraints, and/or changing the objective function.

Definition 6.1 (Auxiliary problem). *Let P be a MIP with n variables. For a polyhedron $N \subseteq \mathbb{Q}^n$ and objective coefficients $c_{aux} \in \mathbb{Q}^n$, a MIP Q defined as*

$$\min \{c_{aux}^t x \mid x \in \mathcal{S}_P \cap N\} \quad (6.1)$$

is called an auxiliary problem of P . The polyhedron N associated with Q is called its neighborhood.

In other words, Q has the same number of variables (columns) as the original MIP P and its solution set \mathcal{S}_Q is a subset of \mathcal{S}_P by construction. Definition 6.1 requires N to be a polyhedron, i.e., it should be expressed by a finite set of inequalities. The definition includes $N = \mathbb{Q}^n$. The auxiliary objective function c_{aux} can be different from the objective function of P .

In total, SCIP features 10 LNS heuristics. However, there are only a few different types of neighborhoods typically used. All LNS heuristics have in common that they solve auxiliary problems around a set of reference points. One of the most common classes of neighborhoods is derived by fixing those integer variables whose solution values agree on a set of reference points.

Definition 6.2 (Fixing neighborhood). Let P be a MIP with n variables and n^{int} integer variables. Let $M \subseteq \mathcal{I}$ and $x^* \in \mathbb{Q}^n$ denote a reference point. A fixing neighborhood

$$N^{fix}(M, x^*) := \{x \in \mathbb{Q}^n \mid x_j = x_j^*, j \in M\}$$

fixes the variables in M to their values in x^* .

Definition 6.3 (Matching set). For $k \geq 1$, let $X = \{x^1, \dots, x^k\} \subset \mathbb{Q}^n$ with $x^i \neq x^{i'}$ for all $i \neq i' \in \{1, \dots, k\}$. The matching set

$$M^=(X) := \{j \in \mathcal{I} \mid x_j^i = x_j^1, i \in \{1, \dots, k\}\}$$

describes all integer variable indices whose values agree on X . We call X the set of reference points.

Definitions 6.2 and 6.3 admit the use of reference points such as LP solutions, which are not (integer) feasible. Note that whenever a set of reference points X contains at least one solution $\tilde{x} \in \mathcal{S}_P$, the auxiliary MIP defined by the fixing neighborhood of the matching set is feasible because $X \subseteq N^{fix}(M^=(X), \tilde{x})$.

The task of finding an improving solution can be easily incorporated into Definition 6.1. Assume that an incumbent solution $\tilde{x}^{best} \in \mathcal{S}_P$ and a dual bound Z^* are already available. For $\delta \in (0, 1)$, every solution $\tilde{x} \in \mathcal{S}_P$ that satisfies

$$c^t \tilde{x} \leq (1 - \delta) \cdot c^t \tilde{x}^{best} + \delta \cdot \underbrace{Z^*}_{< c^t \tilde{x}^{best}} < c^t \tilde{x}^{best}$$

is clearly an improving solution. The set of solutions that are better than \tilde{x}^{best} by at least δ is contained in the *improvement neighborhood*

$$N^{obj}(\delta, \tilde{x}^{best}) := \{x \in \mathbb{Q}^n \mid c^t x \leq (1 - \delta) \cdot c^t \tilde{x}^{best} + \delta \cdot Z^*\}. \quad (6.2)$$

Therefore, whenever an incumbent solution is available, our auxiliary problems are always defined over the combination N' of a neighborhood N with an improvement neighborhood,

$$N' = N \cap N^{obj}(\delta, \tilde{x}^{best}),$$

to filter out all nonimproving solutions regardless of the choice of c_{aux} . The choice of δ is an important control parameter to weigh between the difficulty (and feasibility) of the auxiliary problem and the desired amount of improvement.

6.1.1 Fixing Neighborhood LNS Heuristics

Combining Definitions 6.2 and 6.3 is very popular for constructing auxiliary problems. Starting from a set of reference points $X = \{x^1, \dots, x^k\}$, a fixing neighborhood is

obtained with the help of the matching set of X . Since all points in X agree on their matching set $M^=(X)$, the same fixing neighborhood is obtained regardless of the anchor point

$$N^{\text{fix}}(M^=(X), x^1) = N^{\text{fix}}(M^=(X), x^i) \quad \text{for all } i \in \{1, \dots, k\}.$$

RINS [Danna et al., 2005] *Relaxation induced neighborhood search (RINS)* is one of the first generally applicable LNS approaches. The idea of RINS is to fix integer variables whose solution values agree in the solution \tilde{x} of the LP relaxation at the current local node, and the current incumbent solution \tilde{x}^{best} . The neighborhood of the auxiliary MIP of RINS is

$$N_{\text{RINS}} := N^{\text{fix}}(M^=(\{\tilde{x}, \tilde{x}^{\text{best}}\}), \tilde{x}^{\text{best}}).$$

Crossover [Rothberg, 2007] Another improvement heuristic is the *Crossover* heuristic, which is inspired by the recombination of solutions within genetic algorithms. Crossover selects $k \geq 2$ already known, feasible solutions $Y = \{\tilde{x}^1, \dots, \tilde{x}^k\} \subseteq \mathcal{S}_P$ as reference points, which need not necessarily contain the incumbent. The *crossover neighborhood* fixes

$$N_{\text{Cross}} := N^{\text{fix}}(M^=(Y), \tilde{x}^1).$$

[Rothberg, 2007] suggest to use $k = 2$ solutions that are randomly selected from all available solutions, using a bias towards solutions with better objective.

Mutation Furthermore, in the same article, Rothberg [2007] suggest a second LNS heuristic called *Mutation* in SCIP lingo that fixes a random subset of integer variables of the incumbent solution. For a randomly chosen subset $M^{\text{rand}} \subseteq \mathcal{I}$, the *mutation neighborhood* is defined as

$$N_{\text{Muta}} := N^{\text{fix}}(M^{\text{rand}}, \tilde{x}^{\text{best}}).$$

Mutation is the only LNS heuristic for which the difficulty of the auxiliary problem can be directly controlled, namely by a number or percentage of integer variables that should be fixed. In contrast, all previous neighborhoods depend on the cardinality of their matching set.

RENS [Berthold, 2014a] Starting from an LP relaxation solution \tilde{x} , the relaxation enforced neighborhood search (RENS) neighborhood focusses on the feasible roundings of \tilde{x} and can be written as

$$N_{\text{RENS}} := \{x \in \mathbb{Q}^n \mid \lfloor \tilde{x}_j \rfloor \leq x_j \leq \lceil \tilde{x}_j \rceil, j \in \mathcal{I}\}. \quad (6.3)$$

Similarly to the RINS heuristic, the aim of RENS is to construct feasible solutions that are close to the LP relaxation solution and therefore have a near-optimal solution value.

6.1.2 LNS Heuristics Using Constraints and Auxiliary Objective Functions

All approaches presented so far fix a set of integer variables using one or several reference points. *Local Branching* [Fischetti and Lodi, 2003] is the first LNS heuristic that uses a different neighborhood.

Local Branching [Fischetti and Lodi, 2003] Instead of fixing a set of variables and solving for improving solution values on the remaining variables, the neighborhood of *Local Branching* is restricted to a ball around the current incumbent solution in a special metric. More formally, Let P be a MIP with $\mathcal{B} \neq \emptyset$ binary variables. Based on the Manhattan metric for $x \in \mathbb{Q}^n$, the *binary norm* of x is defined as

$$\|x\|_b := \sum_{j \in \mathcal{B}} |x_j|. 1$$

Let $\tilde{x}^{\text{best}} \in \mathcal{S}_P$ be an incumbent solution for P , and let $d^{\text{max}} > 0$ denote a distance cutoff parameter. The *local branching neighborhood* is the restriction

$$N_{\text{LBranch}} := \left\{ x \in \mathbb{Q}^n \mid \|x - \tilde{x}^{\text{best}}\|_b \leq d^{\text{max}} \right\}$$

The reason for preferring the binary norm over the regular norm or the norm taking all integer variables is practicality. The binary norm can be expressed as a linear constraint without introducing auxiliary variables.

Proximity Search [Fischetti and Monaci, 2014] A dual version of Local Branching has been introduced as *Proximity*. Using the binary norm, Proximity seeks to minimize the binary norm $\|x - \tilde{x}^{\text{best}}\|_b$ through an auxiliary objective coefficient vector c_{Proxi} defined as

$$(c_{\text{Proxi}})_j := \begin{cases} 0 & \text{if } j \notin \mathcal{B} \\ 1 & \text{if } j \in \mathcal{B} \text{ and } \tilde{x}_j^{\text{best}} = 0 \\ -1 & \text{if } j \in \mathcal{B} \text{ and } \tilde{x}_j^{\text{best}} = 1 \end{cases}$$

over the entire set of improving solutions:

$$N_{\text{Proxi}} := N^{\text{obj}}(\delta, \tilde{x}^{\text{best}}).$$

Zero Objective The second LNS heuristic that uses an auxiliary objective function different from the original objective function is *Zero Objective*. As its name suggests, it uses $c_{\text{Zeroobj}} := 0$ as auxiliary objective function. Zero Objective thereby reduces the search for an (improving) solution to a feasibility problem. If an incumbent solution \tilde{x}^{best} is

¹Strictly speaking, $\|\cdot\|_b$ is a seminorm because nonzero vectors can have binary norm of 0.

available, Zero Objective searches the set of improving solutions $N_{\text{Zeroobj}} := N^{\text{obj}}(\delta, \tilde{x}^{\text{best}})$, and $N_{\text{Zeroobj}} = \mathbb{Q}^n$ otherwise.

DINS [Ghosh, 2007] Distance induced neighborhood search (DINS) combines elements of the Crossover, Local Branching, and RINS heuristics. Similarly to RINS, the intuition is that improving solutions are located between the current incumbent solution \tilde{x}^{best} and the solution to the node LP relaxation \check{x} . With the intention of reducing their *integer distance*

$$\|\tilde{x}^{\text{best}} - \check{x}\|_i := \sum_{j \in \mathcal{G}} |\tilde{x}_j^{\text{best}} - \check{x}_j|,$$

let $J := \{j \in \mathcal{G} \mid |\tilde{x}_j^{\text{best}} - \check{x}_j| \geq 0.5\}$ denote the index set of general integer variables with a difference of at least 0.5 between the two reference points. The J -neighborhood of DINS is

$$N_J := \{x \in \mathbb{Q}^n \mid |x_j - \check{x}_j| \leq |\tilde{x}_j^{\text{best}} - \check{x}_j|, j \in J\}.$$

This neighborhood restricts lower and upper bounds of the general integer variables. Let $Y^{\text{DINS}} \subseteq \mathcal{S}_P$ denote a subset of currently available solutions, containing \tilde{x}^{best} , to the MIP at hand. The DINS neighborhood can be written as a combination of a total of four neighborhoods

$$\begin{aligned} N_{\text{DINS}} := & N_J \\ & \cap N^{\text{fix}}(\mathcal{G} \setminus J, \tilde{x}^{\text{best}}) \\ & \cap N^{\text{fix}}(\mathcal{B} \cap M^=(\{\check{x}, \check{x}^{\text{root}}\} \cup Y^{\text{DINS}}), \tilde{x}^{\text{best}}) \\ & \cap N_{\text{LBranch}}. \end{aligned}$$

The set of general integer variables outside J is fixed to the values in the incumbent solution. Binary variables that have not changed between the root LP relaxation solution \check{x}^{root} and \check{x} are fixed if they have taken the same value in all solutions in Y^{DINS} . In our implementation of DINS, we use between $1 \leq |Y^{\text{DINS}}| \leq 5$ available solutions with best objective, depending on how many solutions are available. Finally, the search is further restricted to a certain binary distance around the current incumbent solution through an additional local branching neighborhood.

Remarks There is further work on LNS approaches that are not covered here. Note that the only heuristics that do not use an incumbent solution are RENS and Zero Objective. Fischetti and Lodi [2008] proposed an extension of Local Branching that starts from an infeasible reference point. Such points are quickly produced by rounding or with a few iterations of the Feasibility Pump [Fischetti and Salvagnin, 2009]. In addition to the local branching constraint, the auxiliary problem of [Fischetti and Lodi, 2008] is extended by additional variables to model and penalize the violation of constraints, inspired by the phase 1 of the simplex algorithm. A recent approach called Alternating

Criteria Search [Munguía et al., 2018] also starts from infeasible reference points, and alternates between auxiliary problems with artificial feasibility objective and the original objective function of the input MIP in a parallel setting. The necessary diversification is obtained by fixing subsets of integer variables indexed by a random consecutive index set, which is a variant of Mutation [Rothberg, 2007] discussed in Section 6.1.1. The heuristics presented in [Gamrath et al., 2015a, 2019] formulate and solve auxiliary problems only for the general integer variables as a final post-processing step after fixing all binary variables based on available, global problem structures such as cliques and implications between binary and integer variables.

6.2 Adaptive Large Neighborhood Search for MIP

The proposed Adaptive Large Neighborhood Search heuristic governs the execution of a set \mathcal{Q} of 8 available auxiliary problems from Section 6.1, which has been chosen as a representative set of diverse LNS heuristics from the literature. Table 6.1 gives a quick overview of the auxiliary problems used, as well as their individual preconditions. ALNS is executed periodically during the main search based on a certain, parameter controlled frequency. In addition, ALNS respects an internal budget and previous successes and may decide to delay its execution until later. This dynamic execution schedule is explained in detail in Section 6.2.2. Therefore, we denote only those calls to ALNS as *rounds* that lead to the selection and solution process of an auxiliary problem.

In each such round $t = 1, 2, \dots$, ALNS basically performs the following steps.

1. Select an auxiliary problem $Q_t \in \mathcal{Q}$ via a *bandit selection strategy*.
2. Apply generic (un-)fixing of integer variables depending on the target fixing rate of Q_t . Stop if generic fixing cannot be applied.
3. Setup and solve the auxiliary problem Q_t .
4. Reward the auxiliary problem and update the bandit selection strategy based on the outcome of Q_t .

Different bandit selection strategies and their individual update procedures are explained in Section 6.3. The solution process of the auxiliary problem uses a strict limit on the number of branch-and-bound nodes to keep the overall computational effort small. It may still be very expensive to solve auxiliary problems if the corresponding neighborhood is large, especially since some neighborhoods do not fix integer variables directly. In Section 6.2.1, a generic approach is explained for fixing additional variables to reach any desired target fixing rate and hence reduce the complexity of the auxiliary problem. Details on the dynamic adjustment of the target fixing rate and node limit are given in Section 6.2.2. Finally, Section 6.2.3 introduces the scoring mechanism for rewarding Q_t .

Auxiliary Problem	Description	Preconditions
RINS	Fixes matching values in incumbent and LP relaxation solution	Feasible LP relaxation at current node, incumbent solution
Crossover	Fixes matching values in 2 or more randomly chosen, available solutions	Sufficiently many solutions
Mutation	Fixes random subset of variables to values in incumbent	Incumbent solution
RENS	Restricts the auxiliary problem to the feasible roundings around the current LP solution	Feasible LP relaxation at current node
Local Branching	Limits the maximal binary distance from the incumbent	Incumbent solution, MIP with binary variables
Proximity Search	Minimizes the binary distance from the incumbent	Incumbent solution, MIP with binary variables
Zero Objective	Reduces search for an (improving) solution to a feasibility problem	nonzero objective function
DINS	Sophisticated combination of RINS, Crossover, and Local Branching	Incumbent solution, feasible LP relaxation at current node

Table 6.1: Overview of the auxiliary problems used in ALNS. See Section 6.1 for information and references.

6.2.1 Fixing and Unfixing Variables

Many of the neighborhood definitions in Section 6.1 consist in the fixing of a subset of integer variables to their values in a solution $\tilde{x} \in \mathcal{S}_P$, which can be the incumbent or an inferior solution for P at round t and depends on the selected auxiliary problem Q_t . The *fixed set* of Q_t and its corresponding neighborhood N_{Q_t} is defined as

$$M_{Q_t}^{\text{fix}} := \{j \in \mathcal{B} \cup \mathcal{G} \mid N_{Q_t} \subseteq \{x \in \mathbb{Q}^n \mid x_j = \tilde{x}_j\}\}.$$

The size of the fixed set is denoted by $n_{Q_t}^{\text{fix}} := |M_{Q_t}^{\text{fix}}|$. Every auxiliary problem (action) $Q \in \mathcal{Q}$ operated by ALNS has a *target fixing rate* $\phi_{Q,t} \in [0, 1)$ that changes between rounds as explained in Section 6.2.2. It may happen that Q_t does not reach its specified target fixing rate, i.e.

$$n_{Q_t}^{\text{fix}} < \phi_{Q_t,t} \cdot n^{\text{int}}.$$

For example, the neighborhoods of Zero Objective, Proximity and Local Branching do not fix any variables. It may also happen that $\phi_{Q_t,t}$ is exceeded, which unnecessarily restricts the search space. ALNS treats both cases very similarly by using a generic variable fixing prioritization to sort the set of possible (un)fixings.

6. Adaptive Large Neighborhood Search

In the first case, $\phi_{Q_t,t} \cdot n^{\text{int}} - n_{Q_t}^{\text{fix}}$ additional integer variables from $\overline{M_{Q_t}^{\text{fix}}} = \mathcal{I} \setminus M_{Q_t}^{\text{fix}}$ have to be selected. For two integer variables $j, j' \in \overline{M_{Q_t}^{\text{fix}}}$ with reference solution values $\tilde{x}_j, \tilde{x}_{j'}$, the fixing $x_j = \tilde{x}_j$ is preferred over $x_{j'} = \tilde{x}_{j'}$, if, in decreasing order of priority,

1. x_j has a smaller distance than $x_{j'}$ from $M_{Q_t}^{\text{fix}}$ in the *variable constraint graph* (see below).
2. The reduced costs for fixing $x_j = \tilde{x}_j$ are smaller than those for fixing $x_{j'} = \tilde{x}_{j'}$,

$$\tilde{d}_j \cdot (\tilde{x}_j - \check{x}_j^{\text{root}(j)}) < \tilde{d}_{j'} \cdot (\tilde{x}_{j'} - \check{x}_{j'}^{\text{root}(j')})$$

3. The *pseudo-costs* (see below) for fixing $x_j = \tilde{x}$ are smaller than for $x_{j'} = \tilde{x}_{j'}$,

$$\Psi_j(\tilde{x}_j - \check{x}_j^{\text{root}}) < \Psi_{j'}(\tilde{x}_{j'} - \check{x}_{j'}^{\text{root}}).$$

4. Randomly.

Variable constraint graph The idea behind the variable constraint graph is to maintain several unfixed variables together in some constraints to increase the likelihood that the auxiliary problem contains an improving solution. Intuitively, finding improving solutions requires to alter several solution values per constraint. For a constraint with only a single unfixed variable in the auxiliary problem, it is unlikely that the reference solution value of this variable can be altered in the direction of an improving solution.

For a given MIP P , the variable constraint graph G_P is a bipartite graph with one node for each variable and constraint of P , $V(G_P) = \{v_j \mid j \in \{1, \dots, n\}\} \cup \{w_j \mid j \in \{1, \dots, m\}\}$. Its edges $E(G_P) := \{(v_j, w_i) \mid A_{ij} \neq 0\}$ correspond to the nonzero entries of the matrix A . Distances in G_P are breadth first distances. First, each node has a distance of 0 to itself. Starting from a variable node v_j , all variables with an edge to one of the constraint nodes adjacent to v_j have a distance of 2 from v_j .

All variables which are reachable via another constraint from any of the nodes with distance 2 have a distance of 4, and so on. Since all distances between variables in G_P are even, we divide all breadth-first distances by two.

The distance of a variable node v_j from the nodes corresponding to $M_{Q_t}^{\text{fix}}$ is the minimum distance to any of the variable nodes in $M_{Q_t}^{\text{fix}}$. It is determined by queuing all variable nodes in $M_{Q_t}^{\text{fix}}$ into the initial queue for breadth first search.

If the original problem has block structure, the variable prioritization concentrates additional fixings on those blocks with a nonempty intersection in $M_{Q_t}^{\text{fix}}$. Related ideas are used, e.g., in presolving for detecting independent components of a MIP [Gamrath et al., 2015b], or within the *Graph-Induced Neighborhood Search* (GINS) released with SCIP 4.0 [Maher et al., 2017].

Reduced cost score The cost based scores used as tiebreakers in steps 2 and 3 both penalize a deviation of the potential fixing from an LP solution at the root node of the branch-and-bound search. After the initial LP relaxation at the root of the search, most MIP solvers solve a sequence of further LP relaxations during their cut separation loop.

The first associated penalty uses reduced costs. Recall from Section 2.2 that reduced costs are part of every optimal simplex tableau. At the root node, reduced costs during the LP relaxations are stored per variable, to enable root reduced-cost strengthening (see Section 2.7.1) during the search. For each variable x_j , we initialize its associated *root reduced costs* to those observed in the initial optimal LP solution. At each subsequent LP, each time we encounter higher reduced costs than the recorded reduced costs for x_j , they are stored together with the corresponding LP solution value $\check{x}_j^{\text{root}(j)}$. Therefore, the LP solution values used to compare the potential fixings of j and j' may come from different LP solutions. Ties in the reduced cost comparison can occur if, for example, both variables have a corresponding score of 0, which is always the case if both variables are basic in all LP solutions at the root node.

Reduced costs are 0 for all basic variables in the optimal simplex tableau. The LP solution $\check{x}_j^{\text{root}(j)}$ of each nonbasic variable x_j is either ℓ_j or u_j . Nonbasic variables at their lower bound have a reduced cost coefficient $\tilde{d}_j \geq 0$ and variables at their upper bound have $\tilde{d}_j \leq 0$. In both cases, the product $\tilde{d}_j \cdot (\tilde{x}_j - \check{x}_j^{\text{root}(j)})$ is nonnegative. It is a lower bound on the objective deterioration by fixing $x_j = \tilde{x}_j$, hence the preference for variable fixings with smaller reduced cost scores.

Pseudo-cost score So far, we have introduced *pseudo-costs* [Bénichou et al., 1971, see also Definition 2.8] as a common aggregate of historical branching information on the variables. Recall that the pseudo-cost score is an estimation of the potential dual bound degradation after fixing $x_j = \tilde{x}_j$. As reference serves the final root LP solution \check{x}^{root} on which the solution process started branching. Computing $\Psi_j(\tilde{x}_j - \check{x}_j^{\text{root}})$ considers fixing $x_j = \tilde{x}_j$ as a branching restriction from $\check{x}_j^{\text{root}} \notin \mathbb{Z}$ in the direction of \tilde{x}_j , upwards if $\tilde{x}_j > \check{x}_j^{\text{root}}$ and downwards otherwise. The pseudo-cost score estimates the increase in the dual bound after fixing j . As an example, assume that a binary variable $j \in \mathcal{B}$ has a root LP solution value $\check{x}_j^{\text{root}} = 0.4$ and a reference solution value of $\tilde{x}_j = 1$. Assume that the average dual bound increase has been $\Psi_j^- = 10$ for branching down on x_j and $\Psi_j^+ = 5$ for branching up. The pseudo-costs for the fixing $x_j = \tilde{x}_j$ are calculated as $\Psi_j(\tilde{x}_j - \check{x}_j^{\text{root}}) = \Psi_j^+ \cdot 0.6 = 3$. If the solution value had been 0 instead of one, the corresponding pseudo-cost score is $\Psi_j(0 - \check{x}_j^{\text{root}}) = -\Psi_j^- \cdot (-0.4) = 4$ for branching down on x_j .

The pseudo-cost score summarizes the branching history of a variable. Like reduced cost scores, pseudo-costs are always nonnegative. The pseudo-cost score $\Psi_j(\tilde{x}_j - \check{x}_j^{\text{root}})$ of a fixing is only an estimate of the impact of the fixing of x_j on the objective, in contrast to reduced cost scores. As for reduced costs, we prefer additional fixings with smaller

pseudo-cost scores to increase the likelihood of good solutions in the remaining auxiliary problem.

Unfixing variables Only slight details are changed if the neighborhood was too restrictive, such that $n_{Q_t}^{\text{fix}} - \phi_{Q_t,t} \cdot n^{\text{int}}$ variables from $M_{Q_t}^{\text{fix}}$ should be selected and unfixed (relaxed). Distances are now computed in the variable constraint graph starting from $\overline{M}_{Q_t}^{\text{fix}}$, and variables with a small distance from $\overline{M}_{Q_t}^{\text{fix}}$ are preferably relaxed to keep the auxiliary problem connected. Since we use the cost tiebreakers as estimate of the objective degradation of a fixing in the auxiliary problem, variables are relaxed preferably if they have a large reduced cost score or, in case a tie occurs, a large pseudo-cost score. Finally, if none of the scores discriminate between two variables, the preference is given by a random score assigned to each variable. Generic (un-)fixing within ALNS is only applied if the target fixing rate $\phi_{Q_t,t}$ is missed by a tolerance of 10 %, i.e. only if $n_{Q_t}^{\text{fix}} \notin [(\phi_{Q_t,t} - 0.1) \cdot n^{\text{int}}, (\phi_{Q_t,t} + 0.1) \cdot n^{\text{int}}]$. As the fixings of the RENS neighborhood depend solely on the LP solution, it is the only auxiliary problem for which there is no suitable integer feasible reference solution to use for generic fixings. Generic unfixings, however, are always possible, even for RENS.

6.2.2 Dynamic Limits

Good limits on the computational budget of an LNS heuristic are essential to make it useful inside a MIP solver. To this end, a trade-off must be made between the intensity of the search inside the auxiliary problem and the runtime. For ALNS, the complexity of the auxiliary problem and the budget are adapted dynamically between the individual calls to ALNS.

All the following dynamic decisions consider the auxiliary problem Q_t at the t -th round of ALNS ($t = 0, 1, \dots$) and its *solution status* $\text{stat}(Q_t)$ which can be one of

- *inf*, if Q_t was infeasible,
- *opt*, if Q_t was solved to optimality
- *sol*, if Q_t provided an improving solution for P
- *nosol*, if no improving solution was found searching Q_t .

Target fixing rate $\phi_{Q_t,t}$ The first dynamic adjustment of the auxiliary problem complexity over time is described in [Rothberg, 2007], together with the introduction of the Crossover and Mutation LNS heuristics (cf. Section 6.1), which are available in ALNS. Rothberg [2007] suggests controlling the complexity of an auxiliary problem Q_t by specifying the amount of integer variables that should be fixed before solving the auxiliary problem Q_t . The intuition is that the difficulty of Q_t decreases with increasing fixing rate. In the notation of the present work, the target amount of fixed integer

variables at round t is specified by a target fixing rate $\phi_{Q,t} \in [0, 1)$ for each $Q \in \mathcal{Q}$ of ALNS.

For $Q \in \mathcal{Q}$, let $T_Q(t)$ denote the number of times that Q has been selected, including round t . The fixing rate is modified according to the status in round t as

$$\phi_{Q,t+1} = \begin{cases} \phi_{Q,t}, & \text{if } Q \neq Q_t \text{ or } \text{stat}(Q_t) = \text{sol}, \\ \max\{0.1, \phi_{Q,t} - 0.75^{T_Q(t)} \cdot 0.2\}, & \text{if } \text{stat}(Q_t) \in \{\text{inf}, \text{opt}\} \\ \min\{0.9, \phi_{Q,t} + 0.75^{T_Q(t)} \cdot 0.2\} & \text{if } \text{stat}(Q_t) = \text{nosol} \end{cases}$$

If Q_t was too easy for the solver, i.e. it could be solved to optimality or infeasibility within a given node budget, the fixing rate for the next iteration is decreased. If no new solution was found, the target fixing rate is increased. If a solution was found, but the search could not be completed, the fixing rate is kept. The additive change of the fixing rate is 0.2 initially, which is multiplied with 0.75 after every update step, exactly as in [Rothberg, 2007]. The use of max and min ensures that the target fixing rate stays within 10 % and 90 %. In our implementation, those two values are parametrized and can be individually set for every auxiliary problem. Every target fixing rate is initialized by $\phi_{Q,1} = 0.9$, which represents the most conservative value in the allowed range of the fixing rate², see Section 6.4 for details.

Stall node limit ν_t^{lim} The main budget limitation of ALNS is a limit on the number of consecutive branch-and-bound nodes during which no improving solution is found, the so-called stall node limit. The stall node limit ν_{t+1}^{lim} for the next round of ALNS is adjusted based on the results of the auxiliary problem of round t as follows.

$$\nu_{t+1}^{\text{lim}} = \begin{cases} \nu_t^{\text{lim}}, & \text{if } \text{stat}(Q_t) \in \{\text{opt}, \text{inf}, \text{sol}\} \\ \min\{\lfloor \nu_t^{\text{lim}} \cdot 1.05 \rfloor + 1, 5000\}, & \text{if } \text{stat}(Q_t) = \text{nosol} \end{cases}$$

ALNS uses an affine linear function of the branch-and-bound nodes U^{bb} in the main search to limit the search effort inside auxiliary problems. Let $U^{\text{bb}}(Q_i)$ denote the amount of nodes used for searching the auxiliary problem Q_i at round $1 \leq i \leq t-1$, and let $s(t-1)$ denote the total number of improving solutions found by ALNS until round $t-1$ inclusively. Concretely, the next round t of ALNS is called as soon as the main search nodes U^{bb} have progressed such that

$$\kappa_0 + \frac{s(t-1) + 1}{(t-1) + 1} \cdot \kappa_1 \cdot U^{\text{bb}} - \sum_{i=1}^{t-1} (100 + U^{\text{bb}}(Q_i)) \geq \nu_t^{\text{lim}}. \quad (6.4)$$

Here, κ_0 is an initial budget of ALNS and κ_1 is the node budget relative to U^{bb} . The initial budget κ_0 allows to execute ALNS already early during the tree search when

²controlled via 8 user parameters `heuristics/alns/*/maxfixingrate` all defaulting to 0.9.

U^{bb} is small. When the search progresses and the initial budget κ_0 has been spent after a (usually small) number of ALNS rounds, the relative node budget κ_1 is the main parameter to control ALNS resources relative to the main search. In (6.4), the relative budget is increased or decreased based on the total number of improving solutions that ALNS contributed. With this strategy, ALNS slowly fades out if it does not find improving solutions. The last term expresses the total resources used so far by ALNS, with an additional 100 nodes per round to account for the setup costs of each Q_i .

Recall that SCIP calls primal heuristics based on their frequency parameter 2.6. In our experiments in Section 6.4, our ALNS implementation has its frequency set to 20, which means that it is called at every depth 0, 20, 40 etc., i.e., roughly every 20 nodes. However, because of the budget computation in Equation (6.4), ALNS only performs the next round, i.e., selects and solves an auxiliary problem when, in addition, the budget computation (6.4) allows for the next round. A similar budget computation is used inside the standalone LNS heuristics RINS and Crossover, as well.

In contrast to the target fixing rate, the stall node limit ν_t^{lim} is a global limit independent of the selected auxiliary problem. This design choice has been made because the target fixing rate is supposed to be the main driver to adjust auxiliary problem difficulty.

6.2.3 A Reward Function for Auxiliary Problems

All the bandit selection strategies presented in Section 6.3 require the definition of a suitable reward function. Intuitively, the reward should always be higher for auxiliary problems that lead to improvements over the current incumbent solution and also depend on the achieved objective quality. Furthermore, between unsuccessful auxiliary problems, the reward should still distinguish if the solution process failed fast or if it required a lot of computational resources. In order for some of the selection strategies in the following Section 6.3 to work correctly, we require that a reward should be in the interval $[0, 1]$. A reward of 0 is the worst possible score, i.e., the maximum penalty.

Let $Q_t \in \mathcal{Q}$ denote the selected auxiliary problem in round $t > 0$, and let $Z^{\text{old}} := c^{\text{t}} \tilde{x}^{\text{best}}$ denote the incumbent value before Q_t is solved, if an incumbent solution $\tilde{x}^{\text{best}} \in \mathcal{S}_P$ is available, or $Z^{\text{old}} := \infty$ otherwise. Similarly, let Z^{new} denote the objective of the best known solution after Q_t has been solved. As before, let ν_t^{lim} and $U^{\text{bb}}(Q_t)$ denote the stall node limit and amount of nodes used by Q_t , respectively.

Two reward functions are combined to reward both the presence of a new incumbent solution and the objective improvement. The former is expressed by the *solution reward*

$$r^{\text{sol}}(Q_t, t) := \begin{cases} 1, & \text{if } \text{stat}(Q_t) \in \{\text{opt}, \text{sol}\}, \\ 0, & \text{else.} \end{cases}$$

The improvement in solution quality is measured by the *closed gap reward*

$$r^{\text{gap}}(Q_t, t) := \frac{Z^{\text{old}} - Z^{\text{new}}}{Z^{\text{old}} - Z^*},$$

which evaluates to 0 if no improving solution could be found, and to 1 if the new solution has an objective that is equal to the dual bound (and hence optimal for P). As a convention, the closed gap reward is 1 if Q contributes the first solution to the problem. Since most neighborhoods require a known solution as input (cf. Table 6.1), this is only possible with RENS and Zero Objective.

Since the time measurement in some MIP solvers such as SCIP is not deterministic, we use the number of nodes to introduce the *effort* $\xi(t)$ as

$$\xi(t) := (1 - \phi_{Q_t, t}) \frac{U^{\text{bb}}(Q_t)}{\nu_t^{\text{lim}}}. \quad (6.5)$$

The effort $\xi(t)$ serves as a deterministic approximation of runtime spent on the search in the auxiliary problem. It has the additional property that it lies in the interval $[0, 1]$. In order to compensate for different target fixing rates, $\xi(t)$ uses a scaling by the remaining number of free integer variables. The generic (un)fixing based on the variable prioritization from Section 6.2.1 ensures that the fraction of fixed integer variables in the subproblem is approximately equal to the current target fixing rate. The effort is ≥ 1 if the stall node limit was exhausted ($U^{\text{bb}}(Q_t) \geq \nu_t^{\text{lim}}$) and no integer variables were fixed by the neighborhood of Q_t . If solving Q_t fails to produce a better solution, the last of the three individual reward functions is the *failure reward*

$$r^{\text{fail}}(Q_t, t) := \begin{cases} 1, & \text{if } \text{stat}(Q_t) \in \{\text{opt}, \text{sol}\}, \\ 1 - \min\{\xi(t), 1\}, & \text{else,} \end{cases}$$

which becomes smaller depending on the effort spent in an auxiliary problem, if no improving solution was found.

With two additional convex combination parameters $\lambda_1, \lambda_2 \in [0, 1]$, the reward function of ALNS combines all three rewards into

$$r^{\text{alns}}(Q_t, t) := \lambda_1 r^{\text{fail}}(Q_t, t) + (1 - \lambda_1) \cdot \frac{\lambda_2 r^{\text{sol}}(Q_t, t) + (1 - \lambda_2) r^{\text{gap}}(Q_t, t)}{1 + \xi(t)} \quad (6.6)$$

The first control parameter λ_1 separates the reward between runs that were successful and runs that failed to improve the incumbent solution. The second parameter λ_2 adjusts between the solution and the closed gap rewards. The result, which is again a reward in the interval $[0, 1]$, is scaled by the effort involved to reward fast auxiliary problems more. For the remainder of this work, we propose to use values $\lambda_1 = 0.5$ and $\lambda_2 = 0.8$ as an intuitive choice which reserves the lower half of the reward interval $[0, 1]$ for unsuccessful LNS executions, and the upper half for improvements.

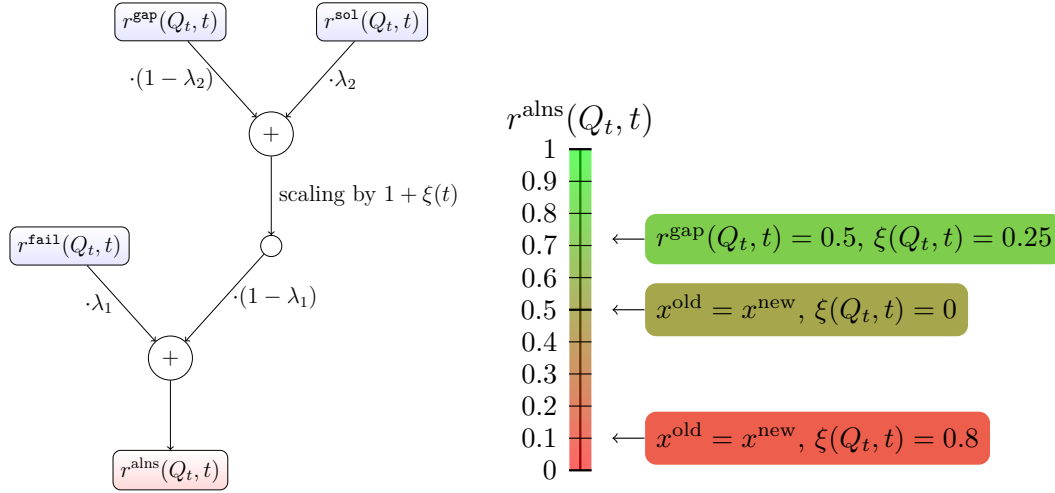


Figure 6.1: Left: Diagram of the proposed reward function. Right: Three hypothetical outcomes and their reward on the $[0, 1]$ -scale.

The left part of Figure 6.1 depicts the individual elements of the ALNS reward function visually. The right part of the figure illustrates three reward examples of hypothetical outcomes after an auxiliary problem has been solved. Starting from the bottom, assume the measured effort (6.5) was 0.8, for example because 80% of the integer variables were fixed and the entire node budget was exhausted. Since no new solution has been found ($x^{\text{old}} = x^{\text{new}}$), the two other rewards $r^{\text{sol}}(Q_t, t), r^{\text{gap}}(Q_t, t)$ are both zero, which yields a reward of $r^{\text{alns}}(Q_t, t) = 0.1$. The middle example does not contribute a new solution, but achieves an effort of 0 and was therefore much faster than the first example. An effort of zero can only be attained if the auxiliary problem could be solved within 0 nodes, i.e. during presolving. Such a case most likely occurs when the auxiliary problem is proven infeasible because of the improvement neighborhood 6.2, which restricts the search space to solutions that improve the primal bound by at least $0 < \delta < 1$. Note that this infeasibility does not mean that no solution for the original MIP P within the improvement neighborhood exists, it only means that the targeted objective improvement cannot be achieved by the fixings in the current auxiliary problem. Therefore, the adaptive fixing rate will be lowered to broaden the search space for the next round in which Q_t is selected again. This outcome receives a reward of 0.5, which is best possible for rounds of ALNS that do not contribute a new solution.

The last hypothetical outcome contributes a new incumbent solution which closes the gap by 50% and therefore achieves a gap reward of 0.5. Every round with a new incumbent solution automatically achieves a solution reward of 1, which has been omitted from the figure for a better readability. In combination with an effort of 0.25, this round achieves a reward of 0.72.

6.3 Selection Strategies for Multi-Armed Bandit Problems

The goal of the present work is a framework that selects among a set of auxiliary problems in Section 6.1 such as to maximize their utility under a shared computing budget. Such a sequential decision process from a finite set of actions (auxiliary problems) with unknown outcome appears in the literature as *Multi-Armed Bandit Problem* [Bubeck and Cesa-Bianchi, 2012].

The basic multi-armed bandit problem can be described as a game, which is played over multiple rounds. In every round $t = 1, 2, \dots$, the player chooses one action $Q_t \in \mathcal{Q}$ from a finite set of available actions. In return for playing Q_t , the player observes a *reward* $r(Q_t, t) \in [0, 1]$ for the selected action. The aim for the player is to maximize their total revenue $\sum_t r(Q_t, t)$. Since only the reward of the selected action can be observed at a time, every suitable algorithmic strategy must find a good balance between exploration across all actions and exploitation of the best action seen so far.

Let $T_Q(t) := \sum_{i=1}^t \mathbb{1}_{\{Q_i=Q\}}$ denote the number of times that action Q has been selected until round t . The *average reward* of Q is

$$\bar{r}_Q(t) := \frac{1}{T_Q(t)} \sum_{i=1}^t \mathbb{1}_{\{Q_i=Q\}} r(Q, i),$$

where we define $\bar{r}_Q(t) = 0$ as long as $T_Q(t) = 0$. The selection strategies below ensure that during the first rounds, all reward averages are meaningfully initialized by playing each action once in randomized order.

Algorithm 3: ε -greedy [Sutton and Barto, 2018]

Input: Set of actions \mathcal{Q} , parameter $\varepsilon \geq 0$

```

1  $t \leftarrow 0$ 
2 while not stopped do
3    $t \leftarrow t + 1$ 
4    $\varepsilon_t \leftarrow \varepsilon \cdot \sqrt{\frac{|\mathcal{Q}|}{t}}$ 
5   Draw  $e_t \sim \mathbb{U}([0, 1])$  /* drawn from uniform distribution */
6   if  $e_t \leq \varepsilon_t$  /* Selection of next action */
7     then
8       | Draw  $Q_t \sim \mathbb{U}(\mathcal{Q})$ 
9     else
10      |  $Q_t \leftarrow \operatorname{argmax}_{Q \in \mathcal{Q}} \bar{r}_Q(t - 1)$ 
11    end
12    Update  $\bar{r}_{Q_t}(t)$  by the observed reward  $r(Q_t, t)$ 
13 end
```

6. Adaptive Large Neighborhood Search

Algorithm 3 [Sutton and Barto, 2018] is a very simple, randomized selection strategy for the multi-armed bandit problem. It uses the short notation $\mathbb{U}(X)$ to denote the *uniform distribution* over a set X . The initial lack of reward information is compensated by a randomized selection of the first few actions. The amount of random selections decreases at the speed of $\frac{1}{\sqrt{t}}$ and can be controlled by the input parameter ϵ . With increasing t , it therefore becomes less and less likely to choose an action at random, whereas the probability of greedily exploiting the best action increases.

A different, more deterministic approach [Auer et al., 2002] uses *upper confidence bounds* (UCB) based on the principle of optimism at the face of uncertainty. Assume that \mathcal{Q} is an ordered $|\mathcal{Q}|$ -uple $(Q_1, Q_2, \dots, Q_{|\mathcal{Q}|})$. The selection strategy α -UCB selects

$$Q_t = \begin{cases} \operatorname{argmax}_{Q \in \mathcal{Q}} \left\{ \bar{r}_Q(t-1) + \sqrt{\frac{\alpha \ln(1+t)}{T_Q(t-1)}} \right\} & \text{if } t > |\mathcal{Q}|, \\ Q_t & \text{if } t \leq |\mathcal{Q}|. \end{cases} \quad (6.7)$$

With the goal to ultimately find the action Q^* with maximum expected reward, the UCB algorithm selects the action that maximizes the sum of the average reward observed so far and its associated *confidence bound*, which depends on the number of times that Q has been selected in proportion to the (logarithmic) overall number of rounds. The rationale behind this is that also inferior actions become more attractive to the algorithm after they have not been selected for a while.

The case distinction in Equation 6.7 is necessary to obtain a meaningful initialization of all sample means and because $T_Q(|\mathcal{Q}|) \geq 1$ for all $Q \in \mathcal{Q}$ is required for the confidence bound in Equation 6.7 to be well-defined. The width of the confidence bound around the average reward is further controlled by a parameter $\alpha \geq 0$. The special case of $\alpha = 0$ yields a completely greedy exploration strategy that does not take into account the upper confidence bound. In the first case of Equation 6.7, eventual occurring ties are broken uniformly at random.

A visual impression of the influence of the parameter α in the α -UCB equation (6.7) is given in Figure 6.2. Assume there are only two actions $\mathcal{Q} = (Q_1, Q_2)$ available, both of which return a constant reward every time they are played, $r(Q_1, t) = \mu_1$ and $r(Q_2, t) = \mu_2$. We assume that $\mu_1 > \mu_2$ such that after the two required rounds to initialize the average reward of each available action, Q_1 will be selected in round 3 because of its better average reward $\bar{r}_{Q_1}(2) = \mu_1$. The question is after how many rounds the weaker action Q_2 is played by α -UCB for the second time, which happens when its UCB score exceeds the UCB score of Q_1 . As already mentioned, for a value of $\alpha = 0$, α -UCB continues to play Q_1 without considering Q_2 again. For positive values of α , Q_2 will be reconsidered depending on the reward difference $\Delta := \mu_1 - \mu_2$. We denote by $T(\Delta, \alpha)$ the round in which Q_2 will be played the second time. Figure 6.2 shows how this function looks like for different values of Δ between 0 and 1 at three distinct values $\alpha \in \{0.01, 0.1, 1\}$. The y -axis uses a logarithmic scale. Values of $T(\cdot, \cdot)$ that exceed 10^6

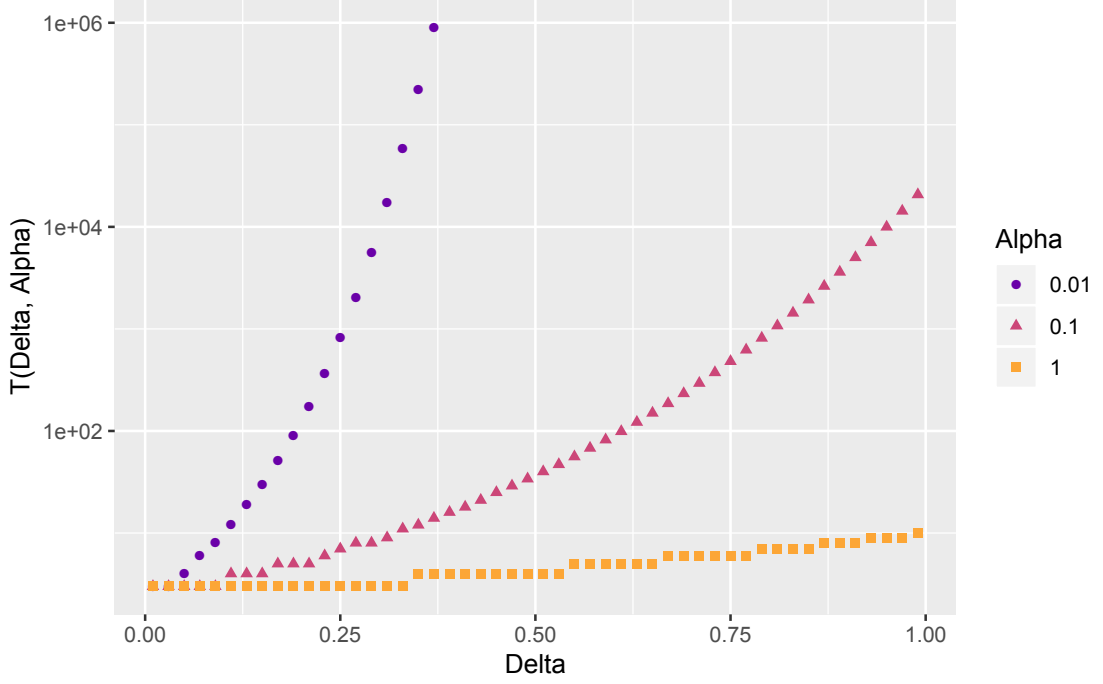


Figure 6.2: Rounds until α -UCB reconsiders the weaker of two actions as a function of the reward difference Δ and the confidence width α .

have been removed. In general, all three curves are increasing with increasing reward difference Δ . At the smallest $\alpha = 0.01$, Q_2 will be selected within the first 100 rounds only if the reward difference Δ is smaller than 0.2, whereas for reward differences larger than 0.35, Q_2 will not be selected within the first 1 million rounds. At a value of $\alpha = 1$, Q_2 will be selected again within the first ten rounds even for $\Delta = 0.99$. In the LNS setting, the situation is more complicated than in this example since we have eight actions to select from and expect nonconstant rewards.

Algorithm 4 [Auer et al., 2003] is a third approach for the multi-armed bandit problem. It is briefly called *Exp.3*, which is an abbreviation of “Exponential Weight Algorithm for Exploration and Exploitation”. In each round t , the next action is selected randomly from a probability distribution defined by marginal probabilities (weights) $p_{Q,t}$ for each $Q \in \mathcal{Q}$. After observing the reward $r(Q_t, t)$, the weight update is performed in two steps. First, the cumulative reward R_{Q_t} of the selected action Q_t is updated in line 7. The cumulative reward divides the observed reward by the probability to select Q_t , thereby emphasizing actions with a high reward compared to their current selection probability. Second, the probabilities for the next iteration $t + 1$ are computed as a convex combination of two probability distributions based on the choice of $\gamma \in [0, 1]$. In the two extreme cases, the algorithm either draws from a uniform distribution ($\gamma = 1$) in the next iteration, or from a distribution defined over the cumulative rewards ($\gamma = 0$), using a softmax normalization. This normalization assigns the largest weights to actions with high cumulative reward, while the probabilities on actions with low cumulative reward vanish fast.

6. Adaptive Large Neighborhood Search

Algorithm 4: Exp.3

Input: Set of actions \mathcal{Q} , convex combination parameter $\gamma \in [0, 1]$

```

1  $p_{Q,1} \leftarrow \frac{1}{|\mathcal{Q}|}$ ,  $R_Q \leftarrow 0$  for all  $Q \in \mathcal{Q}$ 
2  $t \leftarrow 0$ 
3 while not stopped do
4    $t \leftarrow t + 1$ 
5   Draw  $Q_t$  according to probability distribution  $p_{Q,t}$ 
6   Observe reward  $r(Q_t, t)$ 
7    $R_{Q_t} \leftarrow R_{Q_t} + \frac{r(Q_t, t)}{p_{Q_t, t}}$ 
8   foreach  $Q \in \mathcal{Q}$  do
9      $p_{Q, t+1} \leftarrow (1 - \gamma) \frac{\exp(R_Q)}{\sum_{Q' \in \mathcal{Q}} \exp(R_{Q'})} + \frac{\gamma}{|\mathcal{Q}|}$ 
10  end
11 end

```

Remarks Depending on the nature of the reward distribution, two main scenarios of multi-armed bandit problems are distinguished, see, e.g., [Bubeck and Cesa-Bianchi, 2012]. In the *stochastic scenario*, the observable rewards $r(Q, t)$ for every action $Q \in \mathcal{Q}$ are independent, identically distributed (i.i.d.) random draws over time from a probability distribution with unknown *expected reward* $\mu_Q \in [0, 1]$. In the stochastic scenario, a good strategy should play an action Q^* with maximum expected reward $\mu_{Q^*} = \max_{Q' \in \mathcal{Q}} \mu_{Q'}$ as often as possible.

In the *adversarial scenario*, the player faces an opponent that chooses the rewards with the goal to maximize the player's *regret*—the discrepancy between the player's reward and the best possible reward. The opponent may take into account all choices previously made by the player, but does not know the selected action at time t . After the player and the opponent have each made their decisions, the player receives the reward $r(Q_t, t)$ for the selected action only, while the opponent is informed about the player's choice Q_t . It is noteworthy that in the adversarial scenario, the opponent has an incentive to play rewards different from 0 in every round of the game because the player's regret is minimal in every round t where all actions have a reward of 0. For a player in the adversarial scenario, a good strategy must necessarily be randomized in some way because every deterministic algorithm is easily fooled by the opponent, who can minimize the player's total reward by assigning a reward of 0 to the player's deterministic next action, and 1 to all other actions.

Intuitively, the adversarial scenario seems much harder to approach than the stochastic scenario because the latter is indifferent to choices made by the player, and estimates of the expected rewards can be built over time. It turns out that it is possible, even for the adversarial scenario, to create strategies that yield an asymptotically optimal reward in their respective scenario. While the ε -greedy and α -UCB strategies can be made asymptotically optimal for the stochastic scenario, the Exp.3 selection

strategy and its variants are a state-of-the-art strategy for the adversarial scenario. The reader is referred to the survey [Bubeck and Cesa-Bianchi, 2012] for more information about and variants of the discussed selection strategies.

Both ε -greedy and α -UCB address the stochastic scenario, in which the distribution of rewards is fixed across all rounds. This assumption is violated for the proposed reward function (6.6) for LNS auxiliary problems because some ALNS rounds may be executed after an optimal solution has already been found, such that no auxiliary problem can contribute an improving solution and receive a reward of 0.5 or higher anymore. But even after an optimal solution has been found, the proposed reward function prefers quicker fails, preferably detected during the presolving of the auxiliary problem, over actions that consume a lot of resources that may be invested in improving the dual bound at this stage of the main search. Therefore, at each stage of the search, we seek to maximize the reward of the selected actions, although the potential payoff may change over time. An LNS auxiliary problem that was not successful at its first attempt may become useful later during the search, if initialized from a different reference solution.

In particular α -UCB and Exp.3 try to choose inferior actions from time to time, which is desirable in the context of MIP primal heuristics to diversify the search. The advantage of α -UCB in this respect is its explainability. In contrast to Exp.3, α -UCB has a deterministic explanation, the UCB score itself, why it prefers which action in each round.

It should be noted that the Exp.3 algorithm presented here is a classical variant. The parameter γ is necessary to explore each action sufficiently often to prove some regret bounds. Recently, Neu [2015] introduced an Exp.3 variant based on *implicit exploration* that does not require this uniform distribution controlled via γ .

6.4 Computational Results

The proposed ALNS framework has been implemented and tested as an additional plugin on top of SCIP 5.0, using CPLEX 12.7.1 as the underlying LP solver. All 8 auxiliary problems listed in Table 6.1 and their corresponding neighborhoods have been incorporated into ALNS. As instance set, we use the union of three MIPLIB collections 3.0, 2003, and 2010 [Achterberg et al., 2006; Bixby et al., 1998; Koch et al., 2011], and the COR@L [Coral] instance set, totaling to 666 instances. The computational experiments for the present work are split into two parts. The first part is an offline simulation that uses reward information about all auxiliary problems in each call to ALNS. This information is used to compare the performance of auxiliary problems, and to calibrate the parameters of the bandit selection strategies from Section 6.3. Section 6.4.3 describes the results that we obtained with the ALNS framework inside of SCIP using the readily calibrated α -UCB selection strategy. Since a lot of parameters have

6. Adaptive Large Neighborhood Search

Symbol	SCIP parameter(s)	Section Ref.	Simulation	MIP
δ	minimprov{low,high}	6.1	0.01	0.01
k	crossover/nsols	6.1.1	2	2
d^{\max}	not parameterized	6.1.2	$0.2 \cdot n^{\text{bin}}$	$0.2 \cdot n^{\text{bin}}$
$ Y^{\text{DINS}} $	dins/npoolsols	6.1.2	5	5
$\phi_{Q_t,t}$	*/{min,max}fixingrate	6.2.1	$\{0.1, 0.3, \dots, 0.9\}$	dynamic in $[0.3, 0.9]$
ν_1^{lim}	minnodes	6.2.2	50	50
κ_0	nodesofs	6.2.2	500	500
κ_1	nodesquot	6.2.2	0.1	0.5
λ_1	rewardbaseline	6.2.3	0.5	0.5
λ_2	rewardcontrol	6.2.3	0.8	0.8
ϵ	epsilon	6.3	$\in [0, 4]$	–
α	alpha	6.3	$\in [0, 1]$	0.0016
γ	gamma	6.3	$\in [0, 1]$	–
–	banditalgo	6.3	$\{\epsilon\text{-greedy, UCB, Exp.3}\}$	UCB

Table 6.2: Overview of involved parameters and values for the simulation and the MIP experiments. All SCIP parameters are preceded by `heuristics/alns/`. The placeholder `*` must be substituted by the name of a neighborhood, e.g., `rens`.

been introduced in the previous sections, Table 6.2 summarizes the parameter settings used for the simulation and the performance experiments in this section.

6.4.1 Auxiliary Problem Comparison

The first part aims at providing a fair comparison between the auxiliary problems in the ALNS framework. Instead of choosing a single auxiliary problem, all of them are executed one after another at each call to ALNS, and their individual rewards are recorded. In order to ensure fairness, every found improving solution is only used to compute the reward function. However, SCIP does not store these solutions as they could potentially impact the neighborhoods of the subsequent auxiliary problems at this call. All dynamic decisions from Section 6.2 are deactivated for this experiment. The target fixing rate is kept fixed at $\{0.1, \dots, 0.9\}$ in steps of 0.2, with a tolerance of ± 0.1 . Also, the stall node limit is kept fixed at 50 nodes. The budget computation (6.4) skips the dynamic adjustment based on the number of solutions that ALNS found, such that ALNS is executed more statically according to its frequency schedule as soon as the budget computation allows another run. Recall that the additional generic fixings/unfixings are only applied if the obtained fixing rate lies outside the tolerance interval. The experiments have been conducted on a Linux cluster using Ubuntu 16.04, with a time limit of 5h for each instance. All runs are single threaded.

Not all neighborhoods are applicable to all MIP instances. For example, Local Branching and Proximity require instances with binary variables. Zero Objective requires a nonzero objective function. For this simulation experiment, we focus on those instances with binary variables and nonzero objective function such that all neighborhoods are applicable. Furthermore, on an instance which meets those requirements, Crossover

Fixing rate	Total across instances			Instances with ... rounds			
	Rounds	Success	Rate	> 10	> 20	> 30	> 40
0.1	9037	841	0.093	227	169	124	93
0.3	9233	975	0.106	234	172	131	92
0.5	9789	1005	0.103	237	182	140	103
0.7	9925	1196	0.121	241	189	138	97
0.9	10085	1337	0.133	246	187	142	101

Table 6.3: Number of (successful) rounds of ALNS on 494 instances.

requires at least 2 available solutions. RENS and RINS require a feasible LP relaxation at the local node. We wait until enough solutions have been found during the main search before ALNS is executed. Recall from Section 6.2.1 that RENS is special in that generic fixing cannot be applied to RENS because its neighborhood relies solely on the LP solution at the current node, but no feasible reference solution is involved. Whenever RENS does not attain its targeted fixing rate during the simulation, RENS obtains a reward of 0.

In total, our data set comprises 48k records over 494 instances at five tested fixing rates. Each data record contains rewards of all eight auxiliary problems as well as information about the round and instance where it has been recorded. Table 6.3 shows the number of ALNS rounds for every tested fixing rate, where each round consists in searching all eight available auxiliary problems once. The rounds range from 9037 at a fixing rate of 0.1 to 10085 at a fixing rate of 0.9. The number of executed rounds is different for every fixing rate because the auxiliary problems become simpler with increasing fixing rate, such that more rounds of ALNS can be executed during the search. The total number of rounds where at least one of the tested auxiliary problems contributes a solution is shown in the column “Success” and the corresponding proportion in column “Rate”. Across the tested fixing rates, the success rate ranges from 9.3% to 13.3% and increases with the fixing rate. In a round in which none of the auxiliary problems contribute a solution, the selection process is only required to select an auxiliary problem that fails fast, but cannot contribute to the overall search process with an incumbent solution. Therefore, we will report all simulation results on the data set restricted to the successful rounds shown in the column “Success”.

Furthermore, Table 6.3 also shows the numbers of instances for which more than 10, 20, 30, and 40 ALNS rounds were executed during the data collection. For example, more than 90 instances admit at least 40 rounds of ALNS. A certain number of rounds is necessary for the bandit selection strategies. The selection strategy α -UCB, for example, tries every action once during the first eight rounds to initialize the average reward. This α -UCB initialization phase is completed for more than 220 instances for all different fixing rates as shown in the table. On average, ALNS was executed between 18.3 and 20.4 times per instance, depending on the fixing rate.

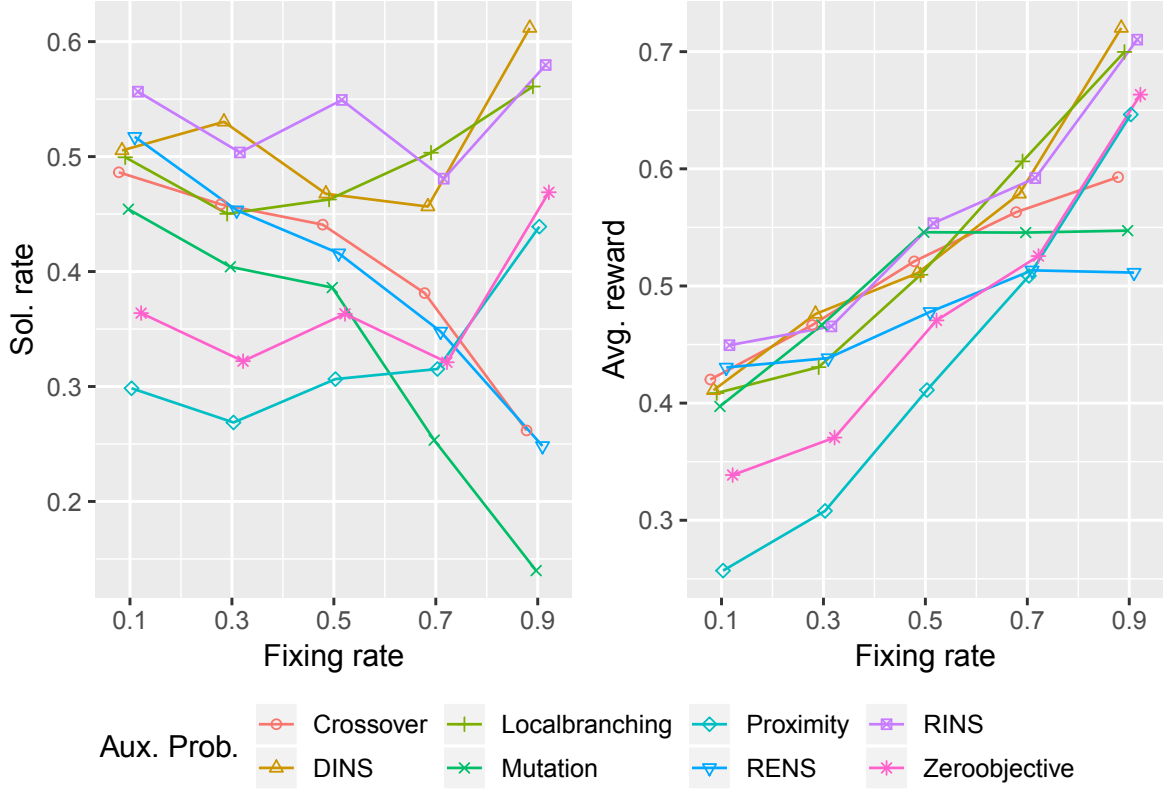


Figure 6.3: Solution rates (left) and average rewards (right) at different fixing rates.

The left part of Figure 6.3 shows the average solution rate of each auxiliary problem at the different tested fixing rates. The solution rate is the fraction of successful executions of an auxiliary problem/selection strategy. In this section, we show the solution rate as additional measure of the quality. When we compare auxiliary problems across fixing rates, the solution rate does not depend on the fixing rate like the reward (6.6). For each fixing rate, we compute the solution rate and average rewards on the subset of records on which at least one of the possible auxiliary problems finds a solution, as shown in the column “Success” of Table 6.3. At the smallest fixing rate of 0.1, RINS has the highest solution rate of approximately 0.56, which means that RINS contributes a solution in 56 % of the 841 cases at this fixing rate in which any auxiliary problem is successfully applied.

In Figure 6.3, we try to detect trends for individual auxiliary problems when the fixing rate is varied. At the same time, these charts allow for comparisons between different LNS techniques. For example, it can be observed that RINS, DINS, and Local Branching are almost consistently the top three methods across all tested fixing rates. All three achieve their highest solution rate at the highest tested fixing rate of 0.9, where DINS has the highest solution rate of 0.62 across all tested techniques and fixing rates. In contrast, the depicted solution rates of Crossover, RENS, and Mutation clearly exhibit a decreasing trend towards higher fixing rates.

The ranking between the auxiliary problems is similar regarding the obtained average rewards shown in the right part of Figure 6.3. As before, we restrict ourselves to the rounds

counted as “Success” in Table 6.3. A higher average reward results from an increased solution frequency, a better solution quality, and/or less effort to solve the corresponding auxiliary problems. In the figure, the average rewards of most auxiliary problems clearly increase with the fixing rate. This is partly because the reward definition (6.6) penalizes neighborhoods of auxiliary problems with a high fixing rate less strictly. At a particular fixing rate, the different rewards can be compared well.

RINS, Local Branching, and DINS also achieve high average rewards. The highest increase in average reward can be observed for Proximity and Zero Objective. At a high fixing rate of 0.9, RENS achieves the smallest average reward. The reward for Mutation only increases up to a fixing rate of 50%. Its reward is almost constant for all fixing rates $\geq 50\%$. RENS lacks a reference solution for additional, generic fixings, which is why it can run less frequently than others. A possible explanation for the decreasing solution rate of Crossover is the random selection of reference solutions. Searching a narrow auxiliary problem around a reference solution far away from the incumbent may lower its chances to find a better solution. The lower solution rates of Mutation are remarkable because RINS and Mutation use the same reference solution, namely the incumbent. RINS may even need additional generic fixings to reach higher target fixing rates, whereas the Mutation scheme always fixes the targeted percentage of integer variables. The large discrepancy in their solution rates indicates that more informed approaches such as the LP driven neighborhoods of RINS or DINS are the most important fixing schemes.

One may ask whether a well-performing auxiliary problem such as RINS entirely dominates the less performant auxiliary problems such as Mutation. Figure 6.4 illustrates the measured rewards for RINS and Mutation in a histogram, which shows the distribution of their reward difference $r^{\text{alns}}(Q_{\text{RINS}}, t) - r^{\text{alns}}(Q_{\text{Mutation}}, t)$ regardless of the fixing rate at which these rewards were recorded. For consistency, only rounds are shown that are marked as “Success” in Table 6.3. If the reward difference is positive, RINS achieves a higher reward than Mutation. RINS has a clear tendency to score higher. However, also the execution of Mutation can be beneficial. Mutation reaches a higher reward in about 30% of the cases. Analogous comparisons for other pairs of neighborhoods yield similar results. Based on these observations, it is reasonable to enable all available auxiliary problems by default, and to rely on the selection mechanism.

6.4.2 Simulation of the Selection Process

The data set from the previous section is now used for an offline calibration of the three bandit selection strategies from Section 6.3. Recall that each of the three bandit selection methods has a single parameter that can be calibrated for the use inside the ALNS selection process. The parameter $\varepsilon \geq 0$ of the ε -greedy strategy controls how long the selection strategy selects uniformly among the auxiliary problems before

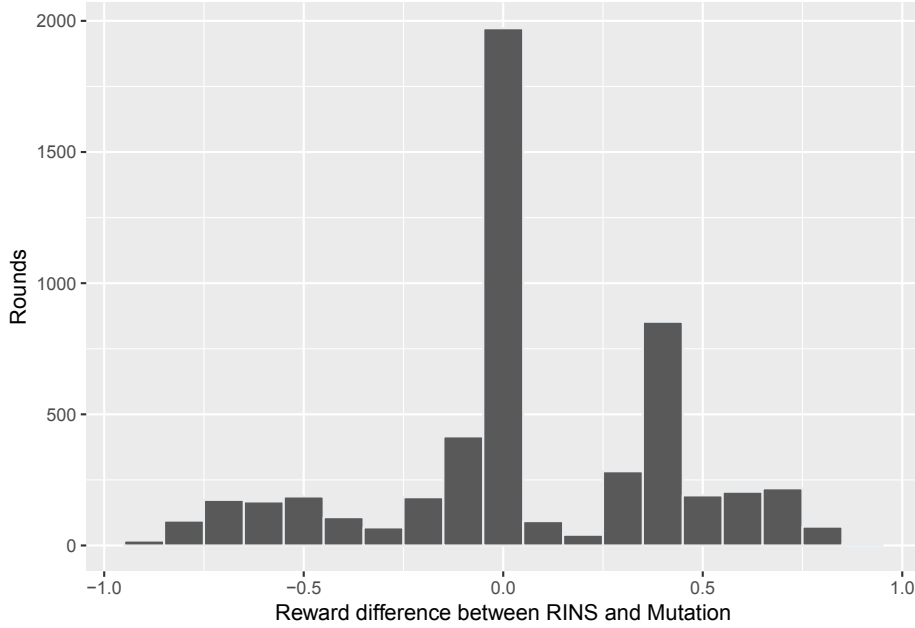


Figure 6.4: Reward comparison of RINS and Mutation.

transitioning into a greedy selection based on the largest average reward observed. The $\alpha \geq 0$ parameter controls the width of the confidence band around the observed average rewards in α -UCB (6.7). Recall that a larger value of α forces α -UCB to select actions with inferior average reward more frequently. Finally, the parameter $0 \leq \gamma \leq 1$ controls the mass of the uniform distribution in the mixed probability density from which Exp.3 makes its selection. All three α -UCB, Exp.3, and ε -greedy are calibrated on the entire data set of 48000 rounds, i.e., including those rounds in which no auxiliary problem contributes a solution.

Since each selection strategy involves some randomized choices, average rewards are computed over 100 repetitions of the experiment. This simulation of the selection routines has been implemented in the programming language R. For the calibration, we call the R function `optimize` and obtain optimal values of $\varepsilon = 0.4685844$, $\alpha = 0.0046$, and $\gamma = 0.07041455$. Ideally, the selection performs better than a pure random selection for instances that allow for a certain number of rounds to initialize the selection process. Note that certain parameter choices of the Exp.3 ($\gamma = 1$) and ε -greedy bandits are equivalent to a uniform random selection.

Figure 6.5 shows the selection quality for each bandit selection strategy in terms of both their solution rate on the left and average reward on the right. While the reward (6.6) is the immediate feedback that the selection strategies receive to adjust their respective ranking of the auxiliary problems and depends in particular on the fixing rate, the solution rate computation is not biased towards higher fixing rates. As in the previous section, the figures in this section summarize only the subset of rounds from the column “Success” in Table 6.3 ranging from 841 to 1337 depending on the fixing rate. This means that the solution rate of an optimal selection strategy would

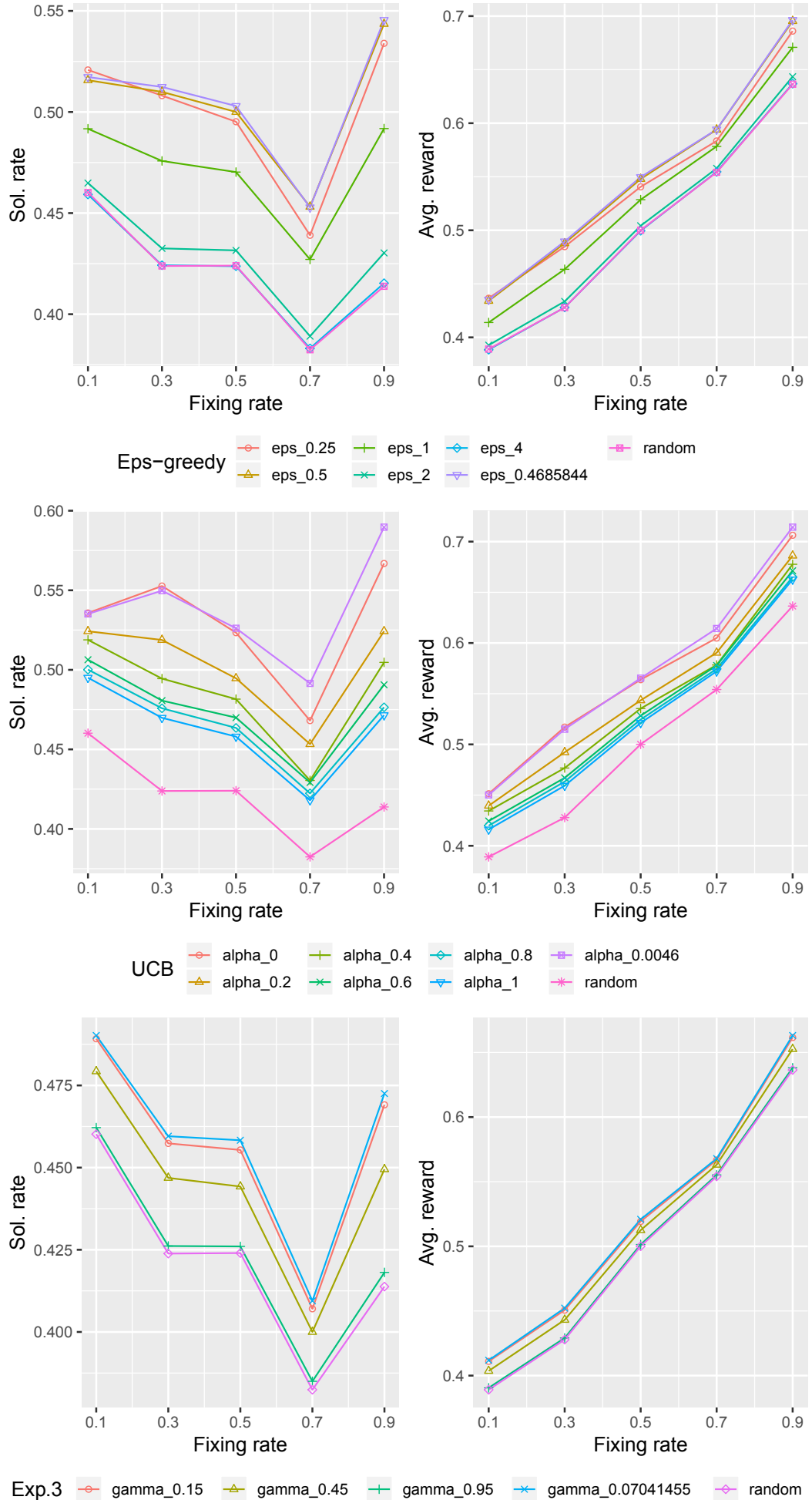


Figure 6.5: Comparison of selection performance for different parameter choices.

be a horizontal line with value 1.0 across all fixing rates. As a reference curve, each of the six plots of Figure 6.5 shows the expected solution rate/reward of a completely randomized selection strategy “random”. This reference curve is computed as the sample average across all eight measured solution rates/rewards at each round. Therefore, it represents the expected solution rate of a uniform random selection strategy.

The first part of the figure shows the solution rate (left) and average reward (right) at every tested fixing rate for the ε -greedy selection strategy shown in Algorithm 3. The best choice for ε computed by R is 0.4685844, which has the largest average reward across all tested fixing rates. Some other manually selected choices of ε are also shown. Recall from Algorithm 3 that at larger values of ε , the selection strategy tends to uniformly select among the actions more frequently. In particular, as long as the quantity ε_t , which decreases with the number of rounds, is larger than 1, every selection is a uniform random selection. In the figure, the results for $\varepsilon = 4$ are indistinguishable from “random” sampling considering both solution rate and average reward. The reason is the limited number of rounds per instance/fixing rate in our data, which is at most 71. At an initial choice of $\varepsilon = 4$, the quantity ε_t is larger than 1 for all rounds of our simulation data such that only random sampling is applied by the ε -greedy bandit. At all smaller choices of the ε -parameter, ε -greedy always improves upon the reference curve “random”.

In the middle row of Figure 6.5, we show solution rate and average reward of the α -UCB bandit for different choices of the α parameter. The average reward of the α -UCB selection strategy has been maximized for the parameter choice of $\alpha = 0.0046$. With this choice of α , α -UCB achieves the highest solution rate and average reward of all three tested bandit strategies. Other choices of α are detrimental especially with respect to the solution rate compared to the calibrated parameter value, but clearly achieve a better solution rate than the reference curve “random”.

The last two plots depict the selection quality of Exp.3 at different values of the γ parameter. Some hand-picked values $\{0.15, 0.45, 0.95\}$ are compared to $\gamma = 0.07041455$, the optimal value for γ as computed by the R function `optimize`, and the reference line “random”. At all tested values of the γ -parameter, the selection quality of Exp.3 is better than purely randomized selection. Furthermore, the experiment reveals that higher values of γ decrease the selection quality across all tested fixing rates. The choice of $\gamma = 0.95$ shows, as expected, almost the same selection quality as a pure random selection. Note that the average selection quality is higher for α -UCB and ε -greedy than for Exp.3.

The improvements in solution rate and average reward of all bandit selection strategies suggest that it is clearly beneficial to incorporate observed rewards into the selection process. The optimal values for the different parameters can be interpreted as follows. The optimal value for the γ -parameter is very close to a purely weight based Exp.3 selection strategy. The optimal value of the α -parameter shows a higher selection quality than the nearby value of $\alpha = 0$, a purely greedy selection. This is seconded by the optimal

value of the ε -parameter. The “near-greedy” optimal values of all three parameters indicate that it suffices to revisit inferior actions only if the reward difference to the best action is small. This shows that learning from past observations clearly helps the selection process at later stages.

Another observation is that the plots of Figure 6.5 seldomly cross, i.e. the ranking between different parameter choices is the same for different fixing rates. This indicates that the selection strategies can be safely combined with an adaptive fixing rate.

Finally, the learning success of the bandit selection methods is depicted in Figure 6.6, in which we draw the solution rate as a function of the number of rounds within the ALNS framework for each selection strategy. Each bandit selection strategy uses its optimized parameter value. As a comparison serves the strategy “random”, which represents, as before, the expected solution rate of a uniformly randomized selection strategy at each round over the entire duration of the search. In order to aid the visual distinction between the solution rates of the strategies, the figure also shows a straight line as the result of a linear regression between round and solution rate. Throughout all rounds, the solution rate of the reference strategy “random” stays relatively constant around 0.3, whereas the solution rate of each bandit selection strategy shows an increasing trend. The ε -greedy selection strategy shown uses the optimized value of ε . Its margin from the baseline solution rate is already visible at rounds 6–8, and keeps improving. As an example, the (arbitrary) mark of a solution rate of 0.6 is first reached after 24 rounds, and reliably surpassed after 40 rounds to the selection routine, as can be seen by the corresponding regression line. The solution rate of the calibrated α -UCB selection strategy reaches the mark of 0.6 after 17 rounds for the first time, and almost consistently after 30 rounds. The regression line of α -UCB is clearly the highest across all strategies. The price for this selection performance is that the first 8 observations must be spread over the 8 auxiliary problems to select from, which is why α -UCB achieves exactly average performance at this early stage. At a later stage, α -UCB reaches a solution rate of 1.0 for the four rightmost observations, i.e., α -UCB can reliably identify and select a well performing auxiliary problem at this stage. Recall that these plots represent average solution rates over 100 repetitions of the experiment. Also for Exp.3, the solution rate for the choice of $\gamma = 0.07041455$ is better than the reference curve “random” after a small number of rounds and has a clear tendency to increase with the number of rounds. Still, Exp.3 is clearly the weakest of the three bandit selection strategies even with an optimized choice of its γ -parameter.

As a conclusion, all three bandit selection algorithms achieve an above average selection performance, as desired. With an increasing initialization time, the learning effect becomes more pronounced. α -UCB achieves the best solution rate, followed by ε -greedy and Exp.3. Arguably, the good solution rate is an indication that the designed reward function, which the bandits actually receive as feedback, captures the ranking between the neighborhoods sufficiently well within the ALNS framework.

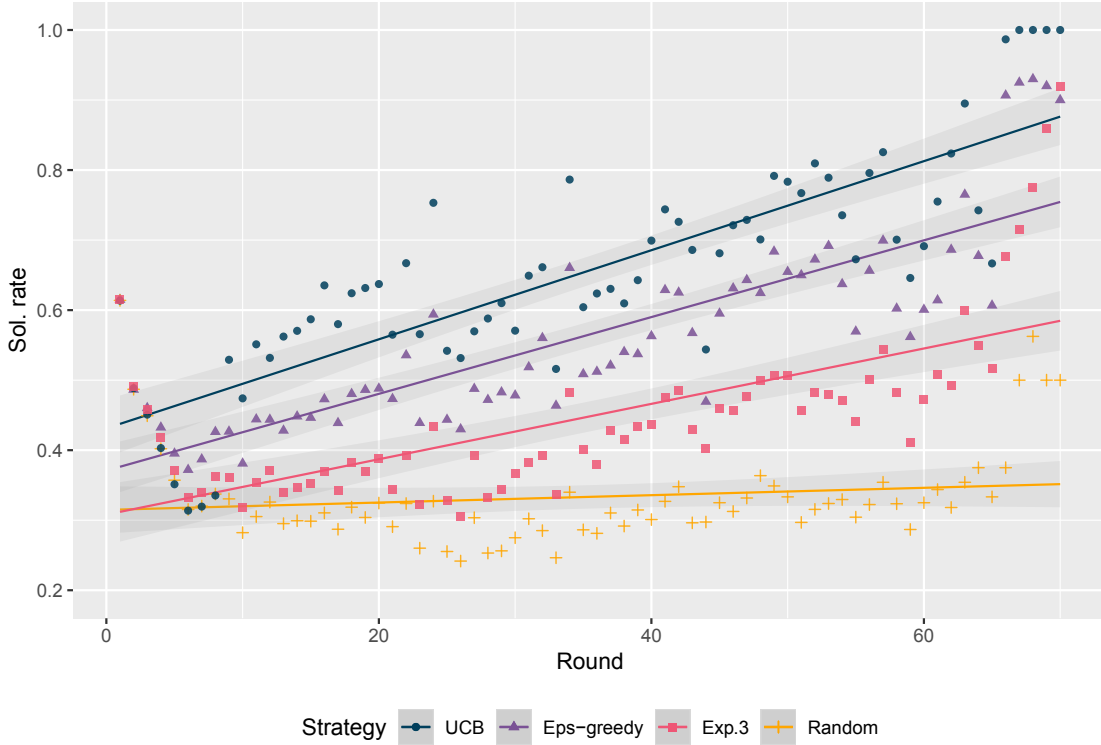


Figure 6.6: Average solution rate as a function of the individual round.

6.4.3 MIP Performance

This section examines the impact of ALNS on MIPLIB 2017 in a real setting where only one auxiliary problem is called at each round. The benchmark set MIPLIB 2017 consists of 240 instances in total and 150 instances that were not part of any of the four existing sets that we used for calibration of the bandit selection strategies.

For the results in this section, we test the newest version of SCIP by the time of this writing, SCIP 7.0.2 [Gamrath et al., 2020], in which ALNS is active by default. We made a couple of minor modifications to the released version of the code:

- We generalize the RENS neighborhood 6.3. As mentioned in Section 6.2.1, RENS is the only neighborhood without a reference solution for generic variable fixings. If RENS does not reach its (tight) target fixing rate of 90 % at its first call, it will be penalized with a reward of 0, which seemed unfair. We introduced fractionality-based fixing for the RENS neighborhood, such that RENS continues to fix variables to their (rounded) LP solution value in the order of least fractionality until it reaches its target fixing rate.
- We implemented a multiple root initialization: In SCIP 7.0.2, ALNS is only called (at most) once at the end of the root node. With the goal of initializing more than one neighborhood early during the search, we allow for multiple calls of ALNS during the root node processing of SCIP.

- We use local reduced costs and pseudo-costs for generic variable fixing: Instead of reduced costs and pseudo-costs relative to the root LP solution, we use local LP solutions (at the nodes where ALNS is called) for generic variable fixings with the goal to diversify the search neighborhoods.
- We modified the computational budget of ALNS: Currently, the budget computation of ALNS (and other LNS heuristics in SCIP) aims at calling the heuristic less and less frequently if it does not find improving solutions. For ALNS, however, the number of target nodes for the auxiliary problems are increased each time the search of an auxiliary problem stalled without finding an improving solution, as explained in Section 6.2.2. These two effects combined led to a very infrequent call strategy for ALNS. In our revised implementation, we remedy this by introducing a lower threshold on the relative number of branch-and-bound nodes that ALNS may spend regardless of its success. In the budget computation (6.4), the ALNS node budget relative to the number of nodes in the main search tree is computed as $\frac{s(t-1)+1}{(t-1)+1} \cdot \kappa_1$, which can converge to zero. In our revised implementation, we use a lower threshold of 0.1 on the above factor, which is equal to the value of κ_1 used for the simulation. As a result, we still use an adaptive budget allocation for ALNS as described in (6.4), but allow at least 10 % of the main search nodes to be additionally spent inside ALNS.
- We introduce a maximum number of calls to ALNS on the same incumbent solution, which we set equal to the number of neighborhoods that are active on an instance. A neighborhood is inactive if the structural information of the MIP (binary variables and/or nonzero objective function) does not admit the neighborhood's fixing scheme. After ALNS has been called with the same incumbent solution (including no solution) once for each of its active neighborhoods, it will immediately return and wait for new input before it attempts to solve the next auxiliary problem.

The simulation in the previous section revealed that α -UCB (6.7) with a suitable choice of its selection parameter α is the best performing bandit selection strategy among the three tested strategies from Section 6.3. Recall that a larger α results in a higher frequency of choosing inferior actions. Besides the good simulation performance, α -UCB also provides the easiest explanation, namely the UCB value itself (6.7), why an auxiliary problem has been selected.

All parameters values (and their SCIP names) for this experiment can be found in Table 6.2. Besides the use of α -UCB as a selection strategy, generic (un-)fixing using the variable prioritization from Section 6.2.1 is enabled, and a dynamic target fixing rate is used. All auxiliary problems have 0.9 as initial, conservative fixing rate, which is only reduced if the auxiliary problem was too easy or too restrictive (if proven infeasible) as explained in Section 6.2.2. The stall node limit is dynamically adjusted as explained in Section 6.2.2, and the budget computation (6.4) considers the number of found solutions

by ALNS, such that the execution schedule is more dynamic than during the simulation in the previous sections.

We compare against the setting **ALNS off**, which uses default settings of SCIP 7.0, but deactivates ALNS.

The existing standalone LNS heuristics RENS, RINS, and Crossover are active independently from ALNS also in the **ALNS** setting, as this combination of ALNS and the standalone LNS heuristics represents the default settings of SCIP 7.0. Standalone RENS is only used at the (end of the) root node of the search. Preliminary experiments have shown that deactivating this single RENS call at the end of the root is detrimental to performance.

ALNS using α -UCB has two places where randomized decisions are used. The first random decision concerns the selection process during the first eight rounds of ALNS where α -UCB tests one previously unseen action per round in a randomized initial order. The second random decision is the use of a randomized score as last tie-breaker in the variable fixing prioritization.

We test both settings using the default plus two nondefault initial random seeds for SCIP to better cope with the huge performance variability that some instances may exhibit. The experiment in this section has been conducted on a Linux cluster of 48 computing nodes equipped with Intel Xeon Gold 5122 at 3.60GHz and 96 GB. The time limit was 1 hour for every instance and seed. In order to measure time as accurately as possible, every job has been scheduled exclusively. As before, all jobs are single threaded.

Table 6.4 shows aggregated results for three performance measures, the solving time to optimality in column **Time** and the primal integral **Integral**, as well as the total number of solved instance/seed combinations. We treat every instance/seed combination as an individual record. This table has been prepared using the Interactive Performance Evaluation Tools, see Section 2.9.5, on the raw SCIP log file output. Individual outcomes for every instance and setting are found in Table C.1 in the appendix. The two measures are presented as shifted geometric mean time using a shift of 1 second and shifted geometric mean primal integral [Berthold, 2013] with respect to the known optimal solution values using a shift of 100, which corresponds to a gap of 100 % for 1 second.

For a better quantitative assessment, the table shows the relative performance for **ALNS off** in columns **TimeQ** and **IntegralQ**. Factors larger than 1 in these columns indicate an improvement using ALNS.

Using three random seeds, our benchmark consists of 720 records in total. Table 6.4 summarizes the performance for the entire test bed in the first row (group **All**) as well as several interesting subgroups. The subgroup **Diff** contains all instance/seed pairs for which the two settings have a different solution path. A change in the path is detected by a change in the number of LP iterations of the main solution process. On the complementary group **Equal**, ALNS does not alter the solving process. We call a record *solvable* if it could be solved by at least one setting. Both groups drop all

Group	Instances	ALNS			ALNS off		
		Time	Integral	Solve	TimeQ	IntegralQ	Solve
All	720	781	7814	340	1.01	1.16	344
Diff	195	171	1110	185	1.06	1.37	189
Equal	155	136	2468	155	0.98	0.98	155
Timeouts	370	3600	34018	0	1.00	1.14	0
[0,3600]	350	155	1589	340	1.02	1.18	344
[100,3600]	219	614	3493	209	1.05	1.20	213

Table 6.4: Performance results of ALNS compared with ALNS off. Numbers in **bold** font indicate where one setting was strictly better than the other.

unsolvable records. This is particularly interesting for the results regarding runtime, which is otherwise partially leveled out by records for which all settings time out and hence contribute equally to the shifted geometric mean time.

As a fourth row, we also show the group **Timeouts**. On this group both ALNS and ALNS off timed out, such that only the primal integral can be different between settings.

The last two rows use the standard bracket notation $[x, 3600]$ for $x \in \{0, 100\}$. A bracket $[x, 3600]$ consists of all solvable records where the slower setting required at least x seconds of solving time. The first bracket $[0, 3600]$ therefore consists of all solvable records. The second bracket $[100, 3600]$ consists of 219 harder but still solvable instances.

We first focus on the differences regarding runtime and primal integral. Overall, ALNS achieves a speedup of 1 % and an improvement in primal integral by 17 %.

On the 195 instances from the group **Diff**, ALNS shows a time improvement by 6 % and an improvement in primal integral by 37 %. The group **Equal** is the only group where ALNS off is faster than ALNS, reducing the time and primal integral by 2 %. This is not surprising since for this group, ALNS only causes overhead, but does not contribute to the solution process.

On the bracket groups, we see a time improvement by 2 % for all solvable instances and 5 % for the harder bracket $[100, 3600]$. The primal integral improvements are 18 % and 20 %, respectively.

We see that ALNS achieves substantial integral improvements in all except the **Equal** group. Overall, its overhead on this group is negligible compared to its benefits on the larger group **Diff**.

However, there are still four records solved less with ALNS enabled on the four groups **All**, **Diff**, $[0, 3600]$, and $[100, 3600]$. The number of instances solved exclusively by ALNS is 6, compared to 10 records solved exclusively by ALNS off. In total, the 16 records that were exclusively solved by one of the two settings are split across 14 instances. This is a first indication that ALNS does not introduce a systematic deterioration that could explain the discrepancy in solved instances.

One may think that ALNS has a significant overhead on those instances, but this is actually never the case. There are cases such as the one of `assign1-5-8`, which can be solved by ALNS off eight seconds before the time limit. However, ALNS being called 13 times only spends 0.3 seconds of runtime in this case.

We tested these 14 instances with seven more different random seeds. In this setup, ALNS solves 3 records more than ALNS off. Therefore, we consider performance variability as the main reason that we observe this discrepancy in solved instances. The effects of new solutions found during the root node, for example, can trigger additional separation rounds in SCIP, which may change the LP solution on which the first branching is performed. Such effects cannot be completely avoided in a realistic experimental setup, but are beyond the scope of ALNS.

We conclude that ALNS achieves the main goal of a MIP primal heuristic, namely the improvement of the primal integral, very effectively. On the set of solvable instances, we see a time improvement by 2 %, which is amplified to 5 % on the set of harder instances.

6.5 Summary

This chapter introduces Adaptive Large Neighborhood Search for MIP, a framework around eight well-known LNS heuristics from the literature. It has been implemented as a primal heuristic in SCIP and is publicly available since SCIP 5.0. The framework combines a selection procedure, which is governed by strategies for the multi-armed bandit problem, and the idea of generic additional variable fixings to adjust the complexity of the auxiliary problems as needed. To rank between auxiliary problems, we propose a reward function that combines the important aspects of solution quality and effort into a single number. We have used a simulation experiment to calibrate each bandit algorithm individually. Training the bandit strategies with this reward function shows a clear trend to improve the solution rate with an increasing number of rounds. As a byproduct of this simulation, we saw clear differences between the auxiliary problems regarding the number of solutions they produce. Two of the auxiliary problems that were most successful in our experiments, DINS and Local Branching, have been previously inactive in SCIP. ALNS with an α -UCB bandit selection strategy has been activated by default in addition to the standalone LNS heuristics RENS, RINS, and Crossover since SCIP 5.0. Because of generic variable (un-)fixing, it represents an extension of these powerful standalone LNS heuristics. Besides, previously disabled techniques such as DINS, Local Branching, and Proximity are activated within ALNS for the first time by default, thereby enriching SCIP's default heuristic strategies. Before ALNS, it was not clear how to best integrate five disabled LNS heuristics into the mix. One of the key points is that all techniques within ALNS share a common computational budget, which makes it possible to easily adjust the overall computational budget spent inside LNS heuristics.

This refactoring also has benefits regarding the future maintainability of the code. A callback infrastructure defining the neighborhoods inside ALNS uses 50 % less code than the individual LNS techniques in their standalone plugins, and allows for an easy integration of new techniques into the ALNS heuristic framework.

We see several future perspectives for this work. Adaptive algorithm selection may also be beneficial in other parts of the solver where the choice between similar methods largely affects the overall performance. In the next chapter, we will present two such results for diving heuristics, and for dynamic switching between different pricing strategies of the dual simplex procedure to maximize the node throughput during the search. Second, we hope that the software design of the introduced ALNS framework proves useful as a development platform for incorporating novel LNS-related heuristic ideas and algorithmic enhancements into SCIP more easily in the future. Finally, it would be interesting to see how the bandit selection strategies compete against other selection mechanisms from the ML community. Attempts with popular ML technology may consider additional features of the problem instance or search statistics to train a more informed selection method.

7

Adaptivity Beyond Large Neighborhood Search

In Chapter 6, we introduced an adaptive selection mechanism for large neighborhood search heuristics that is inspired by selection strategies for the multi-armed bandit problem introduced in Section 6.3. In this chapter, we transfer the developed ideas to two further typical algorithmic components within an LP-based branch-and-bound solver. This chapter is an edited version of the proceedings paper on *Adaptive Algorithmic Behavior for Solving Mixed Integer Programs Using Bandit Algorithms* [Hendel, Miltenberger, and Witzig, 2019], which we extended by additional formulae and explanations.

Recall that a substantial amount of the total solving time of the branch-and-bound algorithm is spent on LP (re-)optimizations to obtain both improving solutions and lower bounds on the instance at hand. As explained in Section 2.2, LP relaxations during the search are usually solved via the dual simplex algorithm, for which several pricing methods exist. In this chapter, we consider three well-known and practically proven methods called *deve*x pricing [Harris, 1973], *steepest edge pricing* [Goldfarb and Reid, 1977], and *quick start steepest edge* [Forrest and Goldfarb, 1992]. The concrete choice of a pricing method can make a substantial difference in the overall solving time (Sec. 7.2).

The largest subclass of primal heuristics in SCIP are diving heuristics (see Sec. 7.1). In total, SCIP in version 5.0 features 12 diving heuristics. For such a number of heuristics, it can be a tedious task to tune all their associated control parameters by hand.

For both classes, namely simplex pricing strategies and diving heuristics, it is desirable to "learn" the best performing algorithms during the solving process. To this end, we investigate computational benefits of adaptive algorithmic behavior, governed by algorithms for the multi-armed bandit problem (Sec. 6.3). Like for LNS heuristics in

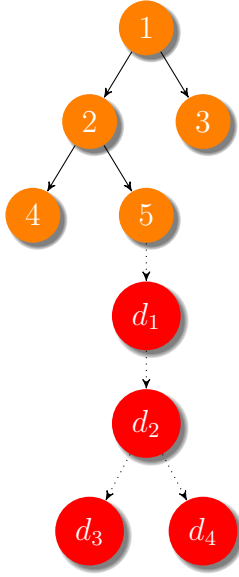


Figure 7.1: Illustration of a diving heuristic in SCIP. The top nodes labeled 1 – 5 are the main branch-and-bound search tree, whereas the nodes d_1, \dots, d_4 represent nodes explored within a diving heuristic. For both node types, the labels represent the order in which the search nodes are explored. Diving heuristics of SCIP explore an auxiliary tree in a depth-first fashion by tightening variable bounds, executing propagation and resolving the associated node LP relaxations until an (in-)feasible leaf is reached. When the diving heuristic finishes, the nodes $d_1 - d_4$ are discarded, and the main search continues. The diving variable and value selections are mainly aimed at providing feasible solutions. Information carried over by diving heuristics consists of variable branching information, conflict clauses from infeasible nodes, and, of course, primal solutions.

the previous chapter, we propose suitable reward functions to measure the success of an algorithm for each class. At the end of each section, we show the individual impact on SCIP with SoPlex as LP solver.

7.1 Diving Heuristics

Diving heuristics are an important class of primal heuristics in modern MIP solvers. Starting from a fractional LP solution, diving heuristics explore an auxiliary search tree in a depth-first fashion. In order to speed up this auxiliary search, some node processing aspects are altered compared to the main branch-and-bound search:

- LP relaxations are not resolved at every diving node, but only after a sufficient amount of integer variable domains has changed through propagation. Most diving heuristics in SCIP use 15 % changed integer variable domains as a threshold to trigger another LP relaxation solve.
- Primal heuristics are disabled during diving. Only a simple rounding procedure attempts to produce feasible solutions from each new LP relaxation solution.
- Cutting plane separation is disabled.

It may happen that the current node is detected infeasible, either by propagation or after solving the LP relaxation. In this case, all diving heuristics perform a *1-level backtrack*, i.e., they undo the most recent branching decision that led to the infeasible node, and continue the diving search in the sibling. In Figure 7.1, the node d_3 is such an infeasible node, after which the depicted diving heuristic continues searching d_4 . If the sibling node is also infeasible the diving heuristic terminates.

The branching rules used in diving heuristics usually tend towards feasibility. In contrast to that, branching rules of the main search process, e.g., reliability

branching [Achterberg et al., 2004], focus on a good subdivision of the problem that helps raising the dual bound quickly. For an overview of the diving heuristics available in SCIP, we refer to [Berthold, 2008].

In SCIP, diving heuristics also provide useful search information. For example, domain propagation is applied after rounding variables, to reduce variable domains or even detect infeasibility. The latter can be analyzed by conflict analysis techniques, e.g., [Achterberg, 2007b; Witzig et al., 2017], to derive additional global information.

7.1.1 Selection Strategy

We have extended SCIP by a new primal heuristic plugin called *adaptive diving* that selects one out of nine available diving heuristics at each call.

Similarly to the budget considerations (6.4) for ALNS in the previous chapter, also the diving heuristics of SCIP measure their effort spent, but not in terms of search nodes, but in terms of LP iterations. Adaptive diving tries to spend its budget on the best performing diving heuristics. The budget computation is influenced by the success of adaptive diving in terms of found solutions, and by the amount of LP iterations it spends relative to the LP iterations of the main branch-and-bound search.

If the budget computation allows for its execution, adaptive diving selects and launches a diving heuristic. Each call where adaptive diving executes a diving heuristic is called a *round* and denoted by $t = 1, 2, \dots$, as in the previous chapter.

Concretely, let κ^{bb} denote the total number of Simplex iterations spent on node LP relaxations during the main branch-and-bound search. Let $s(t-1)$ and $s^{\text{best}}(t-1)$ denote the number of (incumbent) solutions found by adaptive diving until round $t-1$. Let $\kappa^{\text{ad}}(t-1)$ denote the total number of LP iterations consumed by adaptive diving.

The LP iteration budget of adaptive diving is computed relative to the LP iterations κ^{bb} of the main search as

$$1500 + \frac{s(t-1) + 10 \cdot s^{\text{best}}(t-1) + 1}{t} \cdot 0.1 \cdot \kappa^{\text{bb}} - \kappa^{\text{ad}}(t-1). \quad (7.1)$$

In (7.1), $t = 1, 2, \dots$ denotes the next round of adaptive diving. The LP iteration budget of adaptive diving has an initial offset of 1500 iterations. The budget is computed relative to the LP iterations of the main search and depends on the number of solutions found by adaptive diving, where newly found incumbent solutions are given a higher weight. The next round t of adaptive diving is executed when (7.1) becomes positive.

At each round t , we propose a weighted sampling strategy as follows. Let \mathcal{A} denote the set of diving heuristics available to adaptive diving. For a diving heuristic $a \in \mathcal{A}$ and round $t = 1, 2, \dots$, we denote by $\tau_a^{\text{back}}(t)$ the number of backtracks and by $\tau_a^{\text{conf}}(t)$ the number of conflict clauses generated. Clearly, those two numbers are different from zero only in rounds in which a is the selected diving heuristic. Our sampling weight for

7. Adaptivity Beyond Large Neighborhood Search

	#	default			adaptivediving		
		solved	nodes	time	solved	nodes _Q	time _Q
all	491	320	2550	152	327	0.938	0.958
affected	284	274	1120	46	281	0.939	0.945
[10,tilim]	245	210	2693	158	217	0.899	0.922
[100,tilim]	144	109	5821	526	116	0.904	0.909

Table 7.1: Aggregated results for adaptive diving over three random seeds. Columns: shifted geom. mean of generated nodes (**nodes**, shift: 100), solving time in seconds (**time**, shift: 1), and respective quotients (**nodes_Q** and **time_Q**).

diving heuristic a is computed as

$$p_{a,t} \propto \left(\frac{\sum_{t'=1}^{t-1} \tau_a^{\text{back}}(t')}{\sum_{t'=1}^{t-1} \tau_a^{\text{conf}}(t') + 10} + 10^{-4} \right)^{-1} \quad (7.2)$$

The symbol \propto reads “proportional to”, which means that we omit the normalization constant such that the probabilities sum up to one. We use the summation term 10 in the denominator to force a more uniform weight distribution between the available diving heuristics during the first executions of adaptive diving, thereby ensuring that all diving heuristics are selected several times for a meaningful initialization of the score. The second summand 10^{-4} ensures that the weight $p_{a,t}$ is never the reciprocal of zero.

In round $t = 1$, all selection probabilities are equal, such that $p_{a,1}$ represents a uniform distribution among the available diving heuristics. With increasing rounds $t > 1$, the weights (7.2) will prefer those diving heuristics with a small backtracks to conflict ratio. The motivation for this ratio is that a conflict clause represents useful learned information for the main branch-and-bound search even beyond the execution of an individual heuristic. Therefore, our weighted sampling prefers exactly those diving heuristics that provide many useful conflicts with little effort, which we measure by the number of backtracks performed.

7.1.2 Computational Results

In this section, we report the influence of adaptive diving on a large set of heterogeneous MIP benchmark instances. Our experiment is based on a pre-release version of SCIP 6.0 and SoPlex 3.1.1. The experiment has been performed on a test set MMMC of 496 instances combining the benchmark sets MIPLIB 3, MIPLIB 2003, MIPLIB 2010, and COR@L (see [Achterberg et al., 2006; Bixby et al., 1998; Koch et al., 2011] and [Coral]). Each experiment has been conducted on a cluster with identical machines to ensure comparable running time measurements.

Table 7.1 compares the performance of SCIP in its standard configuration (**default**) and with adaptive diving selection (**adaptivediving**) on the MMMC test set with a time limit of one hour. An instance is called “solved” only if it has been solved consistently for each of three tested random seeds. For **default**, we further report the number of branch-and-bound nodes in column “**nodes**” and the solving time in column “**time**” using shifted geometric means shifted by 100 nodes and 1 sec., respectively. For **adaptivediving**, we show node and time shifted geometric means relative to **default** as our baseline in columns “**time_Q**” and “**nodes_Q**”, respectively. We group the instances in different categories, which are shown in different rows of Table 7.1. The first row shows the results for all 491 instances, including those that timed out. The second row “affected” shows the results for instances that could be solved consistently across all three seeds by **default** or **adaptivediving**, and where there was a different solution path taken in at least one of the three tested seeds, measured by the number of simplex iterations of the main search. Note that these do not account for potential additional simplex iterations spent inside of **adaptivediving** or the existing diving heuristics of SCIP. The third and fourth row of Table 7.1 restrict the affected instances further. [10,tilim] filters out all instances that could be solved in less than 10 seconds by both **default** and **adaptivediving**. Similarly, [100,tilim] filters out all instances that could be solved in less than 100 seconds by both **default** and **adaptivediving**.

Using **adaptivediving**, SCIP could solve seven more instances consistently within the time limit. On non-trivial instances in the group [10,tilim] where at least one configuration needs 10 or more seconds, it achieves a speed up of almost 8 %, and a comparable reduction in the number of branch-and-bound nodes. A detailed comparison for all instances in MMMC can be found in Table D.1 in Appendix D.

7.2 Pricing for the Dual Simplex Algorithm

The dual simplex algorithm is one of the most important techniques for LP problems and key for the LP-based branch-and-bound approach. Among the algorithmic choices within the simplex algorithm, one is the determination of the direction to search for a new basic solution, called *pricing step*. In this chapter, we consider three well-known and practically proven methods called *devex pricing* [Harris, 1973], *steepest edge pricing* [Goldfarb and Reid, 1977], and *quick start steepest edge* [Forrest and Goldfarb, 1992]. All methods try to select a direction that is steepest in regard to the dual objective improvement, thereby balancing accuracy of the decision and computational overhead per iteration. Devex pricing requires the least work per iteration but may lead to a higher number of total iterations. On the other hand, steepest edge computes accurate improvement measures that often lead to a considerable smaller iteration count, and an initialization step that can be expensive to compute, depending on the starting basis. While this is less relevant for pure LP solving, in the branch-and-bound context many LP re-optimizations are

performed that start from an advanced basis. Here, quick start steepest edge sacrifices accuracy for a faster initialization. We refer to the literature for an in-depth description of these pricing techniques.

LP relaxations solved during the root node are special. The initial LP relaxation has to be solved from scratch without a starting basis available, which typically requires a considerably longer effort than subsequent LP reoptimizations. During separation, cutting planes refine the initial LP relaxation as additional rows. The corresponding tighter LP relaxations can be resolved using the dual simplex algorithm, starting from the optimal basis for the previous LP relaxation, which is still dual feasible. However, added rows typically require more simplex iterations than resolves inside the branch-and-bound tree, where a child node LP relaxation differs from its parent LP relaxation in only a single bound change, namely the branching decision.

Because of this structural difference between separation loop LP relaxations and LP relaxations solved inside the branch-and-bound search tree, we focus solely on LP relaxations solved inside the branch-and-bound tree, i.e., after the root node has been solved. Therefore, each LP relaxation inside the search tree comprises a round $t = 1, 2 \dots$ in bandit terminology.

7.2.1 Selection Strategies for Simplex Pricing

The set of available LP pricings (actions) consists of $\mathcal{A} := \{\text{devex}, \text{qsteep}, \text{steep}\}$. Here, **devex** denotes devex pricing due to Harris [1973], the default rule used by **SoPlex**, **qsteep** denotes quick start steepest edge pricing introduced by Forrest and Goldfarb [1992], and **steep** denotes classical steepest edge pricing [Goldfarb and Reid, 1977].

In round $t = 1, 2, \dots$ in which a node LP relaxation needs to be reoptimized, a selection strategy first chooses one of the available pricings $a_t \in \mathcal{A}$. Using a_t as pricing strategy, the LP relaxation is solved to optimality, after which we observe the corresponding running time $\tau_{a_t}(t)$ of this LP relaxation solve. The goal of the selection process is to maximize the LP throughput by carefully selecting the fastest among the available pricing strategies. It is not sufficient to consider the number of LP iterations, because the costs per iteration vary between the pricing strategies.

For a subset $\mathcal{A}' \subseteq \mathcal{A}$ of the available pricings, we denote by

$$T_{\mathcal{A}'}(t) := \sum_{t'=1}^t \mathbb{1}_{\{a_{t'} \in \mathcal{A}'\}}$$

the total number of times that one of the pricings in \mathcal{A}' has been selected in rounds 1 through t , and we denote by

$$\bar{\tau}_{\mathcal{A}'}(t) := \sum_{t'=1}^t \mathbb{1}_{\{a_{t'} \in \mathcal{A}'\}} \frac{\tau_{a_{t'}}(t')}{T_{\mathcal{A}'}(t)}$$

the average running time spent on the pricings in \mathcal{A}' . The goal of the following selection strategies is to use the above information to select a pricing that minimizes the running time of the LP relaxations during the search.

UCB As first selection strategy, we use α -UCB (cf. Sec. 6.3). In order to apply the UCB Formula (6.7), it is necessary that the reward always lies within the interval $[0, 1]$.

We propose to transform the observed running time $\tau_{a_t}(t)$ of pricing a_t in round t into the reward

$$r_{a_t}^{\text{lp}}(t) := \left(1 + \frac{\tau_{a_t}(t)}{\bar{\tau}_{\mathcal{A}}(t)}\right)^{-1}. \quad (7.3)$$

The actual observed runtime of the selected pricing a_t in round t appears in the denominator in Formula (7.3). It is scaled by the average runtime across all actions observed so far. First note that $r_{a_t}^{\text{lp}}(t) \in [0, 1]$, as required. Observing a runtime that matches exactly the average runtime observed so far yields a reward of exactly $\frac{1}{2}$. Below average runtimes achieve rewards larger than $\frac{1}{2}$, whereas above average runtimes achieve rewards that are smaller than $\frac{1}{2}$. Recall that UCB requires a parameter α that determines the width of the confidence band. For the experiments in this section, we set the parameter $\alpha = 2$.

greedy We also test a **greedy** strategy that always selects the pricer with minimum *modified average runtime*

$$\bar{\tau}_a^\sigma(t) := \sum_{t'=1}^t \mathbb{1}_{\{a_{t'}=a\}} \frac{\tau_{a_{t'}}(t')}{T_a(t) + \sigma_a}. \quad (7.4)$$

The use of the shift values encourages more exploration among the available pricers at the beginning of the search, such that the modified average runtime is meaningfully initialized for each pricing $a \in \mathcal{A}$. Later during the search, when $T_a(t) \gg \sigma_a$, the modified average runtime approaches the average runtime $\bar{\tau}_a(t)$.

Not all pricings are treated equally by the **greedy** selection strategy. By default, SCIP uses **devex** pricing throughout the search. Therefore, we propose to use **devex** preferably inside the **greedy** selection strategy as follows. We always select **devex** for the first 10 rounds. After that, we simply keep using **devex** if it requires less than 20 LP iterations on average. Second, we assign different shift values $\sigma_{\text{devex}} = 100$ and $\sigma_a = 50$ for $\mathcal{A} \setminus \{\text{devex}\}$. With this choice, a different pricing **devex** $\neq a \in \mathcal{A}$ is only used if a is substantially faster than **devex**. Note that **greedy** does not use any sort of randomization.

weighted As a last variant (**weighted**), we use the modified average runtime (7.4) of the **greedy** selection method as input for a weighted sampling. In each round $t = 1, 2, \dots$,

we make a randomized selection based on the sampling weights

$$p_{a,t} \propto \left(\bar{\tau}_a^\sigma(t) + 10^{-4} \right)^{-1}. \quad (7.5)$$

Again, the symbol \propto reads “proportional to”, which means that we omit the normalization constant such that the probabilities sum up to one.

7.2.2 Computational Results

Due to runtime considerations, we selected a subset of 105 instances from our ground set of 496 instances. A time limit of 900 seconds was used for each solve. Note that we use actual runtime to drive the selection strategy because **SoPlex** provides no deterministic runtime measurement. Therefore, this is the only experiment in this thesis where the solution path of the algorithm may differ even if an instance is solved twice with the same initial conditions. We accomodate for this by repeating the solution process of each instance at 4 different LP random seeds.

In Table 7.2, we compare the obtained results by considering the average LP throughput (LPs per second) and running time of three fixed pricers {**devex**, **qsteep**, **steep**} and the three proposed selection strategies **UCB**, **greedy**, and **weighted** from Section 7.2.1.

Pricer	solved	LPthpt	LPthpt _Q	time	time _Q
devex	64	74.24	1.000	91.82	1.000
steep	65	62.66	<i>0.844</i>	99.41	<i>1.083</i>
qsteep	60	58.00	<i>0.781</i>	101.13	<i>1.101</i>
UCB	63	79.02	1.064	92.80	1.011
weighted	65	71.93	0.969	93.95	1.023
greedy	65	85.06	1.146	89.11	0.970

Table 7.2: Results for LP pricers. Columns: shifted geom. mean LP throughput (**LPthpt**, shift: 1), time in seconds (**time**, shift: 1), and respective quotients (**LPthpt_Q**, **time_Q**). 105 instances, 4 LP seeds, 900 sec. time limit

Column “**LPthpt**” shows the shifted geometric mean LP throughput for each pricing, for which we use a shift value of 1. We also present the LP throughput “**LPthpt_Q**” relative to the baseline **devex**, which is the default of SCIP. This measure, LP throughput, is the direct measure that our proposed selection strategies should maximize.

Among the fixed pricers, **devex** is clearly the one with the highest LP throughput. The throughput can be increased by 6 % when using **UCB**, and even 14 % when using **greedy**. In contrast, **weighted** does not yield an improved LP throughput.

It should be noted that even if the LP throughput of one pricing is higher than for a different pricing, this does not necessarily mean that the solution time of the branch-and-bound procedure is necessarily smaller, which is mainly caused by performance variability. The faster pricing may result in a different optimal LP solution, which

leads to an inferior branching decision and a larger search tree as a result. This can also be observed in Table 7.2. We show in column “**time**” the shifted geometric mean runtime and in column “**time_Q**” the relative results with **devex** as baseline. While the **greedy** strategy even yields a 3 % time improvement, the positive result of **UCB** for the LP throughput is still too marginal to make SCIP consistently solve problems faster on average. A detailed comparison for all 105 instances can be found in Table D.2 in Appendix D.

7.3 Summary

We have extended the adaptive control mechanisms studied in Chapter 6 to two additional classes of algorithms within a MIP solver. For each class, we introduce a suitable reward function to rank the different algorithms.

The adaptive LP pricing intuitively has the nature of a stochastic bandit scenario. This intuition is confirmed by our results, in which the greedy and UCB strategy yield a higher LP throughput than any fixed pricer.

On the contrary, for diving heuristics, it is sufficient to select successful ones more often, which is why a weighted sampling selection strategy can be preferred. In both cases, we obtain considerable performance improvements on a diverse set of general MIP instances.

On the technical side, all diving heuristics of SCIP were refactored to obey a common design, which then allowed the introduction of adaptive diving. While the work on adaptive diving is completed, the adaptive LP pricing is still prototypical. Future work on this requires to replace the measured solving time by a deterministic reward criterion. For the diving heuristic, it is interesting to compare the obtained results with new selection principles that are not based on past rewards.

8

Estimating Search Tree Size

Users of MIP solvers still face the seemingly irremediable curse that comes with solving \mathcal{NP} -hard combinatorial problems: it is very hard to predict how long a solver will take to solve an instance by any other means than to just wait for termination. Since the core algorithm of all state-of-the-art MIP solvers is branch-and-bound, the main problem is that of estimating the size of the B&B tree.

This chapter is an edited copy of the article *Estimating the Size of Branch-And-Bound Trees* [Hendel et al., 2021] which has been accepted for publication. In the article, we formally proof that approximating the size of the B&B tree within a factor of 2 is impossible unless $\mathcal{P} = \mathcal{NP}$.

In this chapter, we focus on online tree size estimation, i.e., during the execution of the algorithm. Therefore, we treat the general problem of online tree size estimation over the entire duration of the search process. The main contributions are:

- a review of state-of-the-art methods which estimate search progress and B&B tree size online;
- a tree size estimation method via time series forecasting, including a new representation technique which we call *adaptive resolution*.
- a new method, based on the frequency of leaf nodes in the B&B tree, which approximates the search progress;
- new tree size estimation methods applying Machine Learning techniques to combine individual methods;
- an efficient implementation and integration of these algorithms in SCIP. Since its newest release [Gamrath et al., 2020], SCIP version 7 displays an estimate of the search progress as a percentage in a new column of the output;
- a computational comparison of all methods.

The experimental results show that at the start of the search, the best ML method improves upon the best individual method in estimation accuracy by a considerable margin and by a factor of 10 compared to a method based on the gap, the de facto progress measure of current MIP solvers.

This chapter is structured as follows. We review existing literature on tree size estimation in Section 8.1. Section 8.2 introduces search completion as the main label which we try to approximate. Afterwards, two types of tree size estimate methods are presented: the ones directly based on approximations of search completion are presented in Section 8.3, and the ones that use time series forecasting techniques are explained in Section 8.4. In Section 8.5, we describe the details of a simulation to calibrate several parameters of the time series forecasting based estimators. In Section 8.6, we give some remarks about the implementation in SCIP. In Section 8.7, we show how to combine the tools from previous sections via Machine Learning techniques. We then perform computational experiments to compare all discussed methods in Section 8.8.

8.1 Search Tree Exploration & Tree Size Estimation

The exponential nature of the branch-and-bound algorithm 2.3 has motivated research on early estimates of final search attributes such as the search tree size, the total amount of time, or the optimal objective value of the given problem. Most previous contributions have been made in the area of tree size prediction, mostly for general search trees. An application inside branch-and-bound is particularly challenging because the changing primal bound results in occasional pruning of huge parts of the search tree.

Knuth [1975] suggested averaging the individual predictions of repeated random probes down the search tree in a Monte Carlo fashion as an unbiased estimate of the search tree size. Knuth’s model assigns probabilities to each potential branch of the backtracking procedure that estimate the relative sizes of the child subtrees in each branch. The method then computes an estimate of the total tree size by sampling root-to-leaf paths subject to these probabilities. Knuth showed that even when naively using uniform probabilities, this sampling procedure produced reasonable results, but the high variance of the method makes it unreliable, especially when the tree is unbalanced [Kilby et al., 2006].

An extension based on partial backtracking has been proposed by Purdom [1978]. As a generalization of Knuth’s method, Chen [1992] introduced the use of stratifiers in order to reduce the variance of the estimate. The concept of stratifiers is also referred to as *type system* [Lelis et al., 2013, 2014; Zahavi et al., 2010]. Chen’s stratified sampling traverses a partial search tree in breadth-first order. Both Knuth and Chen discuss an additional difficulty of branch-and-bound algorithms for tree size prediction: the absence of knowledge about the optimal objective value of the problem at hand.

Recently, Lelis et al. [2013, 2014] suggested several extensions to stratified search. The first, a *two-step stratified sampling*, constructs a set of stratified search trees in the manner of Chen [1992] and then simulates a depth-first search to only visit nodes from the previously sampled trees. To better cope with a decreasing objective when new incumbent solutions are found, Lelis et al. [2013] store additional objective information in the form of histograms. This is inspired by the use of histograms for tree size predictions by Burns and Ruml [2013] in the context of iterative deepening in general search trees. The second approach by Lelis et al. [2014] is called *retentive stratified sampling*. It uses auxiliary data on solution paths gathered during previous probes to model more correct pruning behavior of the actual search algorithm. Retentive stratified sampling produces predictions of similar quality without the exhaustive memory requirements of the two-step stratified search [Lelis et al., 2013].

Kullmann [2009] extended Knuth’s ideas by augmenting the estimation with additional information about the progress of the search. Provided that one has a measure of how much progress is made by a particular decision, a quantity called the τ -value can be derived that can be used to estimate the relative sizes of the child subtrees of a particular search node. Kullmann studied this problem in the context of SAT, where for example, the number of satisfied clauses can be used as a sufficient progress measure. Kullman showed that this method could be used to derive probabilities that reduce the variance exhibited by Knuth’s sampling method.

A similar model was studied for MIP, where a suitable progress measure on the search tree is the dual gap change as branching decisions are made [Le Bodic and Nemhauser, 2017]. They derived a quantity that describes the asymptotic tree sizes, referred to as the ratio φ , which is similar to Kullman’s τ -value. In [Belov et al., 2017], this model is applied to tree size estimation for MIP, where it is shown that the use of the φ value for deriving subtree weights roughly halves the error in the estimate in the sampled tree size compared to using Knuth’s uniform probabilities. All the above methods tackle the problem of *offline* sampling., i.e., before the actual search begins.

While these methods can to some degree be extended to the online case (see e.g. [Belov et al., 2017]), some information is lost in the process. For instance, one way to adapt offline sampling to online tree size prediction is to treat the leaves obtained by the tree search procedure as if they had been obtained randomly. However, while in offline sampling, samples are drawn independently, this does not hold when obtaining leaves online. This phenomenon clearly materializes in the difference between offline and online results of Belov et al. [2017]. Indeed, at any given point in the search, supposing that samples are independent equates to supposing that the first or latest samples observed are equally good predictors of the next samples to be observed. In other words, any such method would ignore possible *trends* in the series of samples. However, we argue that there are multiple types of trends affecting the samples obtained in the B&B. First, since the depth of the tree grows as the search progresses, increasingly deeper leaves are

found, although this is not a monotonic process. Second, and conversely, after a primal solution is found which improves the primal bound, nodes can be pruned at shallower depths than previously. A similar phenomenon occurs with strong conflicts [Achterberg, 2007b].

Other factors such as the phase-specific solving approaches from Chapter 5 contribute to creating varying trends in the amount of resources required to reach a leaf. Hence, in an online setting, while we cannot suppose that samples are independent, capturing trends may mitigate this loss.

In contrast, recent work, including the present chapter, has focused on online estimation methods. These include the Weighted Backtrack Estimator [Kilby et al., 2006], the tree profile estimation [Cornuéjols et al., 2006] and the Sum Of Subtree Gaps [Özaltın et al., 2011], which we use as reference methods for our computational study. They are explained in Sections 8.1.1, 8.2, and 8.3.

Note that tree size estimates have been extensively studied for A^* , see e.g. [Thayer et al., 2012] and references therein. In particular, this reference reviews concepts of progress measures and velocity-based estimates for A^* .

8.1.1 Tree Profile Estimation

This section describes the tree profile estimation [Cornuéjols et al., 2006]. For the tree profile estimation, a depth histogram of the search tree nodes (the “tree profile”) is recorded during the search. A few key characteristics of the tree profile such as the maximum depth are then turned into an estimation of the final tree size.

For search state T_k , let $d_k^{\max} := \max_{v \in V_k^{\text{leaf}} \cup V_k^{\text{inner}}} d(v)$ denote the maximal depth of any solved node at T_k . A *depth profile* \mathcal{D}_k is defined as

$$\mathcal{D}_k := \{|D_{k,i}| : i = 0, \dots, d_k^{\max}\}, \text{ where } D_{k,i} := \{v \in V_k^{\text{leaf}} \cup V_k^{\text{inner}} : d(v) = i\}.$$

The *maximum width depth* is $d_k^{\text{width}} := \operatorname{argmax}_i |D_{k,i}|$ and the *last full depth* $d_k^{\text{full}} := \max \{i : |D_{k,i}| = 2^i\}$. Following the intuition that between a (reasonably initialized) search state T_k and T_U these statistics do not differ too much, the *profile* estimate [Cornuéjols et al., 2006] approximates the growth factors¹

$$\rho_{U,i} := \frac{|D_{U,i}|}{|D_{U,i-1}|}$$

as follows:

$$\rho_{k,i} := \begin{cases} 2, & \text{if } 1 \leq i \leq d_k^{\text{full}}, \\ 1 + \frac{d_k^{\text{width}} - i}{d_k^{\text{width}} - d_k^{\text{full}}}, & \text{if } d_k^{\text{full}} < i \leq d_k^{\text{width}}, \\ 1 - \frac{i - d_k^{\text{width}}}{d_k^{\text{max}} - d_k^{\text{width}}}, & \text{if } d_k^{\text{width}} < i \leq d_k^{\text{max}}. \end{cases}$$

¹The growth factors are called γ -sequence in the original publication [Cornuéjols et al., 2006].

The growth factors are finally turned into an estimation of the final tree size as

$$\hat{U}^{\text{profile}} := 1 + \sum_{i=1}^{d_k^{\max}} \prod_{j=1}^i \rho_{k,i}.$$

8.2 B&B Search States and Search Completion

The B&B algorithm 2 records a sequence of $U + 1$ search states T_0, \dots, T_U . We will refer to T_1, \dots, T_{U-1} as *intermediate* states and to T_U as the *final* state. All states represent trees that have the same root and additional information regarding the primal and dual bounds. Between two successive search states T_k and T_{k+1} at some $k \in \{0, \dots, U - 1\}$, the algorithm permanently removes the current node v from the set of open nodes V_k^{open} to either mark it as a final leaf or split it into subproblems by branching. Concretely, v can be marked to be a final leaf if its dual bound $z_k^*(v)$ exceeds the primal bound Z_k .² This ensures that each search state T_k represents a binary tree.

Throughout the chapter, we assume 2-way branching decisions, which are most common in state-of-the-art MIP solvers. This ensures that each search state T_k represents a binary tree. For $k \geq 0$, this implies in particular that the relation

$$2 \cdot (|V_k^{\text{leaf}}| + |V_k^{\text{open}}|) = |V_k| + 1 \quad (8.1)$$

holds, where we define the set of all nodes $V_k := V_k^{\text{open}} \cup V_k^{\text{inner}} \cup V_k^{\text{leaf}}$.

Equation (8.1) can be shown as follows. First note that the equation trivially holds for a tree that has the root node as the only leaf node. For trees where $|V_k| > 1$, the root node is an internal node with degree 2. Since the leaves have degree 1 and all other internal nodes have degree 3, the sum of the degrees can be written as $(|V_k^{\text{leaf}}| + |V_k^{\text{open}}|) + 3(|V_k| - (|V_k^{\text{leaf}}| + |V_k^{\text{open}}|) - 1) + 2$. The number of edges in a tree is always equal to $|V_k| - 1$. Using that twice the number of edges equals the sum of the degrees of the nodes yields Equation (8.1).

Furthermore, since a solved node enters either V_k^{leaf} or V_k^{inner} , the relations $|V_k^{\text{leaf}}| + |V_k^{\text{inner}}| = k$ and therefore $|V_k| - |V_k^{\text{open}}| = k$ hold at each search state. Note that this implies $|V_U| = U$ when the B&B algorithm terminates.

Definition 8.1 (Search Completion). *Let $T = T_0, \dots, T_U$ be a search state sequence. For every state T_k , $k \in [U]$, in T , we define the search completion as the fraction of solved nodes compared to the size of the final tree*

$$\Gamma_k = \frac{k}{U} = \frac{|V_k^{\text{leaf}}| + |V_k^{\text{inner}}|}{|V_U|}. \quad (8.2)$$

²In practice, SCIP does not report nodes that have been pruned as solved nodes.

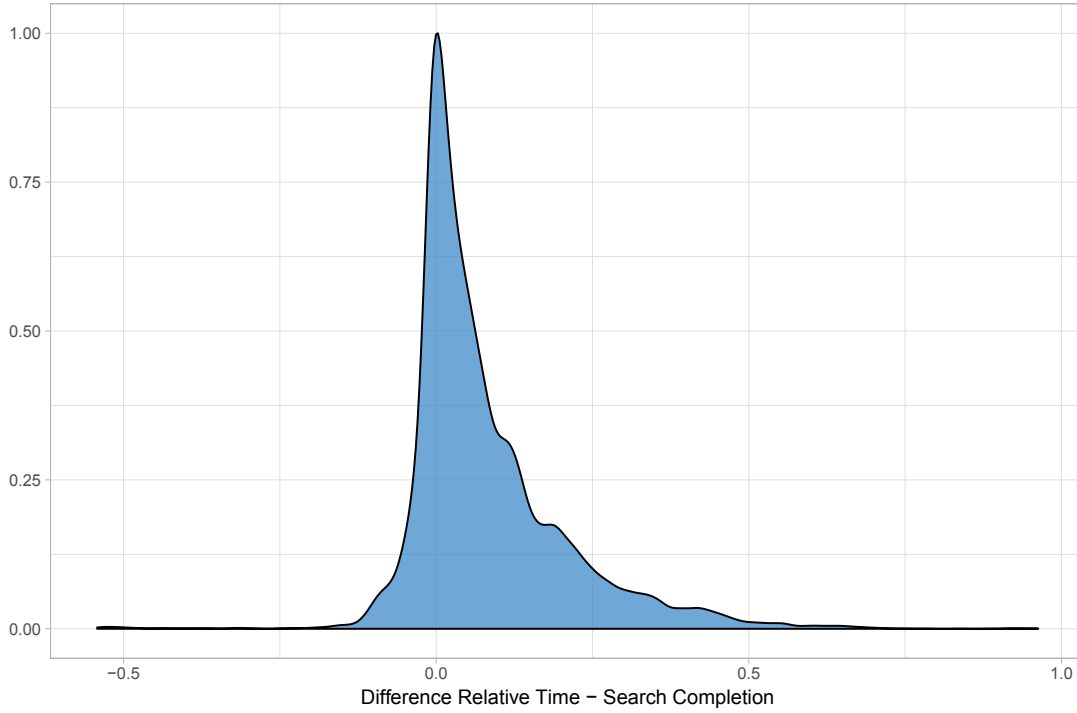


Figure 8.1: Density approximation of the difference between the relative time of the search and the search completion Γ_k

In real optimization scenarios, U and thus Γ_k are only known after the branch-and-bound algorithm terminated. In Sections 8.3 and 8.4 we review measures that can be used to estimate Γ_k and U online.

In practice, the search completion Γ_k also approximates the runtime of a MIP solver quite well. For the data set used in Sections 8.7 and 8.8, which comprises more than 16 000 search states across various MIP instances (see Section 8.7.1 for more information), we show in Figure 8.1 the close correspondence between the relative time of the overall search and the search completion Γ_k . To this end, Figure 8.1 depicts the *data density* of the observed differences between the relative time, i.e., the time normalized to $[0, 1]$, and the search completion in our data set. A high data density at a coordinate on the horizontal axis reflects a high number of observed records near this coordinate. The density has a peak at zero, in which case the search completion approximates the actual fraction of time exactly. Since deviations from zero are mostly positive, the search completion is slightly pessimistic as it often overestimates the remaining time. This can be explained by the fact that like most solvers, SCIP does not spend its entire time on the tree search itself, but also spends significant time on presolving and root node processing. In conclusion, it seems generally reasonable to use the search completion Γ_k as a surrogate for the runtime.

Throughout this chapter, since estimations may over- or underestimate the actual tree size at termination, we measure the error of a tree size estimation \hat{U} using the

normalized ratio

$$E(\hat{U}, U) = \max \left\{ e(\hat{U}, U), \frac{1}{e(\hat{U}, U)} \right\}, \quad \text{where } e(\hat{U}, U) := \frac{\hat{U}}{U}, \quad (8.3)$$

which penalizes under- and overestimations \hat{U} of U by the same factor (see, e.g., [Kilby et al., 2006; Özaltın et al., 2011]).

8.3 Approximations of Search Completion

Let T_k for $k \in [U]$ denote a search state during the B&B algorithm. Any approximation $\hat{\Gamma}_k > 0$ of the search completion Γ_k from Definition 8.1 provides a simple way to estimate the search tree size as

$$\hat{U} := \frac{k}{\hat{\Gamma}_k}, \quad (8.4)$$

since k is equal to the number of solved nodes at T_k . Clearly, good approximations of Γ_k can be expected to give accurate estimates of the final tree size.

In the following, we review four approximate measures of Γ_k and the related weighted backtrack estimation [Kilby et al., 2006]. We start with the well-known gap.

8.3.1 Gap

Every B&B solver reports a relative gap between the primal and dual bound in one form or another, which makes the gap the most common progress measure for B&B.

We are interested in the gap at a search state T_k , taking into account the primal bound Z_k and the dual bound Z_k^* . The gap monotonically decreases from $1 = \gamma(Z_0, Z_0^*)$ to $0 = \gamma(Z_U, Z_U^*)$. Therefore, taking $1 - \gamma(Z_k, Z_k^*)$ yields a monotone approximation of Γ_k .

In the absence of any dedicated progress measure in the output of MIP solvers, the gap serves as the *de facto* progress measure. However, as we will see in our experiments, it provides the poorest approximation of all search progress measures defined in this section. One shortcoming of the gap is that for infeasible MIPs, the gap is equal to 1 at all search states before T_U , and assumes 0 only at the last search state. Even for feasible MIPs, if the primal bound does not change, only improvements to the global dual bound are reflected by the gap. For instances for which the difficulty lies in finding an optimal solution, or towards the end of the B&B search, where the absolute gap is small, changes in the global dual bound occur very infrequently. The Sum of Subtree Gaps, presented in the next section, has been designed to take into account the change of dual bound at every node, rather than only the change of the worst dual bound.

8.3.2 Sum of Subtree Gaps (SSG)

The *Sum of Subtree Gaps* (SSG) has been proposed by Özaltın et al. [2011] as a better runtime estimate than the gap. The main idea is to split the overall search tree into a disjoint family of subtrees, and to average the gaps of the different subtrees. First, recall from Definition 2.4 that at each search state T_k , we have a node dual bound $z_k^*(v)$ available for every node $v \in V_k$. Each node v also represents its own subtree $\text{subtree}(v) \subseteq V_k$, which consists of v itself and all its direct and indirect descendants. If $v \in V_k^{\text{leaf}} \cup V_k^{\text{open}}$, then $\text{subtree}(v) = \{v\}$. In all other cases, v is an inner node with at least two descendants. If v is an inner node, $z_k^*(v)$ is equal to the smallest dual bound across the open nodes in $\text{subtree}(v)$, because final leaf nodes have a node dual bound of infinity.

Consider a set of nodes $V' \subseteq V_k$ with the property that for all $v \neq v' \in V'$ $\text{subtree}(v) \cap \text{subtree}(v') = \emptyset$. In other words, v and v' are the roots of disjoint subtrees of V_k . Note that V_k^{open} satisfies this property from search state T_k onwards, i.e., for all search states $k' \geq k$. We compute

$$f(V') := \sum_{v \in V'} \gamma(Z_k, z_k^*(v))$$

as the (unscaled) sum of subtree gaps. Further, let

$$\text{pred}^Z(k) := \min \{k' \in [k] : Z_{k'} = Z_k\}$$

denote the last iteration of B&B up to k in which the primal bound improved. The SSG uses a special set of subtree roots $V' \subseteq V_k$ to compute f , namely the $|V_{\text{pred}^Z(k)}^{\text{open}}|$ open nodes $V_{\text{pred}^Z(k)}^{\text{open}}$ at the time at which the primal bound has improved last. For $v \in V_k$, let $\text{subtree}(v) \subseteq V_k$ denote the set containing v and its descendants. The SSG is defined as

$$\gamma^{\text{SSG}}(T_k) := \phi_k \cdot f(V_k^{\text{SSG}})$$

over the set

$$V_k^{\text{SSG}} := V_{\text{pred}^Z(k)}^{\text{open}}$$

of open nodes when the incumbent changed last, with a scaling factor ϕ_k defined as

$$\phi_k := \begin{cases} 1, & \text{if } k = 0, \\ \phi_{k-1}, & \text{if } k > 0 \text{ and } Z_k = Z_{k-1}, \\ \phi_{k-1} \cdot \frac{f(V_{k-1}^{\text{SSG}})}{f(V_k^{\text{SSG}})} & \text{if } k > 0 \text{ and } Z_k < Z_{k-1}. \end{cases}$$

The scaling factor changes every time a new incumbent solution is found. Özaltın et al. [2011] prove that due to this scaling factor the SSG $\gamma^{\text{ssg}}(T_k)$ is monotonically decreasing from $\gamma^{\text{ssg}}(T_0) = 1$ to $\gamma^{\text{ssg}}(T_U) = 0$, like the gap. Therefore, an approximation of search completion based on SSG is given by

$$\hat{\Gamma}_k^{\text{ssg}} = 1 - \gamma^{\text{ssg}}(T_k).$$

The SSG represents a generalization of the gap that takes into account the gaps of individual subtrees, which cover the remaining open portion of the search tree. A dual bound change in any of the involved subtrees is reflected by the SSG, but not necessarily by the gap, because the gap only changes with the global dual bound.

One limitation of both the gap and the SSG is that changes in the dual bounds are not reflected in absence of a primal bound. The progress measures presented in Sections 8.3.3, 8.3.4 and 8.3.5 are not based on the primal or dual bound. They can thus be used for feasibility problems and more generally for other tree search algorithms.

8.3.3 Tree Weight

The tree weight has been first used by Kilby et al. [2006] as the denominator of the weighted backtrack estimator (studied in Section 8.3.4). Its use as a progress measure was proposed in [Anderson et al., 2018]. This measure assigns to each node $v \in V_k$ a *weight* $2^{-d(v)}$, where $d(v)$ is the depth of v . Note that every parent weight equals the sum of the weights of its children. We call

$$\omega(T_k) := \sum_{v \in V_k^{\text{leaf}}} 2^{-d(v)}$$

the *tree weight* at state T_k .

Proposition 8.2. *Let T be a nonempty sequence of $U + 1$ search states of a B&B tree, and let $\omega : T \rightarrow [0, 1]$ be the tree weight as above. It holds that*

$$0 = \omega(T_0) \leq \dots \leq \omega(T_{U-1}) < \omega(T_U) = 1$$

Proof. We show inductively that for each search state

$$\underbrace{\sum_{v \in V_k^{\text{leaf}}} 2^{-d(v)}}_{= \omega(T_k)} + \sum_{v \in V_k^{\text{open}}} 2^{-d(v)} = 1. \quad (8.5)$$

At the initial search state T_0 , the set of leaf nodes V_0^{leaf} is empty and the set of open nodes V_0^{open} contains the root node at depth 0 as only node. From this, it follows that $\omega(T_0) = 0$.

Assume that the invariant (8.5) holds for any intermediate search state T_k . When moving to T_{k+1} , there are two cases. Let $v_k \in V_k^{\text{open}}$ denote the node processed in iteration $k+1$ of the branch-and-bound algorithm 2. At the end of iteration $k+1$, v_k is removed from the set of open nodes and enters either V_{k+1}^{leaf} or V_{k+1}^{inner} . In the first case, it is trivial to see that the invariant (8.5) still holds, because the contribution of v_k moves from the right summand to the left. In the other where v_k becomes an inner node its two children v', v'' are added to the set of open nodes. We therefore have

$$\begin{aligned} \sum_{v \in V_{k+1}^{\text{leaf}}} 2^{-d(v)} + \sum_{v \in V_{k+1}^{\text{open}}} 2^{-d(v)} &= \sum_{v \in V_k^{\text{leaf}}} 2^{-d(v)} + \sum_{v \in V_k^{\text{open}}} 2^{-d(v)} - 2^{-d(v_k)} + 2^{-d(v')} + 2^{-d(v'')} \\ &= \sum_{v \in V_k^{\text{leaf}}} 2^{-d(v)} + \sum_{v \in V_k^{\text{open}}} 2^{-d(v)} \underbrace{-2^{-d(v_k)} + 2 \cdot 2^{-d(v_k)-1}}_{=0} \\ &= 1, \end{aligned}$$

such that the invariant also holds at search state T_{k+1} .

Since the final search state T_U is the only search state in T at which the open nodes are empty, the invariant (8.5) also shows that $\omega(T_{U-1}) < \omega(T_U)$ and that $\omega(T_U) = 1$.

The tree weight provides a non-decreasing measure of the progress of a B&B search.

At the start of the search, $V_0^{\text{leaf}} = \emptyset$, thus $\omega(T_0) = 0$. After every final leaf node, the tree weight strictly increases and the search ends at step U with a tree weight of 1. With those properties, the tree weight ω can be directly used as approximation of search completion,

$$\hat{\Gamma}_k^{\text{tree weight}} := \omega(T_k).$$

Consider the example B&B search tree depicted in Figure 8.2. The tree has 9 nodes, which are numbered as if they have been traversed from left to right in depth-first order. The leaf weights are $w_4 = w_5 = 0.125$ and $w_6 = w_8 = w_9 = 0.25$. From step 0 to 9, the tree weight therefore assumes the 6 distinct values $\omega(T_0) = \omega(T_1) = \omega(T_2) = \omega(T_3) = 0$, $\omega(T_4) = 0.125$, $\omega(T_5) = 0.25$, $\omega(T_6) = \omega(T_7) = 0.5$, $\omega(T_8) = 0.75$, $\omega(T_9) = 1$.

Remarks In this chapter, we consider uniform node weights between a node and its sibling that only depend on the depth of the node. The proof of Proposition 8.2 shows that the tree weight is generalizable to nonuniform node weights. In fact, every weight scheme in which the parent weight is distributed across its children in a branching step yields a tree weight with the desired properties that can be used as approximation of search completion. The ideal weight scheme distributes the parent weight relative to the sizes of the respective subtrees of the children. Knuth [1975] has shown that this yields a perfect estimation of tree size with zero variance at the first leaf node. Obviously, equipped with knowledge of the sizes of each subtree, a perfect tree size estimation

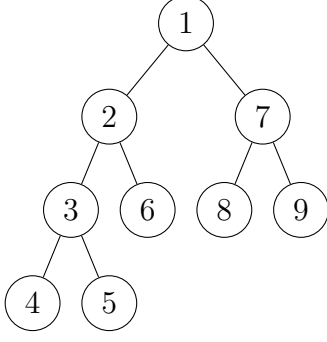


Figure 8.2: Example search state at termination ($U = 9$, $|V_9^{\text{leaf}}| = 5$, $|V_9^{\text{open}}| = 0$). We assume that the tree was traversed from left to right in depth-first order. The final leaves are $V_9^{\text{leaf}} = \{4, 5, 6, 8, 9\}$ and internal nodes are $V_9^{\text{inner}} = \{1, 2, 3, 7\}$.

is available before the search even begins. Also Belov et al. [2017] experimented with nonuniform node weights based on the individual improvements in the dual bound of the children. Like estimations of the subtree sizes, also dual bound improvements in the children are not readily available during branching in general, and need to be estimated. From an implementation point of view, a disadvantage of nonuniform node weights is an additional memory requirement to store the individual weights. In preliminary experiments, we did not obtain satisfying results with the methods described by Belov et al. [2017] in our online setting, which is different from the offline sampling that they apply. We therefore do not consider nonuniform node weights for the remainder of this thesis.

8.3.4 Weighted Backtrack Estimator (WBE)

The *Weighted Backtrack Estimator* (WBE) by Kilby et al. [2006] is an online method for binary search trees explored by depth-first search. For each probe down the tree until a leaf node at depth d is reached, the estimate for the size of the tree considering only this single probe is $2^{d+1} - 1$, while the probability of reaching this particular leaf by randomly choosing whether to go left or right at every depth is 2^{-d} , which coincides with the node weight and which Kilby et al. [2006] combine into a weighted mean. Therefore, in our notation, the WBE is a projection of the current tree weight to estimate the number of leaf nodes at completion. To this end, at search state T_k with positive tree weight $\omega(T_k) > 0$, the weighted backtrack estimate is computed as

$$\hat{U}_k^{\text{wbe}} := 2 \cdot \frac{|V_k^{\text{leaf}}|}{\omega(T_k)} - 1. \quad (8.6)$$

Note that the tree size estimation (8.6) is not equivalent to the one obtained when using $\hat{\Gamma}_k^{\text{tree weight}} = \omega(T_k)$ within (8.4), which yields

$$\hat{U}_k^{\text{tree weight}} := \frac{|V_k^{\text{leaf}}| + |V_k^{\text{inner}}|}{\omega(T_k)}.$$

8. Estimating Search Tree Size

k	1	2	3	4	5	6	7	8	9
$ V_k^{\text{leaf}} $	0	0	0	1	2	3	3	4	5
$\lambda(T_k)$	-0.5	-0.25	-0.17	0.125	0.3	0.42	0.36	0.44	0.5

Table 8.1: Values of the leaf frequency for the running example in Figure 8.2.

One noticeable difference is that \hat{U}_k^{wbe} is only sensitive to the creation of final leaves, not open nodes. Kilby et al. [2006] consider the case where all leaves have depth $d \gg 1$, except the left child of the root node, which is a leaf. Suppose the leaf with depth 1 is visited first. After two leaf nodes, one with depth 1, the other d , WBE computes a tree size estimate of $\frac{4}{2^{-1}+2^{-d}} - 1 \approx 7$, which can be arbitrarily far from the $d + 1$ nodes already traversed to reach the leaf at depth d . Until the number of samples approaches 2^{d-1} , the sample of depth 1 will render WBE essentially useless: for a large enough depth d , the estimate is approximately $4|D| - 1$. As pointed out in [Le Bodic and Nemhauser, 2017], the 0.5 weight of the left child encodes an initial implicit assumption that both sides of the tree have the same size. After finding a sample at depth d on the right side of the tree, it should become clear that the assumption was incorrect. However, this is not taken into account by the WBE other than by the slow incorporation of other samples with virtually insignificant individual weight.

8.3.5 Leaf Frequency

In this subsection we introduce a new progress measure called *Leaf Frequency*, based on the well-known observation that due to Equation (8.1), $|V_U^{\text{leaf}}|/U \approx 1/2$, i.e., the final leaf nodes comprise about one half of the overall tree at the end of the search. In order to extend this to a search progress measure at intermediate search states T_k , we define the *leaf frequency* as

$$\lambda(T_k) := \frac{1}{k} \left(|V_k^{\text{leaf}}| - \frac{1}{2} \right). \quad (8.7)$$

The subtraction of $\frac{1}{2}$ in the formula above is the necessary correction such that $\lambda(T_U) = \frac{1}{2}$ at termination. More generally, by combining (8.1) and the relation $|V_k| - |V_k^{\text{open}}| = k$, we obtain

$$\lambda(T_k) = \frac{1}{k} \left(|V_k^{\text{leaf}}| - \frac{1}{2} \right) = \frac{1}{k} \left(\frac{1}{2}(|V_k| - 2|V_k^{\text{open}}| + 1) - \frac{1}{2} \right) = \frac{1}{2} - \frac{|V_k^{\text{open}}|}{2k}.$$

From this equation, we derive that $\lambda(T_k) \leq \frac{1}{2}$ and that $\lambda(T_k) = \frac{1}{2}$ only for $k = U$. Moreover, $\lambda(T_k) \geq -\frac{1}{2}$ holds for all k and $\lambda(T_k) > 0$ after the first final leaf, i.e., as soon as $|V_k^{\text{leaf}}| \geq 1$.

Table 8.1 lists the leaf frequency for all $U = 9$ iterations of the running example from Figure 8.2. This example shows that the leaf frequency $\lambda(T_k)$ is not necessarily

monotone in contrast to the other measures discussed in this section.

We propose to transform the leaf frequency into an approximation of search completion via

$$\hat{\Gamma}_k^{\text{leaf-freq}} = 2 \cdot \max\{0, \lambda(T_k)\}.$$

We prefer this transformation over its alternative $\lambda(T_k) + \frac{1}{2}$, which has the disadvantage that it assumes a search completion of at least 50 % after the first final leaf node, which is often too optimistic.

8.4 Estimation of Tree Size via Time Series Forecasting

In the previous section, we presented approximations of the search completion based on four measures: gap, SSG, tree weight, and leaf frequency, each of which can be translated into an estimate of the final search tree size using (8.4). In particular the tree weight and the related WBE suffer from a typical limitation if applied to optimization problems. When an improving solution is found, new samples will reflect it, but no mechanism revisits the estimates provided by previous samples, despite the fact that in practice, many nodes are pruned by bound, hence they would have been pruned at shallower depth, yielding smaller estimates. This can be fixed if the entire tree is kept in memory and reprocessed to compute new estimates upon improvement of the primal bound. However, this is more memory than the solving process itself keeps, which only requires the open parts of the tree, and would therefore create a substantial overhead to the search.

In this section, we present a different approach which uses Double Exponential Smoothing (DES), a time series forecasting technique, for each search progress measure. DES was also the approach used in the original paper [Özaltın et al., 2011] that introduced the SSG.

8.4.1 General Definition of Double Exponential Smoothing (DES)

Given a time series $(Y_t)_{t=0,1,2,\dots}$, DES estimates the *level* q_t of the time series, representing a fitted value of Y_t , and its *trend* s_t , which has the role of a slope. Both are computed as weighted averages of the training data, with exponentially decaying weights on older observations, as follows. Let $0 < \alpha, \beta < 1$ and denote by q_0 and s_0 initial level and trend values. For $t = 1, 2, 3, \dots$, DES fits the level and trend component to the data

recursively via

$$\begin{aligned} q_t &= \alpha Y_t + (1 - \alpha)(q_{t-1} + s_{t-1}), \\ s_t &= \beta (q_t - q_{t-1}) + (1 - \beta)s_{t-1}. \end{aligned} \tag{8.8}$$

At time step t^* , the current level and trend components q_{t^*} and s_{t^*} are used to compute a forecast of $h \in \mathbb{N}$ steps into the future as

$$\hat{Y}_{t^*+h} := q_{t^*} + h \cdot s_{t^*}. \tag{8.9}$$

Note that (8.8) is a simple variant of DES [Holt, 2004], but more complex models of DES exist [Hyndman et al., 2008].

8.4.2 Double Exponential Smoothing for Tree Size Estimation

As values Y_t , we use the four measures from Section 8.3 (not their corresponding search completion approximations), namely tree weight $\omega(T_k)$, leaf frequency $\lambda(T_k)$, gap $\gamma(Z_k, z_k^*(V_k))$, and SSG $\gamma^{\text{ssg}}(T_k)$. As a fifth measure, we consider the number of open nodes $|V_k^{\text{open}}|$, which is strictly positive at all intermediate steps $0 < k < U$ and reaches 0 at $k = U$.

We know that, by definition of the tree weight, the search is complete when it reaches a state T_U that satisfies $\omega(T_U) = 1$, hence we define and denote the target value of this time series to be $Y_U = 1$. To use DES at a time series step t^* , we therefore compute (8.9) to find the time step h^* in the future at which the forecast $\hat{Y}_{t^*+h^*}$ reaches $Y_U = 1$ as

$$h^* := \frac{Y_U - q_{t^*}}{s_{t^*}}. \tag{8.10}$$

We do the same for the four other time series, with target values $Y_U = 0.5$ for leaf frequency $\lambda(T_k)$, 0 for gap $\gamma(Z_k, z_k^*(V_k))$ as well as SSG $\gamma^{\text{ssg}}(T_k)$, and 0 for open nodes $|V_k^{\text{open}}|$.

Note that for non-monotone time series (leaf frequency and open nodes), it may happen that the trend s_{t^*} has the “wrong” sign. Indeed, we have observed that the open nodes time series often roughly follows unimodal behaviour, with a maximum attained around the middle of the search. For monotone time series, s_{t^*} may be zero if the time series plateaus for sufficiently many consecutive observations. In either case, the forecast cannot reach the target value, which means that DES cannot provide a tree size estimation. For any time series for which this occurs, we do not use DES and instead report twice the number of solved nodes at t^* as tree size estimation. For the open nodes time series, this corresponds to the observation that while the trend is not decreasing, the midpoint of the search has not been reached yet.

8.4.3 Time Series Steps and Adaptive Resolution

Parts of the process to design, implement and calibrate the methods we present throughout the chapter require storing the time series produced during a solve. As B&B trees can grow very large, we looked for ways to reduce the size of the input for memory and runtime purposes. A first improvement consists in only recording search completion measures or open nodes at search states where a new leaf node was found. To further reduce the number of points in the time series, we introduce a compression method that does not create a data point at every search state T_k , $k \in [U]$, but compresses observations made at multiple consecutive states into a single data point. While a compression of a fixed number of search states into a single data point would be the simplest approach, the number of data points would still grow linearly, but more importantly, it would reduce the number of data points available to make predictions at the start of the search, where they are most needed.

Instead, we use a process which we call *Adaptive Resolution*. In adaptive resolution, the number of data points is upper bounded by a parameter C , which stays constant during the search, and instead the number of search state compressed into a single data point increases as the tree grows. As a result, at the start of the search, every search state provides a data point in the time series. In order to not exceed C records, every time we reach the maximum capacity C , we compress the C records into $C/2$ re-processed records, thereby changing the resolution.

More formally, consider the *leaf index set*

$$\mathcal{K}^{\text{leaf}} := \{k \in [U] : |V_k^{\text{leaf}}| = |V_{k-1}^{\text{leaf}}| + 1\} =: \{k_1, \dots, k_{|V_U^{\text{leaf}}|}\}.$$

The leaf index set considers only half as many observations as the full index set $[U]$. It always contains the terminal step U . Restricted to the leaf index set, the tree weight time series becomes strictly monotone, which can be an advantage for forecasting because the trend component is always positive. Thus, we will use the leaf index set $\mathcal{K}^{\text{leaf}}$ from now on.

DES will assign most of the weight to the most recent individual leaf nodes. However, the information used by an estimation method at a single leaf can have a high variance. For instance, in the case of the tree weight, the relative difference between two leaf weights is exponential in the difference of their respective depth. We overcome this deficiency by essentially creating batches of 2^r leaves as a single time step, starting with $r = 0$, and by adaptively increasing r over time, as more data becomes available. More precisely, for $r \in \mathbb{Z}_{\geq 0}$ we denote the *index set at resolution r* by

$$\mathcal{K}^r := \{k_{2^r}, k_{2 \cdot 2^r}, k_{3 \cdot 2^r}, \dots, k_{R \cdot 2^r}\}, \quad \text{where } R := \left\lfloor \frac{|V_U^{\text{leaf}}|}{2^r} \right\rfloor.$$

For $r \geq 0$, we always batch 2^r leaves together into a single time series step. Time series step $t = 1, 2, \dots, R$ corresponds to search state $T_{k \cdot 2^r}$. At T_k , the time series has only been recorded for all $\mathcal{K}^r \cap [k]$.

The use of adaptive resolution mainly affects the estimation from DES. With an increased resolution $r \geq 0$, the meaning of a step into the future changes. At the beginning of the search, a forecast of 10 steps corresponds to 10 leaf nodes. Later during the search, a single forecast step can comprise many leaf nodes. For each search state $k \geq 0$ for which $V_k^{\text{leaf}} > 0$, there exists a well-defined time series step $t(k)$, corresponding to the most recent leaf in \mathcal{K}^r prior to (and including) k . For this time series step $t(k)$, we first make a prediction of the remaining number of time series steps h^* according to (8.10). We then take into account the current resolution to rescale $t(k) + h^*$ and estimate the total tree size U at termination via

$$\hat{U} := 2 \cdot 2^r (t(k) + h^*) - 1, \quad (8.11)$$

where the term in parentheses is an estimation of the final number of terminal nodes $|V_U^{\text{leaf}}|$ and therefore turned into an estimate of the total tree size via (8.1).

8.5 Calibration of Double Exponential Smoothing

The quality of double exponential smoothing highly depends on the choice of the level and trend parameters α and β in (8.8). This section describes the process used to optimize the parameters of DES and the adaptive resolution capacity C in particular over a large set of MIP instances using multiple snapshots for each instance. In the experimental evaluations of Section 8.8, α , β , and C are then fixed for all instances.

A direct calibration technique for α and β could consist, for example, in a grid search, where at every grid point we evaluate the accuracy of estimates on a set of MIP instances. In order to avoid repeating the same MIP runs, we solve each instance once and record the entire search, on which we can evaluate tree size estimation methods via an offline simulation as in [Belov et al., 2017]. The simulation code is written in R [R Core Team, 2018] using the package `forecast` [Hyndman et al., 2019; Hyndman and Khandakar, 2008]. It is publicly available from <https://github.com/GregorCH/treesize-estimation>.

8.5.1 Simulation Setup

As training set, we use the combination of the following four MIP benchmark sets: MIPLIB 3 [Bixby et al., 1998], MIPLIB 2003 [Achterberg et al., 2006], MIPLIB 2010 [Koch et al., 2011], and COR@L [Coral]. Each of these 496 instances is solved using SCIP 6.0 [Gleixner et al., 2018] and the entire search tree is recorded as a VBC file [Leipert, 1996]. Instances not solved to optimality within 2 hours or

with fewer than 10 leaves are discarded, leaving 233 instances. The input data for every tree size estimation method for all leaves is then extracted from the VBC trees, using a customized version of the Python code <https://github.com/pierre-lebodic/bnb-mip-estimates/>. Our simulation code applies adaptive resolution to this offline data exactly as if it had been collected online. Since this calibration simulation is based on recorded VBC trees, we have *all* leaf nodes of the tree available.

Tree size estimation methods need to be evaluated throughout the entire search process, rather than at the end of the search. For each simulated tree, we record the normalized ratio at up to 95 points during the search. Specifically, we create a point in the time series for each moment at which $\omega(T_k)$ first exceeds $\{0.01, 0.02, \dots, 0.95\}$. There can be fewer than 95 estimations as multiple weight levels can be first reached with a single leaf. As an example, the number of considered records for the **pigeon-10** tree is 91. At a resolution capacity $C = 1024$, the total number of considered records over the trees of all 233 instances that we use below is 10 322.

To conduct the simulation, we filter the initial record set to only the required leaf nodes that our SCIP implementation with adaptive resolution would use. Concretely, at a chosen capacity of $C = 1024$, these are all first 1024 leaf nodes $\{k_1, \dots, k_{1024}\}$. After this, the resolution changes to $r = 1$, such that the 512 leaf nodes $\{k_{1026}, k_{1028}, \dots, k_{2048}\}$ are kept. At k_{2048} , the resolution is increased the next time, and we only keep the 512 leaf nodes $\{k_{2052}, k_{2056}, \dots, k_{4096}\}$, and so on. Next, the accumulated tree weight is used to identify at most 95 out of all those leaf nodes at which the tree weight level reaches another full percent. Such a leaf might be k_{4000} , with its current resolution of $r = 2$, thereby folding four leaf nodes into one time series step. For computing the single prediction for DES parameters α, β at k_{4000} , we collect all the 1000 records including k_{4000} corresponding to the resolution $r = 2$, namely $\{k_4, k_8, k_{12}, \dots, k_{4000}\}$, which is exactly the data our SCIP implementation has available at runtime.

For example, the largest tree in our data set belongs to the instance **pigeon-10** and has almost 14 million leaf nodes. Its compressed simulation data contains 7507 recorded leaf nodes and their time series data, at a maximum resolution of $r = 13$. Note that this record number depends on the selected capacity C .

8.5.2 Calibration of the Adaptive Resolution Capacity C

We dynamically change the resolution each time the current step t reaches a maximum capacity denoted by C time series steps. Using powers of 2 for the resolution provides an efficient way to update the time series data during the search. For each time series we store at most C values at resolution $r = 0$. When we record the C 'th leaf node at search state T_{k_C} , we drop all odd records Y_1, Y_3, \dots, Y_{C-1} such that our record history contains $C/2$ values and set the resolution to $r = 1$. With the resolution update, we essentially change the meaning of a time series step. The previously recorded Y_2 is now

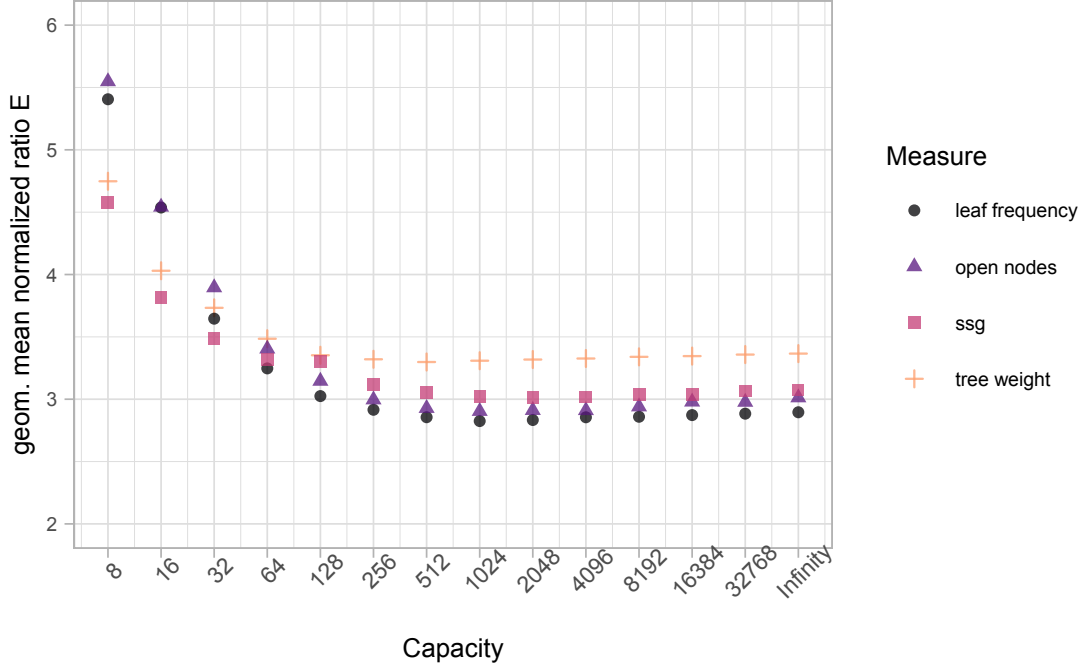


Figure 8.3: Normalized ratio at different capacities C used by adaptive resolution.

Y_1 , Y_4 is now Y_2 , such that we have the compressed search history at the new resolution readily available. From now on, we record the time series value of every 2nd leaf node until we reach the $2C$ 'th leaf node.

For a general resolution $r \geq 0$, as long as the storage size is not exhausted (i.e., $|\mathcal{K}^r \cap [k]| < C$), we extend the time series at each 2^r th leaf by 1 step, and the DES level and trend values are incrementally updated using (8.8). When the storage size is reached, we increase the current resolution by 1 and drop all odd observations. This effectively shrinks the number of stored time series values to half the capacity $\frac{C}{2}$. We then recompute the DES fit for the compressed index set from scratch. At a resolution of $r = 10$, each time step $t = 1, 2, 3, \dots$ represents $2^8 = 256$ search states.

Adjusting the resolution capacity C can lower the variance of the observations made at leaves by adaptively batching them. While we would like to optimize C , α and β together, this is computationally expensive. Instead, we first optimize C by a line search. For every value of C , we use ETS [R.J. Hyndman, 2018], which considers errors, trend, and seasonality components of a time series, hence the name ETS.³ The key property of ETS for our simulation is that it optimizes α and β at each record to the available training data, thereby allowing for a calibration of C independent of the DES parameters.

We test capacities between $C = 2^3$ and $C = 2^{15}$, for which we compute the geometric mean normalized ratio E of the estimation obtained by an ETS forecast for each of the four measures tree weight, leaf frequency, open nodes, and SSG.

³We use the `forecast`-package of R, which features the ETS method.

Time Series	α	β	E	E ETS
Tree Weight	0.65	0.15	3.32	3.30
Leaf Frequency	0.30	0.33	2.71	2.82
SSG	0.60	0.15	2.71	3.01
Open Nodes	0.60	0.15	2.69	2.90

Table 8.2: DES parameters that minimize the geometric mean normalized ratio $E(\hat{U}, U)$ for each time series on the training set.

The effect of compressing multiple observations into a single data point is illustrated by Figure 8.3, where the effect of changing C is measured as the geometric mean normalized ratio (8.3). Unsurprisingly, a very small capacity (e.g. 8) leads to a relatively high error, while an arbitrarily large capacity (i.e. no compression) leads to a relatively small error. The most interesting takeaway is that there are finite values of C for which no accuracy is lost, and in fact DES produces slightly more accurate results.

The rightmost entry uses an infinite capacity, which corresponds to not using adaptive resolution. At a capacity of 1024, all four measures achieve their best or second best normalized ratio over all tested capacities. Therefore, we consider the choice of 1024 as a good compromise between a coarse view on the entire search process and a fine view on the local, recent behavior. Note that in particular, the obtained normalized ratios at $C = 1024$ are consistently better than without adaptive resolution ($C = \infty$).

8.5.3 Calibration of the DES Parameters α and β

For every time series, we calibrate the values of α and β that minimize the geometric mean normalized ratio. We conduct a grid search by varying α in steps of 0.05 between 0.1 and 0.95 and β in steps of $\frac{0.05}{\alpha}$ between $\frac{0.1}{\alpha}$ and 1. Table 8.2 shows the obtained values of α and β per time series, their corresponding geometric mean normalized ratio E as well as the obtained normalized ratios using ETS instead (at $C = 1024$), which fits the parameters α and β adaptively to the time series at hand. Note that we did not explicitly optimize the parameters for the gap time series, for which we use the same parameters as for the SSG.

Interestingly, the calibrated parameters from Table 8.2 result in a lower (and hence, better) normalized ratio over the entire data set than could be obtained by ETS, despite the additional degrees of freedom of ETS. We explain this surprising result by the fact that ETS is too sensitive to the past behavior of the time series compared here, which yields a larger normalized ratio if an unpredictable event such as a new incumbent solution alters the search process significantly. The parameters from Table 8.2 may not fit the training data as accurately, but produce more robust estimations on average.

8.5.4 Example Instance `danooint`

In order to illustrate the DES based tree size estimates, we discuss their behavior for the MIPLIB instance `danooint` in detail. In Figure 8.4, we visualize the discussed time series and resulting tree size estimations. The data for the plots have been obtained with SCIP 6.0, which we extended by the necessary functionality to record all discussed time series and estimations; more details on the implementation are given in Section 8.6. The time series estimations are all obtained by a DES forecast. We record all five time series tree weight, gap, SSG, leaf frequency, and open nodes with an adaptive resolution as explained in Section 8.4.3. We increase the resolution every time we reach $C = 1024$ observations. The parameters α and β have been calibrated for each of the five time series, individually, see Appendix 8.5. Independently of adaptive resolution and to help our later computational experiments across instances, our implementation outputs at most 100 “estimation snapshots” per instance, namely at every leaf node at which another percent of tree weight has been reached. We compare five tree size estimations. The tree weight, leaf frequency, gap, SSG, and open nodes are illustrated as a function of the number of leaf nodes with respect to the left y -axis (“Value”) of Figure 8.4.⁴ Note that the number of open nodes is the only time series with values outside the interval $[0, 1]$. The dashed curves illustrate how the corresponding estimations evolve with an increasing number of leaf nodes. The estimations are obtained by first computing the number of remaining time series steps h^* as explained in Equation (8.10), which are then converted into an estimation of the final tree size using Equation (8.11) to compensate for the adaptive resolution.

Figure 8.4 depicts tree weight values that appear to increase almost linearly with the number of leaves. The corresponding tree weight estimations start steep at the beginning, yielding underestimations of the final tree size during the first 10 000 leaves. After approximately 20 000 leaf nodes, the estimations stays very close to the true final tree size of 1 million nodes, which is visualized as a black horizontal line. The tree weight estimation is consistently closer to the actual tree size than the other estimations during most of the search.

The values of the leaf frequency rise very steeply at the beginning of the search, but stabilize very soon and increase at a lower pace afterwards. The steep incline yields underestimations of the actual tree size at first. Later, it shows an overestimation of the actual tree size that is comparable to the SSG estimation. The leaf frequency is not monotonic in general, although it appears strictly increasing in the top part of Figure 8.4.

The number of open nodes increases during the first half of the search and drops during the second half until it reaches zero at termination. (We observe this roughly unimodal behavior in most instances.) Linear forecasts such as DES cannot cope with

⁴For technical reasons, we show the complement $1 - \gamma(Z_k, z_k^*(V_k))$ of the gap.

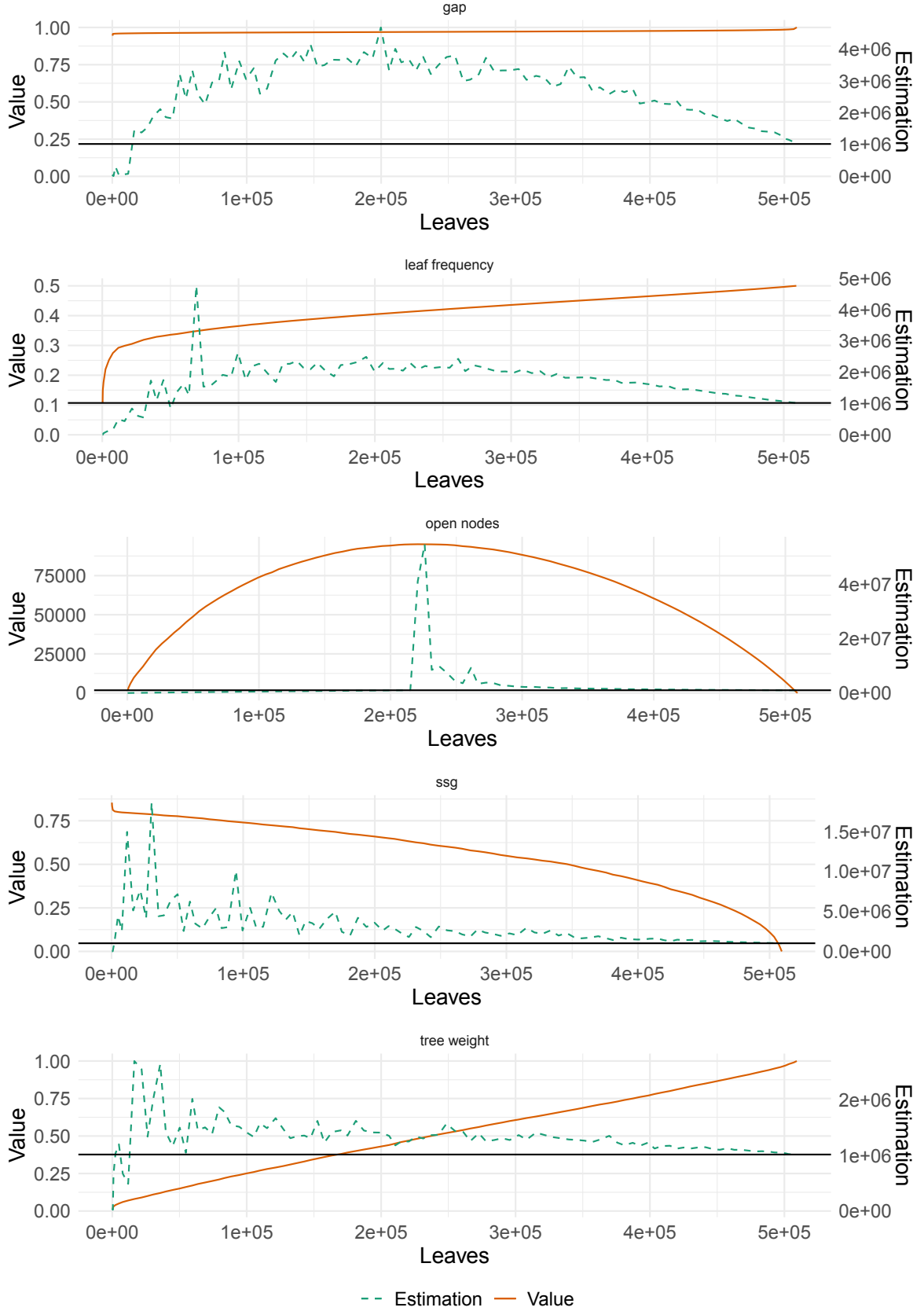


Figure 8.4: Examples of the five time series and their DES estimates as a function of leaves for instance `danoint`. For technical reasons, we show the complement $1 - \gamma(Z_k, z_k^*(V_k))$ of the gap.

this reversing trend by themselves. As long as the number of open nodes increases, also the corresponding trend into the future stays positive, in which case we use twice the number of already explored nodes as estimation, as mentioned in Section 8.4.2. When the number of open nodes starts to decrease, and hence the trend starts to become negative, the first few forecasts overestimate the final tree size, especially at the turning point. For the remainder of the search, the estimations converge to the true tree size.

This example shows that the proposed time series estimations on tree weight and leaf frequency can yield quite accurate predictions early during the search.

8.6 Implementation in SCIP

With the aim to provide a unified and understandable theoretical presentation, the B&B measures from Section 8.3 and the forecasting methods from Section 8.4 have been introduced as functions of the B&B search state T_k . The SSG as well as the time series methods require information from past search states, the former in form of a primal bound updating search state, the latter because double exponential smoothing takes into account the entire search sequence for forecasting.

Clearly, it is computationally prohibitive to explicitly save the entire search state sequence. Furthermore, the measures and time series can be efficiently incremented during the search and need not be recomputed from scratch. This section provides some details about our SCIP implementation of the estimation methods. Our code is encapsulated as an event handler plugin that reacts on node events of the main search. It is invoked when a branching occurs in line 16 in Algorithm 2 or when a node becomes a final leaf in line 14. We extended the node event system of SCIP in order to capture such events even for open nodes that are pruned (e.g., because of a new primal bound). The event handler maintains the statistics $|V_k^{\text{leaf}}|$, $|V_k^{\text{inner}}|$, $|V_k^{\text{open}}|$ for an internal model tree that counts all internal or final leaves as solved. We hence ensure that our model tree is in fact binary, so that our assumptions regarding the tree weight $\omega(T_U) = 1$ or the leaf frequency measure hold at the end of the search.

Some further remarks about specific implementations of the measures and estimations:

Tree Weight, Leaf Frequency, Gap The values of all three measures tree weight $\omega(T_k)$, leaf frequency $\lambda(T_k)$ and gap $\gamma(Z_k, z_k^*(V_k))$ are updated in constant time from their values at T_{k-1} .

SSG The SSG [Özaltın et al., 2011] is more involved than the previous measures because it requires efficient updates of the individual subtree dual bounds for the different subtrees rooted at $V_{\text{pred}^Z(k)}^{\text{open}}$, i.e. the open nodes at the last primal bound improvement. For each subtree, we keep a priority queue of all open nodes $v \in V_k^{\text{open}}$ sorted by their dual bound. This allows for an efficient removal or addition of nodes with

at most logarithmic effort $\mathcal{O}(\log(\max\{|subtree(v)| : v \in V_{\text{pred}^Z(k)}^{\text{open}}\}))$. The value of the SSG only changes if a dual bound defining node is removed from its respective subtree after the node has been branched or pruned. The priority queues are reinitialized at each update of the primal bound.

WBE The weighted backtrack estimation [Kilby et al., 2006] can be computed in constant time from the value of the tree weight measure and the statistic $|V_k^{\text{leaf}}|$ of the model tree.

Profile Estimation For the profile estimation [Cornuéjols et al., 2006], we incrementally update the depth profile \mathcal{D}_k in constant time by adding 1 to the entry $d(v)$ corresponding to the selected node v . The computation of \hat{U}^{profile} can be done in $\mathcal{O}(d_k^{\text{max}})$. In order to save time, we store the statistics $(d_{k'}^{\text{full}}, d_{k'}^{\text{max}}, d_{k'}^{\text{width}})$ at the last step $k' < k$ when the estimation was computed, as well as the estimated tree size. If the statistics have not changed between k' and k , we report the same estimation.

8.7 Learning the Search Completion

We now show how Machine Learning models can be trained to approximate the search completion Γ_k , using measures we introduced in Sections 8.3 and 8.4 as features. Unlike the previous section, we are interested in good approximations of the search completion $\Gamma_k \in [0, 1]$, as our preliminary experiments showed that using the final tree size instead as labels for training does not generalize as well to unseen instances. Using the approximated search completion, a tree size estimate can then be retrieved as we have before, using (8.4).

8.7.1 Data Set

We use the MIP instance sets MIPLIB 3 [Bixby et al., 1998], MIPLIB 2003 [Achterberg et al., 2006], MIPLIB 2010 [Koch et al., 2011], MIPLIB 2017 (benchmark version 2) from Chapter 3, and COR@L [Coral], resulting in a set of 671 unique instances. For the collection of the data, we use our SCIP implementation with all search completion measures and time series forecasts enabled. Our code periodically outputs an estimation snapshot of all methods, namely each time that the tree weight reaches another percent, from which we obtain up to 99 estimation snapshots per instance. We consider the 276 instances which can be solved within 2 hours and require at least 100 nodes, resulting in a data set comprising 16k estimation snapshots as data points.

In order to validate the generalization of the learners on unseen instances, we randomly split the instances of our data set into 80 % training and 20 % test set, such that all records for one instance are fully contained in either the training or the test set.

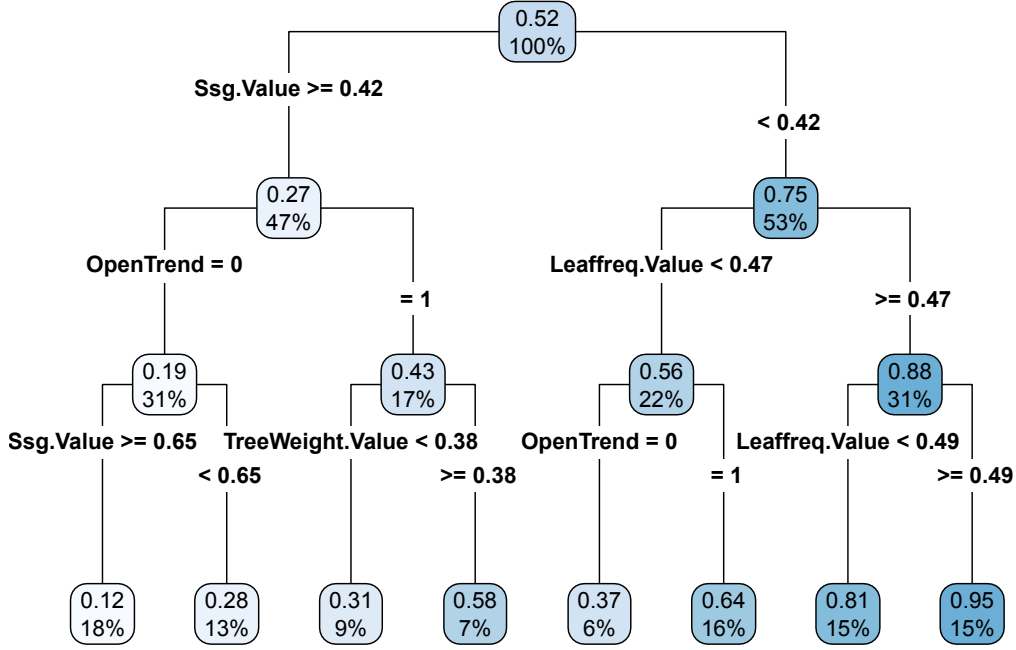


Figure 8.5: A visualization of the learned regression tree. In contrast to the linear models, this tree can only output one out of 8 distinct values as approximation of the search completion.

8.7.2 Training

We formulate the problem to approximate the search completion based on the available measurements as a regression task, i.e., we seek to find a good approximation

$$f(X_k) \sim \Gamma_k$$

that maps features X_k available at *any* search state T_k to the actual, a posteriori search completion Γ_k . To this end, we use nine features: at estimation snapshot T_k (to which there exists a unique time series step $t(k) \in \{1, \dots, C\}$, where C is the adaptive resolution capacity), we take both the measured values and the DES trends of the

- tree weight: $\omega(T_k), s_{t(k)}^{\text{tree weight}}$
- SSG: $\gamma^{\text{ssg}}(T_k), s_{t(k)}^{\text{ssg}}$,
- leaf frequency: $\lambda(T_k), s_{t(k)}^{\text{leaf-freq}}$
- gap: $\gamma(Z_k, z_k^*(V_k)), s_{t(k)}^{\text{gap}}$.

As a ninth feature, we use a Boolean indicator equal to 1 if the time series of the number of open nodes has a decreasing trend component, and 0 otherwise.

We train four different regression models on all nine features, unless indicated otherwise, to learn the search completion:

1. **linear model**: a linear regression;
2. **linear monotone**: a linear regression on only two of the nine features: tree weight $\omega(T_k)$ and SSG search completion $\hat{\Gamma}_k^{\text{ssg}}$, which are both monotone;
3. **regression tree**: a single regression tree;
4. **random forest**: a random forest regression.

A regression tree [Breiman et al., 1983] partitions the training data recursively by selecting a feature and a value to split such that the variance after the split is minimized. The learned regression tree is depicted in Figure 8.5.

Random Forests [Breiman, 2001] are ensembles of regression trees. For a random forest, a (predefined) number N of regression trees is trained on independently bootstrapped subsets of the training data samples, which contain approx. 60 % of training data. Also, the number of features to consider at each split is limited. These choices prevent a random forest from overfitting. An approximation by a random forest regression is the mean approximation of its N regression trees. We conduct the learning procedure with the R package **randomForest**. This package also allows to set a limit on the number of samples at the terminal nodes of the tree, to prevent overfitting. We train a large random forest **random forest big** with $N = 200$ trees and a minimum terminal node size of 25 as well as one called **random forest reasonable** with $N = 100$ trees and a minimum node size of 75. The trained weights of the linear monotone regression on tree weight and SSG yields a mixture of roughly 40 % tree weight and 60 % SSG. After normalization such that the sum equals 1, the corresponding weights of 0.4 and 0.6 ensure that the search completion approximation is monotone increasing.

Remarks. We have also experimented with other regression techniques such as gradient boosted trees [Friedman, 2001] and neural networks to approximate search completion. Furthermore, we also tried to combine the individual tree size estimates (including WBE and profile). In all cases, we omit the obtained results for brevity, which were comparable, but inferior to the accuracy of the presented random forests. In conclusion, it seems that attempting to approximate search completion instead of the actual tree size indeed works far better, since the wide variance of tree size estimations makes fitting a linear model, particularly regression, to the data very difficult and ineffective.

8.8 Evaluation

In this section, we evaluate the estimation quality of all the techniques discussed in this chapter. In Section 8.8.1, we evaluate the search completion estimation techniques from Section 8.3 and 8.7. Then, in Section 8.8.2, we put everything together and evaluate the corresponding tree size estimations against those produced by DES in Section 8.4 and several reference methods from the literature. Lastly, Section 8.8.3 compares the quality of the tree size estimation methods on several of the larger model groups from

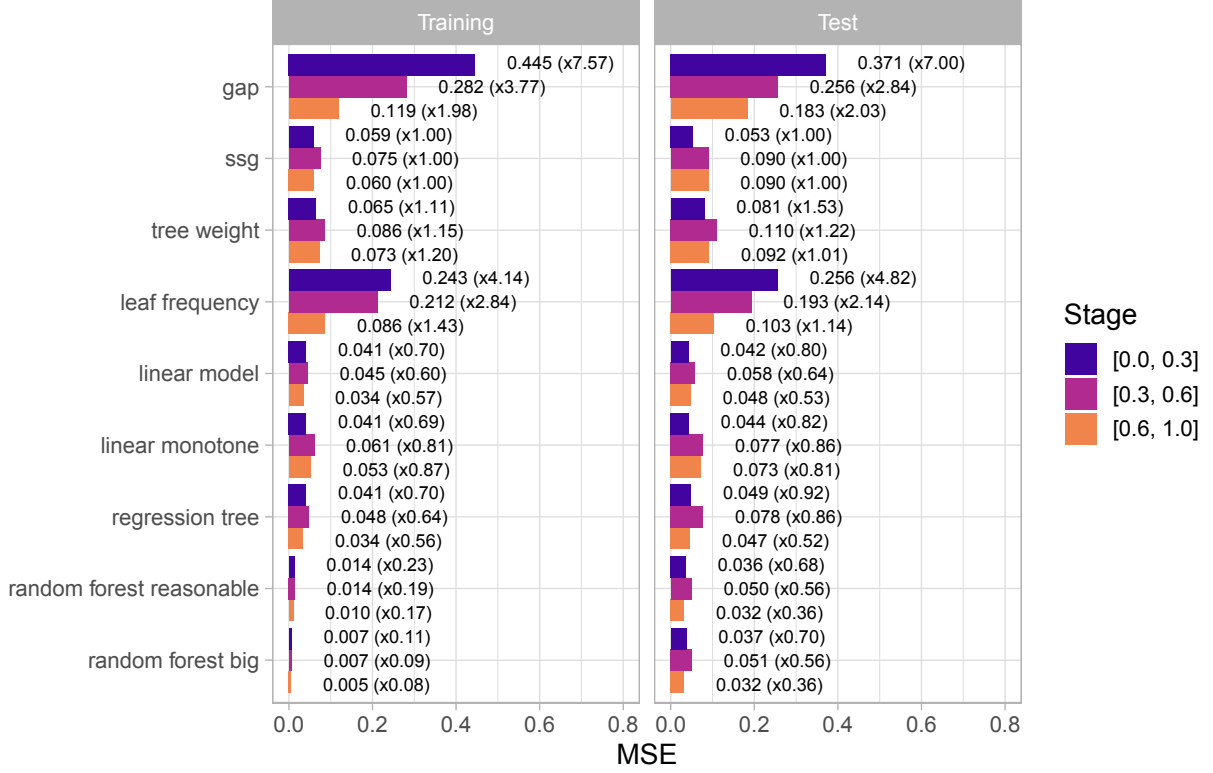


Figure 8.6: MSE between the search completion Γ_k and its approximations.

the MIPLIB 2017 submission pool, (see also Section 3.4), with and without individual training on the model groups.

8.8.1 Comparison of Search Completion Estimation Methods

Figure 8.6 provides a comparison of the quality of the learned approximations from Section 8.7 compared to the quality of the untrained measures tree weight, SSG, leaf frequency, and gap from Section 8.3. We report the *Mean Squared Error* (MSE) $(\hat{\Gamma} - \Gamma_k)^2$ between the approximation and the actual value, which is the target function that the regression procedure attempts to minimize. We report the MSE separately for the training and test set. In parentheses, we show the ratio between the corresponding MSE and the MSE obtained by the SSG method, chosen as a baseline as it is the best performing method among the individual (untrained) measures of search completion. We further categorize the data points into an early, intermediate, and late stage based on the value of the tree weight measure $\omega(T_k)$: we call a data point *early* if $0 \leq \omega(T_k) \leq 0.3$, *intermediate* for $0.3 < \omega(T_k) \leq 0.6$, and otherwise *late*. The four untrained search completion approximations based on gap $\hat{\Gamma}_k^{\text{gap}}$, SSG $\hat{\Gamma}_k^{\text{ssg}}$, tree weight $\hat{\Gamma}_k^{\text{tree weight}}$, and leaf frequency $\hat{\Gamma}_k^{\text{leaf-freq}}$ serve as comparison baselines. Figure 8.6 clearly shows that the gap alone offers the worst individual approximation quality, especially during the early and intermediate stages. Instead, the tree weight and SSG measures provide substantially

closer approximations of search completion throughout all stages, with a slight advantage for the SSG.

Moreover, Figure 8.6 demonstrates that all learned methods improve upon the results of the best individual method SSG throughout all three stages on the test set. The monotone linear regression, which combines the SSG and tree weight measures, and the linear regression on all nine features improve upon the SSG by relative factors between 15–20 % and 20–50 %, respectively. Interestingly, a single regression tree achieves a comparable performance to a linear regression during the late stage, although the regression tree only uses eight different labels (see Figure 8.5).

The two trained random forest models achieve the best test results by a significant margin on the training set. On the test set, they improve upon the SSG by 30 %–64 % in the MSE.

The reason for the better performance of the regression forests compared to linear regression is that the feature values are only used indirectly by the forests, namely for deciding the “bucket” into which a data point falls, but the actual value that the regression forest assigns to an input is not computed as a weighted combination of the input as by linear regression. This allows to better integrate a logical feature such as the decreasing/increasing trend of the open nodes.

Among the tested methods, a random forest regression outperforms all other methods in all respects on both the training and the test set. Regarding the linear methods, it is surprising how well even the monotone estimation based on only tree weight and SSG works. The next section compares the estimation performance of these learned search completions to the individual methods from the previous sections.

8.8.2 Comparison of Tree Size Estimation Methods

We are now ready to compare the tree size estimation quality of all discussed methods. We categorize the discussed methods into four groups based on how the estimation is derived.

1. The first group comprises the four search progress measures from Section 8.3. We treat (a suitable transformation of) each measure as approximation of the search completion Γ_k and compute a tree size estimation using (8.4).
2. Two reference methods, the profile estimation [Cornuéjols et al., 2006] and the WBE [Kilby et al., 2006], are treated as a separate group because they use different approaches.
3. For the third group, all estimates are computed via DES, see Section 8.4. As explained in Section 8.4.3, the considered time series use an adaptive resolution with maximum capacity of $C = 1024$ and are indexed over the leaf index set $\mathcal{K}^{\text{leaf}}$. For each time series, the corresponding parameters α and β from Table 8.2 in Appendix 8.5 are

Method	n	MSE	E	2-Acc	3-Acc	4-Acc
Search Completion Approximation						
gap	861	0.371	36.305	20.7 %	34.7 %	40.0 %
SSG	861	0.053	5.702	43.0 %	58.9 %	66.3 %
tree weight	861	0.081	4.134	30.5 %	45.5 %	61.6 %
leaf frequency	861	0.256	6.590	29.0 %	44.5 %	52.5 %
Custom Estimations						
profile	861	–	79.930	19.5 %	30.3 %	35.7 %
WBE	861	–	4.686	30.4 %	46.9 %	59.1 %
Double Exponential Smoothing						
open nodes	861	–	5.717	40.9 %	52.1 %	58.1 %
gap	861	–	5.686	36.6 %	50.4 %	58.5 %
SSG	861	–	4.695	39.1 %	55.4 %	62.6 %
tree weight	861	–	4.532	35.1 %	51.9 %	60.7 %
leaf frequency	861	–	4.876	42.4 %	56.9 %	61.9 %
Learned Methods						
linear model	861	0.042	7.580	47.0 %	59.7 %	67.4 %
linear monotone	861	0.044	4.088	50.1 %	62.8 %	70.6 %
regression tree	861	0.049	3.447	49.6 %	62.0 %	69.9 %
random forest big	861	0.037	2.835	51.9 %	71.0 %	75.8 %
random forest reasonable	861	0.036	2.830	54.2 %	70.7 %	75.7 %

Table 8.3: Estimate comparison during the early stage ($0 \leq \omega(T_k) \leq 0.3$) on test set.

used to update DES and produce estimations. For the gap time series, we use the same parameters as for the SSG.

4. The last group comprises five learned approximations of search completion using linear or forest regression, see Section 8.7.

For the two groups that approximate search completion, the computed approximation is corrected via $\tilde{\Gamma} = \max\{\hat{\Gamma}, 10^{-6}\}$, which is necessary because the gap and SSG approximations may still be at 0, and the linear regressions may even yield negative approximations in rare cases. As data set, we use the same 16k estimation snapshots as in Section 8.7.1 and the same training/test split. We use the learned predictions based on the training set of Section 8.7 and apply them to the test set. We omit the results for the training set, on which the learned methods from Section 8.7 already have an advantage, and focus on the test set, which comprises a total of 3452 records.

We summarize the obtained estimation ratios on the test set records in three Tables 8.3–8.5, where we distinguish an early, intermediate, and late stage depending on the tree weight value. For every estimation method, we report the geometric mean normalized ratio E (8.3) of its tree size estimation. This value is bounded from below by 1, which would be a perfect estimation. Nine of the tested methods derive their estimation from an approximation of the search completion. For those, we report the MSE between the approximation and the actual search completion. As in [Özaltın et al.,

Method	n	MSE	E	2-Acc	3-Acc	4-Acc
Search Completion Approximation						
gap	1016	0.256	11.142	41.3 %	52.4 %	60.3 %
SSG	1016	0.090	3.891	62.7 %	73.1 %	76.2 %
tree weight	1016	0.110	3.034	53.4 %	75.7 %	81.2 %
leaf frequency	1016	0.193	3.196	54.8 %	62.8 %	69.7 %
Custom Estimations						
profile	1016	–	28.961	34.2 %	38.6 %	44.9 %
WBE	1016	–	3.242	57.1 %	70.5 %	78.7 %
Double Exponential Smoothing						
open nodes	1016	–	3.049	59.0 %	69.9 %	74.3 %
gap	1016	–	3.634	47.7 %	64.0 %	71.2 %
SSG	1016	–	2.924	54.8 %	70.1 %	78.0 %
tree weight	1016	–	3.069	56.2 %	71.2 %	76.2 %
leaf frequency	1016	–	2.926	59.0 %	72.2 %	78.9 %
Learned Methods						
linear model	1016	0.058	2.495	65.6 %	73.8 %	81.4 %
linear monotone	1016	0.077	2.678	62.5 %	75.3 %	78.3 %
regression tree	1016	0.078	2.554	60.4 %	73.5 %	78.9 %
random forest big	1016	0.051	2.273	65.2 %	79.3 %	84.4 %
random forest reasonable	1016	0.050	2.292	65.2 %	79.3 %	83.4 %

Table 8.4: Estimate comparison during the intermediate stage ($0.3 < \omega(T_k) \leq 0.6$) on test set.

Method	n	MSE	E	2-Acc	3-Acc	4-Acc
Search Completion Approximation						
gap	1575	0.183	3.835	65.1 %	76.5 %	81.4 %
SSG	1575	0.090	2.118	78.9 %	84.6 %	89.4 %
tree weight	1575	0.092	1.688	81.4 %	87.2 %	88.5 %
leaf frequency	1575	0.103	1.662	77.8 %	85.3 %	88.8 %
Custom Estimations						
profile	1575	–	31.143	47.9 %	55.6 %	58.2 %
WBE	1575	–	1.714	79.6 %	86.6 %	88.4 %
Double Exponential Smoothing						
open nodes	1575	–	1.637	81.0 %	86.5 %	89.0 %
gap	1575	–	2.175	70.7 %	80.0 %	84.1 %
SSG	1575	–	1.689	78.0 %	84.4 %	87.7 %
tree weight	1575	–	1.681	81.0 %	85.6 %	87.7 %
leaf frequency	1575	–	1.662	81.5 %	88.4 %	90.9 %
Learned Methods						
linear model	1575	0.048	1.581	83.1 %	88.6 %	90.7 %
linear monotone	1575	0.073	1.669	80.2 %	85.3 %	89.7 %
regression tree	1575	0.047	1.511	85.0 %	89.2 %	91.2 %
random forest big	1575	0.032	1.443	86.9 %	90.9 %	92.4 %
random forest reasonable	1575	0.032	1.446	87.2 %	90.8 %	92.6 %

Table 8.5: Estimate comparison during the late stage ($0.6 < \omega(T_k)$) on test set.

2011], we also present the percentage of data points that are ϵ -accurate, i.e., which satisfy $E(\hat{U}, U) \leq \epsilon$ for $\epsilon = 2, 3, 4$.

Analysis of the Search Completion Measures. Among the search completion approximations, the methods **SSG** and **tree weight**, which already showed to have a better approximation quality of the actual search completion than their counterparts **gap** and **leaf frequency**, are also better in terms of normalized ratio. Interestingly, the **tree weight** estimation yields considerably better estimations than **SSG** despite its worse approximation quality, as measured by the MSE.

The **tree weight** search completion achieves the lowest or second-lowest ratio across all search completion methods across stages. Only during the late stage, the **leaf frequency** approximation of search completion yields a better normalized ratio. The two estimations **tree weight** and **WBE** are comparable in that they use the same measure, namely the tree weight measure ω , from which they compute an estimated number of nodes (**tree weight**) or leaves (**WBE**) of the final search tree. However, the tree weight estimation yields better estimations during all stages.

Analysis of Double Exponential Smoothing Forecasts. Throughout all stages, the measures **SSG** and **gap** yield much better estimations within a time series approach than by projecting them as a measure of search completion. Both as search completion

and as time series, a direct comparison between the **SSG** and **gap** estimations always shows a favorable behavior of the **SSG** method. This confirms the findings by Özaltın et al. [2011] that a time series forecasting using **SSG** has a substantially better tree size estimation accuracy than the **gap**. The results also show that all three **gap**, **leaf frequency**, and **SSG** benefit from the use of DES. In contrast, the **tree weight** measure is best used as an approximation of search completion.

The custom method **profile** has by far the largest normalized ratio across all stages. A closer look at the individual records reveals that the **profile** estimation is the only method with serious overestimations. Note that the **profile** estimation does not necessarily converge to the actual tree size at termination, unlike all other tested methods.

The most extreme overestimation of the **profile** method is on instance **ns1952667**, where the estimated tree size of $7 \cdot 10^{44}$ overestimates the actual, moderate tree size of 19k nodes by 40 orders of magnitude.

The instance **ns1952667** is a pure feasibility model, i.e., it has an objective function of zero. Therefore, nodes can only be marked as final if they are proven infeasible. The search finishes as soon as SCIP finds a feasible solution, which happens inside the B&B tree after about 12,000 nodes.

The two measures based on the objective function, namely **gap** and **SSG**, jump from one to zero when the search terminates. The **tree weight** measure is close to zero throughout the search because the final leaf nodes are encountered rather deep in the search tree. This also affects the **WBE**.

Therefore, all DES-based estimates except for **leaf frequency** have to use the fallback strategy introduced in Section 8.4.2 to predict twice the number of currently visited nodes during most of the search. **Leaf frequency** is the most informative measure on this instance because it captures the regularity of terminal nodes during the tree search.

Analysis of Learned Approximations. As one may expect, for all time series, the estimate becomes more accurate at later stages with more data being available. As for the learned methods, the random forests outperform all other tested methods by a considerable margin in terms of ϵ -accuracy and E , as could be expected from their approximation quality measured by the MSE. The large normalized ratio during the early stage of the **linear regression** mostly arises from a few negative approximations of search completion, and despite the applied correction that maps these approximations into the allowed range. Note that the regression forest approximation is always positive, since it is the mean value of a subset of training labels, which are themselves positive search completions.

Despite their good performance, regression forests also have disadvantages. Their estimation cost grows linearly with the number of trees in the forest. In contrast, the

Set	Example	Instances	solved	Records
chromaticindex ⁶	chromaticindex1024-7	121	105	1287
generated	gen-ip002	92	81	7772
iis ⁷	iis-glass-cov	92	69	6765
map ⁸	map06	180	180	12240
network-flow ⁹	g200x740	168	156	3536
opm2	opm2-z12-s8	150	92	1874

Table 8.6: Data collected on homogeneous instance sets.

costs to compute one linear (monotone) regression approximation in the nine-dimensional feature space is negligible.⁵ Random Forests are therefore especially advantageous if the estimates do not have to be computed at every node, but infrequently during the search.

In this section, we have used estimation snapshots across a variety of publicly available MIP benchmark sets, which have been compiled to cover a broad range of MIP applications. For instances from one specific type of application, the learning procedure should ideally be repeated to capture the search process of the B&B algorithm on those instances better than by our pretrained general purpose approximations.

Remarks. One important influence on the behavior of a B&B-solver is the way in which improving solutions are found during the search. One can suspect that initializing the solution process with an optimal solution improves the quality of the predictions of the B&B-tree size. However, evaluating the normalized ratios as above shows that they actually increase slightly. This happens, because fewer records in the last phase of the search are available. If one factors this influence out, the ratios slightly decrease as expected. We do not present detailed results, because our goal was to predict the size of the B&B-tree for a general solution run, in which an optimal solution is not available.

8.8.3 Comparison on Homogeneous Instance Sets

In this section, we compare the estimation accuracy on six different homogeneous sets of MIP instances. All instance sets have been obtained from the public git repository of submissions to MIPLIB 2017.¹⁰ While the MIPLIB 2017 collection of 1065 instances contains at most five instances from each such set, there are many more instances available from the repository. The selection of the six sets, which are shown in Table 8.6, has been made with respect to the following criteria:

⁵Also note that the monotonicity of the linear monotone regression can be slightly more pleasing when used as a progress bar during the search, although it is less accurate on average.

⁶[Le Bodic and Nemhauser, 2015]

⁷[Pfetsch, 2008], 23 instances \times 4 different random seeds.

⁸[Ahmadizadeh et al., 2010]

⁹[Ortega and Wolsey, 2003], 42 instances \times 4 different random seeds.

¹⁰<https://git.zib.de/miplib2017/revised-submissions-final.git>

1. sufficient number of instances,
2. solvability with SCIP during the performance evaluation for MIPLIB 2017, (see Section 3.5.5 for details),
3. large enough trees.

A sufficient number of instances guarantees that we can obtain a meaningful separation into training and test set. For the sets of **iis** [Pfetsch, 2008] and **network-flow** instances, we performed runs with four (default+3) random seeds to inflate the actual instance sets of 23 and 42 instances to 92 and 168 instance-seed combinations. As before, for each set, we first discard instances that could not be solved. Second, we split the obtained records into 80 % instances for training and 20 % for testing. If multiple random seeds were used, the split ensures that the test set contains only unseen instances (and their respective seeds). For each set, we train independent search completion approximations on the obtained training sets.

Figure 8.7 summarizes the accuracy of all discussed methods in terms of the geometric mean normalized ratio E for all test records per instance set. For simplicity, we report the accuracy over the entire search process regardless of the early, intermediate, and late stages measured by tree weight. As in the previous section, we classify the tree size estimates into four groups. **Search Completion** comprises the four measures of search completion from Section 8.3. **DES** comprises five estimates that are derived from DES forecasts as explained in Section 8.4.2. All four measures of search completion also allow a DES forecast and hence appear twice in the figure. The group **Custom** comprises two further reference methods, **WBE** [Kilby et al., 2006] and **profile** [Cornuéjols et al., 2006]. The last group **Learned** finally comprises five different learned search completion methods as presented in Section 8.7. In addition, the last group is enriched by **random forest miplib**, which corresponds to the random forest model **random forest big** from the previous sections 8.7 and 8.8 without further training on the application-specific data sets.

The learned random forests consistently achieve the smallest ratios among the methods. The random forest estimates are particularly accurate on the **map** instances, for which they achieve (almost) best possible ratios of 1.00 and 1.01, respectively. This very good result is possible because the instance set of 180 instances can be further grouped into 9 different subsets of instances. For each subset, the SCIP solution process is identical, because there is only little variation in the input data. The random forests can reliably recognize the similarities in the solution process and always assign the correct search completion even on unseen instances from the test set.

It is noteworthy that the individual estimates based on forecasting or search completion approximation vary substantially between the different sets. For instances from the set **opm2**, all individual forecasts and search completion methods yield ratios between 2.9 and 5.1, but can be effectively combined into a random forest with an

acceptable ratio $E < 2$. Unsurprisingly, the random forests with application specific training always outperform the general random forest `random forest miplib`, while the latter one still achieves a superior performance to the DES forecasting estimates on all tested instance sets, and outperforms the monotone linear regression on four and the linear regression estimate even on five of the six tested instance sets.

8.9 Summary and Outlook

In this chapter, we discussed and compared all state-of-the-art online methods and new methods to predict the size of the B&B tree at termination. We grouped the presented tree size estimates into approximations of the search completion on the one side and time series based estimates on the other side. We improved the time series forecasts by introducing adaptive resolution.

By far the best estimation quality is achieved by combining value and trend components of the individual time series into a learned regression forest that approximates search completion better than any individual method and yields superior estimation accuracy even on unseen test instances. As an additional validation, our study on six different homogeneous instance sets showed that the performance of a random forest can be significantly improved by training on a particular instance set. Nevertheless, using heterogeneous general training data generalizes quite well to outperform most of the individual estimates.

Our results provide clear evidence that accurate tree size estimation requires a combination of several atomic measures such as tree weight and SSG to compensate for their individual weaknesses.

An efficient implementation of these estimates is included into SCIP as of version 7.0. It manifests during the solution process as a new display column of approximate search completion. By default, the display column shows the monotone linear regression because its approximation is easy to explain, improves upon the accuracy of the tree weight and SSG, and can be computed faster than its regression forest counterpart. For an even more accurate search completion approximation, user regression forests can be trained from SCIP log files via an external R script and input into SCIP.

The present work can be extended in various ways. Several of our methods require that the underlying tree is a binary B&B tree, which is the default in most state-of-the-art solvers. Measures such as the tree weight are easily generalized to nonbinary trees at the cost of additional memory and computational effort [Belov et al., 2017].

The time series forecasting methods we used are mainly linear, investigating nonlinear functions or more advanced forecasting techniques might be beneficial. Finally, we hope that this work also inspires the use of the presented methods for algorithmic improvements. One promising direction might be the use of search completion proxies in a massively parallel solver such as UG [Shinano et al., 2012] for better load balancing.



Figure 8.7: Geometric mean normalized ratio for six homogeneous MIP test sets.

We investigate a first algorithmic application of tree size estimation, namely the decision during the tree search whether it may be worthwhile restarting, in the next chapter.

Clairvoyant Restarts

A standard technique in backtracking search consists in restarting the search at the root node, retaining as much information about the previous tree as efficiently possible. This is common practice in Constraint Programming (CP) and Satisfiability (SAT) solvers, for instance. In contrast, restarts are not so commonly used in modern MIP solvers. Instead, the entire search is often performed within a single B&B tree without restarting. Before version 7.0, into which we integrated the work presented in this chapter, SCIP only performed restarts at the end of or sometimes during the root node based on the fraction of fixed integer variables.

We started this study with a simple experiment: in SCIP, we forced a restart after 1000 nodes. On a benchmark consisting of 496 MIPLIB and COR@L instances, this strategy yielded an average 4.7% slow down. However, for the 106 (resp. 88, 72, 48) instances where default SCIP (without the restart) required at least 50k (resp. 100k, 200k, 500k) nodes, this forced restart produced a speed-up of 7% (resp. 9, 17, 18%). Indeed, for instances that require relatively small trees, poor decisions at the top of the tree have a smaller impact, and thus restarts are generally not beneficial, but this trade-off becomes interesting for instances that require large search trees. A rough early estimate of the search tree size could be used to detect such cases.

In this chapter we will use the tree size estimates from Chapter 8 as a criterion for deciding restarts, thereby significantly improving the overall performance of SCIP. As a reference to the predictive nature of tree size estimation on the one hand and the French background of one of the authors of the original conference proceedings [Anderson et al., 2018], on which this chapter is based, on the other hand, we call this new restart methodology *clairvoyant*. This chapter is a rewritten version of the original proceedings paper in the notation of this thesis. The algorithmic description has been reworked, and a broader computational study has been conducted using the tree-size estimates from the previous chapter.

The clairvoyant restart strategy is truly online: it observes the default search and *may* decide to restart. If it does not, there is almost no measurable memory or time overhead.¹ Note that this is different from algorithm selection [Kilby et al., 2006; Lobjois and Lemaitre, 1998], wherein online estimates are used to select an algorithm before the actual search, and always incur a fixed overhead. Our new computational results on MIPLIB 2017 in Section 9.3 show that clairvoyant restarts improve the runtime by 7 % on affected instances, and 20 % on “hard” instances. These results, together with the simplicity of the method, demonstrate the general potential of clairvoyant algorithms.

This chapter is organized as follows. We first discuss the potential algorithmic benefits of restarts during the B&B search in Section 9.1. We describe our clairvoyant restart algorithm in Section 9.2 and present computational results in Section 9.3.

9.1 Current Restart Strategies in MIP Solvers

Restart strategies of the commercial MIP solvers CPLEX, FICO Xpress, and Gurobi are kept as trade secrets. As already mentioned, SCIP only used to restart the solution process at the root node up to version 6.x by default.

At the root node, information obtained from the initial solution to the LP relaxation, valid cutting planes, or improving solutions may lead to many variable domain reductions, for example as a result of reduced-cost tightening (cf. Section 2.7.1), that could not be detected during the initial presolving. If the percentage of fixed integer variables exceeds a threshold (by default, 2.5 % in SCIP), this justifies a restart of the solution process. This criterion ensures that presolvers would fix some variables after a restart, which is expected to have a positive cascading effect on the rest of the search. Hence, restarts were then classified as a type of presolving technique. During a restart, SCIP preserves variable domain reductions, solutions, valid cuts, conflict clauses, and branching history information including pseudo-costs. By default, SCIP may perform arbitrarily many restarts at the root node. After the root node, the B&B algorithm of SCIP commits to a single search tree, no more restarts are performed.

However, besides presolving, such restarts during the search may be beneficial for branching strategies. The branching decisions taken at the start of the B&B search have a significant impact on the number of nodes of the search tree. With the available mechanisms, these decisions are quite uninformed, as strong branching is usually performed with an iteration limit on a small subset of candidate variables, see Section 4.1. Moreover, these candidates are sorted with respect to their hybrid score (2.18), which predominantly uses pseudo-costs. But pseudo-costs are uninitialized at the beginning of the (first) tree search. Therefore, better decisions at the top of the tree can be expected after an in-tree restart, as more branching information is available then.

¹In fact, this statement does not hold for all tree size estimation methods used, as we will see in this chapter.

As far as we are aware of, such restarts during the B&B search in MIP have only been studied by Achterberg [2007a], again using as a criterion the number of globally fixed variables. However, using this criterion, restarts appeared to be detrimental to performance. The results, however, were inferior compared to the SCIP default strategy. As Achterberg [2007a] pointed out, additional global variable fixings seem to occur mostly at the final stage of the solution process, when the computational overhead of rebuilding a new search tree from scratch becomes too high. The author concluded that “in order to make good use of delayed restarts, one has to invent different criteria for their application”. Our clairvoyant restart strategy addresses this disadvantage by using tree size estimation instead of the number of global variable fixings.

9.2 A Clairvoyant Restart Strategy

We present our clairvoyant restart strategy in Algorithm 5 as an extension of the B&B Algorithm 2, which it uses as subroutine to advance the search. After each iteration, it may decide based on tree size estimation to reset the recorded search state to the root node and construct a new search tree.

Concretely, besides a MIP P as its first and obvious input, Algorithm 5 receives additional input parameters that control how often and when the search is reset to the root node. The most important parameter is the limit ρ on the number of restarts. If ρ is set to zero, we conduct a B&B search without an in-tree restart. If ρ is larger than zero, three additional ρ -dimensional parameter vectors control the details of the restart.

At the beginning, Algorithm 5 initializes a search state sequence T . In addition, two counters are initialized. r is initialized to one, always indicating the number of the next restart that the search should perform. The second counter l is incremented every time the current tree size estimation exceeds the acceptable limit. In line 6, the B&B algorithm is called to advance the current search state sequence T by one additional search state. Assuming that we haven’t reached the limit ρ yet, a tree size estimation \hat{U} is computed from T .

In round $r \leq \rho$ of Algorithm 5, the following parameters are used for the restart decision:

- κ_r^{init} is an initial number of search states to wait before a restart decision.
- κ_r^{lim} is a limit on the number of consecutive large tree size estimations to trigger a restart.
- ϕ_r^{clair} is a factor to compute the threshold for an estimated tree size to be large, relative to the index k of the current search state.

A threshold on the estimated tree size in round r of Algorithm 5 is given as factor ϕ_r^{clair} relative to the current iteration k . If \hat{U} exceeds k by at least ϕ_r^{clair} , the second

9. Clairvoyant Restarts

counter l is increased. If l exceeds the limit κ_r^{lim} , the search state sequence T is reset to the initial search state except that the current primal bound Z_k and solution pool S_k are not reset. In practice, cuts, conflict clauses, and branching history information such as pseudo-costs are also preserved during a restart.

This resetting of the tree to its root state is a restart of the B&B search. From the next iteration on, the calls of the B&B algorithm in line 6 will consecutively expand a new search tree in round $r + 1$, and eventually trigger another restart, if the limit ρ has not been reached, yet.

Algorithm 5: branch-and-bound search with clairvoyant restarts

Input: MIP P , restart parameters: limit on the number of restarts $\rho \in \mathbb{N}$.
initialization thresholds $\kappa^{\text{init}} \in \mathbb{N}^\rho$, limits $\kappa^{\text{lim}} \in \mathbb{N}^\rho$, factors $\phi^{\text{clair}} \in \mathbb{R}^\rho$,
Output: Optimal objective value Z^{opt} for P

```

1  $k \leftarrow 0, V_0^{\text{leaf}} \leftarrow \emptyset, V_0^{\text{inner}} \leftarrow \emptyset, V_0^{\text{open}} \leftarrow \{v_0\};$ 
2  $Z_0 \leftarrow \infty, z_0^*(v_0) \leftarrow -\infty, S_0 = \emptyset;$ 
3  $T \leftarrow \{(V_0^{\text{inner}}, V_0^{\text{leaf}}, V_0^{\text{open}}, Z_0, z_0^*, S_0)\};$  // initialize search state sequence
4  $r \leftarrow 1, l \leftarrow 0;$ 
5 while  $Z_k > Z_k^*$  do
6    $T \leftarrow \text{branch-and-bound}(T, 1);$  // perform a single B&B iteration
7    $k \leftarrow k + 1;$ 
8   if  $r \leq \rho$  then
9     if  $k \geq \kappa_r^{\text{init}}$  then
10      Compute tree size estimation  $\hat{U}$  from  $T$ ;
11      if  $\hat{U} \geq \phi_r^{\text{clair}} \cdot k$  then
12         $l \leftarrow l + 1;$ 
13      else
14         $l \leftarrow 0;$  // reset  $l$ 
15      end
16      if  $l \geq \kappa_r^{\text{lim}}$  then // perform a restart
17         $T \leftarrow \{(\emptyset, \emptyset, v_0, Z_k, z_0^*, S_k)\};$  // reset search state sequence
18         $k \leftarrow 0, l \leftarrow 0, r \leftarrow r + 1;$ 
19      end
20    end
21  end
22 end
23 end
24 return  $Z_k;$ 

```

The correctness of Algorithm 5 follows immediately from the correctness of the B&B algorithm and the fact that after at most ρ restarts, Algorithm 5 conducts an uninterrupted restart-free B&B search. Depending on the difficulty of P and the choice of clairvoyant parameters, the algorithm may already finish in a round prior to the final round.

The interested reader may wonder whether and where the aforementioned root restarts based on integer variable fixings occur in Algorithm 5. We consider root restarts as internal part of the root node processing at iteration $k = 1$ of Algorithm 2, i.e., we allow one or even several root restarts as part of the call to the B&B algorithm in line 6. The only search state we observe and record is the state after the final root restart and consecutive (re-)presolve and processing of the root node, which either terminates the search or results in the first branching.

What are the potential benefits of restarting the search at an intermediate state of the tree? Briefly, after an in-tree restart, the B&B algorithm can use multiple sources of valuable search information to produce a better, i.e., smaller search tree. First, just as for root restarts, presolving may also use global variable domain reductions from the previous search tree more intensely, e.g., during probing. Also, the current incumbent solution and in particular its objective value Z can be used by presolving to deduce further variable fixings.

Another very important piece of information are pseudo-costs. We have argued already that one of the major drawbacks of pseudo-costs is that they are not available at the beginning of the search, when the topmost, most relevant branching decisions of the search must be decided. Even strong branching is no complete remedy here, because in practice strong branching is often applied to only a handful of candidates near the top of the search due to performance considerations.

After an in-tree restart, however, the B&B search has much more historical branching information available for many variables, which can be used for a fine a priori ranking of the fractional candidates, such that strong branching only needs to decide between a few pre-selected candidates.

Historical branching information is not restricted to pseudo-costs, but also includes inference information and conflict clauses learned during previous search tree(s) as part of the hybrid score (2.18). Finally, learned information from the adaptive heuristic strategies presented in Chapters 6 and 7 can be readily carried over during an in-tree restart.

9.3 Computational Results

In practice, we will restrict ourselves to $\rho = 1$ restarts to keep the experimental setup reasonable. The parameters κ^{init} and κ^{lim} have the role of safeguards against the potentially high variance of \hat{U} . First, a restart is only triggered after the condition in line 11 is satisfied for κ^{lim} consecutive k 's. Second, no restart is performed until $k \geq \kappa^{\text{init}}$, to allow for a suitable initialization of the tree size estimation. In our experiments, ϕ^{clair} is set to $\{10, 25, 50\}$. Together with $\kappa^{\text{init}} = 1000$, this means that we may only restart trees with $\hat{U} \geq \{10 \cdot 1000, 25 \cdot 1000, 50 \cdot 1000\}$ nodes.

We test our clairvoyant restart strategy with six different types of tree size estimation presented in Chapter 8: **monotone** turns the learned monotone approximation of search completion into an estimation of the remaining tree size as shown in Equation (8.4). **reg forest** uses the smaller of the two trained random forests from Chapter 8 (the so-called **random forest reasonable**) as approximation of search completion, which is then transformed into an estimation of tree size again according to Equation (8.4). Recall that the conversion between an estimated search completion and an estimated tree size is straightforward. A clairvoyant restart is performed if the estimated tree size exceeds the current iteration k by at least a factor of ϕ^{clair} , which is equivalent to stating an estimated search completion of at most $\frac{1}{\phi^{\text{clair}}}$. The largest factor ϕ^{clair} of 50 that we use in this experiment therefore corresponds to an estimated search completion of 2 % or less.

We test four more types of clairvoyant restarts based on different double exponential smoothing (DES) tree size estimations using adaptive resolution as explained in Chapter 8. Concretely, we test tree size estimations based on the gap time series with setting **gap**, based on the SSG time series with setting **ssg**, based on the tree weight time series with setting **tree weight**, and finally based on the leaf frequency time series with setting **leaf freq**. Each of these four types of tree size estimations observes the corresponding value, e.g., tree weight, and extrapolates after how many additional time series steps the time series reaches its target value, such as 1 in the case of the tree weight.

Recall that with increasing tree size, adaptive resolution compresses several branch-and-bound iterations into a single time step. The estimated remaining number of time series steps is turned into a tree size estimation using Equation (8.11). We point out that this DES tree size estimation is fundamentally different from an approximation of search completion. For example, the tree weight may already exceed 2 % early in the search, because some shallow nodes of the branch-and-bound search tree could be pruned early. If afterwards, the tree weight keeps stalling for a while, the DES method will reflect this by increasing its tree size estimation. Therefore, for all four types of DES based estimation, a clairvoyant restart may still be performed even though the corresponding search completion already exceeds $\frac{1}{\phi^{\text{clair}}}$.

The two learned search completion approximations as well as the four DES tree size estimations are combined with three different values of $\phi^{\text{clair}} = \{10, 25, 50\}$ into 18 different settings in total. We use the newest SCIP version 7.0.2, which had just been released by the time of this writing. Clairvoyant restarts have been enabled since SCIP version 7.0 [Gamrath et al., 2020]. Since then, the default version of SCIP corresponds to the setting **tree weight** together with $\phi^{\text{clair}} = 50$.

In order to show the impact of clairvoyant restarts onto the search, we compare the battery of candidate settings with two baseline settings: **no clairvoyant** disables clairvoyant restarts by setting $\rho = 0$. Within this **no clairvoyant** setting, restarts at

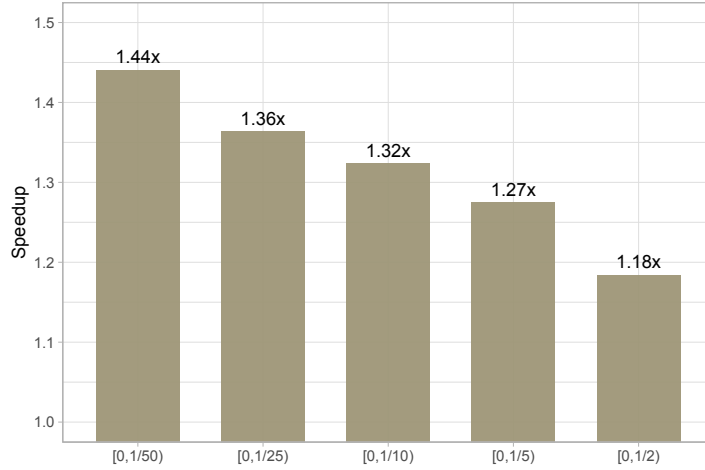


Figure 9.1: Speedup factors for different a-posteriori restart ratios $U^{\text{restart}}/U^{\text{bb}}$.

the end of the root based on the fraction of fixed integer variables may still be performed by SCIP.

A second baseline setting, **0-restart**, disables both root and clairvoyant restarts entirely. The setting **no clairvoyant** represents the default settings of SCIP prior to version 7.0. As test set, we use the benchmark set of MIPLIB 2017 comprising 240 instances, see also 3, using the default + two nondefault random seeds, yielding a total of 720 instances. All experiments are conducted on a cluster with 48 nodes equipped with Intel Xeon Gold 5122 at 3.60GHz and 96GB RAM. Jobs were run exclusively on a node. The time limit was 1h. Details for each problem, seed, and setting of this experiment can be found in Table E.1 in the Appendix.

Clairvoyant restarts are based on an estimation of the current search completion. If a clairvoyant restart was performed at U^{restart} nodes (search states) of Algorithm 5, we can evaluate the actual search completion at the moment of this restart exactly by comparing it with the total number of nodes U^{restart} required by **no clairvoyant** as $U^{\text{restart}}/U^{\text{bb}}$.

Figure 9.1 shows the obtained speedup factors for different such a-posteriori search completions. Concretely, we record for any combination (instance, seed, setting) for which a clairvoyant restart occurred the number of nodes U^{restart} after which the method restarted. We compare U^{restart} against the total number of B&B nodes U^{bb} that the setting **no clairvoyant** without clairvoyant restarts consumes. We show the obtained speedups in shifted geometric mean solving time if the search completion $U^{\text{restart}}/U^{\text{bb}}$ falls into one of the different bins shown on the horizontal axis.

We only include instances (problem-seed combinations) for which a clairvoyant restart was executed, for which the measured search completion falls into one of the categories, and for which either the included case or the setting **no clairvoyant** finished within the time limit. Some of the intervals match the tested options for ϕ^{clair} .

Figure 9.1 shows clear performance benefits in shifted geometric mean if a restart was triggered at a search completion of at most 50 %. The sooner the restart happens,

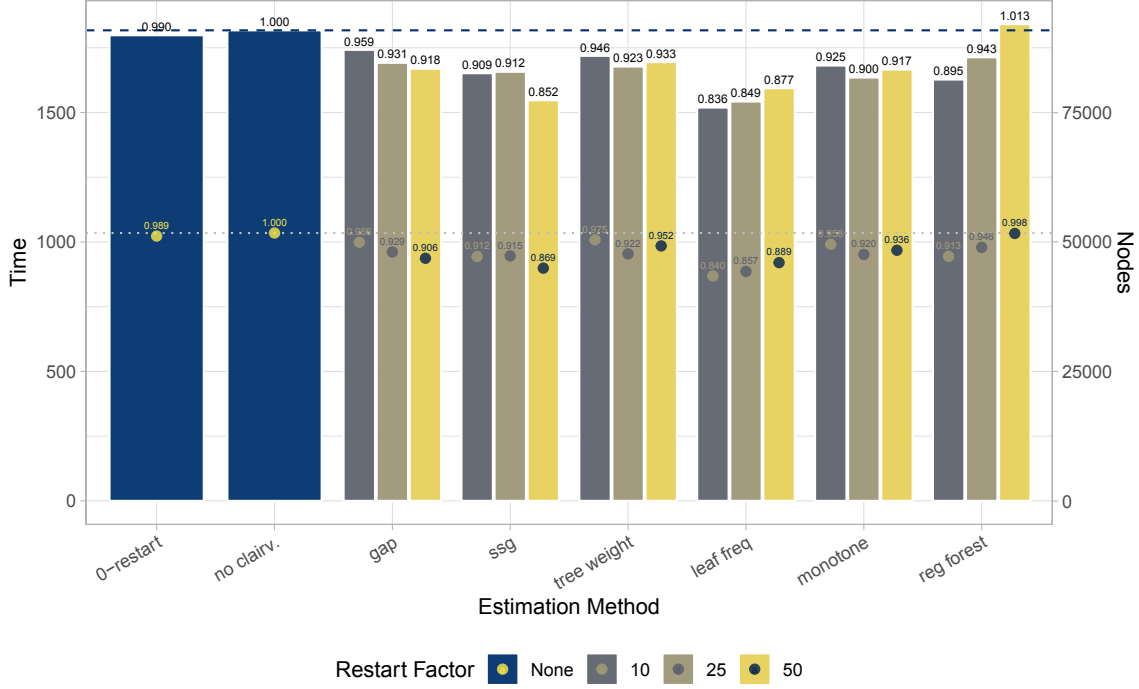


Figure 9.2: Computational results for instances where one setting requires at least 1000 seconds for solving. Each bar has the shifted geometric mean time as height, whereas its text label shows the time relative to `no clairvoyant` as our baseline setting.

the larger is the speedup on average. In the leftmost category of at most 2% search completion before the restart, we see a time improvement of 44%.

Note that there is a subtle bias in the evaluation of this result against the setting `no clairvoyant`. Namely, in order to fall into the leftmost category, the setting `no clairvoyant` must require a search tree of 50k or more nodes.

A possible explanation for the speedups is that the search algorithm may lack sufficient information to make good branching choices at the top of the search, but can use such information from the first search tree after the restart to build a better tree.

For this performance comparison, we conducted a total of 14400 jobs, where one job corresponds to one of the 240×3 (problem \times seed) combinations solved with one of the 20 compared settings, including the baseline settings `no clairvoyant` and `0-restart`. We present a glimpse into this performance data in Figure 9.2.

As in previous chapters, we denote the combination of an input MIP and a random seed as one instance. We give the obtained results for all 20 settings on the subset of solvable instances for which at least one setting required 1000 seconds or more, also denoted by $[1000, \text{tilim}]$ in bracket notation. This subset represents the hardest, yet solvable instances in our test bed.

For this subset, we show in Figure 9.2 the obtained time and node results per setting. The individual numbers are aggregated by a shifted geometric mean, with a shift of 1 for time and 100 for nodes. The vertical bars correspond to the shifted geometric mean runtime, whose scale is shown on the vertical axis on the left.

Above each bar, we indicate the time performance relative to the baseline setting `no clairvoyant`, whose particular height is extended across the figure via a dashed horizontal line. Numbers smaller than one represent an improvement in time.

Each of the clairvoyant restart methods comes in three versions, depending on the setting of the restart factor $\phi^{\text{clair}} \in \{10, 25, 50\}$. We immediately notice that the baseline setting is the slowest setting in this test except for one clairvoyant method `reg forest`, which is 1.3% slower.

Perhaps surprisingly, `0-restart` is on par with `no clairvoyant` in terms of the solving time and solving nodes. We have reported a similar observation already in [Anderson et al., 2018] with a previous version of the clairvoyant restart code based on SCIP 6.0. The most likely explanation is that the default restart parameters were more efficient in a version of SCIP that predates versions 6.0 and 7.0, but have not been re-calibrated.

The best timing results are obtained with `leaf freq`, which speeds up the solving time for this subset by 12–17%. A comparably good speedup of 15% is obtained using `sbg` in combination with a restart factor of 50.

Also `reg forest` achieves a speedup by more than 10%, albeit with the opposite, i.e., least conservative restart factor tested. The DES variants using `gap` and `tree weight` are inferior to `sbg` and `leaf freq`. The other baseline setting in this experiment is `0-restart`, which performs neither clairvoyant restarts nor restarts at the root.

Absolute node results are shown as filled circles with respect to the vertical scale on the right. To be absolutely fair, for the clairvoyant methods, the node results include the first search tree before the restart was performed. As for time, the more interesting relative comparison with `no clairvoyant` is indicated above each circle.

Overall, the relative node results are strikingly correlated with the observed speedups. This is a remarkable result because all clairvoyant methods explore a search tree of sometimes several thousand nodes prior to their restart decision. Admittedly, Figure 9.2 does not give a clear indication whether more conservative/larger restart factors are preferable, the best choice depends on the estimation method with which it is used.

We show the number of performed tree restarts in Figure 9.3. In contrast to the previous figure, here we summarize the number of tree restarts for all 720 tested instances. We also include the nonclairvoyant settings `no clairvoyant` and `0-restart` for clarity.

One surprising result is that `reg forest` in its most conservative setting with $\phi^{\text{clair}} = 50$ performs no tree restarts at all. The slight performance degradation that we saw in Figure 9.2 for this setting compared with `no clairvoyant` therefore indicates a nonnegligible overhead caused by the evaluation of the regression forest after each search state.

All other settings perform a substantial number of tree restarts. As one might expect, the number of restarts consistently decreases if ϕ^{clair} is increased. We observe that the settings `sbg` and `tree weight` perform restarts much more often than the rest.

9. Clairvoyant Restarts

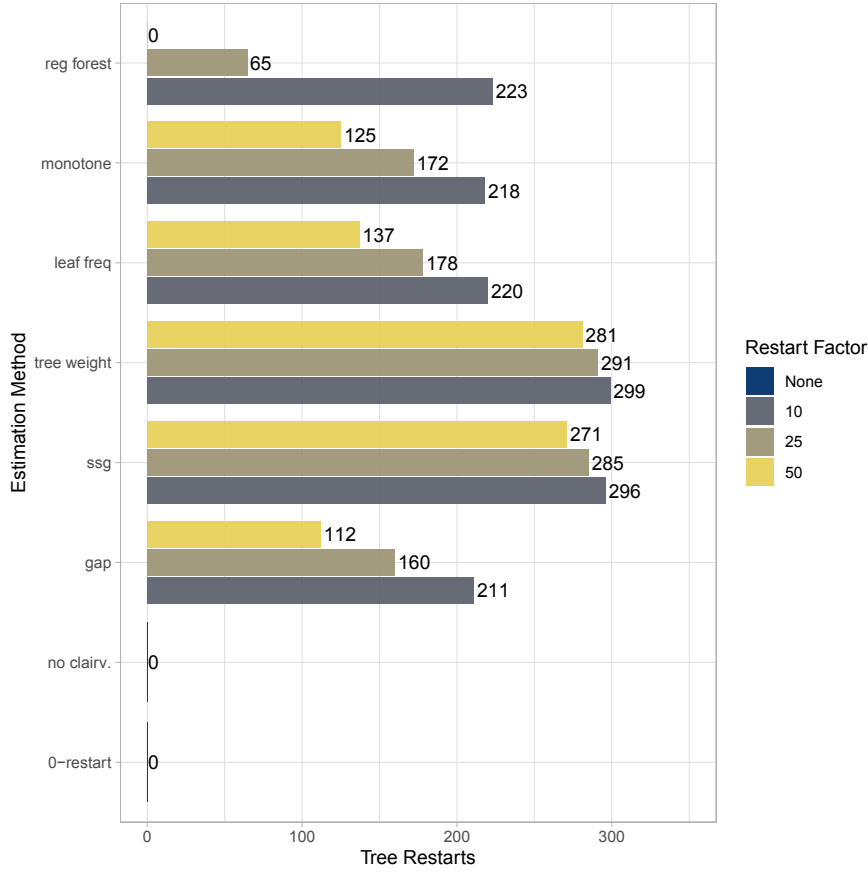


Figure 9.3: Number of performed Tree Restarts for each estimation method and restart factor ϕ^{clair} .

Furthermore, for these two settings, the choice of ϕ^{clair} has less influence on the number of tree restarts than for other settings.

In the previously shown Figure 9.2, the clairvoyant method **leaf freq** emerged as clear winner for the hard subset of instances. We focus on the results with this setting in more detail in Table 9.1. In this table, we give the time and node results obtained with **no clairvoyant** as our baseline setting, and the relative time and node results for the competing settings **0-restart** and **leaf freq**.

Table 9.1 shows the solving time in seconds and the number of solving nodes in columns **time** and **nodes**. As already mentioned, if a restart was performed, **nodes** is the sum of explored nodes in both search trees. For a better comparison, the two columns **time_Q** and **nodes_Q** give the relative performance compared to the baseline setting **no clairvoyant**.

Furthermore, the table shows the number of solved instances in column **solved**, and the total number of root restarts and clairvoyant restarts across all instances in the two columns **root** and **tree**, respectively. It is possible that the solution process of a single instance contributes multiple root restarts, but never more than one clairvoyant restart.

We consider in Table 9.1 five different groups of instances, each in a separate row: All 720 instances are compared in the first row. The second row “affected” summarizes the

	#	no clairvoyant				0-restart			leaf freq ($\phi^{\text{clair}} = 10$)				
		solved	time	nodes	root	solved	time _Q	nodes _Q	solved	time _Q	nodes _Q	tree	root
all	720	338	798	4764	175	340	1.002	1.005	344	0.972*	0.967	220	189
affected	289	122	1244	53608	175	124	1.007	1.012	128	0.936	0.923	220	189
[1000,tilim]	90	82	1950	60588	20	84	0.997	1.001	88	0.809*	0.809*	49	23
$[0, 999]^{U^{\text{bb}}}$	166	166	43	94	84	166	1.000	1.070	166	0.998	1.000	0	84
$[1000, \infty]^{U^{\text{bb}}}$	180	172	508	34624	41	174	1.006	1.040	178	0.895*	0.887*	79	48

Table 9.1: Computational Results for the best performing clairvoyant setting **leaf freq** with a restart factor $\phi^{\text{clair}} = 10$. The best results in a category (**solved/time/nodes**) are highlighted in **bold**. Results that are significant according to a Wilcoxon signed rank test are indicated **separately***.

289 instances on which a restart occurred. The third row gives the results for instances falling into the time bracket [1000,tilim] as explained above, for which one setting must have used 1000 seconds or more. Note that in contrast to Figure 9.2, we only use the three displayed settings to filter the instances for this bracket, such that the 90 instances compared in this table are now a subset of the instance bracket used for Figure 9.2. Similarly to the brackets with respect to solving time, the last two rows filter instances based on the size of the largest search tree. The row $[0, 999]^{U^{\text{bb}}}$ summarizes the subset of solvable instances for which all search trees stayed within 999 nodes, whereas the row $[1000, \infty]^{U^{\text{bb}}}$ shows results on the complementary set of solvable instances where at least one of three settings required 1000 nodes.

We see that overall, **leaf freq** clearly wins in four out of the five groups with respect to time, nodes, and number of solved instances. We highlight results that are significant according to a Wilcoxon signed rank test at a threshold of $p < 0.01$. The time and node results of the group “affected” have p -values of 0.011 and 0.025, respectively. Although they do not meet our very conservatively chosen confidence threshold of 1 %, we consider also the results on “affected” as “borderline significant”. Interestingly, we observe that **leaf freq** has the highest number of root restarts. Of course, it performs at least as many root restarts as **no clairvoyant**. The higher number of root restarts accounts for additional restarts at the root node of the second search tree after a clairvoyant restart. Since root restarts are triggered if a certain fraction of integer variables have been fixed during the (first or second) root, this indicates that clairvoyant restarts sometimes help to reduce the remaining search space further, which is one of the reasons why the second search tree is often significantly smaller than the first.

Summarizing our findings, almost all clairvoyant restart strategies achieve a considerable speedup compared to the previously available root restart strategy in SCIP. The speedup is particularly high on the set of very hard instances. We saw that almost all settings using clairvoyant restarts have a positive effect on the total tree size. Recall that the node related columns **nodes** and **nodes_Q** are based on the total number of nodes of both branch-and-bound search trees before and after the clairvoyant restart. The best performance has been achieved with the new leaf frequency measure that

we introduced in Chapter 8. The most accurate tree size estimation method from the previous chapter, `reg forest`, did not perform as well in our experiment. In its most conservative setting, no restart was performed at all. On the contrary, since no restart was performed by `reg forest`, we observed a measurable overhead in node processing time compared to `no clairvoyant`.

9.4 Summary

In this chapter, we have introduced clairvoyant restarts for MIP, during which we reset the branch-and-bound algorithm 2 if the remaining tree size is estimated to exceed the current number of visited nodes by a large factor. We have formalized our clairvoyant algorithm 5 and tested it for six different types of tree size estimation that were developed in Chapter 8. Especially on harder instances that require a substantial number of search tree nodes and/or a significant time to solve, we obtained substantial improvements with our clairvoyant algorithm. We emphasize again that our clairvoyant algorithm achieves a substantial node reduction even when both search tree sizes are added together.

As we have seen, the precision of the individual estimates plays a minor role; a rough estimation suffices. Our most accurate tree size estimation method, namely a search completion approximation based on random forest regression, is outperformed by the tree size estimations based on double exponential smoothing, which are also faster to compute.

The double exponential smoothing estimations can cope better with changing trends observed during the tree search and are able to adapt their tree size estimation accordingly.

An interesting future direction might be to try to smoothen the search completion approximation from the random forest, and to apply adaptive resolution to it to reduce the number of random forest evaluations when the search tree size increases. Another possible improvement could consist of different settings that are applied before and after a clairvoyant restart, similarly to the different settings used during the three solving phases introduced in Chapter 5. For example, the strong branching effort spent before and after a clairvoyant restart may be recalibrated such that the search tree before the restart is traversed quicker. After the restart, the B&B algorithm may try to generate more cutting planes at the root node if branching has proven less efficient at reducing the gap than cuts have.

Our presented Clairvoyant Algorithm 5 already incorporates the possibility to perform multiple restarts, which has proven beneficial in particular in CP and SAT. The number of explored search nodes between restarts need not be constant, but could either be geometrically increased after each restart, or restarts could be scheduled according to the restart sequence introduced by Luby et al. [1993]. Obviously, if the search tree might

get restarted more than once, this allows for even more options how to fine-tune the search parameters.

A last direction for future work could be the study of partial restarts of the search, that reset the search state sequence to a previous search state other than the root node, at which a decision was taken that should be revised.

10

Conclusion

We started this journey with the goal of turning SCIP into an intelligent solver that adapts to the MIP instance at hand. To this end, we first introduced new notions of reliability, namely hypothesis- and relative error reliability. Both of them take the variance of pseudo-costs into account to dynamically modulate the required strong branching effort at a node and improve the dual integral by up to 15 %. We introduced new adaptive heuristic frameworks around large neighborhood search and diving heuristics that quickly learn which of the available heuristics should be used more frequently. Enabling these heuristic frameworks showed time improvements by up to 10 % and primal integral improvements by up to 37 %. We also introduced an adaptive LP pricing strategy, which improved the LP throughput by 14 %.

Furthermore, we proposed two modifications of the B&B algorithm itself. The first modification considers the search as a sequence of three distinct solving phases. By dynamically adapting the solver parameters to the goal of each phase, we achieved a speed-up of up to 15 % on hard instances, which we showed to be near optimal. The second enhancement of the B&B search is a tree-size estimation based restarting technology called clairvoyant. Therefore, we first combined existing and newly proposed measures of the search progress into an estimation of the final search tree size, which is more accurate than the de-facto standard measure, the gap, by one order of magnitude.

Performance results of the clairvoyant restarts, which improve the solver run time by 20 % on hard instances, thereby underline that this technique is a real enhancement over the classical “one-tree-fits-it-all” B&B algorithm. We see our clairvoyant restarts as an answer to a conjecture by Achterberg [2007a, p. 163], that “in order to make good use of delayed restarts one has to invent new criteria for their application”. We believe that clairvoyant restarts, in combination with the underlying tree-size estimation, may find widespread application in MIP solver technology. Since a recent release, FICO Xpress also performs restarts during the search. As another result of this work, the

periodic output log of SCIP reports on the estimated search completion. This measure is (on average) an order of magnitude more accurate for predicting termination than the classical gap between the primal and dual bounds.

Algorithmic developments often come hand-in-hand with rather technical, infrastructural innovations under the hood of SCIP, which play a major role for its maintainability. From now on, additional techniques introduced into the heuristic frameworks around large neighborhood search and diving heuristics will be calibrated automatically within their allocated global computational budget.

Besides algorithmic advancements, this thesis describes methodological contributions reaching beyond SCIP as a solver. MIPLIB 2017 has broadened the availability of public MIP benchmark sets. It is not only larger than its predecessors and more challenging for today's codes, but has also been derived in a transparent, data-driven process for the first time in the history of the Mixed-Integer Programming Library. Additionally, it comes in a modern design on the web. Since its release in 2018, MIPLIB 2017 has served researchers around the globe as a platform to provide improving solutions to challenging MIP instances from numerous different academic and real-world applications, many of which are unsolved to the present day.

IPET, which I started as a hobby project to produce tables for my bachelor thesis, has matured over the years into an indispensable tool for evaluating solver benchmarks. It has also found its way into many articles and some doctoral theses. Today, IPET is applied daily as the evaluation engine of choice on the continuous integration servers of the SCIP group.

After reconsidering this compilation of adaptive algorithmic strategies, I see some new aspects that could be improved. This is in parts due to the heuristic nature of the methods and their complex interaction with and within the solver. There might always be a better confidence level for hypothesis reliability for a specific use case, a better initial fixing rate for the RENS neighborhood inside ALNS, or a better number of initial nodes to wait before the first decision when to perform a clairvoyant restart. Having re-read and compiled the different chapters of this thesis, I feel inspired to challenge certain design choices that were made in the original work on the topic, to tweak the methods to improve them a little more, and to further analyze the rich data sets that were collected in the preparation of many of our methods.

The good news: This is entirely possible for me as well as you, the reader. Just download SCIP and start to play with its source code, which contains all of the algorithmic technology described in this thesis.

Happy SCIP'ing!

References

- T. Achterberg. *Constraint Integer Programming*. PhD thesis, Technische Universität Berlin, 2007a.
- T. Achterberg. Conflict analysis in mixed integer programming. *Discrete Optimization*, 4(1):4 – 20, 2007b. ISSN 1572-5286. DOI:10.1016/j.disopt.2006.10.006.
- T. Achterberg. SCIP: Solving Constraint Integer Programs. *Mathematical Programming Computation*, 1(1):1–41, 2009.
- T. Achterberg and T. Berthold. Improving the feasibility pump. *Discrete Optimization*, Special Issue 4 (1):77 – 86, 2007.
- T. Achterberg and T. Berthold. Hybrid branching. In W. J. van Hoes and J. N. Hooker, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 6th International Conference, CPAIOR 2009*, volume 5547 of *Lecture Notes in Computer Science*, pages 309–311. Springer, 2009.
- T. Achterberg and R. Wunderling. Mixed integer programming: Analyzing 12 years of progress. In M. Jünger and G. Reinelt, editors, *Facets of Combinatorial Optimization*, pages 449–481. Springer, 2013. ISBN 978-3-642-38189-8. DOI:10.1007/978-3-642-38189-8_18.
- T. Achterberg, T. Koch, and A. Martin. Branching rules revisited. *Operations Research Letters*, 33(1): 42–54, 2004.
- T. Achterberg, T. Koch, and A. Martin. MIPLIB 2003. *Operations Research Letters*, 34(4):361–372, 2006.
- T. Achterberg, T. Berthold, and G. Hendel. Rounding and propagation heuristics for mixed integer programming. In D. Klatte, H.-J. Lüthi, and K. Schmedders, editors, *Operations Research Proceedings 2011*, pages 71–76. Springer Berlin Heidelberg, 2012.
- T. Achterberg, R. E. Bixby, Z. Gu, E. Rothberg, and D. Weninger. Presolve reductions in mixed integer programming. *INFORMS Journal on Computing*, 2019. DOI:10.1287/ijoc.2018.0857.
- K. Ahmadizadeh, B. Dilkina, C. P. Gomes, and A. Sabharwal. An empirical study of optimization for maximizing diffusion in networks. In *Principles and Practice of Constraint Programming*, volume 6308 of *Lecture Notes in Computer Science*, pages 514–521, 2010.
- A. M. Alvarez, Q. Louveaux, and L. Wehenkel. A machine learning-based approximation of strong branching. Technical report, Universite de Liege, 2015.
- A. M. Alvarez, Q. Louveaux, and L. Wehenkel. Online learning for strong branching approximation in branch-and-bound. Technical report, Universite de Liege, 2016.

REFERENCES

- D. Anderson, G. Hendel, P. L. Bodic, and J. M. Viernickel. Clairvoyant restarts in branch-and-bound search using online tree-size estimation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1427–1434, 2018. DOI:10.1609/aaai.v33i01.33011427.
- D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook. On the solution of traveling salesman problems. *Documenta Mathematica Journal der deutschen Mathematikervereinigung*, pages 645–656, 1998. URL <http://emis.de/journals/DMJDMV/xvol-icm/17/Cook.MAN.html>.
- A. Atamtürk, G. L. Nemhauser, and M. W. Savelsbergh. Conflict graphs in solving integer programming problems. *European Journal of Operational Research*, 121(1):40–55, 2000.
- P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2–3):235–256, may 2002. ISSN 0885-6125. DOI:10.1023/A:1013689704352.
- P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The Nonstochastic Multiarmed Bandit Problem. *SIAM Journal On Computing*, 32(1):48–77, jan 2003. ISSN 0097-5397. DOI:10.1137/S0097539701398375.
- E. M. L. Beale. Cycling in the dual simplex algorithm. *Naval Research Logistics Quarterly*, 2(4), 1955. DOI:10.1002/nav.3800020406.
- J. E. Beasley. OR-Library: distributing test problems by electronic mail. *Journal of the operational research society*, 41(11):1069–1072, 1990.
- G. Belov, S. Esler, D. Fernando, P. Le Bodic, and G. L. Nemhauser. Estimating the size of search trees by sampling with domain knowledge. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 473–479, 2017. DOI:10.24963/ijcai.2017/67.
- M. Bénichou, J.-M. Gauthier, P. Girodet, G. Hentges, G. Ribière, and O. Vincent. Experiments in mixed-integer programming. *Mathematical Programming*, 1:76–94, 1971.
- L. Bertacco, M. Fischetti, and A. Lodi. A feasibility pump heuristic for general mixed-integer problems. *Discrete Optimization*, 4(1):63 – 76, 2007. ISSN 1572-5286. DOI:10.1016/j.disopt.2006.10.001. URL <http://www.sciencedirect.com/science/article/pii/S1572528606000855>. Mixed Integer Programming.
- T. Berthold. Heuristics of the Branch-Cut-and-Price-Framework SCIP. In J. Kalcsics and S. Nickel, editors, *Operations Research Proceedings 2007*, pages 31–36, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-77903-2. DOI:10.1007/978-3-540-77903-2_5.
- T. Berthold. Measuring the impact of primal heuristics. *Operations Research Letters*, 41(6):611–614, 2013. DOI:10.1016/j.orl.2013.08.007.
- T. Berthold. RENS—the optimal rounding. *Mathematical Programming Computation*, 6(1):33–54, 2014a. DOI:10.1007/s12532-013-0060-9.
- T. Berthold. *Heuristic algorithms in global MINLP solvers*. PhD thesis, Technische Universität Berlin, 2014b.
- T. Berthold and G. Hendel. Shift-and-propagate. *Journal of Heuristics*, 21(1):73 – 106, 2014. DOI:10.1007/s10732-014-9271-0.
- T. Berthold and G. Hendel. Learning to scale. In *AAAI-21: Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*, 2021. accepted for publication.

- T. Berthold and D. Salvagnin. Cloud branching. In C. Gomes and M. Sellmann, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 7874, pages 28 – 43. 2013. DOI:10.1007/978-3-642-38171-3_3.
- T. Berthold, A. M. Gleixner, S. Heinz, and S. Vigerske. Extending scip for solving miqcps. In P. Bonami, L. Liberti, A. J. Miller, and A. Sartenaer, editors, *Proceedings of the European Workshop on Mixed Integer Nonlinear Programming*, pages 181–196, 2010. URL <http://www.lix.polytechnique.fr/~liberti/ewminlp/ewminlp-proceedings.pdf>.
- T. Berthold, G. Gamrath, and D. Salvagnin. Cloud branching. Presentation slides from Mixed Integer Programming Workshop at Ohio State University. https://mip2014.engineering.osu.edu/sites/mip2014.engineering.osu.edu/files/uploads/Berthold_MIP2014_Cloud.pdf, 2014.
- T. Berthold, G. Hendel, and T. Koch. From feasibility to improvement to proof: three phases of solving mixed-integer programs. *Optimization Methods and Software*, 33(3):499–517, 2018. DOI:10.1080/10556788.2017.1392519.
- T. Berthold, G. Gamrath, and D. Salvagnin. Exploiting dual degeneracy in branching. Technical Report 19-17, ZIB, Takustr. 7, 14195 Berlin, 2019a.
- T. Berthold, A. Lodi, and D. Salvagnin. Ten years of feasibility pump, and counting. *EURO Journal on Computational Optimization*, 7(1):1–14, March 2019b. DOI:10.1007/s13675-018-0109-7. URL https://ideas.repec.org/a/spr/eurjco/v7y2019i1d10.1007_s13675-018-0109-7.html.
- K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is “nearest neighbor” meaningful? In C. Beeri and P. Buneman, editors, *Database Theory — ICDT’99*, pages 217–235. Springer Berlin Heidelberg, 1999. DOI:10.1007/3-540-49257-7_15.
- B. Bischl, P. Kerschke, L. Kotthoff, M. Lindauer, Y. Malitsky, A. Fréchette, H. Hoos, F. Hutter, K. Leyton-Brown, K. Tierney, et al. ASlib: A benchmark library for algorithm selection. *Artificial Intelligence*, 237:41–58, 2016.
- R. E. Bixby. Solving real-world linear programs: A decade and more of progress. *Operations Research*, 50(1):3–15, 2002. DOI:10.1287/opre.50.1.3.17780.
- R. E. Bixby, E. A. Boyd, and R. R. Indovina. MIPLIB: A test set of mixed integer programming problems. *SIAM News*, 25:16, 1992.
- R. E. Bixby, S. Ceria, C. M. McZeal, and M. W. Savelsbergh. An updated mixed integer programming library: MIPLIB 3.0. *Optima*, 58:12–15, 1998.
- G. E. P. Box. NON-NORMALITY AND TESTS ON VARIANCES. *Biometrika*, 40(3-4):318–335, 1953. ISSN 0006-3444. DOI:10.1093/biomet/40.3-4.318.
- A. Brearley, G. Mitra, and H. Williams. Analysis of mathematical programming problems prior to applying the simplex algorithm. *Mathematical Programming*, 8:54–83, 1975.
- L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, Oct 2001. ISSN 1573-0565. DOI:10.1023/A:1010933404324.
- L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Chapman & Hall CRC, 1983.

REFERENCES

- S. Browne, J. Dongarra, E. Grosse, and T. Rowan. The Netlib mathematical software repository. *D-lib Magazine*, 1(9), 1995.
- S. Bubeck and N. Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *CoRR*, abs/1204.5721, 2012. URL <http://arxiv.org/abs/1204.5721>.
- E. Burns and W. Ruml. Iterative-deepening search with on-line tree size prediction. *Annals of Mathematics and Artificial Intelligence*, 69(2):183–205, 2013. ISSN 1012-2443. DOI:10.1007/s10472-013-9347-9.
- M. R. Bussieck, A. S. Drud, and A. Meeraus. MINLPLib—a collection of test models for mixed-integer nonlinear programming. *INFORMS Journal on Computing*, 15(1):114–119, 2003.
- P. C. Chen. Heuristic sampling: A method for predicting the performance of tree searching programs. *SIAM Journal on Computing*, 21(2):295–315, 1992. ISSN 0097-5397. DOI:10.1137/0221022.
- V. Chvátal. *Linear Programming*. Freeman, New York, 1983.
- W. Cook. Markowitz and Manne + Eastman + Land and Doig = Branch and Bound. *Documenta Mathematica*, Extra Volume: Optimization Stories:227–238, 2012.
- Coral. Coral MIP benchmark library, 2016. <http://coral.ise.lehigh.edu/data-sets/mixed-integer-instances>.
- G. Cornuéjols. Valid inequalities for mixed integer programs. *Mathematical Programming*, 112:3–44, 2008.
- G. Cornuéjols, M. Karamanov, and Y. Li. Early estimates of the size of branch-and-bound trees. *INFORMS Journal on Computing*, 18(1):86–96, 2006. ISSN 1526-5528. DOI:10.1287/ijoc.1040.0107.
- H. P. Crowder, R. S. Dembo, and J. M. Mulvey. Reporting computational experiments in mathematical programming. *Mathematical Programming*, 15(1):316–329, 1978.
- R. J. Dakin. A tree-search algorithm for mixed integer programming problems. *The Computer Journal*, 8(3):250–255, 1965.
- E. Danna. Performance variability in mixed integer programming. Presentation slides from MIP workshop in New York City. <http://coral.ie.lehigh.edu/~jeff/mip-2008/program.pdf>, 2008.
- E. Danna, E. Rothberg, and C. L. Pape. Exploring relaxation induced neighborhoods to improve MIP solutions. *Mathematical Programming*, 102(1):71–90, 2005. ISSN 1436-4646. DOI:10.1007/s10107-004-0518-7.
- G. Dantzig, R. Fulkerson, and S. Johnson. Solution of a large scale traveling salesman problem. Technical Report P-510, RAND Corporation, Santa Monica, California, USA, 1954.
- G. B. Dantzig. *Activity Analysis of Production and Allocation*, chapter Maximization of a linear function of variables subject to linear inequalities, pages 339–347. John Wiley & Sons, 1951.
- G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111, 1960. DOI:10.1287/opre.8.1.101.
- S. S. Dey, A. Iroume, M. Molinaro, and D. Salvagnin. Improving the randomization step in feasibility pump. *SIAM Journal on Optimization*, pages 355–378, 2018.

- J. Eckstein. Parallel branch-and-bound algorithms for general mixed integer programming on the cm-5. *SIAM Journal on Optimization*, 4(4):794–814, 1994. DOI:10.1137/0804046.
- M. Falk, R. Becker, and F. Marohn. *Angewandte Statistik: Eine Einführung mit Programmbeispielen in SAS*. Springer-Verlag GmbH, 2003. ISBN 9783540405801.
- M. Fischetti and A. Lodi. Local branching. *Mathematical Programming*, 98(1-3):23–47, 2003. ISSN 0025-5610.
- M. Fischetti and A. Lodi. Repairing MILP infeasibility through local branching. 35:1436–1445, 05 2008.
- M. Fischetti and A. Lodi. Heuristics in mixed integer programming. In J. J. Cochran, L. A. Cox, P. Keskinocak, J. P. Kharoufeh, and J. C. Smith, editors, *Wiley Encyclopedia of Operations Research and Management Science*. John Wiley & Sons, Inc., 2010. Online publication.
- M. Fischetti and M. Monaci. Backdoor Branching. In O. Günlück and G. J. Woeginger, editors, *Integer Programming and Combinatorial Optimization*, volume 6655 of *Lecture Notes in Computer Science*, pages 183–191. Springer Berlin / Heidelberg, 2011.
- M. Fischetti and M. Monaci. Branching on nonchimerical fractionalities. *OR Letters*, 40(3):159–164, 2012.
- M. Fischetti and M. Monaci. Proximity search for 0-1 mixed-integer convex programming. *Journal of Heuristics*, 20(6):709–731, Dec 2014. ISSN 1572-9397. DOI:10.1007/s10732-014-9266-x.
- M. Fischetti and D. Salvagnin. Feasibility pump 2.0. *Mathematical Programming Computation*, 1(2):201–222, Oct 2009. ISSN 1867-2957. DOI:10.1007/s12532-009-0007-3.
- M. Fischetti, F. Glover, and A. Lodi. The feasibility pump. *Math. Program.*, 104(1):91–104, Sept. 2005. ISSN 0025-5610.
- M. Fischetti, A. Lodi, and G. Zarpellon. Learning MILP resolution outcomes before reaching time-limit. In L.-M. Rousseau and K. Stergiou, editors, *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 275–291. Springer, Cham, 2019.
- J. J. Forrest and D. Goldfarb. Steepest-edge simplex algorithms for linear programming. *Mathematical Programming*, 57(1):341–374, May 1992. DOI:10.1007/BF01581089.
- J. J. H. Forrest and J. A. Tomlin. Updated triangular factors of the basis to maintain sparsity in the product form simplex method. *Mathematical Programming*, 2:263–278, 1972. DOI:10.1007/BF01584548.
- J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Ann. Statist.*, 29(5):1189–1232, 10 2001. DOI:10.1214/aos/1013203451.
- F. Furini, E. Traversi, P. Belotti, A. Frangioni, A. Gleixner, N. Gould, L. Liberti, A. Lodi, R. Misener, H. Mittelmann, et al. QPLIB: a library of quadratic programming instances. *Mathematical Programming Computation*, 11(2):237–265, 2019.
- G. Gamrath. Improving strong branching by propagation. In C. Gomes and M. Sellmann, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 7874 of *Lecture Notes in Computer Science*, pages 347–354. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-38170-6. DOI:10.1007/978-3-642-38171-3_25.

REFERENCES

- G. Gamrath. *Enhanced Predictions and Structure Exploitation in Branch-and-Bound*. PhD thesis, Technische Universität Berlin, 2020.
- G. Gamrath and M. Lübbecke. Experiments with a generic Dantzig-Wolfe decomposition for integer programs. In P. Festa, editor, *Experimental Algorithms*, volume 6049 of *Lect. Notes in Comp. Sci.*, pages 239–252, Berlin, 2010. Springer-Verlag. DOI:10.1007/978-3-642-13193-6_21.
- G. Gamrath, T. Berthold, S. Heinz, and M. Winkler. Structure-based primal heuristics for mixed integer programming. In *Optimization in the Real World*, volume 13, pages 37 – 53. 2015a. DOI:10.1007/978-4-431-55420-2_3.
- G. Gamrath, T. Koch, A. Martin, M. Miltenberger, and D. Weninger. Progress in presolving for mixed integer programming. *Mathematical Programming Computation*, 7(4):367–398, 2015b. DOI:10.1007/s12532-015-0083-5.
- G. Gamrath, T. Berthold, S. Heinz, and M. Winkler. Structure-driven fix-and-propagate heuristics for mixed integer programming. *Mathematical Programming Computation*, 11(4):675 – 702, 2019. DOI:10.1007/s12532-019-00159-1.
- G. Gamrath, D. Anderson, K. Bestuzheva, W.-K. Chen, L. Eifler, M. Gasse, P. Gemander, A. Gleixner, L. Gottwald, K. Halbig, G. Hendel, C. Hojny, T. Koch, P. Le Bodic, S. J. Maher, F. Matter, M. Miltenberger, E. Mühmer, B. Müller, M. E. Pfetsch, F. Schlösser, F. Serrano, Y. Shinano, C. Tawfik, S. Vigerske, F. Wegscheider, D. Weninger, and J. Witzig. The SCIP Optimization Suite 7.0. ZIB-Report 20-10, Zuse Institute Berlin, March 2020. URL <http://nbn-resolving.de/urn:nbn:de:0297-zib-78023>.
- M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979. ISBN 0716710447.
- M. Gasse, D. Chetelat, N. Ferroni, L. Charlin, and A. Lodi. Exact combinatorial optimization with graph convolutional neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 15580–15592. Curran Associates, Inc., 2019. URL <https://proceedings.neurips.cc/paper/2019/file/d14c2267d848abeb81fd590f371d39bd-Paper.pdf>.
- J.-M. Gauthier and G. Ribière. Experiments in mixed-integer linear programming using pseudo-costs. *Mathematical Programming*, 12(1):26–47, 1977. ISSN 0025-5610. DOI:10.1007/BF01593767.
- A. Georges, A. Gleixner, G. Gojic, R. L. Gottwald, D. Haley, G. Hendel, and B. Matejczyk. Feature-based algorithm selection for mixed integer programming. ZIB-Report 18-17, Zuse Institute Berlin, 2018.
- S. Ghosh. DINS, a MIP Improvement Heuristic. In M. Fischetti and D. P. Williamson, editors, *Integer Programming and Combinatorial Optimization: 12th International IPCO Conference, Ithaca, NY, USA, June 25-27, 2007. Proceedings*, pages 310–323, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-72792-7. DOI:10.1007/978-3-540-72792-7_24.
- A. Gilpin and T. Sandholm. Information-theoretic approaches to branching in search. *Discrete Optimization*, 8(2):147 – 159, 2011. ISSN 1572-5286. DOI:10.1016/j.disopt.2010.07.001. URL <http://www.sciencedirect.com/science/article/pii/S1572528610000423>.

- A. Gleixner, M. Bastubbe, L. Eifler, T. Gally, G. Gamrath, R. L. Gottwald, G. Hendel, C. Hojny, T. Koch, M. E. Lübbecke, S. J. Maher, M. Miltenberger, B. Müller, M. E. Pfetsch, C. Puchert, D. Rehfeldt, F. Schlösser, C. Schubert, F. Serrano, Y. Shinano, J. M. Viernickel, M. Walter, F. Wegscheider, J. T. Witt, and J. Witzig. The SCIP Optimization Suite 6.0. Technical report, Optimization Online, July 2018. URL http://www.optimization-online.org/DB_HTML/2018/07/6692.html.
- A. Gleixner, G. Hendel, G. Gamrath, T. Achterberg, M. Bastubbe, T. Berthold, P. M. Christophel, K. Jarck, T. Koch, J. Linderoth, M. Lübbecke, H. D. Mittelmann, D. Ozyurt, T. K. Ralphs, D. Salvagnin, and Y. Shinano. MIPLIB 2017: Data-Driven Compilation of the 6th Mixed-Integer Programming Library. *Mathematical Programming Computation*, 2021. DOI:doi.org/10.1007/s12532-020-00194-3.
- D. Goldfarb and J. K. Reid. A practicable steepest-edge simplex algorithm. *Mathematical Programming*, 12(1):361–371, 1977. DOI:[10.1007/BF01593804](https://doi.org/10.1007/BF01593804).
- T. Granlund and the GMP development team. *GNU MP: The GNU Multiple Precision Arithmetic Library*, 6.1.2 edition, 2016. URL <http://gmplib.org/>.
- P. M. J. Harris. Pivot selection methods of the devex lp code. *Mathematical Programming*, 5(1):1–28, Dec 1973. ISSN 1436-4646. DOI:[10.1007/BF01580108](https://doi.org/10.1007/BF01580108).
- J. A. Hartigan and M. A. Wong. Algorithm AS 136: A K-Means clustering algorithm. *Applied Statistics*, 28(1):100–108, 1979. DOI:[10.2307/2346830](https://doi.org/10.2307/2346830).
- H. He, H. D. III, and J. Eisner. Learning to search in branch-and-bound algorithms. In *NIPS 2014*, 2014.
- G. Hendel. IPET interactive performance evaluation tools. <https://github.com/GregorCH/ipet>.
- G. Hendel. Empirical analysis of solving phases in mixed integer programming. Master thesis, Technische Universität Berlin, 2014.
- G. Hendel. Enhancing MIP branching decisions by using the sample variance of pseudo costs. In L. Michel, editor, *Integration of AI and OR Techniques in Constraint Programming*, volume 9075, pages 199–214. Springer International Publishing, 2015. DOI:[10.1007/978-3-319-18008-3_14](https://doi.org/10.1007/978-3-319-18008-3_14).
- G. Hendel. Adaptive large neighborhood search for mixed integer programming. ZIB-Report 18-60, Zuse Institute Berlin, 2018.
- G. Hendel, M. Miltenberger, and J. Witzig. Adaptive algorithmic behavior for solving mixed integer programs using bandit algorithms. In B. Fortz and M. Labbé, editors, *Operations Research Proceedings 2018*, pages 513–519. Springer International Publishing, 2019. ISBN 978-3-030-18500-8. DOI:[10.1007/978-3-030-18500-8_64](https://doi.org/10.1007/978-3-030-18500-8_64).
- G. Hendel, D. Anderson, P. L. Bodic, and M. Pfetsch. Estimating the size of branch-and-bound trees. *Inform. Journal on Computing*, 2021. Accepted for publication.
- A. Hoffman, M. Mannos, D. Sokolowsky, and N. Wiegmann. Computational experience in solving linear programs. *Journal of the Society for Industrial and Applied Mathematics*, 1(1):17–33, 1953.
- C. C. Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting*, 20(1):5 – 10, 2004. ISSN 0169-2070. DOI:[10.1016/j.ijforecast.2003.09.015](https://doi.org/10.1016/j.ijforecast.2003.09.015). URL <http://www.sciencedirect.com/science/article/pii/S0169207003001134>.

REFERENCES

- J. Hooker. Needed: An empirical science of algorithms. *Operations Research*, 42(2):201–212, 1994.
- F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In C. A. C. Coello, editor, *Learning and Intelligent Optimization*, pages 507–523, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. DOI:10.1007/978-3-642-25566-3_40.
- R. Hyndman, A. Koehler, J. Ord, and R. Snyder. *Forecasting with exponential smoothing: the state space approach*. Springer-Verlag, 2008.
- R. Hyndman, G. Athanasopoulos, C. Bergmeir, G. Caceres, L. Chhay, M. O’Hara-Wild, F. Petropoulos, S. Razbash, E. Wang, and F. Yasmeen. *forecast: Forecasting functions for time series and linear models*, 2019. URL <http://pkg.robjhyndman.com/forecast>. R package version 8.5.
- R. J. Hyndman and Y. Khandakar. Automatic time series forecasting: the forecast package for R. *Journal of Statistical Software*, 26(3):1–22, 2008. URL <http://www.jstatsoft.org/article/view/v027i03>.
- Mathematical Programming System/360 Version 2, Linear and Separable Programming – User’s Manual*. IBM Corporation, White Plains, N.Y., 3 edition, October 1969. Publication H20-0476-2.
- R. H. F. Jackson, P. T. Boggs, S. G. Nash, and S. Powell. Guidelines for reporting results of computational experiments. Report of the ad hoc committee. *Mathematical Programming*, 49:413–425, 1991.
- T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’02, page 133–142, New York, NY, USA, 2002. Association for Computing Machinery. ISBN 158113567X. DOI:10.1145/775047.775067.
- S. Kadioglu, E. O’Mahony, P. Refalo, and M. Sellmann. Incorporating variance in impact-based search. In *Proceedings of the 17th International Conference on Principles and Practice of Constraint Programming*, CP’11, pages 470–477, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-23785-0. URL <http://dl.acm.org/citation.cfm?id=2041160>.2041199.
- N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, Dec. 1984. ISSN 0209-9683. DOI:10.1007/BF02579150.
- R. M. Karp. *Complexity of Computer Computations*, chapter Reducibility among Combinatorial Problems. The IBM Research Symposia Series. Springer, Boston, MA, 1972. DOI:10.1007/978-1-4684-2001-2_9.
- B. W. Kernighan and D. M. Ritchie. *The C Programming Language*. Prentice Hall Professional Technical Reference, 2nd edition, 1988. ISBN 0131103709.
- L. G. Khachiyan. A polynomial algorithm in linear programming. *Doklady Akademii Nauk SSSR*, 244(5):1093–1096, 1979. english translation in Soviet Math. Dokl. 20(1):191–194, 1979.
- E. B. Khalil, P. L. Bodic, L. Song, G. Nemhauser, and B. Dilkina. Learning to branch in mixed integer programming. In *AAAI-16*, 2016.
- E. B. Khalil, B. Dilkina, G. Nemhauser, S. Ahmed, and Y. Shao. Learning to run heuristics in tree search. In *26th International Joint Conference on Artificial Intelligence (IJCAI)*, 2017.
- P. Kilby, J. K. Slaney, S. Thiébaux, and T. Walsh. Estimating search tree size. In *AAAI*, pages 1014–1019. AAAI Press, 2006.

- V. Klee and G. Minty. How good is the simplex algorithm. 1970.
- D. E. Knuth. Estimating the Efficiency of Backtrack Programs. *Mathematics of Computation*, 29(129): 121–136, 1975.
- T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R. E. Bixby, E. Danna, G. Gamrath, A. M. Gleixner, S. Heinz, A. Lodi, H. Mittelman, T. Ralphs, D. Salvagnin, D. E. Steffy, and K. Wolter. MIPLIB 2010. *Mathematical Programming Computation*, 3(2):103–163, 2011. ISSN 1867-2949. DOI:10.1007/s12532-011-0025-9.
- T. Koch, T. Ralphs, and Y. Shinano. Could we use a million cores to solve an integer program? *Mathematical Methods of Operations Research*, 76(1):67 – 93, 2012. DOI:10.1007/s00186-012-0390-9.
- T. Koch, A. Martin, and M. E. Pfetsch. Progress in academic computational integer programming. In M. Jünger and G. Reinelt, editors, *Facets of Combinatorial Optimization*, pages 483–506. Springer, 2013. ISBN 978-3-642-38188-1. DOI:10.1007/978-3-642-38189-8_19.
- O. Kullmann. Fundaments of branching heuristics. In A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 205–244. IOS Press, 2009. ISBN 978-1-58603-929-5. DOI:10.3233/978-1-58603-929-5-205.
- F. Kılınç Karzan, G. L. Nemhauser, and M. W. P. Savelsbergh. Information-based branching schemes for binary linear mixed integer problems. *Mathematical Programming Computation*, 1(4):249–293, 2009. ISSN 1867-2949. DOI:10.1007/s12532-009-0009-1.
- P. Laborie and D. Godard. Self-adapting large neighborhood search: Application to single-mode scheduling problems. In P. Baptiste, G. Kendall, A. Munier, and F. Sourd, editors, *MISTA-07*, 08 2007.
- A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960.
- R. Laundry, M. Perregaard, G. Tavares, H. Tipi, and A. Vazacopoulos. Solving hard mixed integer programming problems with Xpress-MP: A MIPLIB 2003 case study. *INFORMS Journal on Computing*, 21:304–319, 2009.
- P. Le Bodic and G. Nemhauser. An abstract model for branching and its application to mixed integer programming. *Mathematical Programming*, 166(1):369–405, 2017. ISSN 1436-4646. DOI: 10.1007/s10107-016-1101-8.
- P. Le Bodic and G. L. Nemhauser. How important are branching decisions: Fooling MIP solvers. *Operations Research Letters*, 43(3):273–278, 2015. DOI:10.1016/j.orl.2015.03.003.
- S. Legg and M. Hutter. Universal intelligence: A definition of machine intelligence. *Minds and Machines*, 17, 01 2008. DOI:10.1007/s11023-007-9079-x.
- S. Leipert. The tree interface – version 1.0 user manual. Technical report, Zentrum für Angewandte Informatik Köln, Lehrstuhl Jünger, 1996.
- L. H. S. Lelis, L. Otten, and R. Dechter. Predicting the size of depth-first branch and bound search trees. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI ’13, pages 594–600. AAAI Press, 2013. ISBN 978-1-57735-633-2. URL <http://dl.acm.org/citation.cfm?id=2540128.2540215>.

REFERENCES

- L. H. S. Lelis, L. Otten, and R. Dechter. Memory-efficient tree size prediction for depth-first search in graphical models. In *Principles and Practice of Constraint Programming - 20th International Conference, CP 2014, Lyon, France, September 8-12, 2014. Proceedings*, pages 481–496, 2014. DOI:10.1007/978-3-319-10428-7_36.
- C. E. Lemke. The dual method of solving the linear programming problem. *Naval Research Logistics Quarterly*, 1(1):36–47, 1954. DOI:10.1002/nav.3800010107. URL <https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800010107>.
- L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018. URL <http://jmlr.org/papers/v18/16-558.html>.
- P. Liberatore. On the complexity of choosing the branching literal in dpll. *Artificial Intelligence*, 116(1):315–326, 2000. ISSN 0004-3702. DOI:10.1016/S0004-3702(99)00097-1. URL <https://www.sciencedirect.com/science/article/pii/S0004370299000971>.
- J. T. Linderoth and T. K. Ralphs. Noncommercial software for mixed-integer linear programming. *Integer programming: theory and practice*, 3(253-303):144–189, 2005.
- J. T. Linderoth and M. W. P. Savelsbergh. A computational study of search strategies for mixed integer programming. *INFORMS Journal on Computing*, 11(2):173–187, 1999.
- L. Lobjois and M. Lemaitre. Branch and bound algorithm selection by performance prediction. In *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence, AAAI '98/IAAI '98*, pages 353–358, Menlo Park, CA, USA, 1998. American Association for Artificial Intelligence. ISBN 0-262-51098-7. URL <http://dl.acm.org/citation.cfm?id=295240.295633>.
- A. Lodi. MIP computation. In M. Jünger, T. Liebling, D. Naddef, G. Nemhauser, W. Pulleyblank, G. Reinelt, G. Rinaldi, and L. Wolsey, editors, *50 Years of Integer Programming 1958-2008*, pages 619–645. Springer, 2009.
- M. Luby, A. Sinclair, and D. Zuckerman. Optimal speedup of las vegas algorithms. *Inf. Process. Lett.*, 47(4):173–180, Sept. 1993. ISSN 0020-0190. DOI:10.1016/0020-0190(93)90029-9.
- M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, M. Birattari, and T. Stützle. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016. ISSN 2214-7160. DOI:10.1016/j.orp.2016.09.002.
- S. J. Maher, T. Fischer, T. Gally, G. Gamrath, A. Gleixner, R. L. Gottwald, G. Hendel, T. Koch, M. E. Lübbecke, M. Miltenberger, B. Müller, M. E. Pfetsch, C. Puchert, D. Rehfeldt, S. Schenker, R. Schwarz, F. Serrano, Y. Shinano, D. Weninger, J. T. Witt, and J. Witzig. The scip optimization suite 4.0. Technical Report 17-12, ZIB, Takustr. 7, 14195 Berlin, 2017.
- H. Marchand, A. Martin, R. Weismantel, and L. A. Wolsey. Cutting planes in integer and mixed integer programming. *Discrete Applied Mathematics*, 123/124:391–440, 2002.
- C. C. McGeoch. Toward an experimental method for algorithm simulation. *INFORMS Journal on Computing*, 8(1):1–15, 1996.
- C. C. McGeoch. Experimental analysis of algorithms. *Notices of the AMS*, 48(3):304–311, 2001.

- MIPLIB. MIPLIB 2.0. URL <http://miplib2010.zib.de/miplib2/miplib2.html>.
- H. Mittelmann. Benchmarks for optimization software. URL <http://plato.asu.edu/bench.html>.
- D. R. Morrison, S. H. Jacobson, J. J. Sauppe, and E. C. Sewell. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, 19:79–102, 2016. ISSN 1572-5286. DOI:10.1016/j.disopt.2016.01.005. URL <https://www.sciencedirect.com/science/article/pii/S1572528616000062>.
- M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient sat solver. In *Proceedings of the 38th Annual Design Automation Conference, DAC '01*, page 530–535, New York, NY, USA, 2001. Association for Computing Machinery. ISBN 1581132972. DOI:10.1145/378239.379017.
- L.-M. Munguía, S. Ahmed, D. A. Bader, G. L. Nemhauser, and Y. Shao. Alternating criteria search: a parallel large neighborhood search algorithm for mixed integer programs. *Computational Optimization and Applications*, 69(1):1–24, Jan 2018. DOI:10.1007/s10589-017-9934-5.
- V. Nair, S. Bartunov, F. Gimeno, I. von Glehn, P. Lichocki, I. Lobov, B. O'Donoghue, N. Sonnerat, C. Tjandraatmadja, P. Wang, R. Addanki, T. Hapuarachchi, T. Keck, J. Keeling, P. Kohli, I. Ktena, Y. Li, O. Vinyals, and Y. Zwols. Solving mixed integer programs using neural networks, 2020.
- J. L. Nazareth. *Computer Solutions of Linear Programs*. Monographs on numerical analysis. Oxford University Press, 1987.
- G. Neu. Explore no more: Improved high-probability regret bounds for non-stochastic bandits. In *NIPS*, 2015.
- F. Ortega and L. A. Wolsey. A branch-and-cut algorithm for the single-commodity, uncapacitated, fixed-charge network flow problem. *Networks*, 41(3):143–158, 2003.
- O. Y. Özaltın, B. Hunsaker, and A. J. Schaefer. Predicting the solution time of branch-and-bound algorithms for mixed-integer programs. *INFORMS J. on Computing*, 23(3):392–403, 2011. ISSN 1526-5528. DOI:10.1287/ijoc.1100.0405.
- M. E. Pfetsch. Branch-and-cut for the maximum feasible subsystem problem. *SIAM Journal on Optimization*, 19:21–38, 2008.
- D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8):2403 – 2435, 2007. ISSN 0305-0548. DOI:10.1016/j.cor.2005.09.012.
- F. A. Potra and S. J. Wright. Interior-point methods. *Journal of Computational and Applied Mathematics*, 124(1):281–302, 2000. ISSN 0377-0427. DOI:10.1016/S0377-0427(00)00433-7. Numerical Analysis 2000. Vol. IV: Optimization and Nonlinear Equations.
- J. Pryor and J. W. Chinneck. Faster integer-feasibility in mixed-integer linear programs by branching to force change. *Computers & Operations Research*, 38(8):1143 – 1152, 2011. ISSN 0305-0548. DOI:10.1016/j.cor.2010.10.025. URL <http://www.sciencedirect.com/science/article/pii/S0305054810002546>.
- P. W. Purdom. Tree Size by Partial Backtracking. *SIAM Journal on Computing*, 7(4):481–491, 1978. DOI:10.1137/0207038.

REFERENCES

- R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2018. URL <https://www.R-project.org/>.
- P. Refalo. *Principles and Practice of Constraint Programming – CP 2004: 10th International Conference, CP 2004, Toronto, Canada, September 27 -October 1, 2004. Proceedings*, chapter Impact-Based Search Strategies for Constraint Programming, pages 557–571. Springer Berlin Heidelberg, 2004. DOI:10.1007/978-3-540-30201-8_41. URL 10.1007/978-3-540-30201-8_41.
- J. R. Rice. The algorithm selection problem. In *Advances in computers*, volume 15, pages 65–118. Elsevier, 1976.
- G. A. R.J. Hyndman. *Forecasting: Principles and Practice*. OTexts: Melbourne, Australia, 2 edition, 2018.
- E. Rothberg. An Evolutionary Algorithm for Polishing Mixed Integer Programming Solutions. *INFORMS Journal on Computing*, 19(4):534–541, 2007. DOI:10.1287/ijoc.1060.0189.
- G. G. Roussas. *A Course in Mathematical Statistics, Third Edition*. Elsevier Science & Technology Books, 2014. ISBN 9780120884469. URL <http://books.google.de/books?id=WsssuAAACAAJ>.
- D. M. Ryan and B. A. Foster. An integer programming approach to scheduling. In A. Wren, editor, *Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling*, pages 269–280. North Holland, Amsterdam, 1981.
- A. Sabharwal, H. Samulowitz, and C. Reddy. Guiding combinatorial optimization with UCT. In N. Beldiceanu, N. Jussien, and E. Pinson, editors, *CPAIOR*, volume 7298 of *Lecture Notes in Computer Science*, pages 356–361. Springer, 2012.
- A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., USA, 1986. ISBN 0471908541.
- Y. Shinano. The Ubiquity Generator framework: 7 years of progress in parallelizing branch-and-bound. In N. Kliewer, J. F. Ehmke, and R. Borndörfer, editors, *Operations Research Proceedings 2017*, pages 143–149, Cham, 2018. Springer International Publishing. DOI:10.1007/978-3-319-89920-6_20.
- Y. Shinano, T. Achterberg, T. Berthold, S. Heinz, and T. Koch. ParaSCIP – a parallel extension of SCIP. In C. Bischof, H.-G. Hegering, W. E. Nagel, and G. Wittum, editors, *Competence in High Performance Computing 2010*, pages 135–148. Springer, 2012.
- Y. Shinano, T. Achterberg, T. Berthold, S. Heinz, T. Koch, and M. Winkler. Solving Open MIP Instances with ParaSCIP on Supercomputers using up to 80,000 Cores. In *Proc. of 30th IEEE International Parallel & Distributed Processing Symposium*, 2016. DOI:10.1109/IPDPS.2016.56.
- K. Smith-Miles and S. Bowly. Generating new test instances by evolving in instance space. *Computers & Operations Research*, 63:102–113, 2015.
- K. Smith-Miles, D. Baatar, B. Wreford, and R. Lewis. Towards objective measures of algorithm performance across instance space. *Computers & Operations Research*, 45:12–24, 2014.
- Soplex. SoPlex. An open source LP solver implementing the revised simplex algorithm. <http://soplex.zib.de/>, 2016.
- R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL <http://incompleteideas.net/book/the-book-2nd.html>.

- Y. Tang, S. Agrawal, and Y. Faenza. Reinforcement learning for integer programming: Learning to cut. Technical report, 2020.
- J. T. Thayer, R. Stern, and L. H. S. Lelis. Are we there yet? - estimating search progress. In D. Borrajo, A. Felner, R. E. Korf, M. Likhachev, C. L. López, W. Ruml, and N. R. Sturtevant, editors, *Proceedings of the Fifth Annual Symposium on Combinatorial Search, SOCS 2012, Niagara Falls, Ontario, Canada, July 19-21, 2012*. AAAI Press, 2012. URL <http://www.aaai.org/ocs/index.php/SOCS/SOCS12/paper/view/5381>.
- L. van der Maaten and G. Hinton. Visualizing high-dimensional data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008. ISSN 1532-4435.
- R. J. Vanderbei. *Linear Programming*, volume 196 of *International Series in Operations Research & Management Science*. Springer US, 2014. DOI:10.1007/978-1-4614-7630-6.
- S. Vigerske. *Decomposition of Multistage Stochastic Programs and a Constraint Integer Programming Approach to Mixed-Integer Nonlinear Programming*. PhD thesis, Humboldt-Universität zu Berlin, 2013. URL <http://nbn-resolving.de/urn:nbn:de:kobv:11-100208240>.
- B. L. Welch. THE GENERALIZATION OF ‘STUDENT’S’ PROBLEM WHEN SEVERAL DIFFERENT POPULATION VARIANCES ARE INVOLVED. *Biometrika*, 34(1-2):28–35, 01 1947. ISSN 0006-3444. DOI:10.1093/biomet/34.1-2.28.
- J. Witzig and A. Gleixner. Conflict-driven heuristics for mixed integer programming. *INFORMS Journal on Computing*, 2020. DOI:10.1287/ijoc.2020.0973. epub ahead of print.
- J. Witzig, T. Berthold, and S. Heinz. Experiments with Conflict Analysis in Mixed Integer Programming. In D. Salvagnin and M. Lombardi, editors, *Integration of AI and OR Techniques in Constraint Programming*, pages 211–220, Cham, 2017. Springer International Publishing. ISBN 978-3-319-59776-8. DOI:10.1007/978-3-319-59776-8_17.
- D. T. Wojtaszek and J. W. Chinneck. Faster mip solutions via new node selection rules. *Computers & Operations Research*, 37(9):1544–1556, 2010. ISSN 0305-0548.
- L. Wolsey. *Integer Programming*. John Wiley & Sons, Inc., second edition edition, 2020.
- A. S. Xavier, F. Qiu, and S. Ahmed. Learning to solve large-scale security-constrained unit commitment problems, 2019.
- D. York, N. M. Evensen, M. L. Martínez, and J. De Basabe Delgado. Unified equations for the slope, intercept, and standard errors of the best straight line. *American Journal of Physics*, 72(3):367–375, 2004. DOI:10.1119/1.1632486.
- U. Zahavi, A. Felner, N. Burch, and R. C. Holte. Predicting the performance of ida* using conditional distributions. *J. Artif. Intell. Res. (JAIR)*, 37:41–83, 2010. DOI:10.1613/jair.2890.

Abstract

This thesis addresses general-purpose solution techniques for mixed-integer programs (MIPs), a paradigm which captures formulations of countless real-world optimization problems. Most state-of-the-art MIP solvers employ a version of the branch-and-bound (B&B) algorithm to solve a MIP instance to proven optimality, supported by numerous auxiliary components that contribute new solutions or improve the dual convergence.

One cannot expect that all such components are equally effective on all possible instances from the tremendous range of MIP applications. Ideally, a solver adapts to a given MIP instance by concentrating the available computational budget on those components that work best. In this thesis, we develop adaptive algorithmic behavior for several such MIP solver components as well as the B&B search itself.

We develop new notions of pseudo-cost reliability, namely *relative-error reliability* and *hypothesis reliability*, by computing confidence intervals and pairwise *t*-tests on branching candidates to dynamically decide if strong branching is necessary. We develop two heuristic frameworks, *adaptive large neighborhood search* and *adaptive diving* that learn the most effective primal heuristics inspired by selection strategies for the multi-armed bandit problem. The presented ideas are transferred to *adaptive LP pricing* to maximize LP throughout by learning the pricing strategy for the dual simplex algorithms online during the search.

Our proposed adaptive algorithmic behavior extends beyond individual solving components to the B&B search as a whole. To this end, we partition the B&B search into a *feasibility phase*, an *improvement phase*, and a heuristically detected *proof phase*. We improve solver performance by emphasizing different components and search strategies in each phase. We propose new estimation techniques for the progress of the B&B search based on forecasting and machine learning techniques. We turn this *tree-size estimation* into a novel restart strategy of the B&B algorithm called *clairvoyant*.

As a methodological contribution, we describe the selection process of *MIPLIB 2017*, which is the current state-of-the-art library for benchmarking MIP solver performance. All of the discussed algorithmic approaches are evaluated within the MIP solver SCIP, for which they show clear performance benefits. They are publicly available as part of SCIP and have been adopted by state-of-the-art commercial solvers such as FICO Xpress.

Zusammenfassung

Im Rahmen dieser Arbeit befassen wir uns mit anwendungsunabhängigen Lösetechniken für gemischt-ganzzahlige Optimierungsprobleme (*engl. mixed-integer programs (MIPs)*), ein Paradigma mit unzähligen praktischen Anwendungsbeispielen. Moderne Lösertechnologie basiert zumeist auf dem Branch-and-Bound-(B&B)-Algorithmus, welcher auf algorithmische Hilfskomponenten zur schnelleren Lösungsfindung und der Konvergenz der dualen Schranke angewiesen ist.

Die Vielfalt an Anwendungsgebieten für MIP lässt schon erwarten, dass nicht alle B&B-Komponenten gleichermaßen erfolgreich auf allen erdenklichen MIP-Anwendungsgebieten sind. Idealerweise passt sich ein MIP-Löser zur Laufzeit an eine gegebene Instanz an, indem er eine bestmögliche Verteilung des zur Verfügung stehenden Budgets auf die einzelnen Komponenten erlernt. Im Rahmen dieser Arbeit werden neue adaptive Lernverfahren sowohl für einzelne Komponenten als auch für den B&B-Algorithmus vorgestellt.

Die neu vorgestellten Entwicklungen in dieser Arbeit umfassen

- verallgemeinerte Pseudokostenzuverlässigkeitsbestimmungen anhand von Konfidenzintervallen und statistischer Tests zur dynamischen Entscheidung, ob Strong Branching erforderlich ist,
- heuristische Frameworks *Adaptive Large Neighborhood Search* und *Adaptive Diving* mit dem Ziel, die effektivsten Primalheuristiken im Laufe des Löseprozesses zu erlernen,
- *Adaptive LP Pricing* zur Verbesserung des Knotendurchsatzes während der B&B-Suche durch dynamische Wahl der besten Pricing-Strategie für das Simplexverfahren,
- eine Einteilung des Löseprozesses in drei Phasen: eine *Zulässigkeits-*, eine *Verbesserungs-*, und eine *Beweisphase*, und die Ausnutzung von Phaseninformation zur gezielten Steuerung des B&B-Verfahrens.
- eine Abschätzung der verbleibenden Baumgröße und des Fortschritts des B&B-Verfahrens basierend auf Forecasting-Techniken und Methoden des Maschinellen Lernens.

- die anschließende Ausnutzung dieser Schätzverfahren für *Clairvoyant Restarts* der B&B-Suche bei einer pessimistischen Prognose.

Ein weiterer, methodischer Beitrag besteht in der detaillierten Vorstellung des Auswahlverfahrens der MIPLIB 2017, dem derzeitigen Standardbenchmarkset zur Evaluierung von MIP-Lösern.

Alle neu vorgestellten Algorithmen wurden im Rahmen dieser Arbeit in den nichtkommerziellen MIP-Löser SCIP integriert und auch praktisch anhand von Rechenexperimenten ausgewertet. Sie sind somit als Teil von SCIP 7.0 öffentlich verfügbar, und wurden teilweise bereits in kommerzielle Löser wie FICO Xpress integriert.



Appendix A

Table A.1: Instancewise results of the experiment in Section 5.4.2. Each column except for **default** denotes a run with a particular phase transition criterion. For each instance, we present the values of the dual integral **integral***, the number of branch-and-bound nodes **nodes**, the achieved primal bound **objective** at termination, the value **integral** of the primal integral, and the overall solving time **time**.

Settings		default	estim	oracle	rank-1
Problem					
10teams	integral*	58.4	61.6	60.9	60.8
	nodes	1272.0	1612.0	1552.0	1612.0
	objective	924.0	924.0	924.0	924.0
	integral	1459.7	648.5	599.0	584.4
	time	25.6	35.4	33.3	33.3
30n20b8	integral*	9555.0	8011.9	6959.1	8029.3
	nodes	14.0	6.0	4.0	6.0
	objective	302.0	302.0	302.0	302.0
	integral	8418.0	8330.0	8317.0	8410.0
	time	183.5	151.7	135.6	152.0
a1c1s1	integral*	124305.2	118916.9	118644.8	162765.1
	nodes	718786.0	626920.0	640772.0	1815029.0
	objective	11642.1	11701.0	11701.0	11754.2
	integral	14318.3	14326.3	14270.7	25160.9
	time	7200.0	7200.0	7200.0	7200.0
acc-tight5	integral*	210.0	210.0	200.0	220.0
	nodes	16931.0	608.0	608.0	608.0
	objective	0.0	0.0	0.0	0.0

continued on next page

A. Appendix A

Table A.1: Instancewise results of the experiment in Section 5.4.2. **integral***: dual integral, **nodes**: number of branch-and-bound nodes, **objective**: primal bound at termination, **integral**: primal integral, **time**: overall solving time.

Settings		default	estim	oracle	rank-1
Problem					
aflow30a	integral	143661.0	6170.0	6160.0	6220.0
	time	1436.6	61.7	61.6	62.2
	integral*	89.8	93.1	87.7	89.1
	nodes	2118.0	2506.0	1852.0	1976.0
	objective	1158.0	1158.0	1158.0	1158.0
	integral	385.8	398.1	388.3	390.8
aflow40b	time	13.9	14.5	13.3	13.5
	integral*	2589.6	3114.8	2798.2	6318.6
	nodes	151103.0	112411.0	69617.0	323752.0
	objective	1168.0	1168.0	1168.0	1168.0
	integral	1850.7	2421.6	2420.7	4094.9
	time	912.9	941.0	681.0	1328.9
air03	integral*	70.2	80.2	70.2	80.1
	nodes	1.0	1.0	1.0	1.0
	objective	340160.0	340160.0	340160.0	340160.0
	integral	81.5	88.4	82.8	84.4
	time	1.0	1.1	1.0	1.1
	integral*	453.3	441.4	442.4	452.2
air04	nodes	213.0	146.0	156.0	156.0
	objective	56137.0	56137.0	56137.0	56137.0
	integral	756.8	740.2	742.1	747.4
	time	70.4	64.1	65.6	65.5
	integral*	166.5	189.7	209.6	189.9
	nodes	181.0	309.0	299.0	365.0
air05	objective	26374.0	26374.0	26374.0	26374.0
	integral	415.9	529.4	552.5	530.7
	time	37.5	39.3	39.3	39.4
	integral*	52653.8	63944.0	52923.0	52774.2
	nodes	427.0	429.0	306.0	246.0
	objective	-41.0	-41.0	-41.0	-41.0
app1-2	integral	28924.9	28904.7	28710.0	29076.8
	time	1118.2	1270.6	976.2	964.7
	integral*	56.9	93.7	99.8	103.4
	nodes	1166439.0	915413.0	960617.0	997478.0
	objective	7580813.0	7580813.0	7580813.0	7580813.0

continued on next page

Table A.1: Instancewise results of the experiment in Section 5.4.2. **integral***: dual integral, **nodes**: number of branch-and-bound nodes, **objective**: primal bound at termination, **integral**: primal integral, **time**: overall solving time.

Settings		default	estim	oracle	rank-1
Problem					
ash608gpia-3col	integral	474.5	595.1	606.4	606.2
	time	7200.0	7200.0	7200.0	7200.0
	integral*	2330.0	3140.0	3150.0	3160.0
	nodes	10.0	27.0	27.0	27.0
	integral	2330.0	3140.0	3150.0	3160.0
atlanta-ip	time	23.3	31.4	31.5	31.6
	integral*	55112.1	66498.4	66528.0	70865.4
	nodes	14375.0	9861.0	9852.0	260171.0
	objective	91.0	95.0	95.0	97.0
	integral	77572.2	60360.0	60360.0	72558.8
bab5	time	7200.0	7200.0	7200.0	7200.0
	integral*	7585.0	13818.2	13765.1	13844.4
	nodes	25819.0	17781.0	17837.0	17728.0
	objective	-106205.7	-106207.2	-106207.2	-106207.2
	integral	23143.2	20850.2	20707.9	20733.8
beasleyC3	time	7200.0	7200.0	7200.0	7200.0
	integral*	91701.2	95215.9	95218.1	107829.5
	nodes	796130.0	1106432.0	1106852.0	2065757.0
	objective	761.0	759.0	759.0	832.0
	integral	9542.5	6425.8	6431.9	67604.0
bell3a	time	7200.0	7200.0	7200.0	7200.0
	integral*	2.3	2.1	2.2	2.3
	nodes	22487.0	23064.0	22579.0	23083.0
	objective	878430.3	878430.3	878430.3	878430.3
	integral	0.0	0.0	0.0	0.0
bell5	time	6.1	5.7	4.2	4.5
	integral*	10.0	0.0	10.0	10.0
	nodes	1140.0	1226.0	1218.0	1224.0
	objective	8966406.5	8966406.5	8966406.5	8966406.5
	integral	6.2	0.2	6.2	6.2
biella1	time	0.8	0.4	0.6	0.6
	integral*	649.4	625.1	614.8	727.5
	nodes	2133.0	2538.0	2436.0	14468.0
	objective	3065005.8	3065005.8	3065005.8	3065005.8
	integral	6224.9	3602.3	3609.6	3983.0

continued on next page

A. Appendix A

Table A.1: Instancewise results of the experiment in Section 5.4.2. **integral***: dual integral, **nodes**: number of branch-and-bound nodes, **objective**: primal bound at termination, **integral**: primal integral, **time**: overall solving time.

Settings		default	estim	oracle	rank-1
Problem					
bienst2	time	781.4	578.7	575.2	1395.4
	integral*	6024.9	6081.0	8201.8	9499.7
	nodes	93988.0	93988.0	92643.0	106272.0
	objective	54.6	54.6	54.6	54.6
	integral	250.7	256.4	246.0	203.6
binkar10_1	time	297.4	297.8	301.6	291.8
	integral*	52.8	59.4	73.4	82.9
	nodes	138787.0	120843.0	119374.0	120533.0
	objective	6742.2	6742.2	6742.2	6742.2
	integral	66.1	61.3	64.6	69.1
blend2	time	177.2	158.3	147.8	137.2
	integral*	16.5	20.9	20.9	19.0
	nodes	412.0	932.0	933.0	634.0
	objective	7.6	7.6	7.6	7.6
	integral	40.7	41.8	41.8	41.4
bley_xl1	time	0.9	1.4	1.4	1.2
	integral*	33176.4	33376.4	33076.4	33268.6
	nodes	20.0	20.0	20.0	20.0
	objective	190.0	190.0	190.0	190.0
	integral	37128.9	37425.6	36974.8	37198.1
bnatt350	time	430.6	434.6	429.0	431.4
	integral*	80.0	70.0	70.0	80.0
	nodes	21343.0	14092.0	14092.0	14092.0
	objective	0.0	0.0	0.0	0.0
	integral	147700.0	81900.0	81894.0	82867.0
cap6000	time	1477.1	819.2	818.9	828.6
	integral*	40.0	50.0	40.0	50.0
	nodes	3788.0	4064.0	4080.0	4561.0
	objective	-2451377.0	-2451377.0	-2451377.0	-2451377.0
	integral	36.6	46.2	36.4	46.2
core2536-691	time	2.7	3.1	2.7	2.8
	integral*	980.3	990.5	1030.7	980.5
	nodes	218.0	218.0	218.0	481.0
	objective	689.0	689.0	689.0	689.0
	integral	1278.3	1309.6	1355.2	1299.6

continued on next page

Table A.1: Instancewise results of the experiment in Section 5.4.2. **integral***: dual integral, **nodes**: number of branch-and-bound nodes, **objective**: primal bound at termination, **integral**: primal integral, **time**: overall solving time.

Settings		default	estim	oracle	rank-1
Problem					
cov1075	time	318.1	321.2	324.1	319.6
	integral*	62170.0	62169.9	90051.2	76188.9
	nodes	1557428.0	1559145.0	1635309.0	1854530.0
	objective	20.0	20.0	20.0	20.0
	integral	250.8	243.1	242.6	239.0
csched010	time	7200.0	7200.0	6923.5	7200.0
	integral*	47458.5	76490.1	63870.3	82644.7
	nodes	931270.0	1049236.0	997781.0	1128300.0
	objective	408.0	409.0	408.0	410.0
	integral	8987.5	10338.6	9311.0	16636.5
danoint	time	7200.0	7200.0	7200.0	7200.0
	integral*	16556.3	15726.8	16750.4	19819.0
	nodes	1050040.0	966643.0	881640.0	1089980.0
	objective	65.7	65.7	65.7	65.7
	integral	1010.9	291.5	294.9	2374.2
dcmulti	time	5078.2	4785.0	3836.3	4451.6
	integral*	1.0	0.9	0.9	0.9
	nodes	322.0	316.0	306.0	261.0
	objective	188182.0	188182.0	188182.0	188182.0
	integral	119.5	61.0	61.0	61.0
dfn-gwin-UUM	time	1.6	1.2	1.2	1.2
	integral*	658.8	669.3	780.1	733.7
	nodes	66936.0	61708.0	63462.0	60074.0
	objective	38752.0	38752.0	38752.0	38752.0
	integral	436.6	437.8	430.8	434.5
disctom	time	139.6	133.4	113.9	115.1
	integral*	190.0	200.0	200.0	200.0
	nodes	1.0	1.0	1.0	1.0
	objective	-5000.0	-5000.0	-5000.0	-5000.0
	integral	366.0	390.0	398.0	386.0
ds	time	3.6	3.9	3.9	3.8
	integral*	273079.9	274607.1	274604.6	274595.1
	nodes	523.0	542.0	546.0	546.0
	objective	361.0	316.6	316.6	316.6
	integral	571215.7	544517.3	544297.5	544375.4

continued on next page

A. Appendix A

Table A.1: Instancewise results of the experiment in Section 5.4.2. **integral***: dual integral, **nodes**: number of branch-and-bound nodes, **objective**: primal bound at termination, **integral**: primal integral, **time**: overall solving time.

Settings		default	estim	oracle	rank-1
Problem					
dsbmip	time	7200.0	7200.0	7200.0	7200.0
	integral*	30.0	40.0	50.0	60.0
	nodes	15.0	11.0	11.0	11.0
	objective	-305.2	-305.2	-305.2	-305.2
	integral	75.0	75.9	91.0	89.0
egout	time	1.2	1.1	1.4	1.3
	integral*	0.0	0.0	0.0	0.0
	nodes	1.0	1.0	1.0	1.0
	objective	568.1	568.1	568.1	568.1
	integral	0.0	0.0	0.0	0.0
eil33-2	time	0.0	0.0	0.0	0.0
	integral*	700.9	728.0	969.4	1149.3
	nodes	735.0	851.0	1235.0	1339.0
	objective	934.0	934.0	934.0	934.0
	integral	518.0	501.3	519.8	595.2
eilB101	time	52.8	57.3	78.9	96.7
	integral*	2920.5	2997.4	2971.5	2276.5
	nodes	8028.0	8283.0	8357.0	6776.0
	objective	1216.9	1216.9	1216.9	1216.9
	integral	1262.9	1781.2	1757.0	822.5
enigma	time	436.2	477.7	474.1	323.3
	integral*	0.0	0.0	0.0	0.0
	nodes	954.0	2759.0	2759.0	2759.0
	objective	0.0	0.0	0.0	0.0
	integral	50.0	68.0	60.0	78.0
enlight13	time	0.5	0.6	0.6	0.7
	integral*	769.1	1386.2	10777.8	970.0
	nodes	13479.0	30211.0	270890.0	23151.0
	objective	71.0	71.0	71.0	71.0
	integral	0.0	0.0	0.0	0.0
enlight14	time	8.6	16.8	119.5	10.7
	integral*	0.0	0.0	0.0	0.0
	nodes	1.0	1.0	1.0	1.0
	integral	0.0	0.0	0.0	0.0
	time	0.0	0.0	0.0	0.0

continued on next page

Table A.1: Instancewise results of the experiment in Section 5.4.2. **integral***: dual integral, **nodes**: number of branch-and-bound nodes, **objective**: primal bound at termination, **integral**: primal integral, **time**: overall solving time.

Settings		default	estim	oracle	rank-1
Problem					
ex9	integral*	3820.0	3730.0	3950.0	3750.0
	nodes	1.0	1.0	1.0	1.0
	objective	81.0	81.0	81.0	81.0
	integral	3820.0	3728.0	3950.0	3746.0
	time	38.2	37.2	39.5	37.4
fast0507	integral*	1250.4	1258.6	1278.5	1287.0
	nodes	1376.0	1376.0	1348.0	1862.0
	objective	174.0	174.0	174.0	174.0
	integral	907.7	932.2	947.8	937.1
	time	576.8	574.8	574.8	615.3
fiber	integral*	10.6	7.5	22.1	18.5
	nodes	8.0	8.0	5.0	8.0
	objective	405935.2	405935.2	405935.2	405935.2
	integral	40.8	28.6	50.6	50.1
	time	1.6	1.5	3.4	1.7
fixnet6	integral*	15.8	26.3	29.6	25.9
	nodes	9.0	9.0	8.0	9.0
	objective	3983.0	3983.0	3983.0	3983.0
	integral	13.8	23.8	23.8	22.6
	time	3.0	3.3	7.5	3.3
flugpl	integral*	0.0	0.2	0.0	0.2
	nodes	251.0	115.0	115.0	174.0
	objective	1201500.0	1201500.0	1201500.0	1201500.0
	integral	0.0	0.3	0.0	0.3
	time	0.0	0.0	0.0	0.0
gen	integral*	10.0	10.0	10.0	10.0
	nodes	1.0	1.0	1.0	1.0
	objective	112313.4	112313.4	112313.4	112313.4
	integral	6.0	6.0	7.0	6.0
	time	0.1	0.1	0.1	0.1
gesa2	integral*	10.1	10.2	10.2	10.2
	nodes	2.0	2.0	2.0	2.0
	objective	25779856.4	25779856.4	25779856.4	25779856.4
	integral	69.8	96.2	89.4	80.8
	time	0.9	1.2	1.1	1.0

continued on next page

Table A.1: Instancewise results of the experiment in Section 5.4.2. **integral***: dual integral, **nodes**: number of branch-and-bound nodes, **objective**: primal bound at termination, **integral**: primal integral, **time**: overall solving time.

Settings		default	estim	oracle	rank-1
Problem					
gesa2-o	integral*	10.3	10.4	10.4	10.4
	nodes	5.0	5.0	5.0	5.0
	objective	25779856.4	25779856.4	25779856.4	25779856.4
	integral	23.2	23.8	23.8	25.7
	time	1.2	1.3	1.2	1.6
gesa3	integral*	10.1	10.2	10.2	10.2
	nodes	7.0	7.0	6.0	7.0
	objective	27991042.6	27991042.6	27991042.6	27991042.6
	integral	13.8	15.2	15.7	26.0
	time	1.6	1.6	2.4	2.0
gesa3_o	integral*	10.1	10.2	10.3	10.2
	nodes	8.0	8.0	8.0	8.0
	objective	27991042.6	27991042.6	27991042.6	27991042.6
	integral	15.5	16.5	27.4	24.7
	time	1.6	1.8	1.9	1.7
glass4	integral*	158051.6	210964.4	211095.6	240008.6
	nodes	16199130.0	14938613.0	14873922.0	19924545.0
	objective	1550013650.0	1575014925.0	1575014925.0	1566682704.5
	integral	165720.8	178427.0	178445.2	175851.2
	time	7200.0	7200.0	7200.0	7200.0
gmu-35-40	integral*	64.9	71.0	71.0	74.9
	nodes	13065327.0	14149261.0	14168881.0	22385757.0
	objective	-2406528.8	-2406328.9	-2406328.9	-2404683.6
	integral	118.5	159.7	159.5	635.5
	time	7200.0	7200.0	7200.0	7200.0
gt2	integral*	0.0	0.2	0.2	0.2
	nodes	1.0	1.0	1.0	1.0
	objective	21166.0	21166.0	21166.0	21166.0
	integral	1.0	5.9	5.9	5.9
	time	0.0	0.1	0.1	0.1
harp2	integral*	56.7	48.7	48.7	1592.9
	nodes	12630591.0	11703033.0	11722399.0	22409557.0
	objective	-73899798.0	-73899798.0	-73899798.0	-73899797.0
	integral	25.4	26.3	26.5	615.4
	time	3700.6	4479.6	4430.6	7200.0

continued on next page

Table A.1: Instancewise results of the experiment in Section 5.4.2. **integral***: dual integral, **nodes**: number of branch-and-bound nodes, **objective**: primal bound at termination, **integral**: primal integral, **time**: overall solving time.

Settings		default	estim	oracle	rank-1
Problem					
iis-100-0-cov	integral*	29247.4	30216.5	39767.6	45359.1
	nodes	102734.0	105711.0	85533.0	89706.0
	objective	29.0	29.0	29.0	29.0
	integral	623.8	651.6	658.5	782.8
	time	1663.9	1722.3	1305.2	1358.8
iis-bupa-cov	integral*	65009.9	67794.8	104947.8	114341.6
	nodes	182534.0	179812.0	172416.0	182329.0
	objective	36.0	36.0	36.0	36.0
	integral	1397.0	1155.4	1115.4	2337.0
	time	6142.9	6512.3	5434.2	5552.2
iis-pima-cov	integral*	19588.8	19722.4	19735.1	10262.7
	nodes	20364.0	20278.0	20296.0	7935.0
	objective	33.0	33.0	33.0	33.0
	integral	4259.5	4391.8	4382.1	1156.2
	time	1383.2	1388.3	1388.6	610.8
khb05250	integral*	0.1	1.1	1.1	1.1
	nodes	3.0	3.0	2.0	3.0
	objective	106940226.0	106940226.0	106940226.0	106940226.0
	integral	2.5	3.2	3.2	3.2
	time	0.5	0.6	1.0	0.6
l152lav	integral*	12.9	13.5	13.8	23.7
	nodes	49.0	92.0	92.0	92.0
	objective	4722.0	4722.0	4722.0	4722.0
	integral	91.0	57.6	62.0	68.6
	time	2.5	3.0	3.2	3.3
lectsched-4-obj	integral*	237.1	249.8	237.5	237.5
	nodes	24222.0	8296.0	9683.0	9683.0
	objective	4.0	4.0	4.0	4.0
	integral	21706.8	8547.6	15065.8	15061.2
	time	399.0	109.6	200.4	200.1
lseu	integral*	3.5	5.4	5.5	5.0
	nodes	336.0	606.0	602.0	379.0
	objective	1120.0	1120.0	1120.0	1120.0
	integral	2.3	8.5	8.5	8.2
	time	0.6	0.8	0.9	0.8

continued on next page

A. Appendix A

Table A.1: Instancewise results of the experiment in Section 5.4.2. **integral***: dual integral, **nodes**: number of branch-and-bound nodes, **objective**: primal bound at termination, **integral**: primal integral, **time**: overall solving time.

Settings		default	estim	oracle	rank-1
Problem					
m100n500k4r1	integral*	0.0	0.0	0.0	0.0
	nodes	7184542.0	6987522.0	6993419.0	2046170.0
	objective	-24.0	-24.0	-24.0	-24.0
	integral	28957.2	28896.8	28900.4	29028.8
	time	7200.0	7200.0	7200.0	7200.0
macrophage	integral*	146518.1	146136.8	146093.5	146068.1
	nodes	1251604.0	1233931.0	1236445.0	1239753.0
	objective	375.0	376.0	376.0	376.0
	integral	10371.0	5015.6	5011.6	5012.4
	time	7200.0	7200.0	7200.0	7200.0
manna81	integral*	30.4	10.2	30.4	10.3
	nodes	1.0	1.0	1.0	1.0
	objective	-13164.0	-13164.0	-13164.0	-13164.0
	integral	14.7	10.0	14.7	10.0
	time	0.8	0.4	0.8	0.6
map18	integral*	3287.1	3879.2	3494.7	3934.9
	nodes	393.0	315.0	333.0	305.0
	objective	-847.0	-847.0	-847.0	-847.0
	integral	3828.6	4000.9	4020.4	4000.4
	time	424.3	433.5	382.6	438.6
map20	integral*	2763.1	2743.1	2746.8	2707.6
	nodes	299.0	299.0	319.0	315.0
	objective	-922.0	-922.0	-922.0	-922.0
	integral	2792.9	2785.2	2770.5	2780.5
	time	335.0	333.3	333.9	327.6
markshare1	integral*	720000.0	720000.0	720000.0	720000.0
	nodes	73327325.0	76824489.0	75830406.0	43086938.0
	objective	5.0	4.0	4.0	3.0
	integral	602368.8	559538.0	559815.9	503482.0
	time	7200.0	7200.0	7200.0	7200.0
markshare2	integral*	720000.0	720000.0	720000.0	720000.0
	nodes	60920471.0	60781960.0	60892446.0	28720432.0
	objective	13.0	12.0	12.0	13.0
	integral	675988.0	661153.5	661142.1	668024.1
	time	7200.1	7200.1	7200.0	7200.0

continued on next page

Table A.1: Instancewise results of the experiment in Section 5.4.2. **integral***: dual integral, **nodes**: number of branch-and-bound nodes, **objective**: primal bound at termination, **integral**: primal integral, **time**: overall solving time.

Settings		default	estim	oracle	rank-1
Problem					
mas74	integral*	2161.4	2268.0	2805.4	3517.6
	nodes	2834519.0	2767121.0	2760117.0	2594501.0
	objective	11801.2	11801.2	11801.2	11801.2
	integral	79.7	332.6	331.8	32.5
	time	565.4	583.8	516.2	440.1
mas76	integral*	82.0	96.0	278.7	133.5
	nodes	404939.0	471714.0	848147.0	436756.0
	objective	40005.1	40005.1	40005.1	40005.1
	integral	8.5	9.1	9.1	7.6
	time	66.7	79.2	118.7	56.2
mcsched	integral*	1240.3	1116.8	1103.0	1259.6
	nodes	19507.0	15565.0	15471.0	14275.0
	objective	211913.0	211913.0	211913.0	211913.0
	integral	230.5	246.4	237.8	234.9
	time	211.9	172.9	173.4	159.9
mik-250-1-100-1	integral*	744.6	738.2	944.9	1006.2
	nodes	943440.0	943440.0	595707.0	683166.0
	objective	-66729.0	-66729.0	-66729.0	-66729.0
	integral	10.2	0.2	10.2	10.2
	time	365.1	362.7	236.8	278.7
mine-166-5	integral*	431.7	453.0	427.0	425.8
	nodes	2045.0	2045.0	1997.0	2045.0
	objective	-566395707.9	-566395707.9	-566395707.9	-566395707.9
	integral	1654.3	1626.2	1618.3	1608.6
	time	31.0	30.7	30.7	31.0
mine-90-10	integral*	409.8	360.6	382.0	354.7
	nodes	77784.0	68094.0	67313.0	57851.0
	objective	-784302337.6	-784302337.6	-784302337.6	-784302337.6
	integral	1944.8	1813.7	1815.5	1789.9
	time	256.3	228.4	232.2	199.4
misc03	integral*	42.4	38.5	35.0	41.9
	nodes	139.0	123.0	137.0	170.0
	objective	3360.0	3360.0	3360.0	3360.0
	integral	49.0	31.5	19.1	30.8
	time	1.2	1.1	1.0	1.2

continued on next page

Table A.1: Instancewise results of the experiment in Section 5.4.2. **integral***: dual integral, **nodes**: number of branch-and-bound nodes, **objective**: primal bound at termination, **integral**: primal integral, **time**: overall solving time.

Settings		default	estim	oracle	rank-1
Problem					
misc06	integral*	10.0	10.0	10.0	10.0
	nodes	4.0	4.0	6.0	4.0
	objective	12850.9	12850.9	12850.9	12850.9
	integral	6.4	5.5	5.5	5.5
	time	0.7	0.5	0.8	0.5
misc07	integral*	562.5	510.5	557.1	596.4
	nodes	21721.0	20003.0	15439.0	17292.0
	objective	2810.0	2810.0	2810.0	2810.0
	integral	63.6	34.6	52.9	34.6
	time	14.4	13.1	11.2	12.3
mitre	integral*	590.0	580.0	660.0	590.0
	nodes	1.0	1.0	1.0	1.0
	objective	115155.0	115155.0	115155.0	115155.0
	integral	582.4	580.2	652.4	582.4
	time	6.0	5.9	6.7	6.0
mkc	integral*	1306.8	1296.7	1307.1	1340.1
	nodes	2524672.0	2989875.0	2985313.0	3871184.0
	objective	-563.7	-563.6	-563.6	-559.6
	integral	2537.0	2067.8	2080.1	5789.1
	time	7200.0	7200.0	7200.0	7200.0
mod008	integral*	3.2	2.3	3.6	3.6
	nodes	7.0	7.0	4.0	7.0
	objective	307.0	307.0	307.0	307.0
	integral	11.6	8.7	13.1	13.1
	time	0.9	0.7	1.2	1.1
mod010	integral*	20.1	20.1	20.1	20.1
	nodes	2.0	7.0	7.0	7.0
	objective	6548.0	6548.0	6548.0	6548.0
	integral	68.0	60.2	60.0	60.0
	time	0.8	0.7	0.7	0.7
mod011	integral*	491.4	517.1	476.4	464.9
	nodes	1229.0	1229.0	1021.0	1068.0
	objective	-54558535.0	-54558535.0	-54558535.0	-54558535.0
	integral	1540.6	1570.5	1559.4	1571.1
	time	176.8	180.3	152.0	145.8

continued on next page

Table A.1: Instancewise results of the experiment in Section 5.4.2. **integral***: dual integral, **nodes**: number of branch-and-bound nodes, **objective**: primal bound at termination, **integral**: primal integral, **time**: overall solving time.

Settings		default	estim	oracle	rank-1
Problem					
modglob	integral*	0.3	0.4	0.5	0.4
	nodes	905.0	905.0	739.0	820.0
	objective	20740508.1	20740508.1	20740508.1	20740508.1
	integral	0.2	0.2	0.3	0.2
	time	1.4	1.5	1.5	1.3
momentum1	integral*	66647.1	60364.6	60205.1	60150.5
	nodes	44070.0	15148.0	15305.0	15331.0
	objective	115610.8	160511.2	160511.2	160511.2
	integral	113930.9	260453.6	260146.2	260052.7
	time	7200.0	7200.0	7200.0	7200.0
momentum2	integral*	69616.4	94193.3	80807.5	94353.1
	nodes	90508.0	99580.0	86526.0	74211.0
	objective	12314.4	12315.1	12314.6	13813.8
	integral	85974.5	83308.9	82711.0	112704.8
	time	7200.0	7200.0	7200.0	7200.0
msc98-ip	integral*	5430.7	5420.6	5410.6	5410.8
	nodes	3391.0	18438.0	10164.0	10162.0
	objective	21655010.0	22273180.0	22273180.0	22273180.0
	integral	353627.0	147592.2	147355.8	147440.3
	time	7200.0	7200.0	7200.0	7200.0
mspp16	integral*	61662.4	63439.4	78986.1	63230.3
	nodes	51.0	57.0	47.0	57.0
	objective	363.0	363.0	363.0	363.0
	integral	49036.0	49190.0	49030.0	49060.0
	time	2579.3	2841.5	5437.7	2838.0
mzzv11	integral*	4336.8	4216.2	4278.5	4223.1
	nodes	1999.0	1999.0	1908.0	1975.0
	objective	-21718.0	-21718.0	-21718.0	-21718.0
	integral	12450.4	12445.9	12449.1	12349.4
	time	267.9	266.0	258.1	262.7
mzzv42z	integral*	3052.0	3019.6	3050.0	3037.6
	nodes	534.0	534.0	536.0	1012.0
	objective	-20540.0	-20540.0	-20540.0	-20540.0
	integral	12112.4	12014.9	12018.5	11988.8
	time	340.3	337.7	338.4	316.6

continued on next page

A. Appendix A

Table A.1: Instancewise results of the experiment in Section 5.4.2. **integral***: dual integral, **nodes**: number of branch-and-bound nodes, **objective**: primal bound at termination, **integral**: primal integral, **time**: overall solving time.

Settings		default	estim	oracle	rank-1
Problem					
n3div36	integral*	43102.2	43060.7	45913.6	43935.3
	nodes	250934.0	260655.0	372168.0	345004.0
	objective	131000.0	130800.0	130800.0	130800.0
	integral	5793.9	5732.8	5758.1	5974.6
	time	7200.0	7200.0	7200.0	7200.0
n3seq24	integral*	7360.9	7520.3	7430.7	7420.7
	nodes	393.0	392.0	393.0	393.0
	objective	61600.0	61600.0	61600.0	61600.0
	integral	172367.6	174014.4	173563.3	173691.5
	time	7200.0	7200.0	7200.0	7200.0
n4-3	integral*	3286.1	3343.7	3975.2	5487.5
	nodes	32231.0	33154.0	29104.0	44646.0
	objective	8993.0	8993.0	8993.0	8993.0
	integral	1409.3	1439.8	1443.4	1754.8
	time	542.3	564.0	472.5	633.3
neos-1109824	integral*	1397.6	1409.5	1231.8	1462.6
	nodes	21927.0	22678.0	10652.0	14781.0
	objective	378.0	378.0	378.0	378.0
	integral	773.5	994.5	981.8	984.9
	time	156.1	154.0	103.4	123.3
neos-1337307	integral*	1127.4	2995.5	2506.2	1025.0
	nodes	370421.0	553458.0	519383.0	391710.0
	objective	-202319.0	-202319.0	-202319.0	-202319.0
	integral	8200.6	5426.7	5280.8	5281.7
	time	7200.0	7200.0	7200.0	7200.0
neos-1396125	integral*	11732.0	15076.5	15160.0	27068.1
	nodes	61200.0	69115.0	59721.0	70372.0
	objective	3000.0	3000.0	3000.0	3000.0
	integral	4047.4	5430.3	5570.4	5463.6
	time	766.1	925.5	856.8	778.5
neos-1601936	integral*	1433.3	1433.3	1530.0	1423.3
	nodes	6755.0	1615.0	1606.0	1615.0
	objective	4.0	6.0	6.0	6.0
	integral	251274.8	686858.2	688908.0	686958.2
	time	7200.0	7200.0	7200.0	7200.0

continued on next page

Table A.1: Instancewise results of the experiment in Section 5.4.2. **integral***: dual integral, **nodes**: number of branch-and-bound nodes, **objective**: primal bound at termination, **integral**: primal integral, **time**: overall solving time.

Settings		default	estim	oracle	rank-1
Problem					
neos-476283	integral*	7894.7	8034.8	8124.8	7884.8
	nodes	685.0	685.0	667.0	855.0
	objective	406.4	406.4	406.4	406.4
	integral	9009.9	9194.0	9298.1	9030.1
	time	275.9	282.0	279.6	279.8
neos-686190	integral*	1453.1	1825.5	1770.5	1757.6
	nodes	7264.0	10378.0	9405.0	9445.0
	objective	6730.0	6730.0	6730.0	6730.0
	integral	3954.3	1825.0	1866.9	1848.6
	time	93.8	118.6	110.5	109.8
neos-849702	integral*	180.0	160.0	170.0	160.0
	nodes	6115.0	48917.0	48917.0	48917.0
	objective	0.0	0.0	0.0	0.0
	integral	17471.0	55887.0	55967.0	55999.0
	time	174.7	558.8	559.6	559.9
neos-916792	integral*	3591.9	3537.8	3480.3	9305.6
	nodes	106472.0	123066.0	124088.0	210792.0
	objective	31.9	31.9	31.9	31.9
	integral	938.9	923.6	904.0	1075.7
	time	406.2	454.3	399.2	593.9
neos-934278	integral*	3221.1	3221.1	3251.0	3251.0
	nodes	889.0	1133.0	992.0	1095.0
	objective	275.0	271.0	271.0	271.0
	integral	133833.6	333188.8	341132.6	335158.6
	time	7200.0	7200.0	7200.0	7200.0
neos13	integral*	33785.0	41656.5	41652.2	42219.1
	nodes	4422.0	3230.0	3230.0	3209.0
	objective	-95.5	-95.5	-95.5	-95.5
	integral	40072.4	44630.5	44623.2	45119.0
	time	1514.8	1727.5	1725.1	1738.6
neos18	integral*	509.0	538.1	567.4	545.6
	nodes	6778.0	5601.0	5179.0	6249.0
	objective	16.0	16.0	16.0	16.0
	integral	390.2	373.1	363.1	338.7
	time	32.1	29.8	31.9	31.6

continued on next page

A. Appendix A

Table A.1: Instancewise results of the experiment in Section 5.4.2. **integral***: dual integral, **nodes**: number of branch-and-bound nodes, **objective**: primal bound at termination, **integral**: primal integral, **time**: overall solving time.

Settings		default	estim	oracle	rank-1
Problem					
net12	integral*	135566.4	143794.4	155271.1	198889.8
	nodes	3864.0	4985.0	5016.0	4605.0
	objective	214.0	214.0	214.0	214.0
	integral	91859.7	31752.4	31764.8	16073.1
	time	2532.6	2746.2	3018.6	3473.0
netdiversion	integral*	88934.9	86721.9	86057.6	86149.9
	nodes	72.0	119.0	113.0	113.0
	objective	251.0	242.0	242.0	242.0
	integral	358622.1	554572.7	554572.7	554572.7
	time	7200.0	6630.7	6581.5	6580.6
newdano	integral*	83214.8	83697.5	112165.5	153952.0
	nodes	2083404.0	2083404.0	1993781.0	2002198.0
	objective	65.7	65.7	65.7	65.7
	integral	1835.3	1846.3	1842.6	1647.9
	time	3557.7	3573.5	3704.2	3242.0
noswot	integral*	825.1	837.9	814.4	805.1
	nodes	829543.0	829543.0	436956.0	455271.0
	objective	-41.0	-41.0	-41.0	-41.0
	integral	10.0	12.7	14.1	12.9
	time	177.5	180.3	175.1	173.1
ns1208400	integral*	187020.0	127900.0	151980.0	106290.0
	nodes	3118.0	2777.0	2785.0	2772.0
	objective	2.0	2.0	2.0	2.0
	integral	180863.0	26900.0	26900.0	27000.0
	time	1870.2	1279.0	1519.8	1062.9
ns1688347	integral*	12504.0	6524.3	7857.2	7727.9
	nodes	6667.0	2609.0	3905.0	4330.0
	objective	27.0	27.0	27.0	27.0
	integral	33631.8	10151.3	10354.4	10263.8
	time	738.9	275.2	388.7	380.6
ns1758913	integral*	78175.5	78124.6	78053.3	79121.5
	nodes	2.0	2.0	2.0	2.0
	objective	-236.8	-236.8	-236.8	-236.8
	integral	613682.7	613682.2	613665.3	613842.1
	time	7200.0	7200.0	7200.0	7200.0

continued on next page

Table A.1: Instancewise results of the experiment in Section 5.4.2. **integral***: dual integral, **nodes**: number of branch-and-bound nodes, **objective**: primal bound at termination, **integral**: primal integral, **time**: overall solving time.

Settings		default	estim	oracle	rank-1
Problem					
ns1766074	integral*	82440.0	84260.0	84420.0	84170.0
	nodes	942544.0	893992.0	893992.0	893992.0
	integral	82440.0	84260.0	84420.0	84170.0
	time	824.4	842.6	844.2	841.7
	time	824.4	842.6	844.2	841.7
ns1830653	integral*	17278.1	17348.0	15647.9	18374.2
	nodes	41218.0	46638.0	36114.0	46887.0
	objective	20622.0	20622.0	20622.0	20622.0
	integral	8839.0	7288.8	6224.8	7889.9
	time	440.3	371.5	382.9	394.2
nsrand-ipx	integral*	5735.0	5194.8	5216.1	5206.5
	nodes	1599798.0	1763180.0	1758600.0	1730377.0
	objective	51840.0	51360.0	51360.0	51360.0
	integral	11873.2	5928.3	5970.0	5998.2
	time	7200.0	7200.0	7200.0	7200.0
nw04	integral*	967.3	986.4	980.6	997.7
	nodes	11.0	11.0	6.0	11.0
	objective	16862.0	16862.0	16862.0	16862.0
	integral	1144.7	1159.7	1151.2	1174.7
	time	24.5	24.4	25.3	24.9
opm2-z7-s2	integral*	11760.0	11726.0	11834.2	18626.1
	nodes	2092.0	2092.0	2094.0	15231.0
	objective	-10280.0	-10280.0	-10280.0	-10280.0
	integral	8068.2	8138.1	8277.5	8477.0
	time	794.6	789.9	794.6	1220.6
opt1217	integral*	2.8	5.4	7.7	5.4
	nodes	1.0	1.0	1.0	1.0
	objective	-16.0	-16.0	-16.0	-16.0
	integral	0.0	0.0	0.0	0.0
	time	0.3	0.4	0.4	0.4
p0033	integral*	0.3	0.4	0.3	0.3
	nodes	1.0	1.0	1.0	1.0
	objective	3089.0	3089.0	3089.0	3089.0
	integral	0.8	0.8	0.8	0.8
	time	0.1	0.1	0.1	0.1
p0201	integral*	14.6	14.3	14.7	4.7

continued on next page

A. Appendix A

Table A.1: Instancewise results of the experiment in Section 5.4.2. **integral***: dual integral, **nodes**: number of branch-and-bound nodes, **objective**: primal bound at termination, **integral**: primal integral, **time**: overall solving time.

Settings		default	estim	oracle	rank-1
Problem					
p0282	nodes	67.0	67.0	59.0	65.0
	objective	7615.0	7615.0	7615.0	7615.0
	integral	23.9	21.1	24.0	14.9
	time	1.8	1.7	1.8	1.7
	integral*	0.3	0.8	0.8	0.8
p0548	nodes	3.0	3.0	1.0	3.0
	objective	258411.0	258411.0	258411.0	258411.0
	integral	1.0	2.3	2.4	2.3
	time	0.3	0.6	0.8	0.6
	integral*	0.8	10.2	10.2	10.2
p2756	nodes	1.0	1.0	1.0	1.0
	objective	8691.0	8691.0	8691.0	8691.0
	integral	10.1	20.1	20.1	20.1
	time	0.1	0.3	0.3	0.3
	integral*	21.5	11.4	31.9	11.4
pg5_34	nodes	137.0	9.0	9.0	9.0
	objective	3124.0	3124.0	3124.0	3124.0
	integral	26.1	16.2	31.9	16.2
	time	1.6	1.2	1.4	1.2
	integral*	156.6	175.1	175.4	252.0
pigeon-10	nodes	291242.0	291323.0	273355.0	305210.0
	objective	-14339.4	-14339.4	-14339.4	-14339.4
	integral	163.2	183.8	183.8	190.0
	time	1287.1	1338.1	1297.1	1400.7
	integral*	72009.0	72009.0	72009.0	72009.0
pk1	nodes	17116573.0	17457654.0	10565366.0	17502182.0
	objective	-9000.0	-9000.0	-9000.0	-9000.0
	integral	520.0	500.0	500.0	520.0
	time	7200.0	7200.0	7200.0	7200.0
	integral*	3399.4	3453.3	4329.0	4349.3
pp08a	nodes	284323.0	284323.0	281331.0	281341.0
	objective	11.0	11.0	11.0	11.0
	integral	311.4	328.9	343.5	347.9
	time	64.2	65.2	53.8	55.0
	integral*	5.5	3.0	5.7	3.7

continued on next page

Table A.1: Instancewise results of the experiment in Section 5.4.2. **integral***: dual integral, **nodes**: number of branch-and-bound nodes, **objective**: primal bound at termination, **integral**: primal integral, **time**: overall solving time.

Settings		default	estim	oracle	rank-1
Problem					
pp08aCUTS	nodes	221.0	225.0	231.0	253.0
	objective	7350.0	7350.0	7350.0	7350.0
	integral	27.2	15.8	29.7	22.8
	time	1.3	1.1	1.4	1.3
	integral*	2.8	3.4	5.5	3.2
	nodes	194.0	165.0	149.0	153.0
	objective	7350.0	7350.0	7350.0	7350.0
	integral	19.8	30.1	32.8	22.9
	time	1.1	1.4	1.4	1.3
	integral*	146786.0	151335.2	151308.6	151293.3
protfold	nodes	10226.0	11588.0	11587.0	11595.0
	objective	-23.0	-20.0	-20.0	-20.0
	integral	251536.1	300859.4	300804.5	300826.7
	time	7200.0	7200.0	7200.0	7200.0
	integral*	110490.5	111043.5	132077.2	157876.3
	nodes	712713.0	712713.0	368355.0	433819.0
	objective	10.0	10.0	10.0	10.0
	integral	1816.4	1887.4	1878.4	1856.1
	time	3542.8	3550.0	2199.1	2629.2
	integral*	5951.6	6199.1	6272.9	6260.0
qiu	nodes	12604.0	12618.0	12616.0	12629.0
	objective	-132.9	-132.9	-132.9	-132.9
	integral	2123.9	2134.9	2153.9	2083.5
	time	79.9	81.6	77.8	77.2
	integral*	40.1	22.5	35.2	22.7
	nodes	36.0	7.0	7.0	7.0
	objective	16029.7	16029.7	16029.7	16029.7
	integral	97.6	54.6	70.7	54.6
	time	8.3	4.5	5.3	4.5
	integral*	11.0	9.8	18.9	19.1
qnet1_o	nodes	16.0	6.0	6.0	6.0
	objective	16029.7	16029.7	16029.7	16029.7
	integral	62.0	51.8	61.8	62.8
	time	6.6	5.1	5.3	5.4
	integral*	1246.6	1244.4	1230.5	1234.1
rail507	nodes				
	objective				
	integral				
	time				
	integral*				

continued on next page

A. Appendix A

Table A.1: Instancewise results of the experiment in Section 5.4.2. **integral***: dual integral, **nodes**: number of branch-and-bound nodes, **objective**: primal bound at termination, **integral**: primal integral, **time**: overall solving time.

Settings		default	estim	oracle	rank-1
Problem					
ran16x16	nodes	799.0	799.0	855.0	865.0
	objective	174.0	174.0	174.0	174.0
	integral	1380.5	1379.9	1372.0	1372.0
	time	242.4	239.9	235.7	239.6
	integral*	838.1	813.8	857.8	1343.3
rd-rplusc-21	nodes	368022.0	346094.0	265832.0	373581.0
	objective	3823.0	3823.0	3823.0	3823.0
	integral	69.6	105.8	106.6	162.5
	time	291.3	283.9	231.0	276.4
	integral*	719567.3	719567.3	719567.3	719567.3
reblock67	nodes	77078.0	61764.0	60360.0	70998.0
	objective	165935.9	166009.7	166009.7	177205.0
	integral	60818.1	118211.1	120667.1	131330.8
	time	7200.0	7200.0	7200.0	7200.0
	integral*	558.5	549.1	554.0	538.3
rentacar	nodes	109664.0	109664.0	57072.0	105846.0
	objective	-34630648.4	-34630648.4	-34630648.4	-34630648.4
	integral	1629.4	1701.9	1627.8	1617.9
	time	253.3	251.3	172.6	246.5
	integral*	114.2	104.2	118.4	124.0
rgn	nodes	4.0	4.0	4.0	4.0
	objective	30356761.0	30356761.0	30356761.0	30356761.0
	integral	130.0	120.0	126.0	136.0
	time	2.7	2.6	3.5	2.8
	integral*	1.5	4.2	4.2	4.2
rmatr100-p10	nodes	1.0	1.0	1.0	1.0
	objective	82.2	82.2	82.2	82.2
	integral	8.2	24.5	24.5	24.5
	time	0.1	0.3	0.3	0.3
	integral*	1808.9	1806.0	1778.7	1757.8
rmatr100-p5	nodes	851.0	851.0	909.0	909.0
	objective	423.0	423.0	423.0	423.0
	integral	952.9	942.9	957.4	941.8
	time	135.1	135.8	131.0	130.7
	integral*	5293.7	6929.7	6131.5	6391.4

continued on next page

Table A.1: Instancewise results of the experiment in Section 5.4.2. **integral***: dual integral, **nodes**: number of branch-and-bound nodes, **objective**: primal bound at termination, **integral**: primal integral, **time**: overall solving time.

Settings		default	estim	oracle	rank-1
Problem					
rmine6	nodes	420.0	451.0	483.0	439.0
	objective	976.0	976.0	976.0	976.0
	integral	1492.9	1399.0	1392.0	1376.4
	time	302.9	304.2	267.6	280.0
	integral*	665.5	387.9	395.2	385.2
rocII-4-11	nodes	2004491.0	742664.0	736822.0	738018.0
	objective	-457.2	-457.2	-457.2	-457.2
	integral	367.9	343.8	331.7	331.4
	time	6096.9	2287.9	2246.4	2263.0
	integral*	12279.0	7037.1	10788.3	10042.0
rococoC10-001000	nodes	40477.0	11718.0	30330.0	17308.0
	objective	-6.7	-6.7	-6.7	-6.7
	integral	11909.9	3637.1	10136.1	7544.9
	time	463.1	204.7	433.9	299.9
	integral*	5034.0	4558.5	4816.8	11986.9
roll3000	nodes	203201.0	174936.0	135810.0	224776.0
	objective	11460.0	11460.0	11460.0	11460.0
	integral	752.7	920.8	941.2	2245.8
	time	1217.3	1011.1	876.6	1274.5
	integral*	6626.7	3522.8	2892.8	19509.0
rout	nodes	2781398.0	1063691.0	423972.0	2331855.0
	objective	12899.0	12890.0	12890.0	12890.0
	integral	1534.3	1083.3	1054.9	4709.2
	time	7200.0	3082.7	1387.3	5522.8
	integral*	224.3	215.0	204.9	207.2
satellites1-25	nodes	26664.0	20646.0	18547.0	18646.0
	objective	1077.6	1077.6	1077.6	1077.6
	integral	162.8	190.5	184.3	190.1
	time	40.1	32.2	27.9	28.2
	integral*	48757.5	74492.3	57389.7	68402.2
set1ch	nodes	3064.0	2648.0	1588.0	2212.0
	objective	-5.0	-5.0	-5.0	-5.0
	integral	31264.0	32148.0	32108.0	45240.0
	time	660.2	995.5	765.3	913.5
	integral*	2.5	4.6	5.3	4.0

continued on next page

A. Appendix A

Table A.1: Instancewise results of the experiment in Section 5.4.2. **integral***: dual integral, **nodes**: number of branch-and-bound nodes, **objective**: primal bound at termination, **integral**: primal integral, **time**: overall solving time.

Settings		default	estim	oracle	rank-1
Problem					
seymour	nodes	9.0	9.0	9.0	9.0
	objective	54537.8	54537.8	54537.8	54537.8
	integral	13.9	16.7	21.1	13.5
	time	0.7	0.8	0.9	0.7
	integral*	15701.9	15933.6	15928.6	15953.0
sp97ar	nodes	150798.0	146737.0	146590.0	146047.0
	objective	423.0	424.0	424.0	424.0
	integral	4737.7	2925.2	2927.8	2946.7
	time	7200.0	7200.0	7200.0	7200.0
	integral*	15504.2	47703.3	48193.8	47725.6
sp98ic	nodes	6289.0	8065.0	8101.0	7619.0
	objective	710063477.9	696853271.7	696853271.7	710898758.9
	integral	60048.1	72299.2	72495.0	72262.6
	time	7200.0	7200.0	7200.0	7200.0
	integral*	3463.5	3263.9	3265.7	3399.2
sp98ir	nodes	131473.0	137628.0	137322.0	185344.0
	objective	452431468.0	450852689.3	450852689.3	450852689.3
	integral	18843.6	5858.7	5856.4	5888.1
	time	7200.0	7200.0	7200.0	7200.0
	integral*	236.9	224.6	220.6	204.4
stein27	nodes	8210.0	5379.0	5375.0	4630.0
	objective	219676790.4	219676790.4	219676790.4	219676790.4
	integral	378.2	458.9	462.1	442.6
	time	106.6	86.3	81.3	69.2
	integral*	27.8	30.6	33.3	19.4
stein45	nodes	3905.0	3905.0	3973.0	3607.0
	objective	18.0	18.0	18.0	18.0
	integral	0.0	0.0	0.0	0.0
	time	1.0	1.1	1.2	0.7
	integral*	207.5	225.0	268.3	283.9
stp3d	nodes	47352.0	47352.0	50336.0	46693.0
	objective	30.0	30.0	30.0	30.0
	integral	4.5	6.6	5.1	5.4
	time	12.2	13.1	11.3	10.7
	integral*	248584.7	249365.6	248779.9	248389.5

continued on next page

Table A.1: Instancewise results of the experiment in Section 5.4.2. **integral***: dual integral, **nodes**: number of branch-and-bound nodes, **objective**: primal bound at termination, **integral**: primal integral, **time**: overall solving time.

Settings		default	estim	oracle	rank-1
Problem					
swath	nodes	1.0	1.0	1.0	1.0
	integral	720000.0	720000.0	720000.0	720000.0
	time	7200.0	7200.0	7200.0	7200.0
	integral*	105447.4	96980.1	96898.0	110113.1
	nodes	1231280.0	1214072.0	1220123.0	1672194.0
tanglegram1	objective	472.6	476.1	476.1	490.6
	integral	14764.9	19927.5	19873.7	34175.3
	time	7200.0	7200.0	7200.0	7200.0
	integral*	89159.2	87762.2	87629.7	88029.6
	nodes	37.0	37.0	37.0	37.0
tanglegram2	objective	5182.0	5182.0	5182.0	5182.0
	integral	15880.2	15738.7	15706.3	15784.6
	time	1161.2	1138.7	1138.6	1142.8
	integral*	1397.2	1415.6	1377.7	1377.7
	nodes	5.0	5.0	5.0	5.0
timtab1	objective	443.0	443.0	443.0	443.0
	integral	711.1	720.2	699.1	697.8
	time	14.9	15.1	14.7	14.7
	integral*	5932.4	5559.8	7258.5	11280.7
	nodes	870361.0	868207.0	896679.0	965573.0
timtab2	objective	764772.0	764772.0	764772.0	764772.0
	integral	385.4	412.6	380.8	356.2
	time	390.1	388.1	376.2	377.6
	integral*	237001.7	326894.8	245532.2	326896.4
	nodes	9144342.0	14814841.0	9027482.0	14855771.0
tr12-30	objective	1136721.0	1208245.0	1138052.0	1208245.0
	integral	38386.2	67445.2	43432.2	67212.9
	time	7200.0	7200.0	7200.0	7200.0
	integral*	191.9	173.2	196.8	294.2
	nodes	1471731.0	1186814.0	1162158.0	1306275.0
triptim1	objective	130596.0	130596.0	130596.0	130596.0
	integral	84.8	93.4	101.3	102.3
	time	1814.6	1521.3	1505.3	1454.0
	integral*	13000.0	12800.0	12700.0	12700.0
	nodes	47.0	4.0	4.0	4.0

continued on next page

A. Appendix A

Table A.1: Instancewise results of the experiment in Section 5.4.2. **integral***: dual integral, **nodes**: number of branch-and-bound nodes, **objective**: primal bound at termination, **integral**: primal integral, **time**: overall solving time.

Settings		default	estim	oracle	rank-1
Problem					
unitcal_7	objective	22.9	22.9	22.9	22.9
	integral	195940.0	98168.0	99763.0	98172.0
	time	2791.3	981.7	997.6	981.7
	integral*	3015.6	2979.1	3036.7	3050.8
	nodes	23265.0	27125.0	19216.0	19861.0
	objective	19635558.2	19635558.2	19635558.2	19635558.2
vpm1	integral	8237.2	8204.4	8238.8	8257.5
	time	1304.6	1469.5	1230.3	1271.4
	integral*	0.0	0.0	0.0	0.0
	nodes	1.0	1.0	1.0	1.0
	objective	20.0	20.0	20.0	20.0
	integral	0.0	0.0	0.0	0.0
vpm2	time	0.0	0.0	0.0	0.0
	integral*	8.4	6.6	10.5	9.0
	nodes	294.0	218.0	224.0	206.0
	objective	13.8	13.8	13.8	13.8
	integral	20.5	17.5	26.2	20.5
	time	1.3	1.2	1.7	1.4
vpphard	integral*	720000.0	720000.0	720000.0	720000.0
	nodes	7476.0	6321.0	6292.0	6321.0
	objective	9.0	30.0	30.0	30.0
	integral	402227.3	605195.8	605180.1	605182.1
	time	7200.0	7200.0	7200.0	7200.0
	integral*	37084.8	49612.0	58244.4	58701.5
zib54-UUE	nodes	539744.0	706521.0	296047.0	294878.0
	objective	10334015.8	10334015.8	10334015.8	10334015.8
	integral	1071.3	1115.0	1110.9	2147.6
	time	3829.7	5375.5	2703.8	2625.2

B

Appendix B

This appendix contains an instance-wise outcome of our computational experiments described in Section 4.3. For each of the five settings, we present three columns; the measured dual integral **integral**^{*}, the number of nodes **nodes**, and the solving time in seconds **time**. Table B.1 shows the results for instances which we classified as small tree instances, and Table B.2 contains the remaining instances, cf. Tables 4.3 and 4.2, respectively.

B. Appendix B

Settings	(hyp)		(rer)-0.01				(rer)-0.1				(rer)-0.05				(fnt)-5			
	integral*	nodes	time	integral*	nodes	time	integral*	nodes	time	integral*	nodes	time	integral*	nodes	time	integral*	nodes	time
Problem																		
30n20b8	12699.9	13	196.4	14587.7	79	221.4	13000.5	18	200.6	15786.8	120	237.6	12756.5	18	196.3			
air04	1766.8	8	36.2	1842.3	8	37.6	1842.3	8	37.6	1847.3	8	37.7	1862.4	8	37.9			
air05	1275.2	50	25.7	1290.3	52	25.9	1290.4	52	26.0	1275.1	52	25.6	1270.2	62	25.6			
app1-2	110453.1	23	1763.5	110730.3	19	1766.7	109918.4	19	1754.8	109878.9	19	1754.2	55296.9	41	875.6			
ash608gpia-3col	2010.0	7	20.1	2120.0	9	21.2	2060.0	9	20.6	2080.0	9	20.8	2000.0	7	20.0			
blend2	23.4	252	0.6	24.3	117	0.7	28.8	155	0.7	23.4	205	0.6	23.4	240	0.6			
dcmulti	5.5	8	1.2	5.4	14	1.0	5.5	14	1.1	5.5	14	1.1	0.3	8	0.8			
fast0507	1457.3	598	140.5	4264.0	714	150.5	4570.0	722	149.6	1128.5	646	146.8	4444.4	630	147.8			
fiber	7.5	4	1.2	8.0	4	1.2	1.8	4	1.0	7.2	4	1.1	4.1	4	1.0			
fixnet6	14.2	18	2.1	19.7	10	2.2	15.3	10	2.2	20.1	10	2.3	11.5	10	1.9			
gesa2	5.2	3	0.8	5.1	3	0.8	5.2	3	0.7	5.2	3	0.7	5.0	3	0.4			
gesa2-o	5.2	2	1.0	5.2	2	1.1	5.2	2	1.0	5.3	2	1.1	5.1	2	0.9			
gesa3	5.2	7	1.3	5.2	7	1.2	5.2	7	1.3	5.1	7	1.0	5.1	7	1.0			
gesa3_o	10.1	7	1.3	5.1	7	1.2	5.2	7	1.3	5.1	7	1.2	0.1	7	0.9			
khh05250	0.1	4	0.5	0.2	4	0.5	0.2	4	0.5	0.2	4	0.5	0.2	4	0.5			
l152lav	31.2	19	1.2	46.6	19	1.6	41.2	19	1.3	41.5	19	1.5	40.9	19	1.1			
lseu	21.8	58	0.5	21.8	64	0.5	21.8	64	0.5	21.8	64	0.5	5.9	191	0.2			
map18	15193.4	309	291.8	15757.4	275	302.4	15821.5	275	303.6	15777.8	275	302.7	15510.6	285	297.8			
map20	12007.8	265	229.9	12518.0	307	239.4	12523.8	307	239.6	12560.5	307	240.2	12353.0	281	236.3			
misc03	43.4	77	1.4	37.8	23	1.2	36.3	23	1.1	33.1	23	1.0	24.1	80	0.8			
misc06	5.0	4	0.5	5.0	4	0.6	5.0	4	0.6	5.0	4	0.6	5.0	4	0.5			
mod008	2.5	7	0.8	0.0	7	0.8	0.0	7	1.0	3.1	7	1.0	2.3	7	0.8			
mod010	5.1	2	0.5	5.1	2	0.5	5.1	2	0.5	5.1	2	0.5	5.1	2	0.5			
mod011	327.1	671	108.8	329.4	671	109.2	326.5	671	108.6	331.6	853	110.4	333.0	855	116.3			
modglob	5.1	21	0.6	5.1	19	0.5	10.1	23	0.6	10.1	25	0.6	0.1	25	0.5			
mspp16	131239.4	57	2474.8	88730.3	31	1673.2	95963.6	29	1809.6	98487.9	29	1857.2	98434.8	31	1856.2			
mzzv42z	5076.1	132	147.7	5117.1	96	148.4	5101.4	96	147.9	5065.3	96	147.2	5458.9	110	155.5			
neos-476283	1971.2	233	73.8	1961.2	105	70.4	2006.2	175	70.1	1976.3	394	74.9	1971.2	110	70.6			
neos13	876.7	6	31.2	883.0	8	31.2	860.5	8	30.8	866.8	8	30.7	897.8	8	32.1			
ns1208400	33880.0	598	338.8	52850.0	860	528.5	52890.0	860	528.9	52820.0	860	528.2	49760.0	881	497.6			
nw04	476.0	8	20.4	480.0	8	20.1	470.0	8	20.0	465.7	8	20.2	496.4	8	20.7			
p0201	42.9	11	1.3	37.7	9	1.2	27.5	9	1.0	32.3	9	1.0	32.6	9	1.1			
p0282	0.5	3	0.5	0.5	3	0.5	0.5	3	0.5	0.0	3	0.4	0.0	3	0.3			
p2756	1.7	3	0.9	1.8	3	1.1	1.8	3	1.1	6.8	3	1.1	5.1	3	0.7			
pp08a	32.4	53	1.2	27.4	51	1.1	32.4	51	1.2	27.4	51	1.1	21.2	161	0.7			
pp08aCUTS	12.8	53	1.2	23.0	51	1.3	18.1	51	1.3	18.0	51	1.2	1.5	153	0.8			
qnet1	16.7	3	2.0	16.7	3	2.0	11.9	3	2.0	17.3	3	2.2	10.2	3	2.0			
qnet1_o	4.1	4	1.4	0.0	4	1.3	4.1	4	1.4	0.0	4	1.4	0.0	4	1.3			
rail507	760.6	554	145.5	664.3	546	136.9	1207.0	582	144.2	704.4	586	142.6	529.0	644	147.1			
rentacar	71.9	4	3.3	77.1	4	3.4	81.7	4	3.8	80.6	4	3.4	80.6	4	3.4			
rmatr100-p10	6884.6	723	121.2	7004.4	793	123.3	7028.6	793	123.8	6981.6	791	122.9	6821.8	709	120.1			
rmatr100-p5	14864.3	339	245.6	15234.8	367	251.7	15562.9	367	257.1	15224.8	367	251.6	14256.9	349	235.6			
set1ch	0.0	3	0.8	0.0	3	0.8	3.0	3	0.8	0.0	3	0.6	0.0	3	0.6			
stp3d	145498.8	17	7200.0	145872.3	14	7200.0	145433.0	14	7200.0	145579.5	14	7200.0	145433.0	14	7200.0			
tanglegram1	92121.1	31	922.3	77378.6	27	774.7	77468.4	27	775.6	77108.9	27	772.0	99072.9	33	991.9			
tanglegram2	783.1	3	7.9	793.1	3	8.0	783.1	3	7.9	793.1	3	8.0	793.1	3	8.0			
vpm2	19.1	298	0.9	35.3	50	1.3	24.8	50	1.1	25.0	110	1.1	19.6	272	1.0			

Table B.1: Instance-wise experimental outcome for instances requiring at most 1000 nodes to solve.

Settings	(hyp)			(rer)-0.01			(rer)-0.1			(rer)-0.05			(fnt)-5		
	integral [*] nodes	nodes time	time	integral [*] nodes	nodes	time	integral [*] nodes	nodes	time	integral [*] nodes	nodes	time			
Problem															
alc1s1	259882.7	1120506	7200.0	259873.5	1364025	7200.0	259879.2	1685774	7200.0	259876.3	2004990	7200.0	259863.8	2070335	7200.0
afflow30a	508.5	1720	12.6	502.7	1908	12.4	507.3	1908	12.4	511.8	1908	12.4	377.5	1246	10.1
afflow40b	20210.8	104858	596.3	29534.9	172263	856.6	31494.3	179651	893.0	9929.0	116421	647.2	16390.8	143682	536.0
arki001	105.7	1120884	7200.0	105.7	592970	7200.0	105.7	754036	7200.0	105.7	1089108	7200.0	105.7	365939	7200.0
atlanta-ip	70465.2	7972	7200.0	70460.7	10011	7200.0	70460.7	10012	7200.0	70469.4	11789	7200.0	70469.7	8529	7200.0
bab5	6707.5	133720	7200.0	6649.7	110708	7200.0	6623.3	105423	7200.0	6644.2	123080	7200.0	6632.3	96735	7200.0
beasleyC3	94243.8	2408887	7200.0	92905.6	2107529	7200.0	92907.7	1377166	7200.0	92907.5	1849676	7200.0	92907.7	2841834	7200.0
bell3a	0.0	20025	3.0	0.0	20027	3.2	1.3	20261	3.1	11.4	21353	3.2	16.3	22611	2.8
bell5	10.0	1025	0.3	10.0	956	0.3	10.0	956	0.3	10.0	956	0.3	10.0	1152	0.3
biella1	34013.5	7109	679.4	19260.3	3629	384.8	19260.3	3629	384.8	15824.9	2771	316.2	33983.5	6883	678.8
bienst2	10925.9	66579	328.4	10702.5	101007	426.9	10718.1	101007	427.7	15947.8	113088	463.4	29985.7	309529	561.7
binkar10_1	596.4	86776	149.4	645.7	107739	163.3	2009.0	106799	161.7	139.4	100593	157.2	175.9	142580	139.1
cap6000	25.0	984	1.6	25.0	738	1.6	25.0	738	2.1	25.0	738	1.7	30.0	1801	1.5
cov1075	238889.6	2029811	4343.6	276493.1	2302521	5027.3	272692.6	2343053	4958.2	239021.6	2057875	4346.0	265179.6	2274377	4821.6
cschedd10	66451.7	369300	5305.9	84124.4	603007	7200.0	83471.9	595757	7200.0	82408.9	580199	7200.0	84082.4	1035664	7200.0
danoint	28360.6	1587831	6349.2	26431.6	916872	5941.5	32253.7	1563067	7200.0	32305.3	1741600	7200.0	20001.8	1102583	4476.0
dfn-gwin-UUM	1896.5	46931	118.7	1183.9	56297	135.2	1045.1	54593	133.4	1336.6	51457	139.2	1333.5	68085	93.1
ds	273675.0	6949	7200.0	273443.2	8668	7200.0	273433.4	8668	7200.0	273437.3	8554	7200.0	273445.3	7413	7200.0
eil33-2	1769.4	844	34.0	1748.3	1244	38.6	1818.4	1688	40.0	1183.5	1750	36.8	1351.2	340	30.3
eilB101	15408.3	8352	296.7	3224.8	5724	237.5	3264.2	5724	238.4	12732.3	6842	266.3	12660.3	6284	265.1
enlight13	1985.2	35737	21.2	1292.9	19169	14.1	1293.3	19169	14.1	8782.4	212656	97.6	13838.2	522384	158.9
glass4	61870.2	3541756	2966.7	240005.6	6451908	7200.0	70217.8	4150120	3363.6	240005.6	7997381	7200.0	240002.3	12763342	7200.0
gmu-35-40	65.0	21704510	7200.0	65.0	19643305	7200.0	65.0	20060056	7200.0	65.0	20562455	7200.0	60.0	32198403	7200.0
harp2	6822.5	197138	136.7	504.5	243174	161.1	320.6	198128	145.5	6887.6	209303	137.9	233.9	448438	160.7
iis-100-0-cov	19380.1	71291	522.7	26112.1	63349	685.3	30456.6	68587	737.3	25113.1	68335	578.4	24816.6	76155	523.3
iis-bupa-cov	98475.4	176675	2454.5	124920.5	146447	3222.1	65723.4	151437	2874.6	56189.1	153879	2319.6	58125.2	181441	2500.9
iis-pima-cov	14787.3	7531	263.6	13231.9	5821	239.4	14110.9	6339	254.5	13808.1	6059	249.3	13735.8	5869	248.4
macrophage	294853.6	2815744	7200.0	294850.8	1594719	7200.0	294850.8	2450737	7200.0	294854.2	2734477	7200.0	294851.0	2875395	7200.0
markshare1	720000.0	91697134	7200.0	720000.0	101065996	7200.0	720000.0	101801275	7200.0	720000.0	100882365	7200.0	720000.0	102171542	7200.0
markshare2	720000.0	83206634	7200.0	720000.0	88008981	7200.0	720000.0	87343340	7200.0	720000.0	87701931	7200.0	720000.0	88721939	7200.0
mas74	4034.7	2035460	395.3	3886.4	2112420	404.3	3425.2	2476385	355.0	3287.4	2567408	338.0	3306.0	2752472	342.5
mas76	99.8	221799	35.2	143.0	381797	55.0	318.2	395730	54.8	283.6	407343	46.7	244.8	547767	51.4
mcsched	60729.2	101259	1249.8	65736.7	108207	1334.0	56338.3	95019	1160.0	6662.9	60293	826.0	26190.3	46543	603.9
mik-250-1-100-1	1897.8	1072724	434.3	5327.1	3197358	1212.0	2179.0	1076085	438.0	1632.6	1505029	379.8	5995.2	6996847	1372.3
mine-166-5	1724.8	1240	31.1	1196.3	198	22.2	1333.9	328	24.5	1351.9	328	24.8	1250.2	2038	23.1
mine-90-10	2573.4	27903	96.5	8327.5	69247	208.1	10010.1	75221	240.3	8448.2	67032	209.7	20332.3	197329	428.7
misc07	973.8	31204	24.8	1183.4	33968	27.1	1245.4	33905	27.3	1027.9	30592	24.7	1201.8	47624	26.3
mkc	5503.8	1759678	7200.0	5504.2	2780944	7200.0	5503.8	1346673	7200.0	5504.0	1750343	7200.0	5504.2	2175182	7200.0
momentum1	85138.8	224442	7200.0	85144.8	395927	7200.0	85133.0	221818	7200.0	85158.2	264131	7200.0	85140.1	519796	7200.0
momentum2	244940.0	95882	4298.6	146827.4	34806	2723.9	202627.9	218703	7200.0	116352.8	140632	5152.9	226569.4	183471	4366.0
nsc98-ip	5196.4	19531	7200.0	5196.4	8595	7200.0	5186.4	88283	7200.0	5196.4	40128	7200.0	5191.4	58884	7200.0
nzzv11	10377.6	1873	332.0	9712.5	1857	319.5	9701.3	1857	318.2	9716.6	1857	318.3	10185.3	2014	324.5
n3div36	54503.3	305537	7200.0	52926.8	350961	7200.0	52929.1	346823	7200.0	54503.6	362117	7200.0	52928.9	513914	7200.0
n3seq24	4800.8	7497	7200.0	4840.6	10605	7200.0	4820.7	9713	7200.0	4825.7	9711	7200.0	4815.7	8601	7200.0
n4-3	40980.4	37963	742.5	47000.5	44277	850.6	47119.4	42763	852.8	47852.6	43169	866.2	9826.5	46719	557.0
neos-1109824	4752.2	16018	88.0	2215.8	5143	41.3	2644.7	7432	49.2	2493.4	6008	46.5	1674.8	17347	84.2
neos-1337307	2851.4	499306	7200.0	2846.1	537959	7200.0	2851.1	501053	7200.0	2856.0	499990	7200.0	2841.1	714364	7200.0
neos-1396125	165458.3	100396	4394.8	107606.3	57401	2582.8	105651.4	59324	2592.5	108259.0	57159	2594.8	37201.4	88307	836.4
neos-686190	2693.3	2190	44.4	2793.4	2400	46.0	2793.4	2400	46.0	2711.4	2263	44.6	2705.3	1865	44.5
neos-916792	124357.4	1936201	7200.0	115050.5	1637517	7200.0	115058.7	1695767	7200.0	117813.7	1944367	7200.0	117042.3	2065470	7200.0
neos18	1316.9	34892	75.9	2794.3	59418	116.6	1699.9	44814	72.4	4982.4	113871	136.1	20510.8	59329	79.2
net12	225620.4	2616	2776.5	289044.4	3642	3556.3	291354.0	3642	3584.8	188654.2	2273	2321.9	23104.4	3139	2842.7
netdiversion	59360.6	1262	7200.4	59214.5	2450	7200.6	59507.7	2440	7200.2	59409.7	2448	7200.2	59752.9	2430	7200.8
newdano	116175.7	1337377	2695.1	202838.0	2102759	4739.1	166152.1	2281511	3972.2	172475.7	2390353	3843.2	166449.7	3194194	3936.2
noswot	7283.7	759174	139.2	7744.2	541939	148.0	5264.0	349424	100.6	9245.9	937041	176.7	16508.7	1607445	315.5
ns1688347	638.9	216	11.5	666.1	227	11.9	765.2	547	13.7	587.2	132	10.6	690.6	1088	12.4
ns1766074	342820.0	741161	3428.2	339300.0	749137	3393.0	342910.0	718315	3429.1	270620.0	774224	2706.2	237360.0	848597	2373.6
ns1830653	16954.7	51131	396.1	30657.6	44597	427.5	30736.6	44597	428.6	31033.3	46837	432.7	9767.1	22244	219.9
nsrand-1px	11004.5	4470709	7200.0	11118.5	2278695	7200.0	15952.5	3796812	6527.1	11109.1	4328054	7200.0	10543.7	3978440	6502.4
opm2-z7-s2	52472.5	1511	873.2	45429.7	1275	756.0	45772.2	1421	761.7	51655.3	1325	859.6	46319.1	1283	770.8
pg5_34	3338.5	110314	797.7	35711.3	93606	830.2	33931.0	71290	801.6	3277.9	67830	755.8	1535.8	136710	713.1
pigeon-10	72000.0	1903058	7200.0	72000.0	2340172	7200.0	72000.0	2260479	7200.0	72000.0	3267789	7200.0	72000.0	3010254	7200.0
pk1	8503.0	441505	86.6	7850.0	245985	78.5	7230.0	275029	72.3	7341.0	344029	74.6	7282.0	393967	74.0
protfold	169384.3	6724	7200.0	169388.0	3654	7200.0	169388.0	3648	7200.0	169387.9	3643	7200.0	169388.3	6014	7200.0
pw-mycl4	161437.8	504422	2868.5	432078.7	3985751	7200.0	432084.8	3052407	7200.0	432083.7	2963901	7200.0	432083.2	3293759	7200.0
qiu	6514.4	9203	75.4	5908.2	8355	70.0	5790.7	8355	68.6	5799.0	8355	68.7	4489.9	9557	48.7
ran16x16	1579.1	254683	255.4	1497.2	251567	244.4	2064.1	248711	251.3	2212.5	261833	270.0	1222.1	284383	177.9
rd-rplusc-21	719565.9	292482	7200.0	719565.9	365129	7200.0	719565.9	176050	7200.0	719565.9	181324	7200.0	719565.9	563331	7200.0



Appendix C

Table C.1: The MIP performance results from Section 6.4.3 for every instance and random seed. The 5 columns **Timeouts**, **Diff**, and **Equal**, $[0,3600]$, and $[100,3600]$ indicate the group membership of an instance to the respective group in Table 6.4.

ProblemName	Seed	Diff	Equal	Timeouts [0,3600] [100,3600]	Time (sec.)		Prim. Int.	
					ALNS	ALNS off	ALNS	ALNS off
30n20b8	0	✓		✓ ✓	184	184	12 376	12 315
30n20b8	1	✓		✓ ✓	158	156	8 110	8 008
30n20b8	2	✓		✓ ✓	278	279	11 900	11 873
50v-10	0		✓		3 600	3 600	2 229	2 558
50v-10	1		✓		3 600	3 600	1 967	2 986
50v-10	2		✓		3 600	3 600	3 297	2 354
CMS750_4	0	✓		✓ ✓	1 263	1 253	3 955	3 917
CMS750_4	1	✓		✓ ✓	1 444	1 718	9 762	13 732
CMS750_4	2	✓		✓ ✓	1 981	3 600	10 021	13 797
academictimetables	0		✓		3 600	3 600	360 001	360 001
academictimetables	1		✓		3 600	3 600	360 000	360 001
academictimetables	2		✓		3 600	3 600	360 000	360 002
air05	0	✓		✓	31	31	361	405
air05	1	✓		✓	36	35	491	486
air05	2	✓		✓	32	35	434	467
app1-1	0	✓		✓	3	3	50	120
app1-1	1	✓		✓	4	5	203	453
app1-1	2		✓	✓	5	4	177	183
app1-2	0		✓	✓ ✓	211	210	9 015	8 976
app1-2	1		✓	✓ ✓	422	408	19 112	18 498
app1-2	2		✓	✓ ✓	304	294	13 680	13 191
assign1-5-8	0		✓	✓ ✓	3 235	3 231	38	36
assign1-5-8	1	✓		✓ ✓	3 600	3 592	36	35
assign1-5-8	2		✓	✓ ✓	2 499	2 494	43	42
atlanta-ip	0		✓		3 600	3 600	24 097	62 737
atlanta-ip	1		✓		3 600	3 600	17 899	14 976
atlanta-ip	2		✓		3 600	3 600	17 152	40 196

continued on next page ...

C. Appendix C

ProblemName	Seed	Diff	Equal	Timeouts [0,3600] [100,3600]	Time (sec.)		Prim. Int.	
					ALNS	ALNS off	ALNS	ALNS off
b1c1s1	0		✓		3 600	3 600	10 464	10 429
b1c1s1	1		✓		3 600	3 600	17 583	17 595
b1c1s1	2		✓		3 600	3 600	14 519	14 478
bab2	0		✓		3 600	3 600	282 447	360 001
bab2	1		✓		3 600	3 600	360 000	360 001
bab2	2		✓		3 600	3 600	330 971	330 191
bab6	0		✓		3 600	3 600	345 704	341 522
bab6	1		✓		3 600	3 600	203 681	360 000
bab6	2		✓		3 600	3 600	221 004	217 729
beasleyC3	0	✓		✓ ✓	87	115	856	955
beasleyC3	1	✓		✓	40	48	323	447
beasleyC3	2	✓		✓	48	26	400	423
binkar10_1	0	✓		✓	32	24	26	30
binkar10_1	1	✓		✓	28	24	24	29
binkar10_1	2	✓		✓	21	29	14	22
blp-ar98	0		✓		3 600	3 600	10 106	10 369
blp-ar98	1		✓		3 600	3 600	10 668	13 355
blp-ar98	2		✓		3 600	3 600	10 524	10 641
blp-ic98	0		✓		3 600	3 600	12 249	15 958
blp-ic98	1		✓		3 600	3 600	11 708	11 996
blp-ic98	2		✓		3 600	3 600	12 525	18 275
bnatt400	0		✓	✓ ✓	240	240	23 700	23 700
bnatt400	1		✓	✓ ✓	185	186	15 600	15 700
bnatt400	2		✓	✓ ✓	120	119	8 530	8 500
bnatt500	0		✓	✓ ✓	467	467	46 705	46 713
bnatt500	1		✓	✓ ✓	546	547	54 625	54 679
bnatt500	2		✓	✓ ✓	613	612	61 308	61 166
bppc4-08	0		✓		3 600	3 600	409	218
bppc4-08	1		✓		3 600	3 600	223	2 793
bppc4-08	2		✓		3 600	3 600	205	482
brazil3	0		✓		3 600	3 600	343 651	343 066
brazil3	1		✓		3 600	3 600	332 451	331 768
brazil3	2		✓		3 600	3 600	345 191	344 741
buildingenergy	0		✓		3 600	3 600	30 338	28 532
buildingenergy	1		✓		3 600	3 600	14 994	29 038
buildingenergy	2		✓		3 600	3 600	19 091	28 818
cbs-cta	0	✓		✓	7	5	710	470
cbs-cta	1	✓		✓	8	41	810	4 120
cbs-cta	2	✓		✓ ✓	4	1 069	350	106 900
chromaticindex1024-7	0		✓		3 600	3 600	53 800	53 800
chromaticindex1024-7	1		✓		3 600	3 600	47 000	47 100
chromaticindex1024-7	2		✓		3 600	3 600	52 900	53 900
chromaticindex512-7	0	✓		✓ ✓	1 523	1 527	9 570	9 630
chromaticindex512-7	1		✓		3 600	3 600	14 300	14 300
chromaticindex512-7	2		✓		3 600	3 600	11 400	11 300
cmflsp50-24-8-8	0		✓		3 600	3 600	360 001	360 000
cmflsp50-24-8-8	1		✓		3 600	3 600	355 069	352 641
cmflsp50-24-8-8	2		✓		3 600	3 600	117 239	106 145
co-100	0		✓		3 600	3 600	73 648	89 623
co-100	1		✓		3 600	3 600	85 986	97 439
co-100	2		✓		3 600	3 600	58 236	194 063
cod105	0	✓		✓ ✓	427	599	594	7 018
cod105	1		✓	✓ ✓	440	435	6 118	5 993
cod105	2		✓	✓ ✓	513	511	7 168	7 168

continued on next page . . .

ProblemName	Seed	Diff	Equal	Timeouts		Time (sec.)		Prim. Int.	
				[0,3600]	[100,3600]				
					ALNS	ALNS off	ALNS	ALNS off	
comp07-2idx	0		✓			3 600	3 600	344 860	344 679
comp07-2idx	1		✓			3 600	3 600	347 753	347 761
comp07-2idx	2		✓			3 600	3 600	348 437	348 421
comp21-2idx	0		✓			3 600	3 600	234 484	237 115
comp21-2idx	1		✓			3 600	3 600	229 093	245 447
comp21-2idx	2		✓			3 600	3 600	230 310	225 994
cost266-UUE	0	✓		✓	✓	2 337	2 335	699	688
cost266-UUE	1	✓		✓	✓	3 228	3 235	582	579
cost266-UUE	2	✓		✓	✓	3 457	3 448	407	397
cryptanalysisiskb128n5obj14	0		✓			3 600	3 600	360 001	360 001
cryptanalysisiskb128n5obj14	1		✓			3 600	3 600	360 001	360 001
cryptanalysisiskb128n5obj14	2		✓			3 600	3 600	360 000	360 000
cryptanalysisiskb128n5obj16	0	✓		✓	✓	3 059	3 068	305 900	306 800
cryptanalysisiskb128n5obj16	1		✓			3 600	3 600	360 000	360 001
cryptanalysisiskb128n5obj16	2		✓			3 600	3 600	360 002	360 001
csched007	0	✓		✓	✓	1 977	2 059	4 481	13 373
csched007	1	✓		✓	✓	3 033	3 092	12 770	12 788
csched007	2		✓	✓	✓	1 364	1 361	6 449	6 469
csched008	0		✓	✓	✓	532	532	10 222	10 221
csched008	1		✓	✓	✓	623	625	777	730
csched008	2		✓	✓	✓	432	432	948	895
cvs16r128-89	0		✓			3 600	3 600	8 922	9 208
cvs16r128-89	1		✓			3 600	3 600	8 454	7 483
cvs16r128-89	2		✓			3 600	3 600	8 272	9 137
dano3_3	0	✓		✓	✓	112	111	1 346	1 345
dano3_3	1	✓		✓	✓	121	120	1 586	1 585
dano3_3	2	✓		✓		88	87	1 639	1 638
dano3_5	0	✓		✓	✓	271	270	1 016	1 015
dano3_5	1	✓		✓	✓	341	339	1 745	1 743
dano3_5	2	✓		✓	✓	421	418	1 312	1 311
decomp2	0	✓		✓		2	2	171	161
decomp2	1	✓		✓		2	2	172	172
decomp2	2		✓	✓		2	2	172	171
drayage-100-23	0	✓		✓		8	9	482	703
drayage-100-23	1	✓		✓		8	9	609	677
drayage-100-23	2	✓		✓		23	8	546	636
drayage-25-23	0		✓			3 600	3 600	619	830
drayage-25-23	1		✓			3 600	3 600	620	871
drayage-25-23	2		✓			3 600	3 600	605	678
dws008-01	0		✓			3 600	3 600	60 440	59 372
dws008-01	1		✓			3 600	3 600	73 834	75 366
dws008-01	2		✓			3 600	3 600	67 431	66 464
eil33-2	0	✓		✓		78	78	377	502
eil33-2	1	✓		✓		73	77	376	479
eil33-2	2	✓		✓		82	65	435	330
eilA101-2	0		✓			3 600	3 600	63 008	82 544
eilA101-2	1		✓			3 600	3 600	47 060	63 704
eilA101-2	2		✓			3 600	3 600	51 600	101 940
enlight_hard	0	✓		✓		1	1	1	1
enlight_hard	1	✓		✓		1	1	1	1
enlight_hard	2	✓		✓		1	1	1	1
ex10	0	✓		✓	✓	119	119	11 900	11 900
ex10	1	✓		✓	✓	120	117	12 000	11 700
ex10	2	✓		✓	✓	122	122	12 200	12 200

continued on next page ...

C. Appendix C

ProblemName	Seed	Diff	Equal	Timeouts [0,3600] [100,3600]	Time (sec.)		Prim. Int.		
					ALNS	ALNS off	ALNS	ALNS off	
ex9	0		✓	✓	12	12	1 160	1 160	
ex9	1		✓	✓	11	10	1 070	1 050	
ex9	2		✓	✓	12	11	1 160	1 110	
exp-1-500-5-5	0	✓		✓	3	3	59	86	
exp-1-500-5-5	1	✓		✓	2	3	47	96	
exp-1-500-5-5	2	✓		✓	2	3	23	83	
fast0507	0		✓	✓	✓	232	215	838	603
fast0507	1		✓	✓	✓	138	133	502	482
fast0507	2		✓	✓	✓	133	122	580	468
fastxgemm-n2r6s0t2	0		✓	✓	✓	425	419	10 670	10 669
fastxgemm-n2r6s0t2	1		✓	✓	✓	469	467	3 985	3 937
fastxgemm-n2r6s0t2	2		✓	✓	✓	773	769	6 376	6 317
fhnw-binpack4-4	0			✓		3 600	3 600	360 000	360 000
fhnw-binpack4-4	1			✓		3 600	3 600	360 000	360 000
fhnw-binpack4-4	2			✓		3 600	3 600	360 000	360 000
fhnw-binpack4-48	0			✓		3 600	3 600	360 000	360 000
fhnw-binpack4-48	1			✓		3 600	3 600	360 000	360 000
fhnw-binpack4-48	2			✓		3 600	3 600	360 000	360 000
fiball	0	✓		✓	✓	295	87	1 362	985
fiball	1	✓		✓	✓	291	417	1 140	1 672
fiball	2	✓		✓	✓	84	469	922	1 893
gen-ip002	0	✓		✓	✓	1 509	1 510	26	7
gen-ip002	1	✓		✓	✓	1 542	1 465	73	8
gen-ip002	2	✓		✓	✓	1 434	1 796	30	35
gen-ip054	0	✓		✓	✓	1 452	1 180	90	14
gen-ip054	1		✓	✓	✓	894	886	22	22
gen-ip054	2	✓		✓	✓	1 694	1 735	142	167
germanrr	0			✓		3 600	3 600	44 124	44 334
germanrr	1			✓		3 600	3 600	46 993	47 096
germanrr	2			✓		3 600	3 600	52 644	52 441
gfd-schedulen180f7d50m30k18	0			✓		3 600	3 600	430 238	429 270
gfd-schedulen180f7d50m30k18	1			✓		3 600	3 600	363 588	361 798
gfd-schedulen180f7d50m30k18	2			✓		3 600	3 600	418 795	416 114
glass-sc	0			✓		3 600	3 600	217	215
glass-sc	1		✓	✓	✓	3 479	3 482	185	185
glass-sc	2		✓	✓	✓	3 050	3 057	432	431
glass4	0	✓		✓	✓	312	3 600	7 808	74 138
glass4	1	✓		✓	✓	946	1 079	23 762	23 800
glass4	2	✓		✓	✓	170	2 324	3 818	55 400
gmu-35-40	0			✓		3 600	3 600	91	102
gmu-35-40	1			✓		3 600	3 600	137	166
gmu-35-40	2			✓		3 600	3 600	90	168
gmu-35-50	0			✓		3 600	3 600	69	144
gmu-35-50	1			✓		3 600	3 600	171	243
gmu-35-50	2			✓		3 600	3 600	262	108
graph20-20-1rand	0		✓	✓		5	5	101	99
graph20-20-1rand	1	✓		✓		8	8	167	132
graph20-20-1rand	2	✓		✓		7	7	207	221
graphdraw-domain	0	✓		✓	✓	1 077	1 226	523	659
graphdraw-domain	1	✓		✓	✓	1 148	1 127	635	861
graphdraw-domain	2	✓		✓	✓	1 016	1 090	364	373
h80x6320d	0	✓		✓		78	74	567	3 123
h80x6320d	1	✓		✓		84	82	637	2 852
h80x6320d	2	✓		✓		90	66	2 253	3 424

continued on next page . . .

ProblemName	Seed	Diff	Equal	Timeouts [0,3600] [100,3600]	Time (sec.)		Prim. Int.	
					ALNS	ALNS off	ALNS	ALNS off
highschool1-aigio	0		✓		3 600	3 600	360 003	360 003
highschool1-aigio	1		✓		3 600	3 600	360 003	360 003
highschool1-aigio	2		✓		3 600	3 600	360 001	360 000
hypothyroid-k1	0	✓		✓	21	21	2 023	2 023
hypothyroid-k1	1	✓		✓	21	21	2 023	2 042
hypothyroid-k1	2	✓		✓	21	21	2 043	2 043
ic97_potential	0		✓		3 600	3 600	213	206
ic97_potential	1		✓		3 600	3 600	375	451
ic97_potential	2		✓		3 600	3 600	231	122
icir97_tension	0		✓		3 600	3 600	3 585	20 256
icir97_tension	1		✓		3 600	3 600	1 626	12 496
icir97_tension	2	✓		✓ ✓	3 578	3 526	3 947	3 851
irish-electricity	0		✓		3 600	3 600	36 773	37 166
irish-electricity	1		✓		3 600	3 600	40 079	39 436
irish-electricity	2		✓		3 600	3 600	342 333	342 081
irp	0	✓		✓	21	16	145	147
irp	1	✓		✓	22	14	142	152
irp	2	✓		✓	21	15	136	149
istanbul-no-cutoff	0	✓		✓ ✓	92	101	560	1 070
istanbul-no-cutoff	1	✓		✓ ✓	102	96	410	1 005
istanbul-no-cutoff	2	✓		✓ ✓	126	104	1 242	1 051
k1mushroom	0		✓	✓ ✓	2 479	2 489	231 023	232 017
k1mushroom	1		✓	✓ ✓	1 422	1 425	132 351	132 718
k1mushroom	2		✓	✓ ✓	1 963	1 970	170 374	170 961
lectsched-5-obj	0		✓		3 600	3 600	132 909	185 007
lectsched-5-obj	1		✓		3 600	3 600	125 342	198 561
lectsched-5-obj	2		✓		3 600	3 600	140 317	177 578
leo1	0		✓		3 600	3 600	3 302	3 593
leo1	1		✓		3 600	3 600	4 251	2 857
leo1	2		✓		3 600	3 600	2 334	3 220
leo2	0		✓		3 600	3 600	17 437	18 725
leo2	1		✓		3 600	3 600	11 129	22 224
leo2	2		✓		3 600	3 600	21 104	19 901
lotsize	0		✓		3 600	3 600	12 577	24 315
lotsize	1		✓		3 600	3 600	21 497	20 157
lotsize	2		✓		3 600	3 600	17 744	23 690
mad	0		✓		3 600	3 600	138 899	114 278
mad	1		✓		3 600	3 600	177 439	90 935
mad	2		✓		3 600	3 600	197 027	100 098
map10	0	✓		✓ ✓	797	937	7 548	16 072
map10	1	✓		✓ ✓	996	1 267	13 383	10 820
map10	2	✓		✓ ✓	845	943	6 208	11 976
map16715-04	0	✓		✓ ✓	2 015	2 010	1 046	31 063
map16715-04	1	✓		✓ ✓	1 939	2 079	15 963	40 514
map16715-04	2	✓		✓ ✓	1 934	2 024	2 062	22 993
markshare2	0		✓		3 600	3 600	345 334	346 971
markshare2	1		✓		3 600	3 600	343 029	342 789
markshare2	2		✓		3 600	3 600	345 164	339 976
markshare_4_0	0	✓		✓ ✓	281	279	11 054	10 980
markshare_4_0	1	✓		✓ ✓	182	183	8 034	8 101
markshare_4_0	2	✓		✓ ✓	290	290	8 662	8 658
mas74	0	✓		✓ ✓	2 189	1 907	379	887
mas74	1	✓		✓ ✓	2 250	2 061	341	818
mas74	2	✓		✓ ✓	2 138	2 382	452	689

continued on next page . . .

C. Appendix C

ProblemName	Seed	Diff	Equal	Timeouts [0,3600] [100,3600]	Time (sec.)		Prim. Int.		
					ALNS	ALNS off	ALNS	ALNS off	
mas76	0	✓		✓	✓	118	114	1	6
mas76	1	✓		✓	✓	104	119	1	6
mas76	2	✓		✓	✓	121	116	1	14
mc11	0	✓		✓	✓	76	278	205	373
mc11	1	✓		✓	✓	162	101	205	336
mc11	2	✓		✓	✓	133	137	245	379
mcsched	0		✓	✓	✓	288	288	165	164
mcsched	1		✓	✓	✓	285	284	134	134
mcsched	2		✓	✓	✓	233	233	153	152
mik-250-20-75-4	0	✓		✓		40	38	1	1
mik-250-20-75-4	1	✓		✓		33	37	10	1
mik-250-20-75-4	2	✓		✓		45	51	10	1
milov12-6-r2-40-1	0			✓		3 600	3 600	574	1 067
milov12-6-r2-40-1	1			✓		3 600	3 600	497	701
milov12-6-r2-40-1	2	✓		✓	✓	3 600	3 354	607	591
momentum1	0			✓		3 600	3 600	73 371	26 359
momentum1	1			✓		3 600	3 600	80 357	80 444
momentum1	2			✓		3 600	3 600	121 320	154 950
mushroom-best	0			✓		3 600	3 600	23 831	22 997
mushroom-best	1			✓		3 600	3 600	23 327	23 237
mushroom-best	2			✓		3 600	3 600	26 601	26 498
mzzv11	0	✓		✓	✓	391	318	5 476	8 323
mzzv11	1	✓		✓	✓	259	435	6 406	7 062
mzzv11	2	✓		✓	✓	605	531	11 206	20 388
mzzv42z	0	✓		✓	✓	343	353	3 888	10 144
mzzv42z	1	✓		✓	✓	182	337	3 314	7 176
mzzv42z	2	✓		✓	✓	315	281	4 818	6 786
n2seq36q	0	✓		✓	✓	362	173	1 907	2 964
n2seq36q	1	✓		✓	✓	115	696	774	8 570
n2seq36q	2	✓		✓	✓	775	324	2 787	4 851
n3div36	0			✓		3 600	3 600	3 351	3 608
n3div36	1			✓		3 600	3 600	2 900	4 419
n3div36	2			✓		3 600	3 600	3 582	4 243
n5-3	0	✓		✓		41	34	406	511
n5-3	1	✓		✓		32	41	275	581
n5-3	2	✓		✓		33	33	166	613
neos-1122047	0		✓	✓		10	10	1 040	1 030
neos-1122047	1		✓	✓		10	11	1 020	1 050
neos-1122047	2		✓	✓		10	11	1 040	1 060
neos-1171448	0	✓		✓		25	6	294	348
neos-1171448	1	✓		✓		8	8	281	365
neos-1171448	2	✓		✓		11	8	231	382
neos-1171737	0			✓		3 600	3 600	5 091	6 610
neos-1171737	1			✓		3 600	3 600	3 962	4 110
neos-1171737	2			✓		3 600	3 600	4 495	6 260
neos-1354092	0			✓		3 600	3 600	360 000	360 001
neos-1354092	1			✓		3 600	3 600	360 000	360 001
neos-1354092	2			✓		3 600	3 600	360 001	360 000
neos-1445765	0	✓		✓		31	30	835	842
neos-1445765	1	✓		✓		32	34	828	908
neos-1445765	2	✓		✓		35	33	945	968
neos-1456979	0			✓		3 600	3 600	16 767	16 784
neos-1456979	1			✓		3 600	3 600	22 851	22 844
neos-1456979	2			✓		3 600	3 600	15 574	15 634

continued on next page . . .

ProblemName	Seed	Diff	Equal	Timeouts		Time (sec.)		Prim. Int.	
				[0,3600]	[100,3600]				
					ALNS	ALNS off	ALNS	ALNS off	
neos-1582420	0	✓	✓			33	32	2 222	2 182
neos-1582420	1	✓	✓			18	17	1 535	1 505
neos-1582420	2	✓	✓			22	22	627	607
neos-2075418-temuka	0		✓			3 600	3 600	360 002	360 003
neos-2075418-temuka	1		✓			3 600	3 600	360 001	360 002
neos-2075418-temuka	2		✓			3 600	3 600	360 002	360 001
neos-2657525-crna	0		✓			3 600	3 600	73 706	73 705
neos-2657525-crna	1		✓			3 600	3 600	292 343	203 001
neos-2657525-crna	2		✓			3 600	3 600	4 624	245 796
neos-2746589-doon	0		✓			3 600	3 600	118 555	118 450
neos-2746589-doon	1		✓			3 600	3 600	64 053	64 356
neos-2746589-doon	2		✓			3 600	3 600	37 002	37 435
neos-2978193-inde	0		✓			3 600	3 600	244	190
neos-2978193-inde	1		✓			3 600	3 600	380	176
neos-2978193-inde	2	✓		✓	✓	3 600	2 870	330	233
neos-2987310-joes	0	✓	✓			18	18	1 743	1 731
neos-2987310-joes	1	✓	✓			16	17	1 645	1 655
neos-2987310-joes	2	✓	✓			16	16	1 606	1 576
neos-3004026-krka	0	✓	✓	✓		246	244	24 600	24 400
neos-3004026-krka	1	✓	✓	✓		630	632	63 000	63 200
neos-3004026-krka	2	✓	✓	✓		107	105	10 600	10 500
neos-3024952-loue	0	✓		✓	✓	1 155	1 193	79 022	79 032
neos-3024952-loue	1	✓		✓	✓	1 356	1 415	106 030	105 934
neos-3024952-loue	2		✓			3 600	3 600	360 000	360 000
neos-3046615-murg	0		✓			3 600	3 600	3 465	3 753
neos-3046615-murg	1		✓			3 600	3 600	1 026	1 970
neos-3046615-murg	2		✓			3 600	3 600	1 963	2 687
neos-3083819-nubu	0	✓		✓		16	16	32	34
neos-3083819-nubu	1	✓		✓		16	17	34	93
neos-3083819-nubu	2	✓		✓		12	21	32	85
neos-3216931-puriri	0		✓			3 600	3 600	226 999	227 062
neos-3216931-puriri	1		✓			3 600	3 600	237 103	237 003
neos-3216931-puriri	2		✓			3 600	3 600	216 690	216 502
neos-3381206-awhea	0	✓	✓			1	1	120	110
neos-3381206-awhea	1	✓	✓			1	1	110	110
neos-3381206-awhea	2	✓	✓			1	1	120	120
neos-3402294-bobin	0	✓		✓	✓	952	2 161	26 738	52 645
neos-3402294-bobin	1	✓		✓	✓	3 600	813	19 070	16 969
neos-3402294-bobin	2	✓		✓	✓	794	978	20 277	17 942
neos-3402454-bohle	0		✓			3 600	3 600	360 047	360 048
neos-3402454-bohle	1		✓			3 600	3 600	360 037	360 041
neos-3402454-bohle	2		✓			3 600	3 600	360 038	360 040
neos-3555904-turama	0		✓			3 600	3 600	12 300	197 878
neos-3555904-turama	1		✓			3 600	3 600	12 200	56 200
neos-3555904-turama	2		✓			3 600	3 600	160 954	159 569
neos-3627168-kasai	0	✓		✓	✓	1 287	3 600	67	74
neos-3627168-kasai	1		✓			3 600	3 600	63	70
neos-3627168-kasai	2	✓		✓	✓	1 106	3 600	61	160
neos-3656078-kumeu	0		✓			3 600	3 600	178 778	177 149
neos-3656078-kumeu	1		✓			3 600	3 600	263 727	264 089
neos-3656078-kumeu	2		✓			3 600	3 600	360 000	360 000
neos-3754480-nidda	0		✓			3 600	3 600	8 203	8 085
neos-3754480-nidda	1		✓			3 600	3 600	16 191	16 176
neos-3754480-nidda	2		✓			3 600	3 600	8 989	8 991

continued on next page . . .

C. Appendix C

ProblemName	Seed	Diff	Equal	Timeouts [0,3600] [100,3600]	Time (sec.)		Prim. Int.	
					ALNS	ALNS off	ALNS	ALNS off
neos-3988577-wolgan	0		✓		3 600	3 600	360 000	360 001
neos-3988577-wolgan	1		✓		3 600	3 600	360 000	360 001
neos-3988577-wolgan	2		✓		3 600	3 600	360 001	360 000
neos-4300652-rahue	0		✓		3 600	3 600	130 198	130 284
neos-4300652-rahue	1		✓		3 600	3 600	78 372	78 204
neos-4300652-rahue	2		✓		3 600	3 600	108 585	109 071
neos-4338804-snowy	0		✓		3 600	3 600	2 362	2 264
neos-4338804-snowy	1		✓		3 600	3 600	2 076	2 111
neos-4338804-snowy	2		✓		3 600	3 600	2 385	2 433
neos-4387871-tavua	0		✓		3 600	3 600	23 606	23 568
neos-4387871-tavua	1		✓		3 600	3 600	10 210	13 478
neos-4387871-tavua	2		✓		3 600	3 600	12 666	21 334
neos-4413714-turia	0	✓		✓ ✓	508	431	24 212	36 437
neos-4413714-turia	1	✓		✓ ✓	336	378	23 149	32 551
neos-4413714-turia	2	✓		✓ ✓	617	486	25 800	41 957
neos-4532248-waihi	0		✓		3 600	3 600	360 001	360 006
neos-4532248-waihi	1		✓		3 600	3 600	360 001	360 000
neos-4532248-waihi	2		✓		3 600	3 600	360 000	360 000
neos-4647030-tutaki	0		✓		3 600	3 600	7 888	7 330
neos-4647030-tutaki	1		✓		3 600	3 600	4 842	7 303
neos-4647030-tutaki	2		✓		3 600	3 600	4 038	7 107
neos-4722843-widden	0	✓		✓ ✓	761	577	9 642	8 786
neos-4722843-widden	1	✓		✓ ✓	537	397	8 089	7 561
neos-4722843-widden	2	✓		✓ ✓	526	505	9 390	8 907
neos-4738912-atrato	0	✓		✓ ✓	726	901	275	254
neos-4738912-atrato	1	✓		✓ ✓	1 887	962	391	304
neos-4738912-atrato	2	✓		✓ ✓	3 600	402	318	229
neos-4763324-toguru	0		✓		3 600	3 600	45 078	80 709
neos-4763324-toguru	1		✓		3 600	3 600	38 684	68 913
neos-4763324-toguru	2		✓		3 600	3 600	36 719	69 509
neos-4954672-berkel	0		✓		3 600	3 600	3 143	2 621
neos-4954672-berkel	1		✓		3 600	3 600	1 942	2 710
neos-4954672-berkel	2		✓		3 600	3 600	2 236	2 213
neos-5049753-cuanza	0		✓		3 600	3 600	184 236	183 120
neos-5049753-cuanza	1		✓		3 600	3 600	222 857	222 090
neos-5049753-cuanza	2		✓		3 600	3 600	360 004	360 004
neos-5052403-cygnet	0		✓		3 600	3 600	34 662	48 654
neos-5052403-cygnet	1		✓		3 600	3 600	54 106	53 641
neos-5052403-cygnet	2		✓		3 600	3 600	27 012	53 888
neos-5093327-huahum	0		✓		3 600	3 600	28 400	32 052
neos-5093327-huahum	1		✓		3 600	3 600	17 650	17 547
neos-5093327-huahum	2		✓		3 600	3 600	27 519	20 284
neos-5104907-jarama	0		✓		3 600	3 600	360 001	360 003
neos-5104907-jarama	1		✓		3 600	3 600	360 002	360 002
neos-5104907-jarama	2		✓		3 600	3 600	360 001	360 004
neos-5107597-kakapo	0		✓		3 600	3 600	10 639	14 132
neos-5107597-kakapo	1		✓		3 600	3 600	17 260	17 280
neos-5107597-kakapo	2		✓		3 600	3 600	18 832	33 216
neos-5114902-kasavu	0		✓		3 600	3 600	360 007	360 006
neos-5114902-kasavu	1		✓		3 600	3 600	360 006	360 009
neos-5114902-kasavu	2		✓		3 600	3 600	360 006	360 010
neos-5188808-nattai	0	✓		✓ ✓	1 409	1 407	6 332	6 203
neos-5188808-nattai	1	✓		✓ ✓	1 806	1 807	9 300	9 188
neos-5188808-nattai	2	✓		✓ ✓	1 515	1 496	4 460	4 283

continued on next page . . .

ProblemName	Seed	Diff	Equal	Timeouts		Time (sec.)		Prim. Int.	
				[0,3600]	[100,3600]	ALNS	ALNS off	ALNS	ALNS off
neos-5195221-niemur	0	✓		✓	✓	2 328	1 456	33 974	23 181
neos-5195221-niemur	1	✓		✓	✓	1 872	2 006	39 645	62 050
neos-5195221-niemur	2	✓		✓	✓	1 537	1 549	21 436	20 790
neos-631710	0		✓			3 600	3 600	51 421	51 611
neos-631710	1		✓			3 600	3 600	44 344	47 019
neos-631710	2		✓			3 600	3 600	41 357	41 261
neos-662469	0		✓			3 600	3 600	33 921	18 309
neos-662469	1		✓			3 600	3 600	23 802	22 445
neos-662469	2		✓			3 600	3 600	27 440	32 022
neos-787933	0	✓		✓		2	2	188	189
neos-787933	1	✓		✓		2	2	198	188
neos-787933	2	✓		✓		2	2	198	188
neos-827175	0	✓		✓		7	7	573	572
neos-827175	1	✓		✓		8	7	612	622
neos-827175	2	✓		✓		7	7	611	632
neos-848589	0	✓		✓	✓	1 749	1 975	49 799	45 991
neos-848589	1		✓			3 600	3 600	51 591	54 881
neos-848589	2	✓		✓	✓	995	3 600	57 005	51 537
neos-860300	0	✓		✓		26	19	612	1 068
neos-860300	1	✓		✓		23	19	822	1 015
neos-860300	2	✓		✓		17	17	722	1 016
neos-873061	0		✓			3 600	3 600	31 142	38 447
neos-873061	1		✓			3 600	3 600	33 703	33 709
neos-873061	2		✓			3 600	3 600	39 633	39 966
neos-911970	0		✓			3 600	3 600	150	172
neos-911970	1		✓			3 600	3 600	135	199
neos-911970	2		✓			3 600	3 600	254	145
neos-933966	0	✓		✓	✓	3 578	2 206	46 641	147 564
neos-933966	1		✓			3 600	3 600	204 345	317 495
neos-933966	2	✓		✓	✓	3 600	3 475	160 900	69 849
neos-950242	0		✓	✓	✓	157	149	13 233	12 433
neos-950242	1	✓		✓	✓	128	709	5 447	69 160
neos-950242	2	✓		✓	✓	1 854	3 600	176 714	360 000
neos-957323	0	✓		✓	✓	56	188	2 050	2 073
neos-957323	1	✓		✓	✓	128	103	2 008	2 116
neos-957323	2	✓		✓	✓	112	409	2 027	2 235
neos-960392	0	✓		✓	✓	804	608	11 693	31 529
neos-960392	1	✓		✓	✓	729	2 121	5 064	31 295
neos-960392	2	✓		✓	✓	737	282	6 220	24 593
neos17	0	✓		✓		9	10	98	91
neos17	1	✓		✓		7	8	103	92
neos17	2	✓		✓		9	8	100	106
neos5	0	✓		✓	✓	82	126	11	14
neos5	1	✓		✓	✓	145	142	12	18
neos5	2		✓	✓	✓	109	109	28	28
neos8	0	✓		✓		7	7	721	720
neos8	1	✓		✓		5	5	501	511
neos8	2	✓		✓		5	5	513	503
neos859080	0	✓		✓		3	3	252	250
neos859080	1	✓		✓		2	2	192	197
neos859080	2	✓		✓		5	5	459	453
net12	0	✓		✓	✓	367	364	5 090	5 040
net12	1	✓		✓	✓	688	685	9 232	9 198
net12	2	✓		✓	✓	653	654	3 937	3 917

continued on next page ...

C. Appendix C

ProblemName	Seed	Diff	Equal	Timeouts [0,3600] [100,3600]	Time (sec.)		Prim. Int.		
					ALNS	ALNS off	ALNS	ALNS off	
netdiversion	0	✓		✓	✓	2 351	639	61 164	48 944
netdiversion	1	✓		✓	✓	3 600	491	110 238	36 398
netdiversion	2	✓		✓	✓	3 600	977	151 795	56 997
nexp-150-20-8-5	0	✓		✓	✓	3 157	2 665	18 175	25 045
nexp-150-20-8-5	1	✓		✓	✓	3 600	1 936	35 826	25 389
nexp-150-20-8-5	2	✓		✓	✓	3 029	1 889	33 503	20 664
ns1116954	0			✓		3 600	3 600	360 001	360 001
ns1116954	1			✓		3 600	3 600	360 004	360 002
ns1116954	2			✓		3 600	3 600	360 001	360 001
ns1208400	0		✓	✓	✓	1 780	1 757	172 600	170 300
ns1208400	1		✓	✓	✓	3 147	3 132	312 400	311 000
ns1208400	2		✓	✓	✓	1 343	1 334	103 400	102 600
ns1644855	0		✓	✓	✓	2 758	1 615	21 222	13 352
ns1644855	1		✓	✓	✓	744	460	7 400	5 431
ns1644855	2			✓		3 600	3 600	26 991	27 001
ns1760995	0			✓		3 600	3 600	72 867	97 092
ns1760995	1			✓		3 600	3 600	96 779	96 858
ns1760995	2			✓		3 600	3 600	96 936	96 858
ns1830653	0	✓		✓	✓	107	196	2 262	3 601
ns1830653	1		✓	✓	✓	163	162	4 058	4 037
ns1830653	2		✓	✓	✓	148	148	3 771	3 783
ns1952667	0			✓		3 600	3 600	360 078	360 205
ns1952667	1		✓	✓	✓	1 975	1 975	197 500	197 500
ns1952667	2		✓	✓	✓	1 897	1 883	189 700	188 300
nu25-pr12	0	✓		✓		6	6	31	32
nu25-pr12	1	✓		✓		6	6	35	33
nu25-pr12	2	✓		✓		8	6	39	31
nursesched-medium-hint03	0			✓		3 600	3 600	348 178	354 602
nursesched-medium-hint03	1			✓		3 600	3 600	354 821	349 400
nursesched-medium-hint03	2			✓		3 600	3 600	354 751	333 731
nursesched-sprint02	0		✓	✓	✓	126	125	8 869	8 758
nursesched-sprint02	1	✓		✓		71	84	1 674	2 072
nursesched-sprint02	2	✓		✓	✓	142	140	8 531	8 824
nw04	0	✓		✓		29	38	2 136	2 114
nw04	1	✓		✓		52	34	2 525	2 269
nw04	2	✓		✓		31	32	2 133	2 095
opm2-z10-s4	0			✓		3 600	3 600	48 063	48 550
opm2-z10-s4	1			✓		3 600	3 600	52 219	52 696
opm2-z10-s4	2			✓		3 600	3 600	54 255	54 650
p200x1188c	0	✓		✓		1	3	1	11
p200x1188c	1	✓		✓		1	3	2	14
p200x1188c	2	✓		✓		1	3	1	14
peg-solitaire-a3	0			✓		3 600	3 600	360 001	360 000
peg-solitaire-a3	1			✓		3 600	3 600	360 000	360 001
peg-solitaire-a3	2			✓		3 600	3 600	360 000	360 001
pg	0	✓		✓		20	18	220	313
pg	1	✓		✓		23	14	94	527
pg	2	✓		✓		25	16	276	370
pg5_34	0	✓		✓	✓	1 199	3 145	97	110
pg5_34	1	✓		✓	✓	3 113	1 261	82	128
pg5_34	2	✓		✓	✓	1 595	1 507	123	300
physiciansched3-3	0			✓		3 600	3 600	360 001	360 000
physiciansched3-3	1			✓		3 600	3 600	360 000	360 000
physiciansched3-3	2			✓		3 600	3 600	360 001	360 001

continued on next page . . .

ProblemName	Seed	Diff	Equal	Timeouts		Time (sec.)		Prim. Int.	
				[0,3600]					
						ALNS	ALNS off	ALNS	ALNS off
physiciansched6-2	0	✓	✓			73	73	6 590	6 560
physiciansched6-2	1	✓	✓	✓		868	856	54 500	54 200
physiciansched6-2	2	✓	✓			66	66	6 570	6 490
piperout-08	0	✓	✓			43	42	2 216	2 214
piperout-08	1	✓	✓			22	22	1 744	1 733
piperout-08	2	✓	✓			33	33	2 397	2 395
piperout-27	0	✓	✓			34	39	2 569	2 658
piperout-27	1	✓	✓			42	42	2 134	2 120
piperout-27	2	✓	✓			13	13	1 020	1 020
pk1	0	✓		✓	✓	135	124	867	1 279
pk1	1	✓		✓	✓	128	130	1 117	994
pk1	2	✓		✓	✓	124	140	1 229	1 073
proteindesign121hz512p9	0		✓			3 600	3 600	193 597	194 133
proteindesign121hz512p9	1		✓			3 600	3 600	194 497	192 867
proteindesign121hz512p9	2		✓			3 600	3 600	193 894	193 747
proteindesign122trx11p8	0		✓			3 600	3 600	85 650	85 418
proteindesign122trx11p8	1		✓			3 600	3 600	79 342	79 305
proteindesign122trx11p8	2		✓			3 600	3 600	82 042	82 134
qap10	0	✓		✓		43	43	1 197	1 194
qap10	1	✓		✓		72	72	1 465	1 462
qap10	2	✓		✓		77	77	1 471	1 465
radiationm18-12-05	0		✓			3 600	3 600	6 944	6 733
radiationm18-12-05	1		✓			3 600	3 600	10 714	39 012
radiationm18-12-05	2		✓			3 600	3 600	24 711	33 615
radiationm40-10-02	0		✓			3 600	3 600	80 508	112 017
radiationm40-10-02	1		✓			3 600	3 600	33 611	159 230
radiationm40-10-02	2		✓			3 600	3 600	152 810	101 382
rail01	0		✓			3 600	3 600	360 002	360 002
rail01	1		✓			3 600	3 600	360 002	360 002
rail01	2		✓			3 600	3 600	360 002	360 002
rail02	0		✓			3 600	3 600	360 014	360 007
rail02	1		✓			3 600	3 600	360 008	360 007
rail02	2		✓			3 600	3 600	360 004	360 005
rail507	0	✓		✓	✓	402	399	811	816
rail507	1	✓		✓	✓	209	190	777	662
rail507	2	✓		✓	✓	188	180	719	691
ran14x18-disj-8	0	✓		✓	✓	1 323	1 829	1 173	1 219
ran14x18-disj-8	1	✓		✓	✓	2 084	2 247	1 183	1 252
ran14x18-disj-8	2	✓		✓	✓	1 444	1 851	1 083	950
rd-rplusc-21	0		✓			3 600	3 600	360 000	360 000
rd-rplusc-21	1		✓			3 600	3 600	289 593	288 787
rd-rplusc-21	2		✓			3 600	3 600	360 000	360 000
reblock115	0		✓			3 600	3 600	1 506	1 986
reblock115	1		✓			3 600	3 600	1 593	1 491
reblock115	2		✓			3 600	3 600	1 519	1 645
rmatr100-p10	0	✓		✓	✓	127	144	618	365
rmatr100-p10	1	✓		✓	✓	140	143	655	694
rmatr100-p10	2	✓		✓	✓	136	140	1 066	558
rmatr200-p5	0		✓			3 600	3 600	8 995	25 697
rmatr200-p5	1		✓			3 600	3 600	17 800	45 245
rmatr200-p5	2		✓			3 600	3 600	4 209	31 619
rocI-4-11	0	✓		✓		80	80	1 088	1 078
rocI-4-11	1	✓		✓		72	72	951	953
rocI-4-11	2	✓		✓		80	79	1 464	1 462

continued on next page . . .

C. Appendix C

ProblemName	Seed	Diff	Equal	Timeouts [0,3600] [100,3600]	Time (sec.)		Prim. Int.	
					ALNS	ALNS off	ALNS	ALNS off
rocII-5-11	0		✓		3 600	3 600	21 596	21 503
rocII-5-11	1		✓		3 600	3 600	26 745	26 633
rocII-5-11	2		✓		3 600	3 600	20 804	20 829
rococoB10-011000	0		✓		3 600	3 600	7 884	8 856
rococoB10-011000	1		✓		3 600	3 600	10 901	11 943
rococoB10-011000	2		✓		3 600	3 600	11 664	12 308
rococoC10-001000	0	✓		✓ ✓	790	657	829	1 261
rococoC10-001000	1	✓		✓ ✓	1 065	630	1 370	1 835
rococoC10-001000	2	✓		✓ ✓	617	824	1 139	2 109
roi2alpha3n4	0	✓		✓ ✓	1 581	1 510	11 770	13 527
roi2alpha3n4	1	✓		✓ ✓	1 803	1 879	11 972	14 082
roi2alpha3n4	2	✓		✓ ✓	1 700	1 707	11 817	14 576
roi5alpha10n8	0		✓		3 600	3 600	53 185	81 386
roi5alpha10n8	1		✓		3 600	3 600	55 929	55 632
roi5alpha10n8	2		✓		3 600	3 600	63 463	71 625
roll3000	0	✓		✓	34	24	219	346
roll3000	1		✓	✓	37	37	532	501
roll3000	2	✓		✓	30	25	195	554
s100	0		✓		3 600	3 600	306 397	360 010
s100	1		✓		3 600	3 600	257 514	360 011
s100	2		✓		3 600	3 600	237 054	360 009
s250r10	0	✓		✓ ✓	3 600	3 502	35 355	36 663
s250r10	1		✓		3 600	3 600	36 291	37 021
s250r10	2		✓		3 600	3 600	25 672	25 746
satellites2-40	0		✓		3 600	3 600	360 000	360 001
satellites2-40	1		✓		3 600	3 600	360 000	360 000
satellites2-40	2		✓		3 600	3 600	360 001	360 000
satellites2-60-fs	0		✓		3 600	3 600	356 975	356 964
satellites2-60-fs	1		✓		3 600	3 600	360 004	360 007
satellites2-60-fs	2		✓		3 600	3 600	281 368	284 237
savsched1	0		✓		3 600	3 600	324 769	324 684
savsched1	1		✓		3 600	3 600	324 722	324 719
savsched1	2		✓		3 600	3 600	324 788	324 760
sct2	0		✓		3 600	3 600	183	218
sct2	1		✓		3 600	3 600	296	146
sct2	2		✓		3 600	3 600	167	206
seymour	0		✓		3 600	3 600	476	412
seymour	1		✓		3 600	3 600	638	350
seymour	2		✓		3 600	3 600	1 051	1 095
seymour1	0	✓		✓	69	69	130	128
seymour1	1	✓		✓	43	43	63	62
seymour1	2	✓		✓	55	55	108	107
sing326	0		✓		3 600	3 600	10 373	15 270
sing326	1		✓		3 600	3 600	8 925	13 206
sing326	2		✓		3 600	3 600	7 878	20 601
sing44	0		✓		3 600	3 600	7 103	13 870
sing44	1		✓		3 600	3 600	8 637	11 092
sing44	2		✓		3 600	3 600	5 697	11 503
snp-02-004-104	0		✓		3 600	3 600	8 967	8 555
snp-02-004-104	1		✓		3 600	3 600	8 549	8 235
snp-02-004-104	2		✓		3 600	3 600	9 481	9 167
sorrell3	0		✓		3 600	3 600	44 220	43 739
sorrell3	1		✓		3 600	3 600	35 833	35 764
sorrell3	2		✓		3 600	3 600	39 151	39 214

continued on next page . . .

ProblemName	Seed	Diff	Equal	Timeouts [0,3600] [100,3600]	Time (sec.)		Prim. Int.		
					ALNS	ALNS off	ALNS	ALNS off	
sp150x300d	0	✓		✓	1	1	1	1	
sp150x300d	1	✓		✓	1	1	1	1	
sp150x300d	2		✓	✓	1	1	1	1	
sp97ar	0			✓	3 600	3 600	6 415	4 763	
sp97ar	1			✓	3 600	3 600	5 537	7 971	
sp97ar	2			✓	3 600	3 600	5 404	5 597	
sp98ar	0			✓	3 600	3 600	3 071	3 640	
sp98ar	1			✓	3 600	3 600	2 353	3 896	
sp98ar	2			✓	3 600	3 600	2 086	4 596	
splice1k1	0			✓	3 600	3 600	137 848	141 287	
splice1k1	1			✓	3 600	3 600	130 192	132 520	
splice1k1	2			✓	3 600	3 600	157 680	157 122	
square41	0			✓	3 600	3 600	144 161	143 375	
square41	1			✓	3 600	3 600	138 282	138 303	
square41	2			✓	3 600	3 600	106 877	105 645	
square47	0			✓	3 600	3 600	173 767	173 695	
square47	1			✓	3 600	3 600	174 289	174 161	
square47	2			✓	3 600	3 600	174 051	174 015	
supportcase10	0			✓	3 600	3 600	220 212	220 208	
supportcase10	1			✓	3 600	3 600	227 562	227 561	
supportcase10	2			✓	3 600	3 600	192 246	192 251	
supportcase12	0			✓	3 600	3 600	9 092	17 555	
supportcase12	1			✓	3 600	3 600	7 629	19 273	
supportcase12	2			✓	3 600	3 600	7 337	18 918	
supportcase18	0			✓	3 600	3 600	15 019	15 027	
supportcase18	1			✓	3 600	3 600	8 381	8 376	
supportcase18	2			✓	3 600	3 600	15 317	19 121	
supportcase19	0			✓	3 600	3 600	361 583	362 243	
supportcase19	1			✓	3 600	3 600	361 208	360 928	
supportcase19	2			✓	3 600	3 600	360 021	362 131	
supportcase22	0			✓	3 600	3 600	360 001	360 002	
supportcase22	1			✓	3 600	3 600	360 000	360 002	
supportcase22	2			✓	3 600	3 600	360 002	360 001	
supportcase26	0			✓	3 600	3 600	1 727	4 471	
supportcase26	1			✓	3 600	3 600	1 187	1 403	
supportcase26	2			✓	3 600	3 600	1 644	2 758	
supportcase33	0		✓	✓	✓	764	761	6 902	6 874
supportcase33	1		✓	✓	✓	292	291	6 711	6 673
supportcase33	2	✓		✓	✓	315	161	6 532	6 734
supportcase40	0		✓	✓	✓	1 397	1 399	2 250	2 238
supportcase40	1		✓	✓	✓	1 565	1 560	2 436	2 411
supportcase40	2		✓	✓	✓	1 368	1 362	1 894	1 873
supportcase42	0			✓	3 600	3 600	5 201	5 810	
supportcase42	1			✓	3 600	3 600	6 089	8 638	
supportcase42	2			✓	3 600	3 600	4 734	6 058	
supportcase6	0			✓	3 600	3 600	10 156	13 078	
supportcase6	1			✓	3 600	3 600	9 492	10 501	
supportcase6	2			✓	3 600	3 600	11 380	9 510	
supportcase7	0	✓		✓	✓	189	143	8 596	8 232
supportcase7	1	✓		✓	✓	234	132	12 971	8 551
supportcase7	2	✓		✓	✓	151	211	8 910	10 829
swath1	0		✓	✓	12	11	342	322	
swath1	1	✓		✓	15	12	189	175	
swath1	2	✓		✓	12	13	212	224	

continued on next page . . .

C. Appendix C

ProblemName	Seed	Diff	Equal	Timeouts [0,3600] [100,3600]	Time (sec.)		Prim. Int.		
					ALNS	ALNS off	ALNS	ALNS off	
swath3	0		✓	✓	✓	347	345	398	378
swath3	1		✓	✓	✓	191	189	374	354
swath3	2	✓		✓	✓	326	198	378	304
tbf-network	0		✓	✓	✓	829	782	43 520	39 665
tbf-network	1		✓	✓	✓	973	918	61 675	57 799
tbf-network	2		✓	✓	✓	3 142	3 123	71 230	69 753
thor50dday	0			✓		3 600	3 600	84 462	97 852
thor50dday	1			✓		3 600	3 600	83 347	114 108
thor50dday	2			✓		3 600	3 600	71 714	71 476
timtab1	0	✓		✓		77	68	546	560
timtab1	1	✓		✓		59	38	409	375
timtab1	2	✓		✓		59	84	592	588
tr12-30	0	✓		✓	✓	703	717	55	46
tr12-30	1	✓		✓	✓	718	951	56	65
tr12-30	2	✓		✓	✓	711	628	60	29
traininstance2	0			✓		3 600	3 600	41 611	42 163
traininstance2	1			✓		3 600	3 600	24 398	24 371
traininstance2	2			✓		3 600	3 600	23 514	27 709
traininstance6	0			✓		3 600	3 600	6 899	3 201
traininstance6	1			✓		3 600	3 600	559	557
traininstance6	2			✓		3 600	3 600	3 277	3 255
trento1	0			✓		3 600	3 600	48 987	32 784
trento1	1			✓		3 600	3 600	45 809	37 187
trento1	2			✓		3 600	3 600	110 905	57 946
triptim1	0		✓	✓	✓	535	536	18 206	18 306
triptim1	1		✓	✓	✓	1 844	1 829	13 737	13 537
triptim1	2		✓	✓	✓	812	812	11 106	11 206
uccase12	0			✓		3 600	3 600	4 098	4 324
uccase12	1			✓		3 600	3 600	3 673	3 924
uccase12	2			✓		3 600	3 600	4 247	4 526
uccase9	0			✓		3 600	3 600	154 540	123 159
uccase9	1			✓		3 600	3 600	109 948	169 550
uccase9	2			✓		3 600	3 600	174 864	293 936
uct-subprob	0	✓		✓	✓	2 334	2 483	1 761	2 093
uct-subprob	1	✓		✓	✓	2 054	2 471	1 442	1 716
uct-subprob	2	✓		✓	✓	1 539	2 958	1 029	1 837
unitcal_7	0		✓	✓	✓	240	239	14 031	13 831
unitcal_7	1		✓	✓	✓	261	260	10 500	10 400
unitcal_7	2		✓	✓	✓	212	211	12 023	11 923
var-smallemery-m6j6	0			✓		3 600	3 600	1 397	5 624
var-smallemery-m6j6	1			✓		3 600	3 600	1 555	1 524
var-smallemery-m6j6	2			✓		3 600	3 600	2 747	3 747
wachplan	0		✓	✓	✓	2 762	2 760	180	180
wachplan	1		✓	✓	✓	2 311	2 312	180	180
wachplan	2		✓	✓	✓	2 188	2 188	200	200

D

Appendix D

Table D.1: Computational results over three random seeds on MMMC test set comparing SCIP in its default configuration (**default**), without diving heuristics at all (**nodiving**), and extended by the new adaptive diving heuristics (**adaptivediving**). The table shows the absolute and relative numbers of nodes (**nodes**, **nodes_Q**) and solving time (**time**, **time_Q**). In addition, the average number of feasible solutions found by **adaptivediving** is shown (**nsols**). The right-most column (**impr.sols**) indicates whether at least one improving solution was found by **adaptivediving** with at least one seed (✓). Relative changes by at least 5% are highlighted in bold (**improvement**) or italic and red (*deterioration*).

Instance	default		nodiving				adaptivediving					
	nodes	time	nodes	nodes _Q	time	time _Q	nodes	nodes _Q	time	time _Q	nsols	impr.sols
10teams	212.7	14.0	471.8	<i>2.22</i>	15.9	<i>1.14</i>	58.1	0.27	9.2	0.66	–	
22433	3.0	2.0	3.7	<i>1.23</i>	2.2	<i>1.08</i>	3.0	1.01	1.6	0.78	1.0	✓
23588	542.7	4.2	529.6	0.98	4.2	0.99	489.2	0.90	4.2	0.99	–	
30n20b8	43.7	153.6	131.9	<i>3.02</i>	246.2	<i>1.60</i>	33.3	0.76	161.5	<i>1.05</i>	1.0	✓
Test3	1.9	4.2	1.9	1.00	3.9	0.93	1.3	0.69	3.5	0.84	0.3	✓
alc1s1	90230.8	3600.0	88560.4	0.98	3600.0	1.00	88181.4	0.98	3600.0	1.00	7.3	✓
acc-tight5	569.2	99.4	512.9	0.90	72.7	0.73	817.3	<i>1.44</i>	123.7	<i>1.24</i>	–	
aflow30a	533.1	16.1	846.5	<i>1.59</i>	19.5	<i>1.21</i>	678.2	<i>1.27</i>	16.8	1.04	0.7	✓
aflow40b	20770.6	561.1	13399.8	0.65	521.8	0.93	16235.3	0.78	577.8	1.03	–	
air03	2.0	2.1	2.0	1.00	2.0	0.96	2.0	1.00	1.9	0.91	1.3	✓
air04	77.5	52.3	91.1	<i>1.18</i>	43.3	0.83	48.6	0.63	45.6	0.87	0.7	✓
air05	462.1	29.8	363.8	0.79	27.2	0.91	308.2	0.67	29.2	0.98	–	
aligninq	623.6	25.9	1509.8	<i>2.42</i>	25.9	1.00	1658.0	<i>2.66</i>	24.3	0.94	–	
app1-2	19.8	771.0	36.6	<i>1.85</i>	753.4	0.98	23.6	<i>1.19</i>	741.1	0.96	–	
arki001	632928.1	3600.0	569989.2	0.90	3600.0	1.00	538136.2	0.85	3600.0	1.00	4.0	✓
ash608gpia-3col	3.6	28.1	7.8	<i>2.15</i>	26.8	0.95	3.0	0.82	27.9	0.99	–	
atlanta-ip	6341.3	3600.0	5933.9	0.94	3600.0	1.00	4965.6	0.78	3600.0	1.00	–	
bab5	40843.0	3600.0	28727.2	0.70	3600.0	1.00	41647.2	1.02	3600.0	1.00	1.7	✓
bc	15073.0	989.1	16115.8	<i>1.07</i>	1034.9	1.05	17845.8	<i>1.18</i>	1086.5	<i>1.10</i>	0.3	✓
bc1	2074.6	186.1	4620.7	<i>2.23</i>	209.4	<i>1.12</i>	2391.8	<i>1.15</i>	163.5	0.88	0.3	✓
beasleyC3	27.4	28.6	119.6	<i>4.37</i>	34.9	<i>1.22</i>	123.5	<i>4.51</i>	41.9	<i>1.46</i>	14.7	✓
bell3a	2470.4	0.9	1651.6	0.67	0.8	0.81	1307.7	0.53	0.6	0.68	26.0	✓
bell5	367.0	0.5	465.5	<i>1.27</i>	0.5	1.00	605.8	<i>1.65</i>	0.5	<i>1.05</i>	43.3	✓
biella1	4831.4	1027.1	2463.6	0.51	686.8	0.67	3007.6	0.62	744.5	0.72	1.0	✓
bienst1	13512.7	115.3	11775.7	0.87	100.8	0.87	14638.0	<i>1.08</i>	119.5	1.04	4.7	✓
bienst2	70278.9	569.3	53181.0	0.76	434.3	0.76	55481.9	0.79	451.8	0.79	2.0	✓
binkar10_1	2453.0	31.5	2530.8	1.03	28.2	0.90	3255.5	<i>1.33</i>	40.1	<i>1.27</i>	1.3	✓
blend2	1355.4	1.4	742.1	0.55	0.7	0.52	1166.8	0.86	1.2	0.87	–	
bley_xl1	9.6	182.8	8.5	0.89	181.2	0.99	1.7	0.17	168.7	0.92	1.0	✓
bnatt350	9721.7	963.2	4776.2	0.49	467.1	0.48	4615.4	0.47	542.8	0.56	–	
cap6000	2021.9	3.3	2195.9	<i>1.09</i>	3.0	0.89	1937.3	0.96	3.2	0.96	50.7	
core2536-691	164.5	196.3	149.9	0.91	137.3	0.70	83.2	0.51	130.1	0.66	0.3	✓
cov1075	37461.2	108.7	26781.9	0.71	69.8	0.64	21022.7	0.56	71.1	0.65	8.7	✓
csched010	227915.0	3600.0	207252.2	0.91	3600.0	1.00	204593.6	0.90	3505.2	0.97	0.3	✓
d10200	484293.5	3600.0	612265.4	<i>1.26</i>	3600.0	1.00	401617.9	0.83	3600.0	1.00	0.7	✓
d20200	61761.8	3600.0	164217.2	<i>2.66</i>	3600.0	1.00	73368.6	<i>1.19</i>	3600.0	1.00	1.7	✓

cont. on next page

Table D.1 cont.

Instance	default		nodiving				adaptivediving					
	nodes	time	nodes	nodesQ	time	timeQ	nodes	nodesQ	time	timeQ	nsols	impr.sols
dano3_3	12.4	112.9	14.8	1.19	112.7	1.00	6.6	0.54	116.8	1.03	0.7	✓
dano3_4	10.4	136.9	10.2	0.97	134.8	0.98	9.0	0.86	150.6	1.10	1.0	✓
dano3_5	200.7	330.6	192.1	0.96	284.6	0.86	145.8	0.73	275.2	0.83	0.7	✓
dano3mip	460.6	3600.0	1042.9	2.26	3600.0	1.00	495.6	1.08	3600.0	1.00	–	
danoint	1181425.3	3558.1	1125317.4	0.95	3122.3	0.88	1113895.2	0.94	3231.1	0.91	1.7	✓
dcmulti	86.0	2.4	55.9	0.65	2.3	0.97	99.5	1.16	2.0	0.85	1.3	✓
dfn-gwin-UUM	22916.1	99.2	24228.9	1.06	99.8	1.01	23843.1	1.04	102.0	1.03	140.7	✓
disctom	1.0	5.9	1.0	1.00	5.9	1.00	1.0	1.00	5.9	1.00	–	
ds	594.5	3600.0	1019.6	1.72	3600.0	1.00	454.2	0.76	3600.0	1.00	58.3	✓
dsbmip	9.6	1.6	16.8	1.76	1.6	1.02	4.8	0.51	0.9	0.56	1.3	✓
egout	1.0	0.5	1.0	1.00	0.5	1.00	1.0	1.00	0.5	1.00	1.0	✓
eil33-2	631.9	67.2	679.0	1.07	73.8	1.10	729.4	1.15	73.8	1.10	1.3	✓
eilB101	10544.5	239.0	9813.6	0.93	218.8	0.92	17189.5	1.63	336.4	1.41	6.3	✓
enigma	935.3	0.5	431.0	0.46	0.5	0.96	326.1	0.35	0.5	0.96	–	
enlight13	1.0	0.5	1.0	1.00	0.5	1.00	1.0	1.00	0.5	1.00	–	
enlight14	1.0	0.5	1.0	1.00	0.5	1.00	1.0	1.00	0.5	1.00	–	
ex9	1.0	27.5	1.0	1.00	28.1	1.02	1.0	1.00	28.2	1.02	–	
fast0507	1025.5	235.9	803.7	0.78	175.9	0.75	745.5	0.73	189.5	0.80	7.7	✓
fiball	5028.8	2113.6	9336.7	1.86	3491.6	1.65	4384.4	0.87	1847.5	0.87	1.0	✓
fiber	4.0	1.7	4.6	1.16	1.9	1.12	4.0	1.00	1.7	0.98	3.3	✓
fixnet6	3.3	4.7	3.0	0.90	4.6	0.97	3.3	1.00	5.1	1.08	10.0	✓
flugpl	1.0	0.5	1.0	1.00	0.5	1.00	1.0	1.00	0.5	1.00	1.0	✓
gen	1.0	0.5	1.0	1.00	0.5	1.00	1.0	1.00	0.5	1.00	–	
germanrr	1325.5	3600.0	2893.7	2.18	3600.0	1.00	906.8	0.68	3600.0	1.00	3.3	✓
gesa2	1.3	0.5	1.3	1.00	0.5	1.00	1.7	1.25	0.5	1.00	1.0	✓
gesa2-o	2.6	0.9	2.6	1.00	0.8	0.94	2.0	0.76	0.7	0.78	1.0	✓
gesa3	8.8	3.4	6.9	0.79	3.2	0.95	5.9	0.68	3.9	1.15	1.3	✓
gesa3_o	6.6	2.8	7.3	1.10	4.0	1.44	6.0	0.90	3.3	1.18	0.3	✓
glass4	2077361.5	2920.8	2458448.3	1.18	3600.0	1.23	1030344.4	0.50	1401.2	0.48	6.0	✓
gmu-35-40	1461071.3	3600.0	3247969.2	2.22	3600.0	1.00	1687277.3	1.16	3600.0	1.00	2.0	✓
gt2	1.0	0.5	1.0	1.00	0.5	1.00	1.0	1.00	0.5	1.00	–	
haprp	1.0	0.5	1.0	1.00	0.5	1.00	1.0	1.00	0.5	1.00	–	
harp2	2203367.5	1079.0	2492591.7	1.13	1142.8	1.06	1644638.6	0.75	767.6	0.71	22.7	✓
iis-100-0-cov	86589.8	532.6	88130.1	1.02	511.1	0.96	86035.9	0.99	521.9	0.98	9.0	✓
iis-bupa-cov	174929.6	2006.0	158889.8	0.91	1725.9	0.86	160776.5	0.92	1924.5	0.96	10.3	✓
iis-pima-cov	7264.1	311.6	6351.2	0.87	270.2	0.87	6105.1	0.84	272.3	0.87	8.3	✓
khb05250	3.2	0.5	2.0	0.62	0.5	0.99	3.2	1.00	0.5	0.99	2.3	✓
l152lav	32.4	2.2	35.4	1.09	2.5	1.12	56.2	1.74	3.3	1.49	0.7	✓
lectsched-4-obj	1473.1	33.3	243.9	0.17	19.5	0.59	38.6	0.03	12.0	0.36	0.3	✓
leo1	45925.9	3600.0	56743.9	1.24	3600.0	1.00	57568.8	1.25	3600.0	1.00	0.3	✓
leo2	61837.4	3600.0	63021.1	1.02	3600.0	1.00	62135.3	1.00	3600.0	1.00	–	
liu	828433.0	3600.0	931693.0	1.12	3600.0	1.00	665812.4	0.80	3600.0	1.00	31.3	✓
lrn	1134.1	3600.0	1402.7	1.24	3600.0	1.00	1535.3	1.35	3600.0	1.00	3.0	✓
lseu	58.1	0.5	95.4	1.64	0.5	1.05	82.6	1.42	0.6	1.13	0.3	✓

cont. on next page

Table D.1 cont.												
Instance	default		nodiving				adaptivediving					
	nodes	time	nodes	nodes _Q	time	time _Q	nodes	nodes _Q	time	time _Q	nsols	impr.sols
m100n500k4r1	2235185.7	3600.0	2537595.1	1.14	3600.0	1.00	2329569.4	1.04	3600.0	1.00	8.0	✓
macrophage	9683.9	243.4	6345.3	0.66	146.3	0.60	7636.8	0.79	165.3	0.68	1.0	✓
manna81	1.0	0.5	1.0	1.00	0.5	1.00	1.0	1.00	0.5	1.00	–	
map18	266.1	261.6	271.4	1.02	222.6	0.85	256.2	0.96	332.9	1.27	0.7	✓
map20	291.9	254.2	265.4	0.91	186.2	0.73	236.9	0.81	250.4	0.98	0.3	✓
markshare1	19663456.6	3600.0	25464037.1	1.29	3600.0	1.00	20584380.8	1.05	3600.0	1.00	16.3	✓
markshare2	4423023.0	3600.0	10510623.5	2.38	3600.0	1.00	4377939.3	0.99	3600.0	1.00	13.0	✓
mas74	6338096.1	1695.5	4807477.7	0.76	1257.8	0.74	6424904.2	1.01	1644.7	0.97	2.0	✓
mas76	215133.6	89.3	203671.4	0.95	71.9	0.81	265165.3	1.23	97.8	1.09	0.7	✓
mcsched	9526.9	223.0	6921.6	0.73	171.3	0.77	8695.2	0.91	204.5	0.92	1.0	✓
mik-250-1-100-1	17563.0	54.7	30966.9	1.76	82.2	1.50	23661.6	1.35	67.8	1.24	20.7	
mine-166-5	1653.8	60.1	1005.9	0.61	48.6	0.81	570.9	0.34	71.3	1.19	0.7	✓
mine-90-10	25782.5	173.9	21434.8	0.83	188.9	1.09	31446.8	1.22	201.6	1.16	2.3	✓
misc03	31.0	0.7	17.8	0.57	0.7	0.88	25.2	0.81	0.7	0.97	0.3	✓
misc06	3.3	0.6	3.0	0.90	0.5	0.85	3.0	0.90	0.6	0.97	6.7	✓
misc07	8546.6	13.1	5314.6	0.62	9.7	0.73	4884.8	0.57	10.1	0.77	0.3	✓
mitre	1.0	10.2	1.0	1.00	10.2	1.00	1.0	1.00	10.2	1.00	–	
mkc	387333.6	3600.0	384365.8	0.99	3600.0	1.00	435964.5	1.13	3600.0	1.00	0.3	✓
mkc1	28036.1	203.7	320870.5	11.45	1125.8	5.53	455465.8	16.25	1398.1	6.86	1.0	✓
mod008	2.0	0.5	2.0	1.00	0.5	1.00	2.0	1.00	0.5	1.00	9.7	
mod010	2.0	0.5	2.0	1.00	0.5	1.00	2.0	1.00	0.5	1.00	1.0	✓
mod011	643.0	373.1	777.3	1.21	402.4	1.08	615.4	0.96	360.7	0.97	1.3	✓
modglob	2.0	0.5	2.0	1.00	0.5	1.00	2.0	1.00	0.5	1.00	4.3	✓
momentum1	7017.3	3600.0	11426.1	1.63	3600.0	1.00	12250.4	1.75	3600.0	1.00	–	
momentum2	40632.1	3600.0	36985.8	0.91	3600.0	1.00	33270.2	0.82	3600.0	1.00	1.3	✓
momentum3	134.8	3600.0	141.6	1.05	3600.0	1.00	77.3	0.57	3600.0	1.00	–	
msc98-ip	3522.6	3600.0	3635.1	1.03	3600.0	1.00	1850.0	0.53	3600.0	1.00	0.3	✓
mspp16	3.0	428.1	4.3	1.43	402.8	0.94	4.9	1.64	452.7	1.06	–	
mzzv11	1704.6	331.3	1538.3	0.90	305.4	0.92	958.2	0.56	264.1	0.80	1.0	✓
mzzv42z	177.3	161.4	276.4	1.56	167.8	1.04	66.8	0.38	181.0	1.12	1.0	✓
n3div36	88449.9	3600.0	91767.9	1.04	3600.0	1.00	89341.7	1.01	3600.0	1.00	–	
n3seq24	151.2	3600.0	459.2	3.04	3600.0	1.00	74.9	0.49	3600.0	1.00	–	
n4-3	2835.7	178.6	2449.8	0.86	159.7	0.89	3467.0	1.22	220.1	1.23	73.0	✓
nag	31388.4	3600.0	34045.5	1.08	3600.0	1.00	29353.0	0.94	3600.0	1.00	3.7	✓
neos-1053234	418752.8	2172.9	922222.1	2.20	2872.2	1.32	400453.6	0.96	2070.5	0.95	–	
neos-1053591	2377.9	3.8	2540.2	1.07	3.7	0.98	2128.5	0.90	3.3	0.87	1.0	✓
neos-1056905	4956678.8	3235.8	8191129.7	1.65	3600.0	1.11	5087261.1	1.03	2688.0	0.83	3.7	✓
neos-1058477	1.0	0.5	1.0	1.00	0.5	1.00	1.0	1.00	0.5	1.00	–	
neos-1061020	869.7	199.6	1145.4	1.32	221.1	1.11	866.2	1.00	205.0	1.03	–	
neos-1062641	1.0	0.5	1.0	1.00	0.5	1.00	1.0	1.00	0.5	1.00	–	
neos-1067731	75928.2	3600.0	104378.6	1.38	3600.0	1.00	82587.7	1.09	3600.0	1.00	1.3	✓
neos-1096528	3942.2	1766.6	2716.6	0.69	1684.0	0.95	5263.6	1.33	2148.4	1.22	0.3	✓
neos-1109824	107.3	13.5	163.2	1.52	14.1	1.05	435.1	4.05	22.5	1.67	0.3	✓
neos-1112782	907703.2	3600.0	867034.3	0.95	3600.0	1.00	947311.6	1.04	3600.0	1.00	–	
cont. on next page												

Table D.1 cont.

Instance	default		nodiving				adaptivediving					
	nodes	time	nodes	nodes _Q	time	time _Q	nodes	nodes _Q	time	time _Q	nsols	impr.sols
neos-1112787	401700.9	3600.0	423880.7	1.05	3600.0	1.00	409889.8	1.02	3600.0	1.00	–	
neos-1120495	16.2	11.7	10.3	0.64	4.9	0.42	7.4	0.46	5.9	0.50	1.0	✓
neos-1121679	19581278.4	3600.0	25447433.5	1.30	3600.0	1.00	20612095.3	1.05	3600.0	1.00	16.3	✓
neos-1122047	1.0	6.0	1.0	1.00	6.0	1.00	1.0	1.00	6.0	1.01	–	
neos-1126860	4590.7	561.7	4983.3	1.09	479.4	0.85	4643.7	1.01	561.5	1.00	1.0	✓
neos-1140050	549.7	3600.0	407.3	0.74	3600.0	1.00	600.9	1.09	3600.0	1.00	–	
neos-1151496	29.8	10.3	94.0	3.16	19.1	1.85	59.5	2.00	16.7	1.62	–	
neos-1171448	15.1	19.6	79.1	5.25	52.2	2.67	1.0	0.07	13.9	0.71	0.7	✓
neos-1171692	170.2	33.0	4526.5	26.60	2672.2	81.01	8.3	0.05	10.7	0.32	0.7	✓
neos-1171737	1761.3	3600.0	2519.8	1.43	3600.0	1.00	2756.3	1.56	3600.0	1.00	0.3	✓
neos-1173026	1.0	0.5	1.0	1.00	0.5	1.00	1.0	1.00	0.5	1.00	0.7	✓
neos-1200887	5173.7	10.6	5985.8	1.16	10.8	1.02	3588.3	0.69	9.0	0.85	1.0	✓
neos-1208069	1180.1	50.1	1305.2	1.11	53.3	1.06	824.9	0.70	59.0	1.18	–	
neos-1208135	3058.6	291.4	3717.6	1.22	182.2	0.62	2013.0	0.66	132.8	0.46	1.0	✓
neos-1211578	7614.2	4.4	5431.7	0.71	3.3	0.75	5667.8	0.74	3.4	0.79	1.0	✓
neos-1215259	1029.6	39.6	911.4	0.89	36.1	0.91	1141.0	1.11	49.1	1.24	–	
neos-1215891	3055.5	309.3	7638.3	2.50	533.9	1.73	2312.4	0.76	210.7	0.68	–	
neos-1223462	153.5	152.1	236.6	1.54	172.2	1.13	363.3	2.37	182.4	1.20	0.3	✓
neos-1224597	12.2	11.2	24.6	2.02	11.2	1.00	12.2	1.00	10.4	0.93	–	
neos-1225589	3.0	0.5	7.3	2.44	0.5	0.97	7.9	2.64	0.5	1.02	33.7	✓
neos-1228986	24206.5	12.4	45629.4	1.89	18.0	1.45	23074.2	0.95	11.1	0.89	1.0	✓
neos-1281048	41.6	7.2	39.8	0.96	5.7	0.79	35.1	0.84	6.2	0.86	0.3	✓
neos-1311124	7494756.8	3600.0	7226683.0	0.96	3600.0	1.00	7537735.7	1.01	3600.0	1.00	1.0	✓
neos-1324574	17185.2	1457.0	43359.1	2.52	3401.8	2.33	16764.4	0.98	1226.0	0.84	–	
neos-1330346	182663.5	3451.6	146965.9	0.81	3600.0	1.04	198130.5	1.08	2984.5	0.86	–	
neos-1330635	1.0	0.5	1.0	1.00	0.5	1.00	1.3	1.32	0.5	1.00	0.3	✓
neos-1337307	354039.9	3600.0	339961.0	0.96	3600.0	1.00	350512.4	0.99	3600.0	1.00	0.3	✓
neos-1346382	6883767.2	3600.0	7740568.6	1.12	3600.0	1.00	5950992.0	0.86	3600.0	1.00	1.0	✓
neos-1354092	213.7	3600.0	258.0	1.21	3600.0	1.00	144.5	0.68	3600.0	1.00	–	
neos-1367061	1.0	25.9	1.0	1.00	26.1	1.00	1.0	1.00	26.1	1.01	–	
neos-1396125	11385.0	93.1	14679.1	1.29	92.7	1.00	20326.1	1.78	119.4	1.28	0.3	✓
neos-1407044	33.6	3600.0	41.9	1.25	3600.0	1.00	33.6	1.00	3600.0	1.00	–	
neos-1413153	2.3	3.2	2.3	1.00	3.2	1.00	2.9	1.26	3.4	1.06	1.3	✓
neos-1415183	1.7	4.5	1.7	1.00	4.5	1.00	1.7	1.00	4.8	1.08	0.7	✓
neos-1417043	1.0	1141.4	1.0	1.00	1140.7	1.00	1.0	1.00	1135.6	0.99	–	
neos-1420205	6094.2	4.0	10093.1	1.66	5.7	1.42	14549.0	2.39	8.4	2.08	0.3	✓
neos-1420546	836.1	3600.0	1342.0	1.60	3600.0	1.00	686.0	0.82	3600.0	1.00	4.7	✓
neos-1420790	73156.0	3600.0	93275.3	1.27	3600.0	1.00	48949.8	0.67	3600.0	1.00	7.3	✓
neos-1423785	16498.6	3600.0	10194.9	0.62	3600.0	1.00	22543.0	1.37	3600.0	1.00	6.0	✓
neos-1425699	1.0	0.5	1.0	1.00	0.5	1.00	1.0	1.00	0.5	1.00	–	
neos-1426662	2592408.2	3600.0	3174356.0	1.22	3600.0	1.00	3082790.9	1.19	3600.0	1.00	1.0	✓
neos-1427181	1764466.7	1815.7	2450802.5	1.39	3600.0	1.98	958984.3	0.54	1018.7	0.56	1.0	✓
neos-1427261	560816.0	3600.0	1597704.9	2.85	3600.0	1.00	606443.5	1.08	3600.0	1.00	2.3	✓
neos-1429185	2500985.2	3600.0	3368842.0	1.35	3600.0	1.00	2608021.7	1.04	3600.0	1.00	1.3	✓

cont. on next page

Table D.1 cont.												
Instance	default		nodiving				adaptivediving					
	nodes	time	nodes	nodes _Q	time	time _Q	nodes	nodes _Q	time	time _Q	nsols	impr.sols
neos-1429212	1125.7	3600.0	1599.3	1.42	3600.0	1.00	966.5	0.86	3600.0	1.00	–	
neos-1429461	4009112.2	3600.0	4776886.7	1.19	3600.0	1.00	4324122.2	1.08	3600.0	1.00	1.3	✓
neos-1430701	35061.4	27.0	55174.3	1.57	33.5	1.24	43063.8	1.23	30.6	1.13	1.7	✓
neos-1430811	261.0	3600.0	905.8	3.47	3600.0	1.00	380.5	1.46	3600.0	1.00	–	
neos-1436709	2055406.2	3600.0	2410935.7	1.17	3600.0	1.00	2096401.2	1.02	3600.0	1.00	1.0	✓
neos-1436713	356491.2	3600.0	1013352.9	2.84	3600.0	1.00	450314.8	1.26	3600.0	1.00	1.7	✓
neos-1437164	1.0	0.5	1.0	1.00	0.5	1.00	1.0	1.00	0.5	1.00	–	
neos-1439395	322812.0	179.9	609646.7	1.89	297.7	1.66	247735.4	0.77	135.0	0.75	1.0	✓
neos-1440225	2972.8	54.6	5014.9	1.69	83.0	1.52	1873.4	0.63	37.2	0.68	–	
neos-1440447	3342.2	3.9	4243.4	1.27	4.3	1.09	3604.4	1.08	4.6	1.18	1.0	✓
neos-1440457	1869959.8	3600.0	1930525.2	1.03	3600.0	1.00	1743319.3	0.93	3600.0	1.00	1.7	✓
neos-1440460	3845061.5	2496.2	5447207.8	1.42	3600.0	1.44	4703501.0	1.22	3600.0	1.44	1.3	✓
neos-1441553	1.3	1.9	1.3	1.00	1.6	0.84	1.0	0.76	1.4	0.73	0.3	✓
neos-1442119	1631305.1	3600.0	1950306.8	1.20	3600.0	1.00	1568703.4	0.96	3600.0	1.00	1.0	✓
neos-1442657	2337818.1	3600.0	2716061.2	1.16	3600.0	1.00	2305938.3	0.99	3600.0	1.00	1.0	✓
neos-1445532	1244.9	3600.0	2259.8	1.81	3600.0	1.00	2323.8	1.87	3600.0	1.00	1.0	✓
neos-1445738	8033.9	3600.0	7220.6	0.90	3600.0	1.00	9033.8	1.12	3600.0	1.00	4.3	✓
neos-1445743	51.1	44.5	19.3	0.38	51.6	1.16	3.3	0.07	52.2	1.17	4.3	✓
neos-1445755	36.2	45.4	67.0	1.85	49.5	1.09	156.8	4.33	43.2	0.95	5.0	✓
neos-1445765	147.9	41.8	124.1	0.84	40.1	0.96	256.5	1.74	44.7	1.07	6.0	✓
neos-1451294	2755.2	1154.5	4767.2	1.73	1701.7	1.47	1673.6	0.61	850.4	0.74	1.0	✓
neos-1456979	26421.6	3600.0	19765.3	0.75	3600.0	1.00	28764.2	1.09	3600.0	1.00	1.3	✓
neos-1460246	1564054.1	3600.0	3073044.6	1.97	3600.0	1.00	1673384.3	1.07	3600.0	1.00	–	
neos-1460265	113.6	5.5	127.9	1.13	5.9	1.06	99.6	0.88	4.6	0.84	1.0	✓
neos-1460543	4967.7	3600.0	10189.0	2.05	3600.0	1.00	5543.7	1.12	3600.0	1.00	3.0	✓
neos-1460641	318140.4	3600.0	248512.6	0.78	3600.0	1.00	295331.9	0.93	3600.0	1.00	2.7	✓
neos-1461051	2596.5	26.0	2541.2	0.98	25.7	0.99	2017.5	0.78	22.9	0.88	–	
neos-1464762	441568.6	3600.0	427882.2	0.97	3600.0	1.00	257345.7	0.58	3600.0	1.00	1.7	✓
neos-1467067	7214231.6	3600.0	7437876.4	1.03	3600.0	1.00	6794105.7	0.94	3600.0	1.00	1.0	✓
neos-1467371	427774.6	3600.0	325660.7	0.76	3600.0	1.00	377995.4	0.88	3600.0	1.00	0.7	✓
neos-1467467	70309.6	3600.0	89027.4	1.27	3600.0	1.00	38093.5	0.54	3600.0	1.00	0.3	✓
neos-1480121	127.7	0.5	330.4	2.59	5.5	10.97	119.9	0.94	4.2	8.43	1.7	✓
neos-1489999	27.9	2.8	26.9	0.96	2.6	0.94	38.7	1.39	3.7	1.33	0.3	✓
neos-1516309	1.0	0.5	1.0	1.00	0.5	1.01	1.0	1.00	0.5	1.00	–	
neos-1582420	860.1	44.1	486.0	0.56	30.8	0.70	215.7	0.25	29.7	0.67	0.3	✓
neos-1593097	27619.9	3600.0	24674.4	0.89	3600.0	1.00	25109.3	0.91	3600.0	1.00	1.7	✓
neos-1595230	34795.5	237.8	38402.7	1.10	198.6	0.83	24084.1	0.69	177.1	0.74	–	
neos-1597104	17.6	227.6	5.0	0.28	180.6	0.79	11.1	0.63	212.4	0.93	–	
neos-1599274	1.0	0.9	1.0	1.00	0.9	0.96	1.0	1.00	0.9	0.94	1.0	✓
neos-1601936	2893.9	2650.5	2107.6	0.73	2859.4	1.08	962.1	0.33	1534.9	0.58	1.0	✓
neos-1603512	13.0	2.0	13.7	1.05	1.9	0.96	8.8	0.67	1.9	0.92	–	
neos-1603518	20.6	5.8	28.9	1.40	6.4	1.10	26.0	1.26	6.1	1.06	–	
neos-1603965	32040.5	3600.0	46357.4	1.45	3600.0	1.00	40267.8	1.26	3600.0	1.00	1.0	✓
neos-1605061	435.8	3600.0	428.3	0.98	3600.0	1.00	611.6	1.40	3600.0	1.00	–	

cont. on next page

Table D.1 cont.

Instance	default		nodiving				adaptivediving					
	nodes	time	nodes	nodesQ	time	timeQ	nodes	nodesQ	time	timeQ	nsols	impr.sols
neos-1605075	1656.5	3600.0	1200.3	0.72	2733.4	0.76	1028.8	0.62	3600.0	1.00	1.7	✓
neos-1616732	1220610.5	3191.9	1238740.1	1.01	3383.6	<i>1.06</i>	1233346.4	1.01	3199.1	1.00	7.3	✓
neos-1620770	592673.2	3600.0	1065280.6	<i>1.80</i>	3600.0	1.00	610626.9	1.03	3600.0	1.00	–	
neos-1620807	1027.9	5.0	1562.3	<i>1.52</i>	5.7	<i>1.14</i>	2123.4	<i>2.07</i>	7.7	<i>1.53</i>	–	
neos-1622252	884058.2	3227.6	928864.0	<i>1.05</i>	2502.4	0.78	836096.5	0.95	3600.0	<i>1.11</i>	–	
neos-430149	32846.4	30.9	33951.8	1.03	30.3	0.98	23787.7	0.72	23.6	0.76	0.3	✓
neos-476283	419.3	140.1	500.9	<i>1.20</i>	138.8	0.99	638.7	<i>1.52</i>	156.5	<i>1.12</i>	1.3	✓
neos-480878	11810.1	46.3	9748.7	0.82	34.6	0.75	8826.7	0.75	38.3	0.83	29.0	✓
neos-494568	4.9	16.5	6.3	<i>1.28</i>	21.4	<i>1.29</i>	4.1	0.84	17.3	1.05	0.3	✓
neos-495307	71326.2	3600.0	104866.4	<i>1.47</i>	3600.0	1.00	57285.4	0.80	3600.0	1.00	134.7	
neos-498623	40.5	45.5	55.5	<i>1.37</i>	49.0	<i>1.07</i>	6.3	0.16	35.2	0.77	0.3	✓
neos-501453	1.0	0.5	1.0	1.00	0.5	1.00	1.0	1.00	0.5	1.00	–	
neos-501474	2.0	0.5	2.0	1.00	0.5	1.00	2.0	1.00	0.5	1.00	–	
neos-503737	24214.7	536.5	14562.5	0.60	330.4	0.62	4680.7	0.19	225.8	0.42	1.3	✓
neos-504674	6176.5	42.6	5774.2	0.94	38.9	0.91	6190.0	1.00	44.8	<i>1.05</i>	1.0	✓
neos-504815	2257.0	14.4	2934.0	<i>1.30</i>	18.2	<i>1.26</i>	2245.2	0.99	14.8	1.03	1.0	✓
neos-506422	5285.9	48.6	3985.5	0.75	37.6	0.77	2743.3	0.52	35.3	0.72	2.0	✓
neos-506428	265.5	3516.6	186.3	0.70	3600.0	1.02	155.2	0.58	3600.0	1.02	0.7	✓
neos-512201	3177.8	28.3	2450.2	0.77	24.9	0.88	3421.7	<i>1.08</i>	29.0	1.02	1.7	✓
neos-522351	1.0	1.2	1.0	1.00	1.3	1.02	1.0	1.00	1.2	0.99	–	
neos-525149	1.0	3.3	7.8	<i>7.85</i>	4.4	<i>1.35</i>	1.0	1.00	2.6	0.80	1.0	✓
neos-530627	1.0	0.5	1.0	1.00	0.5	1.00	1.0	1.00	0.5	1.00	–	
neos-538867	32928.4	82.0	34850.1	<i>1.06</i>	102.7	<i>1.25</i>	25291.6	0.77	64.8	0.79	1.0	✓
neos-538916	6947.7	36.3	5593.1	0.81	27.9	0.77	7852.7	<i>1.13</i>	33.8	0.93	1.0	✓
neos-544324	13.5	39.5	72.6	<i>5.37</i>	39.7	1.00	15.3	<i>1.13</i>	27.8	0.70	0.3	✓
neos-547911	221.4	28.6	191.7	0.87	19.7	0.69	41.2	0.19	16.9	0.59	–	
neos-548047	13599.1	3600.0	24630.6	<i>1.81</i>	3600.0	1.00	15773.8	<i>1.16</i>	3600.0	1.00	0.3	✓
neos-548251	789491.1	3600.0	2042422.3	<i>2.59</i>	3600.0	1.00	568904.1	0.72	3600.0	1.00	1.7	✓
neos-551991	671.0	139.5	1793.7	<i>2.67</i>	220.4	<i>1.58</i>	609.1	0.91	131.9	0.95	–	
neos-555001	19.8	2.4	1.0	0.05	1.0	0.40	9.0	0.46	1.6	0.64	–	
neos-555298	639.3	65.0	2261.2	<i>3.54</i>	77.9	<i>1.20</i>	134.5	0.21	42.7	0.66	1.3	✓
neos-555343	315086.5	2328.1	276719.5	0.88	2421.1	1.04	273288.3	0.87	2198.4	0.94	1.0	✓
neos-555424	114209.2	1065.6	374022.3	<i>3.27</i>	3600.0	<i>3.38</i>	98479.8	0.86	1043.1	0.98	1.7	✓
neos-555694	16.9	9.0	16.2	0.96	5.1	0.57	1.7	0.10	2.4	0.26	0.7	✓
neos-555771	24.1	6.3	6.3	0.26	2.9	0.45	5.5	0.23	3.4	0.53	1.3	✓
neos-555884	252762.8	3600.0	359810.0	<i>1.42</i>	3600.0	1.00	235609.8	0.93	3600.0	1.00	1.3	✓
neos-555927	1115513.2	3600.0	1156971.4	1.04	3600.0	1.00	1141441.7	1.02	3600.0	1.00	3.3	✓
neos-565672	8.3	3486.6	6.0	0.72	3486.8	1.00	6.3	0.76	3522.9	1.01	1.3	✓
neos-565815	1.0	10.2	1.0	1.00	9.5	0.93	1.0	1.00	8.4	0.82	–	
neos-570431	257.1	9.7	326.3	<i>1.27</i>	10.5	<i>1.08</i>	276.7	<i>1.08</i>	10.4	<i>1.08</i>	0.3	✓
neos-574665	4377366.4	3600.0	4252382.8	0.97	3600.0	1.00	3708991.3	0.85	3600.0	1.00	1.0	✓
neos-578379	1.6	179.6	1.6	1.00	179.6	1.00	1.9	<i>1.18</i>	177.2	0.99	–	
neos-582605	501081.5	3600.0	607130.3	<i>1.21</i>	3600.0	1.00	443537.2	0.89	3600.0	1.00	–	
neos-583731	16.7	6.6	16.7	1.00	6.6	1.00	20.7	<i>1.24</i>	7.1	<i>1.07</i>	–	

cont. on next page

Table D.1 cont.												
Instance	default		nodiving				adaptivediving					
	nodes	time	nodes	nodes _Q	time	time _Q	nodes	nodes _Q	time	time _Q	nsols	impr.sols
neos-584146	858994.0	3600.0	966703.3	1.12	3600.0	1.00	1057800.0	1.23	3600.0	1.00	–	
neos-584851	8.9	6.9	6.9	0.77	7.1	1.03	7.8	0.87	7.3	1.06	0.3	✓
neos-584866	112403.3	3600.0	113208.1	1.01	3600.0	1.00	99188.6	0.88	3600.0	1.00	0.3	✓
neos-585192	749.0	21.2	737.4	0.98	20.9	0.99	785.0	1.05	20.1	0.95	0.7	✓
neos-585467	79.5	8.6	62.2	0.78	8.2	0.95	113.7	1.43	9.4	1.09	1.0	✓
neos-593853	28543.2	71.4	12670.0	0.44	31.6	0.44	57948.9	2.03	120.1	1.68	0.7	✓
neos-595904	2.7	16.2	3.6	1.37	15.3	0.94	3.3	1.24	17.7	1.09	0.3	✓
neos-595905	2.6	2.8	3.0	1.13	2.9	1.03	2.0	0.76	2.6	0.91	1.0	✓
neos-595925	27.6	11.4	252.6	9.15	16.1	1.42	27.6	1.00	12.8	1.13	0.3	✓
neos-598183	5.0	5.9	4.0	0.80	4.8	0.82	6.0	1.21	5.7	0.97	–	
neos-603073	231538.1	1099.6	81134.4	0.35	324.7	0.29	129468.2	0.56	479.2	0.44	1.3	✓
neos-611135	74338.8	3600.0	77543.2	1.04	3600.0	1.00	89741.7	1.21	3600.0	1.00	2.0	✓
neos-611838	713.9	19.1	670.9	0.94	15.9	0.83	668.3	0.94	18.4	0.97	4.0	
neos-612125	310.9	14.1	340.3	1.09	13.4	0.95	280.4	0.90	11.0	0.78	3.7	✓
neos-612143	702.0	17.5	726.5	1.03	15.6	0.89	602.0	0.86	17.8	1.01	3.7	✓
neos-612162	631.0	15.0	606.0	0.96	14.9	0.99	605.5	0.96	16.1	1.07	3.7	✓
neos-619167	65.7	3600.0	65.2	0.99	3600.0	1.00	37.9	0.58	3600.0	1.00	–	
neos-631164	188430.0	3600.0	449023.0	2.38	3600.0	1.00	117213.5	0.62	3600.0	1.00	13.0	✓
neos-631517	167948.0	3600.0	504179.0	3.00	3600.0	1.00	118475.0	0.70	3600.0	1.00	11.7	✓
neos-631694	541659.9	3600.0	161942.9	0.30	1696.0	0.47	12411.9	0.02	629.4	0.17	0.3	✓
neos-631709	763.3	3600.0	694.0	0.91	3600.0	1.00	369.7	0.48	3600.0	1.00	–	
neos-631710	1.0	3600.0	1.0	1.00	3600.0	1.00	1.0	1.00	3600.0	1.00	–	
neos-631784	177973.5	3600.0	18259.6	0.10	650.1	0.18	143580.3	0.81	2975.3	0.83	0.3	✓
neos-632335	193.0	10.4	193.0	1.00	10.4	1.00	193.0	1.00	10.4	1.00	–	
neos-633273	259.0	11.1	259.0	1.00	11.0	0.99	259.0	1.00	11.1	1.00	–	
neos-655508	1.0	1.4	1.0	1.00	1.4	1.01	1.0	1.00	1.4	1.01	–	
neos-662469	19081.9	3600.0	9208.4	0.48	1484.4	0.41	26618.9	1.40	3600.0	1.00	3.3	✓
neos-686190	7938.0	107.4	4368.0	0.55	72.2	0.67	7759.5	0.98	107.6	1.00	–	
neos-691058	3096.9	3600.0	3505.3	1.13	3600.0	1.00	2878.3	0.93	3600.0	1.00	1.0	✓
neos-691073	6801.0	3600.0	7842.4	1.15	3600.0	1.00	5827.3	0.86	3600.0	1.00	0.3	✓
neos-693347	19879.9	2418.9	14909.3	0.75	1811.6	0.75	15391.8	0.77	2426.3	1.00	0.7	✓
neos-702280	1031.5	3600.0	1228.0	1.19	3600.0	1.00	898.9	0.87	3600.0	1.00	9.0	
neos-709469	189.7	0.7	153.0	0.81	0.7	1.04	48.7	0.26	0.5	0.78	–	
neos-717614	8855.5	26.6	1414.6	0.16	7.4	0.28	4045.8	0.46	17.1	0.64	0.3	✓
neos-738098	396.1	3600.0	1052.4	2.66	3600.0	1.00	1011.9	2.55	3600.0	1.00	–	
neos-775946	4.5	8.0	13.4	2.99	8.9	1.11	2.7	0.59	6.7	0.84	1.7	✓
neos-780889	1.0	96.2	1.9	1.92	122.6	1.27	1.0	1.00	90.5	0.94	0.7	✓
neos-785899	28.3	2.3	32.1	1.14	3.0	1.33	15.5	0.55	4.1	1.79	–	
neos-785912	203.2	41.0	271.8	1.34	37.2	0.91	274.1	1.35	57.5	1.40	–	
neos-785914	13.0	8.3	23.8	1.84	9.4	1.13	12.6	0.97	7.8	0.94	–	
neos-787933	1.0	1.6	1.0	1.00	1.6	1.00	1.0	1.00	1.6	1.00	–	
neos-791021	36.1	367.7	195.0	5.41	648.1	1.76	35.6	0.99	343.1	0.93	1.0	✓
neos-796608	141.9	2.1	21041.2	148.30	61.8	29.40	2.7	0.02	0.5	0.24	0.7	✓
neos-799838	1.0	32.0	3.2	3.20	47.1	1.47	1.0	1.00	33.1	1.03	0.7	✓

cont. on next page

Table D.1 cont.

Instance	default		nodiving				adaptivediving					
	nodes	time	nodes	nodes _Q	time	time _Q	nodes	nodes _Q	time	time _Q	nsols	impr.sols
neos-801834	241.0	42.2	179.6	0.74	36.6	0.86	155.7	0.65	39.9	0.94	1.0	✓
neos-803219	12618.8	35.7	14849.1	<i>1.18</i>	34.6	0.97	12244.3	0.97	33.5	0.94	1.7	✓
neos-803220	44266.4	78.7	51057.4	<i>1.15</i>	80.2	1.02	42513.6	0.96	75.5	0.96	2.3	✓
neos-806323	7280.4	32.2	6827.6	0.94	30.2	0.94	7238.3	0.99	34.2	<i>1.06</i>	1.0	✓
neos-807454	1.0	2.0	1.0	1.00	2.0	0.99	1.0	1.00	2.0	0.99	–	
neos-807639	2506.2	15.9	3034.8	<i>1.21</i>	17.4	<i>1.10</i>	2528.7	1.01	15.9	1.00	1.3	✓
neos-807705	3331.6	26.8	5277.8	<i>1.58</i>	30.7	<i>1.15</i>	3326.6	1.00	27.5	1.02	3.7	✓
neos-808072	97.2	19.3	80.9	0.83	16.2	0.84	186.3	<i>1.92</i>	25.7	<i>1.33</i>	–	
neos-808214	1148.6	22.9	1546.0	<i>1.35</i>	24.6	<i>1.08</i>	795.5	0.69	18.8	0.82	–	
neos-810286	132.6	75.8	50.8	0.38	64.9	0.86	81.2	0.61	60.5	0.80	–	
neos-810326	823.9	39.8	513.1	0.62	28.8	0.72	1350.1	<i>1.64</i>	55.5	<i>1.39</i>	0.7	✓
neos-820146	1505505.0	3600.0	1554994.4	1.03	3600.0	1.00	1054142.6	0.70	3600.0	1.00	–	
neos-820157	1180763.2	3600.0	1343696.6	<i>1.14</i>	3600.0	1.00	1080849.3	0.92	3600.0	1.00	–	
neos-820879	256.1	52.8	396.4	<i>1.55</i>	48.3	0.92	178.4	0.70	66.5	<i>1.26</i>	0.7	✓
neos-824661	43.1	1298.1	212.3	<i>4.93</i>	1254.5	0.97	44.7	1.04	1243.0	0.96	–	
neos-824695	8.5	184.3	201.8	<i>23.85</i>	613.8	<i>3.33</i>	20.4	<i>2.42</i>	269.8	<i>1.46</i>	0.7	✓
neos-825075	4.4	1.9	3.8	0.85	1.8	0.94	1.7	0.37	1.7	0.88	0.7	✓
neos-826224	1.0	49.0	22.4	<i>22.45</i>	169.9	<i>3.46</i>	1.0	1.00	50.1	1.02	0.7	✓
neos-826250	7.1	137.9	13.5	<i>1.90</i>	127.5	0.93	1.3	0.19	80.9	0.59	0.3	✓
neos-826650	15642.6	3600.0	18816.8	<i>1.20</i>	3600.0	1.00	9255.5	0.59	3600.0	1.00	0.3	✓
neos-826694	6.5	174.5	39.3	<i>6.07</i>	272.9	<i>1.56</i>	1.0	0.15	102.5	0.59	–	
neos-826812	1.0	61.7	1.0	1.00	39.8	0.65	1.0	1.00	66.8	<i>1.08</i>	0.3	✓
neos-826841	82991.1	3600.0	77077.1	0.93	3600.0	1.00	71126.2	0.86	3600.0	1.00	0.7	✓
neos-827015	292.7	1138.5	269.8	0.92	932.6	0.82	291.3	0.99	1264.9	<i>1.11</i>	–	
neos-827175	1.0	12.1	1.0	1.00	9.1	0.75	1.0	1.00	12.3	1.01	0.7	✓
neos-829552	453.9	530.9	277.9	0.61	268.6	0.51	364.5	0.80	422.6	0.80	0.3	✓
neos-830439	1.0	0.5	1.0	1.00	0.5	1.00	1.0	1.00	0.5	1.00	–	
neos-831188	2682.1	282.8	2837.9	<i>1.06</i>	262.7	0.93	2754.9	1.03	280.2	0.99	–	
neos-839838	23347.8	858.9	31126.9	<i>1.33</i>	1065.9	<i>1.24</i>	24314.9	1.04	873.0	1.02	1.3	✓
neos-839859	3254.8	47.7	2838.9	0.87	45.0	0.94	3342.6	1.03	51.3	<i>1.08</i>	0.3	✓
neos-839894	277.1	3600.0	283.8	1.02	3600.0	1.00	206.5	0.74	3600.0	1.00	–	
neos-841664	18450.7	3600.0	50623.4	<i>2.74</i>	3600.0	1.00	18012.5	0.98	3600.0	1.00	269.3	✓
neos-847302	154746.9	3600.0	104375.8	0.67	3600.0	1.00	195037.0	<i>1.26</i>	3600.0	1.00	1.7	✓
neos-848150	25.4	12.5	41.4	<i>1.63</i>	13.8	<i>1.10</i>	65.3	<i>2.57</i>	13.6	<i>1.08</i>	–	
neos-848198	616.7	3600.0	2026.1	<i>3.29</i>	3600.0	1.00	516.6	0.84	3600.0	1.00	184.3	✓
neos-848589	104.3	3600.0	167.4	<i>1.60</i>	3600.0	1.00	101.2	0.97	3600.0	1.00	0.3	✓
neos-848845	2587.4	155.0	12070.0	<i>4.67</i>	245.9	<i>1.59</i>	14142.9	<i>5.47</i>	283.7	<i>1.83</i>	–	
neos-849702	11836.6	268.5	6239.6	0.53	160.1	0.60	6346.0	0.54	175.8	0.66	–	
neos-850681	1.3	4.2	6.0	<i>4.53</i>	6.4	<i>1.53</i>	1.3	1.00	4.8	<i>1.14</i>	–	
neos-856059	190474.5	3117.3	210304.6	<i>1.10</i>	3389.9	<i>1.09</i>	72797.4	0.38	1405.9	0.45	9.0	✓
neos-859770	2.7	131.2	4.1	<i>1.50</i>	133.2	1.01	3.6	<i>1.34</i>	133.0	1.01	–	
neos-860244	1.3	4.5	1.0	0.76	4.3	0.94	1.0	0.76	4.3	0.94	–	
neos-860300	2.3	15.7	2.7	<i>1.14</i>	15.8	1.00	2.6	<i>1.13</i>	19.5	<i>1.24</i>	1.0	✓
neos-862348	86.7	9.3	87.0	1.00	9.1	0.98	38.8	0.45	9.9	<i>1.07</i>	1.0	✓

cont. on next page

Table D.1 cont.												
Instance	default		nodiving				adaptivediving					
	nodes	time	nodes	nodes _Q	time	time _Q	nodes	nodes _Q	time	time _Q	nsols	impr.sols
neos-863472	44960.0	48.0	38508.1	0.86	43.5	0.91	40379.5	0.90	43.8	0.91	–	
neos-872648	15.9	3600.0	19.9	<i>1.26</i>	3600.0	1.00	14.6	0.92	3600.0	1.00	5.3	✓
neos-873061	27.6	3600.0	36.2	<i>1.31</i>	3600.0	1.00	40.1	<i>1.45</i>	3600.0	1.00	13.0	
neos-876808	207.6	3600.0	459.8	<i>2.21</i>	3600.0	1.00	193.1	0.93	3600.0	1.00	0.7	✓
neos-880324	6.8	1.1	10.9	<i>1.61</i>	1.1	0.99	11.1	<i>1.65</i>	1.1	0.94	0.7	✓
neos-881765	17.9	2.0	17.9	1.00	2.0	0.99	55.8	<i>3.11</i>	2.4	<i>1.20</i>	–	
neos-885086	274.8	3600.0	315.0	<i>1.15</i>	3600.0	1.00	129.3	0.47	1717.3	0.48	–	
neos-885524	4585.4	3018.8	7955.3	<i>1.74</i>	1887.3	0.62	3002.9	0.66	1640.8	0.54	0.3	✓
neos-886822	178390.9	1976.9	116732.0	0.65	1470.8	0.74	177113.1	0.99	2174.5	<i>1.10</i>	0.3	✓
neos-892255	531.0	79.9	622.0	<i>1.17</i>	82.9	1.04	679.7	<i>1.28</i>	146.7	<i>1.83</i>	–	
neos-905856	11281.3	134.5	5478.6	0.49	83.9	0.62	9109.0	0.81	128.8	0.96	–	
neos-906865	16334.8	80.2	22002.9	<i>1.35</i>	116.4	<i>1.45</i>	18494.8	<i>1.13</i>	96.4	<i>1.20</i>	0.7	✓
neos-911880	1159496.9	1709.6	1733877.8	<i>1.50</i>	1959.2	<i>1.15</i>	2928989.1	<i>2.53</i>	3600.0	<i>2.11</i>	3.0	✓
neos-911970	3830083.0	3600.0	635148.8	0.17	1067.4	0.30	624932.8	0.16	938.3	0.26	1.0	✓
neos-912015	46.9	7.5	20.0	0.43	5.5	0.73	117.5	<i>2.50</i>	8.5	<i>1.13</i>	–	
neos-912023	9.1	5.8	10.8	<i>1.19</i>	6.0	1.03	84.2	<i>9.24</i>	8.9	<i>1.52</i>	–	
neos-913984	1.0	12.2	1.0	1.00	12.0	0.98	1.0	1.00	12.1	0.99	–	
neos-914441	1.0	5.0	1.0	1.00	5.0	1.00	1.7	<i>1.66</i>	6.6	<i>1.31</i>	1.3	✓
neos-916173	13027.5	102.8	15257.2	<i>1.17</i>	108.5	<i>1.06</i>	16772.0	<i>1.29</i>	120.0	<i>1.17</i>	0.3	✓
neos-916792	140610.8	1000.5	217878.2	<i>1.55</i>	1565.9	<i>1.56</i>	146682.8	1.04	1242.5	<i>1.24</i>	2.7	✓
neos-930752	864.2	3600.0	2417.9	<i>2.80</i>	3600.0	1.00	1327.7	<i>1.54</i>	3600.0	1.00	1.7	✓
neos-931517	874.7	3600.0	1237.9	<i>1.42</i>	3600.0	1.00	613.4	0.70	3600.0	1.00	1.3	✓
neos-931538	5.0	99.6	5.0	1.00	70.0	0.70	3.5	0.70	70.3	0.71	0.3	✓
neos-932721	18.3	22.1	19.5	<i>1.06</i>	18.8	0.85	2.2	0.12	12.2	0.55	–	
neos-932816	301.9	3600.0	673.4	<i>2.23</i>	3600.0	1.00	323.0	<i>1.07</i>	3600.0	1.00	2.0	✓
neos-933364	17038.6	47.4	19695.6	<i>1.16</i>	57.4	<i>1.21</i>	7555.2	0.44	28.9	0.61	1.0	✓
neos-933550	25.7	3.5	25.7	1.00	3.5	1.00	5.5	0.21	1.8	0.53	0.7	✓
neos-933562	20014.7	3600.0	36440.8	<i>1.82</i>	3600.0	1.00	17020.4	0.85	3600.0	1.00	1.7	✓
neos-933638	152.7	2862.4	477.2	<i>3.13</i>	3600.0	<i>1.26</i>	44.3	0.29	1752.6	0.61	–	
neos-933815	73373.1	89.8	32194.9	0.44	51.2	0.57	38306.8	0.52	92.9	1.03	1.3	✓
neos-933966	133.1	2105.2	1296.0	<i>9.73</i>	2295.3	<i>1.09</i>	158.7	<i>1.19</i>	1558.6	0.74	0.7	✓
neos-934278	220.7	3600.0	761.7	<i>3.45</i>	3600.0	1.00	244.1	<i>1.11</i>	3600.0	1.00	1.3	✓
neos-934441	189.7	3600.0	446.9	<i>2.36</i>	3600.0	1.00	239.5	<i>1.26</i>	3600.0	1.00	1.0	✓
neos-934531	3.6	111.2	4.0	<i>1.12</i>	82.5	0.74	6.5	<i>1.79</i>	94.5	0.85	–	
neos-935234	141.6	3600.0	285.8	<i>2.02</i>	3600.0	1.00	173.7	<i>1.23</i>	3600.0	1.00	1.3	✓
neos-935348	76.6	3600.0	576.9	<i>7.53</i>	3600.0	1.00	291.6	<i>3.81</i>	3600.0	1.00	1.0	✓
neos-935496	641435.5	3600.0	658733.2	1.03	3600.0	1.00	558638.0	0.87	3600.0	1.00	1.3	✓
neos-935627	368.2	3600.0	472.8	<i>1.28</i>	3600.0	1.00	429.6	<i>1.17</i>	3600.0	1.00	0.7	✓
neos-935674	610846.1	3600.0	661005.3	<i>1.08</i>	3600.0	1.00	459041.4	0.75	3600.0	1.00	1.7	✓
neos-935769	420.5	3600.0	1824.5	<i>4.34</i>	3600.0	1.00	273.7	0.65	3600.0	1.00	0.7	✓
neos-936660	511.2	3600.0	489.5	0.96	3600.0	1.00	317.0	0.62	3600.0	1.00	0.7	✓
neos-937446	38.8	1229.1	1107.6	<i>28.56</i>	3600.0	<i>2.93</i>	81.1	<i>2.09</i>	2483.1	<i>2.02</i>	–	
neos-937511	268.2	3324.2	1158.3	<i>4.32</i>	3600.0	<i>1.08</i>	158.3	0.59	2158.6	0.65	2.3	✓
neos-937815	390.6	3600.0	1750.0	<i>4.48</i>	3600.0	1.00	303.9	0.78	3600.0	1.00	2.0	✓

cont. on next page

Table D.1 cont.												
Instance	default		nodiving				adaptivediving					
	nodes	time	nodes	nodesQ	time	timeQ	nodes	nodesQ	time	timeQ	nsols	impr.sols
neos-941262	396.1	3600.0	813.1	2.05	3600.0	1.00	195.6	0.49	3600.0	1.00	1.7	✓
neos-941313	17.8	1731.1	18.8	1.06	1544.7	0.89	12.8	0.72	1554.0	0.90	–	
neos-941698	174.4	13.2	167.0	0.96	12.9	0.98	42.9	0.25	7.3	0.55	–	
neos-941717	614427.9	3600.0	598685.0	0.97	3600.0	1.00	572084.3	0.93	3600.0	1.00	2.0	✓
neos-941782	817458.4	3600.0	694131.5	0.85	3600.0	1.00	704851.4	0.86	3600.0	1.00	1.3	✓
neos-942323	113.0	4.9	113.0	1.00	4.9	1.00	818.2	7.24	8.3	1.69	–	
neos-942830	324943.6	952.8	379201.9	1.17	1166.3	1.22	352690.1	1.08	974.1	1.02	0.3	✓
neos-942886	1.0	0.5	1.0	1.00	0.5	1.00	1.0	1.00	0.5	1.00	–	
neos-948126	298.9	3600.0	557.0	1.86	3600.0	1.00	223.7	0.75	3600.0	1.00	1.3	✓
neos-948268	1.0	12.1	1.0	1.00	12.1	1.00	1.0	1.00	12.1	1.00	–	
neos-948346	17.5	3600.0	45.7	2.60	3600.0	1.00	47.8	2.73	3600.0	1.00	0.3	✓
neos-950242	27.9	173.9	25.5	0.91	158.0	0.91	123.6	4.43	425.6	2.45	1.0	✓
neos-952987	1.0	3600.0	1.0	1.00	3600.0	1.00	1.0	1.00	3600.0	1.00	–	
neos-953928	62.3	400.4	69.7	1.12	373.3	0.93	17.8	0.29	272.9	0.68	–	
neos-954925	105.7	2173.4	570.1	5.39	3600.0	1.66	45.8	0.43	1218.4	0.56	–	
neos-955215	17478.0	18.7	17898.2	1.02	20.6	1.10	14791.5	0.85	15.4	0.83	1.0	✓
neos-955800	1250.8	45.5	1698.1	1.36	49.9	1.10	779.0	0.62	40.4	0.89	1.3	✓
neos-956971	141.5	2392.5	450.7	3.18	3269.7	1.37	43.1	0.30	1311.3	0.55	0.3	✓
neos-957143	129.2	2127.4	2126.6	16.46	3600.0	1.69	22.8	0.18	349.6	0.16	0.7	✓
neos-957270	1.0	2.1	1.0	1.00	2.1	1.00	1.0	1.00	2.1	1.00	–	
neos-957323	2.8	101.0	6.3	2.22	93.9	0.93	5.8	2.06	134.1	1.33	0.7	✓
neos-957389	1.0	12.5	1.0	1.00	12.3	0.99	1.0	1.00	12.4	0.99	–	
neos-960392	29.1	662.3	90.8	3.12	777.2	1.17	25.9	0.89	613.6	0.93	0.3	✓
neos-983171	247.0	3600.0	807.1	3.27	3600.0	1.00	148.2	0.60	3600.0	1.00	–	
neos-984165	343.3	3600.0	504.4	1.47	3600.0	1.00	276.6	0.81	3600.0	1.00	1.3	✓
neos13	18381.8	380.0	187287.8	10.19	2655.1	6.99	15453.4	0.84	350.6	0.92	6.3	✓
neos18	1300.3	18.5	1222.5	0.94	21.8	1.18	948.2	0.73	17.8	0.96	0.3	✓
net12	2375.9	1177.0	2112.9	0.89	818.2	0.69	2130.4	0.90	749.4	0.64	0.3	✓
netdiversion	29.2	1128.7	139.8	4.79	3091.9	2.74	12.7	0.43	1045.3	0.93	0.7	✓
newdano	720667.7	3600.0	642235.4	0.89	3600.0	1.00	730835.1	1.01	3600.0	1.00	1.3	✓
noswot	366545.4	90.8	916511.3	2.50	218.9	2.41	504470.5	1.38	124.7	1.37	0.3	✓
ns1208400	857.8	256.0	932.8	1.09	288.2	1.13	1379.0	1.61	283.5	1.11	0.3	✓
ns1688347	790.2	71.2	834.8	1.06	68.8	0.97	1134.9	1.44	70.5	0.99	0.3	✓
ns1758913	1.3	2064.0	1.3	1.00	2051.7	0.99	1.0	0.76	1759.4	0.85	0.7	✓
ns1766074	891795.1	564.9	892001.5	1.00	562.6	1.00	899371.1	1.01	583.0	1.03	–	
ns1830653	5199.4	110.0	7136.5	1.37	134.2	1.22	5233.1	1.01	117.9	1.07	1.0	✓
nsa	207.8	1.9	208.5	1.00	1.3	0.71	211.9	1.02	1.9	1.01	–	
nsrand-ipx	54359.1	496.3	97719.9	1.80	846.3	1.71	36269.2	0.67	318.8	0.64	0.3	✓
nug08	1.7	63.2	2.0	1.19	52.4	0.83	1.3	0.80	58.4	0.93	0.7	✓
nw04	6.3	24.2	5.3	0.84	21.9	0.91	5.3	0.85	23.8	0.98	0.7	✓
opm2-z7-s2	2179.1	243.8	2734.0	1.25	304.6	1.25	2343.2	1.07	233.2	0.96	1.0	✓
opt1217	1.0	0.5	1.0	1.00	0.5	1.00	1.0	1.00	0.5	1.00	–	
p0033	1.0	0.5	1.0	1.00	0.5	1.00	1.0	1.00	0.5	1.00	–	
p0201	12.7	0.8	9.5	0.75	0.5	0.65	8.1	0.64	0.7	0.91	0.7	✓

cont. on next page

Table D.1 cont.												
Instance	default		nodiving				adaptivediving					
	nodes	time	nodes	nodes _Q	time	time _Q	nodes	nodes _Q	time	time _Q	nsols	impr.sols
p0282	2.3	0.5	2.3	1.00	0.5	0.99	2.3	1.00	0.5	0.94	3.3	✓
p0548	1.0	0.5	1.0	1.00	0.5	1.00	1.0	1.00	0.5	1.00	–	
p2756	13.2	3.3	11.0	0.84	2.5	0.78	5.1	0.39	3.0	0.92	1.3	✓
p6b	323294.8	3600.0	219821.9	0.68	3600.0	1.00	304344.2	0.94	3600.0	1.00	43.0	✓
pg	393.2	17.4	322.1	0.82	13.6	0.78	346.9	0.88	16.0	0.92	1.3	✓
pg5_34	205199.5	2352.5	242974.3	<i>1.18</i>	2456.6	1.04	193454.2	0.94	2203.8	0.94	1.3	✓
pigeon-10	4019215.8	1619.4	11611815.7	<i>2.89</i>	3600.0	<i>2.22</i>	8387118.0	<i>2.09</i>	3046.2	<i>1.88</i>	0.3	✓
pk1	334752.2	105.4	370026.6	<i>1.10</i>	110.2	1.05	307638.2	0.92	99.4	0.94	2.3	✓
pp08a	207.8	1.4	200.2	0.96	1.4	0.99	212.1	1.02	1.5	<i>1.05</i>	10.3	✓
pp08aCUTS	122.2	1.8	133.7	<i>1.09</i>	1.9	<i>1.09</i>	118.3	0.97	1.9	1.04	9.0	✓
prod1	26945.2	18.2	47322.8	<i>1.76</i>	27.6	<i>1.52</i>	43144.3	<i>1.60</i>	29.5	<i>1.62</i>	2.3	✓
prod2	98039.7	104.4	122168.5	<i>1.25</i>	120.3	<i>1.15</i>	110835.1	<i>1.13</i>	111.4	<i>1.07</i>	9.3	
protfold	5453.4	3600.0	6804.1	<i>1.25</i>	3600.0	1.00	4773.2	0.88	3600.0	1.00	0.3	✓
pw-myciel4	299777.0	2220.1	162153.8	0.54	1018.3	0.46	266983.7	0.89	1849.0	0.83	1.0	✓
qap10	5.7	152.3	3.3	0.57	87.9	0.58	1.7	0.29	119.4	0.78	0.7	✓
qiu	2580.1	24.2	2512.8	0.97	20.9	0.86	3918.7	<i>1.52</i>	28.1	<i>1.16</i>	8.3	✓
qnet1	15.7	3.7	9.7	0.62	3.8	1.03	2.6	0.17	1.5	0.40	5.7	✓
qnet1_o	2.0	1.8	5.1	<i>2.54</i>	1.8	0.98	2.3	<i>1.16</i>	2.2	<i>1.23</i>	1.0	✓
rail507	745.7	162.6	929.4	<i>1.25</i>	184.0	<i>1.13</i>	821.0	<i>1.10</i>	200.2	<i>1.23</i>	0.7	✓
ramos3	16.6	3600.0	17.0	1.02	3600.0	1.00	13.0	0.78	3600.0	1.00	5.7	
ran14x18.disj-8	461955.8	1440.7	426190.1	0.92	1415.7	0.98	411637.7	0.89	1294.1	0.90	108.3	✓
ran14x18_1	421575.1	1181.0	379931.0	0.90	986.3	0.83	417531.0	0.99	1058.7	0.90	129.3	
ran16x16	14207.2	77.6	9971.7	0.70	72.7	0.94	10089.0	0.71	73.4	0.94	63.7	✓
rd-rplusc-21	258853.1	3600.0	277132.2	<i>1.07</i>	3600.0	1.00	222275.4	0.86	3600.0	1.00	0.7	✓
reblock67	65100.0	258.7	47291.3	0.73	193.9	0.75	49203.0	0.76	195.4	0.76	1.3	✓
rentacar	6.0	2.2	6.0	1.00	2.1	0.99	5.3	0.88	2.2	1.02	1.3	✓
rgn	1.0	0.5	1.0	1.00	0.5	1.00	1.0	1.00	0.5	1.00	1.0	✓
rlp1	15702666.0	3600.0	16847612.6	<i>1.07</i>	3600.0	1.00	17579882.1	<i>1.12</i>	3600.0	1.00	–	
rmatr100-p10	818.4	166.4	835.2	1.02	149.1	0.90	806.9	0.99	162.3	0.97	2.3	✓
rmatr100-p5	507.7	279.8	473.1	0.93	241.0	0.86	473.6	0.93	285.9	1.02	1.0	✓
rmine6	94404.0	840.9	92116.0	0.98	827.8	0.98	87820.6	0.93	784.0	0.93	0.7	✓
rococoC10-001000	61788.9	747.1	66303.8	<i>1.07</i>	985.3	<i>1.32</i>	70980.9	<i>1.15</i>	892.1	<i>1.19</i>	1.3	✓
roll3000	2515.8	51.3	2254.4	0.90	45.5	0.89	3042.7	<i>1.21</i>	59.2	<i>1.15</i>	0.7	✓
rout	24846.1	82.1	13458.6	0.54	58.8	0.72	18795.8	0.76	67.5	0.82	2.7	✓
roy	1.3	0.5	1.3	1.00	0.5	1.00	1.0	0.76	0.5	1.00	0.3	✓
satellites1-25	225.6	892.7	483.1	<i>2.14</i>	674.2	0.76	105.2	0.47	684.3	0.77	1.0	✓
set1ch	1.0	0.5	1.0	1.00	0.5	1.00	1.0	1.00	0.5	1.00	–	
seymour	112578.7	3600.0	95622.5	0.85	3600.0	1.00	115834.4	1.03	3600.0	1.00	17.7	✓
seymour.disj-10	42724.0	3600.0	38954.1	0.91	3600.0	1.00	40519.8	0.95	3600.0	1.00	23.0	✓
sp97ar	11008.0	3600.0	10240.6	0.93	3600.0	1.00	7795.1	0.71	3600.0	1.00	–	
sp97ic	17051.6	3600.0	19037.2	<i>1.12</i>	3600.0	1.00	23161.4	<i>1.36</i>	3600.0	1.00	0.3	✓
sp98ar	1627.8	3600.0	1825.6	<i>1.12</i>	3600.0	1.00	1596.4	0.98	3600.0	1.00	0.3	✓
sp98ic	29695.4	3600.0	31775.9	<i>1.07</i>	3600.0	1.00	31181.3	<i>1.05</i>	3600.0	1.00	0.3	✓
sp98ir	5204.9	77.1	4859.2	0.93	67.9	0.88	4388.9	0.84	62.9	0.82	1.0	✓

cont. on next page

Table D.1 cont.												
Instance	default		nodiving				adaptivediving					
	nodes	time	nodes	nodes _Q	time	time _Q	nodes	nodes _Q	time	time _Q	nsols	impr.sols
stein27	1273.1	0.6	984.7	0.77	0.6	0.90	1310.8	1.03	0.7	<i>1.06</i>	3.0	
stein45	40030.3	12.7	44536.0	<i>1.11</i>	12.4	0.98	40029.2	1.00	12.4	0.98	3.3	
stp3d	1.0	3600.0	5.9	<i>5.92</i>	3600.0	1.00	1.0	1.00	3600.0	1.00	–	
swath	335083.3	3600.0	407024.3	<i>1.22</i>	3600.0	1.00	300096.1	0.90	3600.0	1.00	0.3	✓
tl1717	2003.0	3600.0	2313.9	<i>1.16</i>	3600.0	1.00	1277.3	0.64	3600.0	1.00	2.3	✓
tanglegram1	34.7	400.0	60.1	<i>1.73</i>	490.5	<i>1.23</i>	42.6	<i>1.23</i>	452.6	<i>1.13</i>	–	
tanglegram2	4.2	6.8	3.0	0.71	5.2	0.77	4.2	1.00	6.3	0.93	0.3	✓
timtab1	47140.5	55.1	30422.2	0.65	42.0	0.76	49461.7	1.05	56.3	1.02	4.0	✓
timtab2	1764544.1	3600.0	1688186.0	0.96	3600.0	1.00	1741605.1	0.99	3600.0	1.00	1.0	✓
tr12-30	384457.8	618.9	387094.5	1.01	603.7	0.97	390286.7	1.01	655.6	<i>1.06</i>	2.0	✓
triptim1	1.7	456.9	8.5	<i>5.13</i>	824.5	<i>1.80</i>	1.7	1.00	511.3	<i>1.12</i>	1.0	✓
unitcal_7	81.1	280.0	185.9	<i>2.29</i>	285.2	1.02	69.1	0.85	262.0	0.94	1.7	✓
vpm1	1.0	0.5	1.0	1.00	0.5	1.00	1.0	1.00	0.5	1.00	–	
vpm2	249.4	1.9	240.6	0.96	1.8	0.93	239.1	0.96	2.0	1.03	1.0	✓
vpphard	1079.3	3600.0	2446.2	<i>2.27</i>	3600.0	1.00	1294.4	<i>1.20</i>	3600.0	1.00	2.3	✓
zib54-UUE	87933.1	1734.3	76307.0	0.87	1363.8	0.79	62251.4	0.71	1382.7	0.80	43.0	✓
geom.	870.56	131.06	1066.52	1.23	132.64	1.01	763.29	0.88	125.22	0.96		
shgeom. (100/1)	2532.32	150.82	2901.74	1.15	152.81	1.01	2375.66	0.94	144.57	0.96		

D. Appendix D

Table D.2: LP pricing results for every tested instance, seed, and pricing method from the experiment in Section 7.2. We use the following abbreviations for the pricing methods (column P): d(evex), q(steep), s(teep), g(reedy), u(UCB), w(eighted). The table shows a total of 105 instances from the MMMC test set, which were selected from the entire list of instances with the bash command **shuf**.

Instance	P	time				LPthpt			
		0	1	2	3	0	1	2	3
alc1s1	d	900.0	900.0	900.0	900.0	42.574	43.040	41.563	40.135
	q	900.0	900.0	900.0	900.0	29.670	24.446	34.993	33.850
	s	900.0	900.0	900.0	900.0	37.213	36.888	36.613	36.993
	g	900.0	900.0	900.0	900.0	43.522	40.259	40.331	39.291
	u	900.0	900.0	900.0	900.0	41.359	31.043	51.633	45.428
	w	900.0	900.0	900.0	900.0	31.718	29.091	29.812	31.108
aflow30a	d	18.6	18.5	18.6	18.6	1244.000	1203.871	1144.785	1097.647
	q	21.9	30.1	30.6	25.8	753.696	525.260	587.282	578.873
	s	17.9	17.8	17.9	17.9	749.794	737.652	737.652	743.673
	g	17.7	18.9	19.7	18.1	1062.092	984.153	971.429	1063.514
	u	22.1	22.6	19.4	19.6	1100.000	980.095	972.340	1000.546
	w	19.2	18.8	19.3	19.2	967.337	902.604	988.462	988.462
app1-2	d	572.9	544.2	561.9	616.5	4.924	4.456	5.420	6.166
	q	900.0	900.0	900.0	900.0	1.269	1.228	4.278	4.655
	s	900.0	900.0	900.0	900.0	1.129	1.131	1.128	1.118
	g	735.7	585.5	526.9	568.9	5.902	4.674	4.229	4.663
	u	595.5	677.6	657.4	584.6	4.020	4.827	6.220	4.297
	w	705.8	651.0	602.2	631.2	5.293	4.607	3.987	4.183
bienst2	d	443.8	544.9	518.4	494.3	194.759	181.017	169.368	175.427
	q	900.0	900.0	900.0	900.0	56.873	37.016	40.996	40.979
	s	757.1	783.2	769.5	768.0	117.247	107.874	96.940	95.447
	g	472.5	557.4	627.0	530.3	189.471	183.868	158.378	170.847
	u	489.1	619.8	576.2	475.5	171.459	172.262	162.833	165.103
	w	508.0	717.4	656.4	858.1	130.264	136.877	119.819	128.774
core2536-691	d	237.8	208.3	134.8	493.0	3.357	1.312	0.434	2.729
	q	238.9	644.7	152.5	222.9	5.058	4.654	5.720	5.577
	s	115.8	88.8	195.2	436.7	10.571	6.546	4.075	9.569
	g	110.9	351.7	128.8	172.5	5.850	6.261	7.453	5.911
	u	388.7	253.3	379.8	112.0	3.427	6.705	2.127	6.726
	w	474.8	148.6	447.7	148.4	3.551	5.340	2.782	8.850
csched010	d	900.0	900.0	900.0	900.0	254.254	266.977	267.753	279.667
	q	900.0	900.0	900.0	900.0	177.648	206.084	200.400	188.400
	s	900.0	900.0	900.0	900.0	260.993	270.149	265.517	274.868
	g	900.0	900.0	900.0	900.0	271.277	287.984	246.270	297.205
	u	900.0	900.0	900.0	900.0	265.855	255.365	263.803	269.094
	w	900.0	900.0	900.0	900.0	247.492	245.468	247.870	239.830
d20200	d	900.0	900.0	900.0	900.0	55.414	80.737	65.443	80.534
	q	900.0	900.0	900.0	900.0	24.929	81.643	22.110	23.427
	s	900.0	900.0	900.0	900.0	33.338	32.400	30.440	33.428
	g	900.0	900.0	900.0	900.0	74.107	67.907	28.248	50.438
	u	900.0	900.0	900.0	900.0	62.490	22.219	22.000	17.345
	w	900.0	900.0	900.0	900.0	21.374	69.536	23.036	23.113
dano3_3	d	575.3	641.8	600.5	529.5	0.063	0.060	0.067	0.065
	q	208.6	169.7	182.0	191.5	0.396	0.265	0.422	0.401
	s	109.6	108.5	95.6	116.3	1.762	1.139	2.335	1.547
	g	136.0	116.2	131.7	131.9	0.771	1.109	0.924	0.842
	u	145.4	113.0	126.7	137.0	0.726	1.010	1.180	1.018
	w	136.7	119.8	135.8	127.6	0.855	1.130	0.870	0.875

cont. on next page

Table D.2: LP pricing results for every tested instance, seed, and pricing method from the experiment in Section 7.2.

Instance	P	time				LPthpt			
		0	1	2	3	0	1	2	3
dano3_5	d	804.6	629.2	694.2	866.1	0.307	0.674	0.344	0.283
	q	445.9	504.2	900.0	683.1	0.782	0.781	0.814	0.765
	s	239.7	213.0	273.7	256.6	4.299	3.247	2.857	4.265
	g	277.7	293.7	286.0	320.1	2.686	2.488	2.632	2.229
	u	337.4	327.8	305.1	277.5	2.527	2.795	2.663	2.560
	w	304.5	282.8	302.1	276.4	2.329	2.699	2.782	2.751
enigma	d	0.5	0.5	0.5	0.5	781.000	631.000	518.000	477.000
	q	0.5	0.5	0.5	0.5	790.000	119.000	817.000	752.000
	s	0.5	0.5	0.5	0.5	434.000	662.000	662.000	434.000
	g	0.5	0.5	0.5	0.5	781.000	631.000	518.000	477.000
	u	0.5	0.5	0.5	0.5	612.000	902.000	219.000	580.000
	w	0.5	0.5	0.5	0.5	781.000	631.000	518.000	477.000
enlight14	d	0.5	0.5	0.5	0.5	1.000	1.000	1.000	1.000
	q	0.5	0.5	0.5	0.5	1.000	1.000	1.000	1.000
	s	0.5	0.5	0.5	0.5	1.000	1.000	1.000	1.000
	g	0.5	0.5	0.5	0.5	1.000	1.000	1.000	1.000
	u	0.5	0.5	0.5	0.5	1.000	1.000	1.000	1.000
	w	0.5	0.5	0.5	0.5	1.000	1.000	1.000	1.000
fast0507	d	198.3	225.6	184.6	225.6	22.817	21.862	17.956	20.936
	q	243.7	168.8	168.7	160.6	28.509	30.203	29.215	32.737
	s	122.6	152.4	148.7	144.7	54.444	53.600	55.053	53.027
	g	123.3	103.4	94.7	122.3	51.129	51.579	50.349	46.272
	u	164.5	118.6	135.5	146.7	42.083	37.173	38.802	40.951
	w	201.1	125.4	215.2	136.0	28.425	35.840	29.073	33.544
gesa3	d	4.0	4.0	4.0	4.0	132.000	132.000	132.000	132.000
	q	3.1	3.1	3.1	3.1	114.000	114.000	114.000	114.000
	s	2.5	2.6	2.6	2.6	133.000	133.000	133.000	133.000
	g	4.0	4.1	4.0	4.0	132.000	132.000	132.000	132.000
	u	4.0	4.0	4.0	4.1	132.000	132.000	132.000	132.000
	w	4.0	4.0	4.0	4.0	132.000	132.000	132.000	132.000
gmu-35-40	d	900.0	900.0	900.0	900.0	4957.472	5117.238	4973.098	4855.290
	q	900.0	900.0	900.0	900.0	4396.990	4325.870	4692.965	4336.761
	s	900.0	900.0	900.0	900.0	1792.424	1820.965	1782.291	1808.548
	g	900.0	900.0	900.0	900.0	4851.232	5191.881	4967.683	4735.952
	u	900.0	900.0	900.0	900.0	5234.216	4843.152	4735.445	4890.141
	w	900.0	900.0	900.0	900.0	4871.580	4980.356	4803.680	4876.100
khhb05250	d	0.5	0.5	0.5	0.5	43.000	43.000	43.000	43.000
	q	0.5	0.5	0.5	0.5	39.000	39.000	39.000	39.000
	s	0.5	0.5	0.5	0.5	35.000	35.000	35.000	35.000
	g	0.5	0.5	0.5	0.5	43.000	43.000	43.000	43.000
	u	0.5	0.5	0.5	0.5	43.000	43.000	43.000	43.000
	w	0.5	0.5	0.5	0.5	43.000	43.000	43.000	43.000
lectsched-4-obj	d	34.1	23.5	21.5	41.3	445.055	567.368	429.545	604.450
	q	29.6	11.6	13.4	14.6	332.069	20.000	19.375	20.000
	s	36.6	35.5	22.0	72.3	354.664	359.259	257.471	81.163
	g	27.7	27.2	24.4	35.3	342.174	599.228	474.771	531.845
	u	47.3	8.0	20.3	31.4	445.113	23.000	445.185	443.388
	w	60.7	21.1	32.9	12.6	215.534	480.303	471.512	21.000
m100n500k4r1	d	900.0	900.0	900.0	900.0	1354.100	1476.484	1418.780	1423.688
	q	900.0	900.0	900.0	900.0	1033.038	1052.989	1006.777	992.487
	s	900.0	900.0	900.0	900.0	971.066	962.994	993.292	972.630
	g	900.0	900.0	900.0	900.0	1415.771	1444.667	1407.778	1371.903

cont. on next page

D. Appendix D

Table D.2: LP pricing results for every tested instance, seed, and pricing method from the experiment in Section 7.2.

Instance	P	time				LPthpt			
		0	1	2	3	0	1	2	3
macrophage	u	900.0	900.0	900.0	900.0	1439.498	1444.715	1458.356	1449.160
	w	900.0	900.0	900.0	900.0	1109.017	1181.933	1129.757	1160.798
	d	200.6	200.5	200.2	200.5	173.980	173.303	174.541	174.321
	q	332.8	331.7	332.0	332.5	57.818	58.348	57.979	58.052
	s	246.8	247.1	246.4	247.4	94.651	95.915	94.975	95.105
	g	170.2	204.1	235.0	286.0	162.862	175.251	163.018	180.481
misc07	u	251.4	199.6	245.6	206.4	165.063	188.914	163.635	113.993
	w	329.0	320.1	298.6	309.6	121.521	140.335	144.938	147.225
	d	13.7	14.2	14.2	14.6	4479.765	4244.744	4155.145	4210.818
	q	15.1	14.4	13.9	15.0	3415.000	3812.658	3702.278	3511.751
	s	10.8	10.7	10.7	10.8	1678.019	1706.875	1650.151	1709.779
	g	13.9	14.2	8.0	14.1	3901.351	3847.135	4240.268	4295.280
mkc	u	10.3	11.1	10.5	8.0	4095.312	3895.139	4101.901	4048.214
	w	14.4	12.8	12.7	14.7	4393.036	4118.452	4478.317	3919.638
	d	900.0	900.0	900.0	900.0	820.557	672.545	474.280	320.387
	q	900.0	900.0	900.0	900.0	513.883	235.949	415.825	505.398
	s	900.0	900.0	900.0	900.0	163.630	139.575	198.332	141.679
	g	900.0	900.0	900.0	900.0	368.789	812.473	494.223	256.341
mod008	u	900.0	900.0	900.0	900.0	646.885	458.266	585.384	309.882
	w	900.0	900.0	900.0	900.0	383.976	415.484	456.326	348.011
	d	0.5	0.5	0.5	0.5	79.000	79.000	79.000	79.000
	q	0.5	0.5	0.5	0.5	61.000	61.000	61.000	61.000
	s	0.5	0.5	0.5	0.5	87.000	87.000	87.000	87.000
	g	0.5	0.5	0.5	0.5	79.000	79.000	79.000	79.000
mspp16	u	0.5	0.5	0.5	0.5	79.000	79.000	79.000	79.000
	w	0.5	0.5	0.5	0.5	79.000	79.000	79.000	79.000
	d	358.7	381.1	350.6	504.0	3.700	3.558	3.724	3.611
	q	432.6	471.4	631.3	542.7	3.339	3.128	3.083	3.079
	s	591.9	585.4	478.5	438.5	2.712	2.694	2.602	2.956
	g	356.4	379.5	352.6	502.6	3.591	3.551	3.466	3.619
n4-3	u	373.4	389.8	353.4	507.9	3.353	3.339	3.367	3.165
	w	358.6	380.6	350.8	497.5	3.433	3.571	3.556	3.493
	d	262.3	181.0	198.4	186.1	70.188	73.857	75.043	67.679
	q	204.1	296.3	296.4	271.8	55.096	42.721	42.514	46.260
	s	258.1	256.3	264.1	281.6	62.316	62.988	63.958	65.060
	g	292.3	207.6	232.7	206.9	60.031	60.679	59.695	71.321
nag	u	198.4	165.4	201.6	217.6	68.387	70.147	65.514	60.673
	w	186.6	180.6	170.6	187.1	68.040	64.525	67.900	66.602
	d	900.0	900.0	900.0	900.0	8.997	19.244	9.468	13.188
	q	900.0	900.0	900.0	900.0	20.089	30.552	20.142	22.371
	s	900.0	900.0	900.0	900.0	17.787	25.765	24.584	20.800
	g	900.0	900.0	900.0	900.0	9.821	27.463	24.979	28.266
neos-1109824	u	900.0	900.0	900.0	900.0	18.922	24.049	27.673	31.339
	w	900.0	900.0	900.0	900.0	12.912	20.503	22.031	18.800
	d	11.4	11.4	11.4	11.3	327.000	327.000	327.000	330.000
	q	53.2	21.3	14.6	28.7	418.059	480.952	449.000	521.008
	s	22.7	22.6	22.6	22.3	412.376	420.283	429.381	428.365
	g	11.4	11.4	11.4	11.4	327.000	327.000	327.000	330.000
neos-1121679	u	14.6	11.7	16.2	14.5	566.000	415.000	558.000	459.000
	w	11.4	11.4	11.4	11.4	327.000	327.000	327.000	330.000
	d	900.0	900.0	900.0	900.0	40503.819	43426.849	44849.477	47961.462
	q	900.0	900.0	900.0	900.0	48650.960	47732.673	43624.325	45397.926

cont. on next page

Table D.2: LP pricing results for every tested instance, seed, and pricing method from the experiment in Section 7.2.

Instance	P	time				LPthpt			
		0	1	2	3	0	1	2	3
neos-1200887	s	900.0	900.0	900.0	900.0	35063.374	32543.903	35453.998	32472.430
	g	900.0	900.0	900.0	900.0	40640.170	42960.902	43832.496	47500.216
	u	900.0	900.0	900.0	900.0	41015.926	40756.995	48113.837	46944.355
	w	900.0	900.0	900.0	900.0	41004.903	43732.617	44336.416	47854.721
	d	8.0	12.5	9.3	13.1	2102.030	2045.566	1935.545	1876.061
	q	26.4	13.6	17.0	13.3	658.871	647.135	639.187	589.316
	s	15.8	10.4	15.9	17.2	1000.460	885.678	995.398	942.756
	g	7.2	11.4	13.0	11.5	2018.889	1901.286	2133.618	2024.335
neos-1208135	u	9.3	9.6	15.8	20.1	2135.784	1847.525	2130.512	1158.499
	w	9.0	14.7	14.1	21.4	1373.050	1450.797	1824.045	1246.951
	d	316.7	112.8	219.0	280.4	29.622	18.307	11.232	14.378
	q	181.6	306.8	153.5	314.5	41.041	51.928	51.888	33.805
	s	96.2	140.9	62.8	110.1	105.838	97.942	68.750	90.397
	g	199.3	168.8	169.0	137.7	50.594	31.172	16.549	97.741
	u	427.1	189.8	406.4	128.1	25.878	38.418	27.066	39.249
	w	128.4	96.3	203.0	288.9	73.620	92.787	28.902	45.782
neos-1211578	d	5.8	5.8	5.8	5.8	10504.348	10778.947	10343.885	9511.842
	q	3.5	3.5	3.5	3.6	4180.000	4570.000	4290.000	4607.000
	s	4.3	3.8	2.2	2.1	4696.324	4202.439	2967.000	2967.000
	g	4.6	4.5	4.6	4.6	7712.821	8355.556	8057.143	7712.821
	u	5.3	5.1	4.9	5.1	8441.007	7005.442	7717.857	8954.074
	w	5.2	5.3	5.6	5.1	8783.846	8494.964	8611.111	7950.694
	d	34.4	87.1	45.1	81.9	175.796	234.231	149.868	209.923
	q	74.7	68.6	86.3	93.1	125.655	130.243	115.306	144.237
neos-1215259	s	60.3	56.6	44.3	73.1	109.423	123.399	111.594	104.384
	g	52.5	104.6	80.1	74.5	135.493	116.500	210.729	148.120
	u	55.2	47.4	145.8	40.5	138.149	174.620	193.459	153.507
	w	35.9	43.1	75.2	54.9	152.980	131.443	158.954	136.264
	d	900.0	900.0	900.0	900.0	4806.782	5129.140	5018.683	4413.842
	q	900.0	900.0	900.0	900.0	2575.227	2640.614	2640.565	2236.559
	s	900.0	900.0	900.0	900.0	2973.421	2813.680	2612.887	2889.603
	g	900.0	900.0	900.0	900.0	4350.394	4305.212	4764.726	5081.017
neos-1346382	u	900.0	900.0	900.0	900.0	4605.106	5281.544	4336.167	5169.299
	w	900.0	900.0	900.0	900.0	4590.540	4645.492	4789.903	4696.517
	d	900.0	900.0	900.0	900.0	0.010	0.037	0.016	0.020
	q	900.0	900.0	900.0	900.0	0.013	0.035	0.019	0.021
	s	900.0	900.0	900.0	900.0	0.040	0.086	0.031	0.055
	g	900.0	900.0	900.0	900.0	0.008	0.044	0.019	0.079
	u	900.0	900.0	900.0	900.0	0.008	0.018	0.020	0.021
	w	900.0	900.0	900.0	900.0	0.008	0.043	0.019	0.014
neos-1354092	d	81.2	112.8	156.7	111.2	479.124	287.480	114.974	313.672
	q	214.9	436.5	313.3	256.6	105.252	101.866	165.177	125.753
	s	213.3	385.7	232.5	133.4	125.778	130.756	212.115	189.280
	g	75.1	63.3	99.8	88.2	763.894	482.770	308.508	569.730
	u	84.6	58.7	189.9	211.2	405.363	530.539	164.979	102.820
	w	132.0	229.8	182.2	102.1	196.664	102.792	198.023	394.209
	d	900.0	900.0	900.0	900.0	28.021	25.730	32.952	38.685
	q	900.0	900.0	900.0	900.0	23.588	29.395	45.820	40.289
neos-1396125	s	900.0	900.0	900.0	900.0	62.108	29.868	23.307	34.735
	g	900.0	900.0	900.0	900.0	27.500	26.432	29.703	15.431
	u	900.0	900.0	900.0	900.0	40.039	34.041	37.078	20.928
	w	900.0	900.0	900.0	900.0	27.193	26.114	29.775	12.988
	d	900.0	900.0	900.0	900.0	28.021	25.730	32.952	38.685
	q	900.0	900.0	900.0	900.0	23.588	29.395	45.820	40.289
	s	900.0	900.0	900.0	900.0	62.108	29.868	23.307	34.735
	g	900.0	900.0	900.0	900.0	27.500	26.432	29.703	15.431
neos-1423785	u	900.0	900.0	900.0	900.0	40.039	34.041	37.078	20.928
	w	900.0	900.0	900.0	900.0	27.193	26.114	29.775	12.988

cont. on next page

D. Appendix D

Table D.2: LP pricing results for every tested instance, seed, and pricing method from the experiment in Section 7.2.

Instance	P	time				LPthpt			
		0	1	2	3	0	1	2	3
neos-1426662	d	900.0	900.0	900.0	900.0	2070.145	1777.964	1590.222	2016.281
	q	900.0	900.0	900.0	900.0	833.052	598.923	728.032	859.918
	s	900.0	900.0	900.0	900.0	922.483	541.844	659.671	1069.955
	g	900.0	900.0	900.0	900.0	1999.173	1705.709	1733.105	1891.412
	u	900.0	900.0	900.0	900.0	1803.051	1851.213	1778.505	1784.021
	w	900.0	900.0	900.0	900.0	1021.541	1796.783	1495.989	1772.495
neos-1427261	d	900.0	900.0	900.0	900.0	282.888	355.196	382.616	424.627
	q	900.0	900.0	900.0	900.0	291.592	284.075	203.952	394.548
	s	900.0	900.0	900.0	900.0	204.745	264.489	247.766	251.151
	g	900.0	900.0	900.0	900.0	399.492	270.395	346.659	304.449
	u	900.0	900.0	900.0	900.0	392.847	341.320	381.315	413.005
	w	900.0	900.0	900.0	900.0	198.909	304.426	264.728	247.440
neos-1437164	d	0.5	0.5	0.5	0.5	1.000	1.000	1.000	1.000
	q	0.5	0.5	0.5	0.5	1.000	1.000	1.000	1.000
	s	0.5	0.5	0.5	0.5	1.000	1.000	1.000	1.000
	g	0.5	0.5	0.5	0.5	1.000	1.000	1.000	1.000
	u	0.5	0.5	0.5	0.5	1.000	1.000	1.000	1.000
	w	0.5	0.5	0.5	0.5	1.000	1.000	1.000	1.000
neos-1440460	d	900.0	900.0	900.0	900.0	2355.926	2292.097	2412.208	3358.013
	q	900.0	900.0	900.0	900.0	670.297	599.011	721.037	657.970
	s	900.0	900.0	900.0	900.0	972.577	907.709	996.631	1051.829
	g	900.0	900.0	900.0	900.0	2255.526	2423.529	2259.589	2303.630
	u	900.0	900.0	900.0	900.0	2235.554	2405.995	2372.504	2244.451
	w	900.0	900.0	900.0	900.0	1291.701	1949.901	2657.026	2589.547
neos-1441553	d	4.0	3.4	1.6	2.7	60.000	44.000	13.000	59.000
	q	2.6	2.4	2.0	2.6	31.000	29.000	14.000	29.000
	s	1.8	1.9	2.0	1.8	13.000	13.000	13.000	13.000
	g	4.0	3.4	1.6	2.6	60.000	44.000	13.000	72.000
	u	3.9	3.4	1.6	2.6	63.000	44.000	13.000	62.000
	w	4.0	3.4	1.6	2.6	63.000	44.000	13.000	72.000
neos-1445755	d	56.8	56.3	56.3	56.3	65.466	65.053	65.745	66.595
	q	51.7	51.9	52.0	51.9	61.522	61.255	59.958	60.991
	s	53.7	53.5	53.5	53.4	42.534	41.654	41.778	41.778
	g	56.8	56.5	57.0	56.7	65.885	65.605	66.739	65.328
	u	57.2	57.0	57.1	57.4	66.883	64.509	66.309	66.167
	w	57.6	57.3	56.9	57.2	65.745	66.595	66.167	65.885
neos-1451294	d	900.0	900.0	900.0	900.0	9.444	3.299	4.403	4.276
	q	900.0	900.0	900.0	900.0	8.285	8.229	12.518	8.968
	s	729.4	900.0	900.0	605.3	18.417	22.086	32.852	20.011
	g	900.0	900.0	900.0	900.0	28.648	24.185	27.060	26.788
	u	378.7	900.0	900.0	900.0	3.559	4.211	21.426	8.309
	w	605.2	900.0	900.0	900.0	6.707	8.179	15.942	7.556
neos-1460543	d	900.0	900.0	900.0	900.0	2.458	2.296	2.496	1.885
	q	900.0	900.0	900.0	900.0	1.679	3.202	1.613	1.535
	s	900.0	900.0	900.0	900.0	2.619	2.706	2.822	2.739
	g	900.0	900.0	900.0	900.0	2.046	2.100	2.413	2.074
	u	900.0	900.0	900.0	900.0	1.622	6.415	4.410	3.031
	w	900.0	900.0	900.0	900.0	2.904	2.514	2.097	3.049
neos-1461051	d	38.5	35.2	34.3	34.1	795.168	830.207	792.931	824.631
	q	34.2	35.4	28.9	44.1	277.847	337.652	322.667	382.118
	s	37.9	42.3	45.2	25.8	445.900	453.740	503.525	432.353
	g	36.6	32.0	34.3	31.4	800.596	806.329	851.208	779.617

cont. on next page

Table D.2: LP pricing results for every tested instance, seed, and pricing method from the experiment in Section 7.2.

Instance	P	time				LPthpt			
		0	1	2	3	0	1	2	3
neos-1467467	u	35.8	43.4	31.5	36.8	714.570	691.525	643.075	758.913
	w	40.7	42.3	38.1	38.2	573.388	571.414	605.011	607.380
	d	900.0	900.0	900.0	900.0	24.860	21.612	49.605	16.639
	q	900.0	900.0	900.0	900.0	12.960	15.793	17.322	22.622
	s	900.0	900.0	900.0	900.0	29.062	40.314	28.997	21.847
	g	900.0	900.0	900.0	900.0	15.579	31.152	32.141	10.119
	u	900.0	900.0	900.0	900.0	16.495	22.170	17.428	15.610
neos-1480121	w	900.0	900.0	900.0	900.0	26.001	28.308	18.951	24.485
	d	13.2	3.5	1.0	24.5	739.000	9639.000	1144.000	793.000
	q	1.1	1.0	7.6	17.1	1335.000	927.000	843.000	1460.000
	s	10.6	11.8	13.1	1.4	956.000	728.000	1075.000	974.000
	g	14.7	0.5	0.9	24.5	6666.355	536.000	900.000	814.000
	u	13.8	0.5	5.5	24.7	4133.000	213.000	12433.594	1001.000
	w	13.2	0.5	1.2	24.7	1628.000	731.000	1212.000	842.000
neos-1595230	d	51.5	205.3	121.8	127.9	497.929	365.620	569.562	559.069
	q	900.0	900.0	118.2	300.3	261.400	157.614	122.322	88.502
	s	236.7	85.0	152.4	248.9	231.259	244.870	256.489	149.807
	g	60.0	90.2	157.2	88.0	522.883	458.828	658.616	529.815
	u	58.3	100.4	149.6	69.5	586.770	417.855	615.374	598.584
	w	73.2	123.7	252.3	113.5	341.075	272.816	445.843	361.392
	d	1.3	1.8	1.6	1.8	34.000	42.000	38.000	33.000
neos-522351	q	2.2	1.3	1.8	1.5	37.000	20.000	33.000	30.000
	s	2.6	2.9	3.0	1.1	34.000	42.000	40.000	19.000
	g	1.3	1.7	1.6	1.7	34.000	42.000	38.000	33.000
	u	1.3	1.7	1.6	1.8	34.000	42.000	38.000	33.000
	w	1.3	1.8	1.6	1.8	34.000	42.000	38.000	33.000
	d	19.5	19.4	19.3	19.4	121.069	120.312	122.611	122.222
	q	27.9	26.5	26.2	22.4	150.000	158.667	155.217	159.000
neos-547911	s	21.4	21.4	21.4	21.3	253.333	285.000	255.224	249.635
	g	22.4	17.0	25.3	19.6	274.272	297.000	256.923	354.206
	u	16.6	26.3	27.3	33.0	131.000	167.495	154.651	158.728
	w	18.2	18.3	18.3	18.1	221.951	233.333	220.161	220.161
	d	75.2	58.9	54.4	56.4	304.229	324.000	153.107	159.509
	q	71.6	95.4	65.6	91.9	166.346	229.055	162.951	267.759
	s	494.3	310.8	349.8	665.6	57.805	45.047	37.519	37.416
neos-555298	g	61.5	73.2	54.2	62.1	214.159	324.876	154.857	246.178
	u	63.9	59.7	63.8	59.8	205.859	225.229	212.400	272.455
	w	64.8	79.5	54.6	75.6	234.194	292.203	153.977	288.330
	d	3.1	2.3	8.1	2.7	45.000	37.000	105.000	38.000
	q	12.1	3.2	4.4	3.6	99.000	30.000	44.000	54.000
	s	6.6	8.3	6.3	4.8	64.000	70.000	65.000	37.000
	g	3.2	2.3	9.0	2.7	45.000	37.000	88.000	38.000
neos-555771	u	3.3	2.3	8.5	2.7	45.000	37.000	101.000	38.000
	w	3.3	2.3	7.7	2.7	45.000	37.000	86.000	38.000
	d	900.0	900.0	900.0	900.0	57.949	111.382	72.739	60.007
	q	900.0	900.0	900.0	900.0	50.697	46.992	43.646	41.174
	s	900.0	900.0	900.0	900.0	50.493	39.694	44.628	53.171
	g	900.0	900.0	900.0	900.0	68.451	82.694	67.083	50.593
	u	900.0	900.0	900.0	900.0	52.217	77.317	88.874	51.036
neos-555884	w	900.0	900.0	900.0	900.0	44.021	69.394	70.947	57.822
	d	172.8	172.5	169.4	172.0	1.000	1.000	1.000	1.000
	q	172.8	172.2	172.5	172.0	1.000	1.000	1.000	1.000

cont. on next page

D. Appendix D

Table D.2: LP pricing results for every tested instance, seed, and pricing method from the experiment in Section 7.2.

Instance	P	time				LPthpt			
		0	1	2	3	0	1	2	3
neos-585192	s	172.3	172.6	172.2	169.2	1.000	1.000	1.000	1.000
	g	171.5	173.2	173.7	173.4	1.000	1.000	1.000	1.000
	u	172.6	172.9	172.3	172.0	1.000	1.000	1.000	1.000
	w	172.3	173.6	169.5	172.2	1.000	1.000	1.000	1.000
	d	16.3	16.4	16.3	16.3	945.665	1097.987	1028.931	1003.681
	q	17.0	17.2	19.1	17.4	967.901	872.973	885.714	958.621
	s	32.5	32.6	32.5	32.7	170.895	174.150	167.979	169.089
	g	16.5	16.7	15.3	16.8	1009.877	1039.375	1176.271	966.860
neos-595904	u	15.3	17.2	15.4	18.7	1007.586	895.588	944.000	977.000
	w	15.3	15.3	15.3	15.3	769.620	856.338	821.622	794.771
	d	11.0	11.0	10.9	11.1	96.000	96.000	96.000	96.000
	q	19.2	19.0	19.1	19.2	75.281	76.136	77.907	80.240
	s	15.0	14.9	15.0	14.9	105.844	101.242	100.617	103.822
	g	11.1	11.0	11.1	11.1	96.000	96.000	96.000	96.000
	u	10.9	10.9	11.0	11.0	96.000	96.000	96.000	96.000
	w	10.9	10.9	10.9	10.9	96.000	96.000	96.000	96.000
neos-595925	d	9.2	9.1	9.1	9.1	210.000	210.000	210.000	210.000
	q	24.9	25.1	25.1	25.1	159.218	148.438	147.668	145.408
	s	21.4	21.4	21.3	21.4	173.786	170.476	172.115	185.492
	g	9.1	9.1	9.1	9.1	210.000	210.000	210.000	210.000
	u	9.1	9.1	9.1	9.1	210.000	210.000	210.000	210.000
	w	9.1	9.1	9.1	9.1	210.000	210.000	210.000	210.000
	d	5.0	5.0	5.1	5.1	133.000	133.000	133.000	133.000
	q	7.1	7.0	7.2	5.8	168.000	159.000	179.000	145.000
neos-598183	s	6.7	6.7	6.7	6.7	177.000	177.000	177.000	177.000
	g	5.1	5.1	5.1	5.1	133.000	133.000	133.000	133.000
	u	4.9	5.1	6.7	6.8	112.000	139.000	188.000	207.000
	w	5.1	5.1	5.1	5.1	133.000	133.000	133.000	133.000
	d	95.6	94.6	123.5	96.2	303.071	307.264	362.171	309.267
	q	85.5	126.3	99.6	110.6	214.449	258.581	241.304	255.463
	s	254.4	900.0	900.0	220.5	344.801	434.353	422.650	351.205
	g	155.5	900.0	900.0	74.6	437.833	555.799	620.531	366.135
neos-603073	u	900.0	154.2	80.3	900.0	733.273	458.701	296.176	560.516
	w	900.0	900.0	900.0	246.4	520.833	451.044	549.142	322.889
	d	16.8	16.8	16.7	16.7	302.632	294.872	288.945	291.878
	q	24.4	25.4	24.7	25.4	181.095	179.469	182.447	185.028
	s	26.4	26.1	26.2	26.1	79.427	79.343	79.008	80.193
	g	16.4	16.6	17.7	18.4	276.562	278.740	282.609	283.161
	u	17.7	19.3	18.6	19.4	271.852	263.196	252.632	259.179
	w	23.5	21.9	22.0	23.5	202.305	209.315	211.170	204.480
neos-612162	d	900.0	900.0	900.0	900.0	193.034	283.865	232.537	161.725
	q	900.0	900.0	900.0	900.0	183.926	153.697	164.938	249.458
	s	900.0	900.0	900.0	900.0	168.039	170.057	185.994	194.218
	g	900.0	900.0	900.0	900.0	234.724	232.544	208.382	238.271
	u	900.0	900.0	900.0	900.0	226.462	240.406	264.352	193.520
	w	900.0	900.0	900.0	900.0	202.895	205.003	226.754	208.135
	d	900.0	900.0	900.0	900.0	0.001	0.001	0.001	0.001
	q	900.0	900.0	900.0	900.0	0.001	0.001	0.002	0.001
neos-631694	s	900.0	900.0	900.0	900.0	0.001	0.001	0.001	0.001
	g	900.0	900.0	900.0	900.0	0.001	0.001	0.001	0.001
	u	900.0	900.0	900.0	900.0	0.001	0.001	0.001	0.001
	w	900.0	900.0	900.0	900.0	0.001	0.001	0.001	0.001
	d	900.0	900.0	900.0	900.0	0.001	0.001	0.001	0.001
	q	900.0	900.0	900.0	900.0	0.001	0.001	0.001	0.001
	s	900.0	900.0	900.0	900.0	0.001	0.001	0.001	0.001
	g	900.0	900.0	900.0	900.0	0.001	0.001	0.001	0.001
neos-631710	u	900.0	900.0	900.0	900.0	0.001	0.001	0.001	0.001
	w	900.0	900.0	900.0	900.0	0.001	0.001	0.001	0.001
	d	900.0	900.0	900.0	900.0	0.001	0.001	0.001	0.001
	q	900.0	900.0	900.0	900.0	0.001	0.001	0.001	0.001
	s	900.0	900.0	900.0	900.0	0.001	0.001	0.001	0.001
	g	900.0	900.0	900.0	900.0	0.001	0.001	0.001	0.001
	u	900.0	900.0	900.0	900.0	0.001	0.001	0.001	0.001
	w	900.0	900.0	900.0	900.0	0.001	0.001	0.001	0.001

cont. on next page

Table D.2: LP pricing results for every tested instance, seed, and pricing method from the experiment in Section 7.2.

Instance	P	time				LPthpt			
		0	1	2	3	0	1	2	3
neos-686190	d	97.2	96.8	96.7	97.2	493.897	493.897	493.432	494.675
	q	83.1	96.0	143.2	125.8	403.563	435.837	404.192	428.286
	s	411.7	498.0	156.3	610.9	136.759	147.133	122.868	132.091
	g	75.7	86.7	84.7	89.6	509.108	505.475	522.610	489.540
	u	82.2	143.5	67.8	132.2	512.571	512.401	503.409	475.985
	w	109.8	100.0	92.9	92.1	463.514	413.791	459.636	461.516
neos-691073	d	900.0	900.0	900.0	900.0	3.376	3.196	1.919	3.056
	q	900.0	900.0	900.0	900.0	0.123	0.118	0.141	0.113
	s	900.0	900.0	900.0	900.0	0.728	0.835	0.859	0.921
	g	900.0	900.0	900.0	900.0	2.343	2.857	2.049	1.601
	u	900.0	900.0	900.0	900.0	3.337	4.875	2.125	2.972
	w	900.0	900.0	900.0	900.0	1.478	2.365	1.502	1.388
neos-717614	d	6.4	6.5	6.4	6.5	2411.000	2411.000	2411.000	2411.000
	q	8.0	8.0	8.1	8.1	1090.000	1090.000	1090.000	1090.000
	s	12.4	12.4	12.4	12.2	392.929	413.830	419.784	424.364
	g	40.9	41.0	41.2	47.8	2704.805	2725.759	2720.374	2730.295
	u	4.3	7.2	7.2	199.6	804.000	2703.731	2308.725	2745.518
	w	8.5	8.5	8.5	8.5	2252.976	2610.345	2557.432	2441.935
neos-791021	d	900.0	900.0	900.0	900.0	0.063	0.054	0.055	0.073
	q	308.1	62.9	76.5	82.7	0.692	0.486	0.465	0.499
	s	34.8	28.9	26.9	40.3	1.248	1.259	1.461	1.762
	g	250.4	310.1	289.4	298.0	1.355	1.177	1.019	0.940
	u	900.0	665.3	900.0	900.0	0.243	0.255	0.139	0.186
	w	452.1	495.9	401.0	658.3	0.472	0.700	0.525	0.555
neos-803219	d	37.0	42.1	34.5	29.8	1554.326	1637.955	1596.640	1512.301
	q	38.6	27.4	40.4	42.0	942.002	936.317	1041.804	1030.062
	s	42.0	41.9	45.3	37.0	1019.961	1006.568	978.178	1044.681
	g	36.7	42.0	34.3	29.4	1595.604	1637.955	1593.822	1533.441
	u	36.5	30.3	30.9	33.5	1458.618	1472.386	1545.419	1497.676
	w	37.0	42.1	34.4	29.6	1537.864	1677.905	1640.145	1602.360
neos-803220	d	81.5	76.8	72.5	79.6	2771.021	2901.084	2811.629	2740.566
	q	92.6	87.7	108.9	81.4	1756.395	1764.438	1682.540	2105.215
	s	103.4	97.3	98.5	98.7	1753.464	1754.584	1787.896	1716.346
	g	81.9	77.7	72.7	82.6	2763.752	2898.351	2789.727	2706.704
	u	77.7	75.3	76.5	97.4	2738.143	2809.409	2710.878	2702.465
	w	82.0	77.1	72.9	84.2	2752.918	2843.438	2765.492	2744.632
neos-807454	d	2.1	1.9	1.8	2.3	4.000	5.000	5.000	4.000
	q	3.3	3.5	3.2	2.2	4.000	4.902	5.000	4.000
	s	4.7	4.2	3.4	2.4	4.000	4.000	5.000	4.000
	g	2.1	1.9	1.8	2.3	4.000	5.000	5.000	4.000
	u	2.1	1.9	1.8	2.3	4.000	5.000	5.000	4.000
	w	2.1	1.9	1.8	2.3	4.000	5.000	5.000	4.000
neos-808072	d	22.9	18.8	22.8	18.0	93.074	62.059	97.030	80.786
	q	25.5	27.4	23.4	37.1	37.225	74.960	63.287	71.250
	s	36.5	32.4	39.8	31.6	60.000	54.791	65.738	55.313
	g	26.3	20.8	23.0	22.0	64.378	67.246	102.148	56.500
	u	20.0	30.5	29.8	21.9	86.799	94.422	66.007	76.803
	w	21.4	21.0	26.3	25.8	90.411	62.784	87.634	72.080
neos-820146	d	900.0	900.0	900.0	900.0	1200.173	1340.065	1475.637	1278.118
	q	900.0	900.0	900.0	900.0	479.881	412.138	338.140	430.549
	s	900.0	900.0	900.0	900.0	524.444	531.321	509.587	546.731
	g	900.0	900.0	900.0	900.0	1375.069	1347.657	1483.491	1517.417

cont. on next page

D. Appendix D

Table D.2: LP pricing results for every tested instance, seed, and pricing method from the experiment in Section 7.2.

Instance	P	time				LPthpt			
		0	1	2	3	0	1	2	3
neos-825075	u	900.0	900.0	900.0	900.0	1175.832	1316.960	1314.416	1282.637
	w	900.0	900.0	900.0	900.0	905.074	814.951	889.472	885.397
	d	1.2	1.4	1.4	1.9	14.000	19.000	33.000	30.000
	q	2.3	2.0	1.4	1.8	22.000	32.000	10.000	33.000
	s	1.5	1.5	1.8	2.1	20.000	18.000	26.000	46.000
	g	1.2	1.4	1.4	1.9	14.000	19.000	30.000	30.000
	u	1.4	1.4	1.4	1.9	14.000	28.000	29.000	30.000
neos-826650	w	1.2	1.4	1.5	1.9	14.000	19.000	38.000	30.000
	d	900.0	900.0	900.0	900.0	1.410	2.381	1.282	1.624
	q	900.0	900.0	900.0	900.0	4.532	7.315	4.930	9.240
	s	900.0	900.0	900.0	900.0	18.415	23.573	19.578	15.642
	g	900.0	900.0	900.0	900.0	19.221	21.803	20.262	20.733
	u	900.0	900.0	900.0	900.0	0.687	0.762	2.770	1.676
	w	900.0	900.0	900.0	900.0	4.557	4.262	4.093	2.082
neos-827015	d	900.0	900.0	900.0	900.0	0.832	0.457	0.632	0.634
	q	900.0	900.0	900.0	900.0	2.000	1.864	1.914	2.067
	s	900.0	900.0	900.0	900.0	3.060	2.532	2.922	2.532
	g	731.0	741.7	802.5	900.0	1.920	1.698	1.643	0.527
	u	676.9	900.0	900.0	900.0	2.073	2.284	1.701	1.793
	w	900.0	900.0	900.0	900.0	1.868	2.001	1.147	1.754
	d	11.4	14.8	14.1	12.2	47.000	169.600	165.000	57.000
neos-848150	q	16.4	18.3	21.1	17.8	33.884	38.168	66.190	84.293
	s	18.0	21.1	25.3	17.6	75.000	82.301	132.618	55.000
	g	12.6	16.7	17.0	12.4	48.000	124.378	199.398	50.000
	u	15.8	15.5	16.0	15.1	147.799	39.344	51.000	116.260
	w	15.2	16.1	14.7	15.1	84.158	71.569	49.505	37.000
	d	2.8	2.6	3.1	2.4	10.000	15.686	10.606	16.000
	q	3.4	2.6	11.5	2.6	22.000	15.000	35.000	18.000
neos-850681	s	17.7	30.5	2.4	9.7	33.858	43.605	16.000	31.000
	g	2.8	2.6	3.2	2.4	9.677	15.686	10.448	16.000
	u	2.8	2.6	3.2	2.4	9.917	15.842	10.294	16.000
	w	2.8	2.6	3.1	2.4	10.000	16.000	10.294	16.000
	d	0.7	1.4	0.8	1.2	127.000	174.000	129.000	169.000
	q	1.0	0.9	1.0	1.2	160.000	132.000	125.000	175.000
	s	1.4	1.4	1.4	1.7	166.000	176.000	173.000	185.000
neos-880324	g	0.6	1.4	0.8	1.2	127.000	174.000	129.000	169.000
	u	0.7	1.4	0.8	1.3	135.000	188.000	129.000	193.000
	w	0.7	1.4	0.8	1.2	127.000	174.000	129.000	169.000
	d	900.0	139.2	900.0	900.0	0.059	2.444	0.094	0.065
	q	900.0	900.0	900.0	696.2	0.205	0.027	0.047	1.480
	s	900.0	900.0	900.0	900.0	0.336	0.369	0.423	0.373
	g	900.0	900.0	900.0	900.0	0.183	2.210	0.162	0.205
neos-885086	u	900.0	138.7	900.0	900.0	0.082	2.459	0.081	0.080
	w	900.0	900.0	900.0	900.0	0.130	0.156	0.260	0.122
	d	74.3	170.4	79.6	77.6	22.350	7.585	21.216	22.399
	q	127.1	418.9	138.2	193.8	16.333	5.851	10.332	12.536
	s	44.6	43.0	64.9	40.1	50.414	43.019	38.648	56.081
	g	62.5	280.3	57.8	84.2	34.596	8.507	34.235	28.529
	u	117.1	167.2	133.2	900.0	15.552	12.495	11.608	13.577
neos-905856	w	70.5	226.9	136.8	128.0	30.003	7.681	13.821	16.324
	d	175.7	10.8	632.1	310.2	205.616	68.000	162.907	183.593
	q	900.0	535.5	64.9	189.7	71.534	68.995	79.381	134.994

cont. on next page

Table D.2: LP pricing results for every tested instance, seed, and pricing method from the experiment in Section 7.2.

Instance	P	time				LPthpt			
		0	1	2	3	0	1	2	3
neos-912015	s	303.6	392.5	559.0	117.0	133.597	234.365	158.411	165.356
	g	186.8	57.5	402.5	264.3	189.721	165.786	222.647	217.985
	u	244.8	20.2	88.5	73.3	220.540	166.165	188.405	116.952
	w	192.4	66.8	182.9	541.5	175.281	152.415	150.450	138.977
	d	11.4	9.0	7.1	13.9	309.155	190.000	123.000	371.134
	q	8.3	15.3	16.7	12.3	36.000	160.252	158.537	216.575
	s	12.2	8.0	12.8	9.3	159.000	84.000	168.000	145.000
	g	9.3	9.6	6.0	5.1	290.000	327.000	76.000	33.000
neos-912023	u	11.4	11.3	10.1	6.6	348.503	342.017	267.647	54.000
	w	12.1	5.2	11.5	9.5	233.158	43.000	192.623	97.000
	d	4.7	9.7	7.3	6.9	23.000	236.000	77.000	99.000
	q	10.4	7.7	6.8	10.3	77.000	31.000	22.000	22.000
	s	6.2	14.9	7.1	5.7	23.000	53.000	25.000	20.000
	g	5.2	10.7	8.1	10.1	23.000	397.030	116.000	220.000
	u	4.8	9.9	10.7	6.3	23.000	117.000	244.000	32.000
	w	5.1	11.7	8.9	7.5	23.000	206.618	111.000	71.000
neos-930752	d	900.0	900.0	900.0	900.0	0.042	0.063	0.069	0.054
	q	900.0	900.0	900.0	900.0	1.080	1.414	1.170	1.654
	s	900.0	900.0	900.0	900.0	3.447	3.614	3.859	4.244
	g	900.0	900.0	900.0	900.0	1.613	2.375	2.045	1.800
	u	900.0	900.0	900.0	900.0	0.210	0.256	0.161	0.135
	w	900.0	900.0	900.0	900.0	0.721	0.319	0.153	0.358
	d	900.0	900.0	900.0	900.0	0.666	0.754	0.834	0.883
	q	900.0	900.0	900.0	900.0	0.450	0.661	0.528	0.523
neos-931517	s	900.0	900.0	900.0	900.0	1.542	1.590	1.389	1.609
	g	900.0	900.0	900.0	900.0	0.811	0.660	1.065	0.904
	u	900.0	900.0	900.0	900.0	0.827	0.615	0.906	0.937
	w	900.0	900.0	900.0	900.0	1.248	0.731	1.037	1.060
	d	38.1	34.6	24.4	30.4	2134.055	1269.845	985.203	1128.291
	q	52.1	48.0	185.5	37.7	1156.879	915.906	1652.640	1330.435
	s	58.9	48.2	37.9	41.0	599.461	600.460	457.606	507.455
	g	20.5	28.2	29.2	39.3	1287.197	1069.605	1686.247	1300.957
neos-933815	u	34.0	113.7	31.6	35.2	1416.495	1936.163	1096.829	1136.715
	w	24.2	34.9	41.0	26.4	791.304	1119.696	1139.925	1024.032
	d	564.3	27.0	35.6	33.2	3447.969	2260.145	2139.712	1808.087
	q	62.9	900.0	75.9	77.5	2474.940	2623.024	2807.937	2634.489
	s	47.1	65.0	33.8	35.6	720.135	851.177	634.921	731.229
	g	470.3	36.4	56.4	41.7	2813.633	2666.760	3165.503	2752.853
	u	428.5	39.5	31.9	18.7	2749.309	2146.325	2755.150	1332.865
	w	608.7	25.6	48.4	30.9	3054.541	2133.871	3046.770	1969.260
neos-935496	d	900.0	900.0	900.0	900.0	370.102	319.893	287.201	391.776
	q	900.0	900.0	900.0	900.0	110.251	136.301	231.008	214.681
	s	900.0	900.0	900.0	900.0	182.856	137.957	165.705	175.276
	g	900.0	900.0	900.0	900.0	370.773	158.636	156.460	305.160
	u	900.0	900.0	900.0	900.0	278.231	302.628	227.144	292.852
	w	900.0	900.0	900.0	900.0	293.130	185.492	222.155	203.056
	d	900.0	900.0	900.0	900.0	243.108	261.205	226.507	308.857
	q	900.0	900.0	900.0	900.0	227.703	142.471	152.902	161.751
neos-935674	s	900.0	900.0	900.0	900.0	143.850	153.672	155.693	147.455
	g	900.0	900.0	900.0	900.0	334.405	382.729	308.491	286.157
	u	900.0	900.0	900.0	900.0	331.086	345.513	236.860	270.209
	w	900.0	900.0	900.0	900.0	183.610	229.557	239.872	156.314

cont. on next page

D. Appendix D

Table D.2: LP pricing results for every tested instance, seed, and pricing method from the experiment in Section 7.2.

Instance	P	time				LPthpt			
		0	1	2	3	0	1	2	3
neos-937815	d	900.0	900.0	900.0	900.0	0.019	0.014	0.016	0.017
	q	900.0	900.0	900.0	900.0	0.276	0.296	0.268	0.205
	s	900.0	900.0	900.0	900.0	0.310	0.551	0.399	0.432
	g	900.0	900.0	900.0	900.0	0.231	0.144	0.279	0.135
	u	900.0	900.0	900.0	900.0	0.059	0.188	0.095	0.125
	w	900.0	900.0	900.0	900.0	0.218	0.099	0.145	0.106
neos-948126	d	900.0	900.0	900.0	900.0	0.017	0.016	0.015	0.015
	q	900.0	900.0	900.0	900.0	0.227	0.194	0.204	0.134
	s	900.0	900.0	900.0	900.0	0.460	0.613	0.453	0.478
	g	900.0	900.0	900.0	900.0	0.171	0.093	0.120	0.153
	u	900.0	900.0	900.0	900.0	0.072	0.079	0.082	0.070
	w	900.0	900.0	900.0	900.0	0.167	0.109	0.101	0.115
neos-948268	d	9.4	11.2	14.6	8.2	0.208	0.292	0.196	0.204
	q	10.3	9.7	13.2	16.2	0.212	0.292	0.195	0.204
	s	16.8	11.5	188.1	14.1	0.211	0.289	3.166	0.203
	g	9.4	12.6	14.7	8.2	0.210	0.292	0.194	0.204
	u	9.4	12.6	14.6	8.2	0.210	0.289	0.194	0.205
	w	9.4	12.6	14.7	8.2	0.210	0.289	0.194	0.206
neos-956971	d	900.0	900.0	900.0	900.0	0.008	0.008	0.008	0.008
	q	856.3	900.0	900.0	900.0	0.161	0.194	0.119	0.350
	s	900.0	900.0	900.0	900.0	1.241	0.790	0.775	0.958
	g	900.0	900.0	900.0	900.0	0.150	0.113	0.118	0.187
	u	900.0	900.0	900.0	900.0	0.028	0.087	0.056	0.083
	w	900.0	900.0	900.0	900.0	0.064	0.081	0.079	0.089
neos-960392	d	900.0	900.0	900.0	900.0	0.008	0.008	0.010	0.012
	q	900.0	900.0	780.3	900.0	0.177	0.266	0.213	0.247
	s	645.5	436.8	505.8	415.9	1.279	1.467	1.609	1.648
	g	900.0	900.0	900.0	900.0	0.053	0.098	0.078	0.092
	u	900.0	900.0	900.0	900.0	0.070	0.136	0.109	0.098
	w	900.0	900.0	900.0	900.0	0.052	0.093	0.082	0.067
neos13	d	222.9	223.6	223.8	222.9	497.091	489.964	502.574	492.169
	q	731.4	896.3	499.4	401.3	296.185	313.900	398.259	368.434
	s	514.8	510.6	508.9	510.2	396.987	399.495	399.944	403.401
	g	215.1	138.0	147.4	136.9	511.487	399.098	428.857	354.859
	u	296.0	259.0	447.1	190.1	508.402	475.102	506.231	358.667
	w	797.7	799.9	803.2	796.6	423.524	420.887	424.123	426.081
newdano	d	900.0	900.0	900.0	900.0	208.515	190.594	209.000	198.997
	q	900.0	900.0	900.0	900.0	40.456	40.865	40.965	36.666
	s	900.0	900.0	900.0	900.0	98.065	112.632	108.475	101.520
	g	900.0	900.0	900.0	900.0	183.198	185.652	205.476	185.849
	u	900.0	900.0	900.0	900.0	184.251	199.377	201.902	188.435
	w	900.0	900.0	900.0	900.0	127.921	128.197	129.864	119.249
nw04	d	26.5	26.6	26.5	26.5	157.000	157.000	157.000	157.000
	q	41.3	32.1	34.5	33.5	155.618	108.148	114.024	107.463
	s	24.7	24.8	24.8	24.7	91.000	91.000	91.000	91.000
	g	26.6	26.6	26.6	26.6	157.000	157.000	157.000	157.000
	u	26.6	26.6	26.6	26.6	157.000	157.000	157.000	157.000
	w	26.6	26.6	26.7	26.6	157.000	157.000	157.000	157.000
opt1217	d	0.5	0.5	0.5	0.5	17.000	24.000	21.000	24.000
	q	0.5	0.5	0.5	0.5	19.000	27.000	22.000	22.000
	s	0.5	0.5	0.5	0.5	17.000	22.000	28.000	21.000
	g	0.5	0.5	0.5	0.5	17.000	24.000	21.000	24.000

cont. on next page

Table D.2: LP pricing results for every tested instance, seed, and pricing method from the experiment in Section 7.2.

Instance	P	time				LPthpt			
		0	1	2	3	0	1	2	3
p0548	u	0.5	0.5	0.5	0.5	17.000	24.000	21.000	24.000
	w	0.5	0.5	0.5	0.5	17.000	24.000	21.000	24.000
	d	0.5	0.5	0.5	0.5	33.000	33.000	33.000	33.000
	q	0.5	0.5	0.5	0.5	35.000	35.000	35.000	35.000
	s	0.5	0.5	0.5	0.5	37.000	37.000	37.000	37.000
	g	0.5	0.5	0.5	0.5	33.000	33.000	33.000	33.000
prod1	u	0.5	0.5	0.5	0.5	33.000	33.000	33.000	33.000
	w	0.5	0.5	0.5	0.5	33.000	33.000	33.000	33.000
	d	17.1	17.1	17.4	16.8	7156.893	6928.694	7037.421	7220.705
	q	21.2	21.4	20.5	21.2	5728.772	5847.674	6457.744	5580.614
	s	22.4	22.3	22.2	22.1	3748.904	3819.333	3761.978	3676.364
	g	17.4	17.8	17.6	17.9	6890.989	7079.148	7561.538	6167.129
pw-myciel4	u	17.2	20.1	78.6	51.0	6227.021	6585.849	6304.552	6875.552
	w	20.5	19.5	19.4	22.4	6991.248	6764.394	7258.812	6526.979
	d	900.0	900.0	900.0	900.0	364.061	296.295	299.537	342.828
	q	900.0	900.0	900.0	900.0	53.698	70.231	49.005	43.480
	s	900.0	900.0	900.0	900.0	130.578	140.226	149.225	117.235
	g	900.0	900.0	900.0	900.0	318.923	283.605	310.498	302.086
rentacar	u	900.0	900.0	900.0	900.0	282.694	284.017	350.265	299.743
	w	900.0	900.0	900.0	900.0	150.701	172.503	168.401	177.257
	d	2.1	2.0	2.1	2.1	30.000	30.000	30.000	30.000
	q	1.9	1.9	1.9	1.9	32.000	32.000	32.000	32.000
	s	2.3	2.2	2.3	2.3	36.000	36.000	36.000	36.000
	g	2.1	2.1	2.0	2.0	30.000	30.000	30.000	30.000
rmine6	u	2.1	2.1	2.1	2.0	30.000	30.000	30.000	30.000
	w	2.1	2.0	2.1	2.1	30.000	30.000	30.000	30.000
	d	900.0	900.0	900.0	900.0	849.114	851.783	845.140	853.474
	q	900.0	900.0	676.7	800.6	538.530	488.593	524.223	462.907
	s	900.0	900.0	900.0	900.0	550.833	530.828	521.660	547.573
	g	900.0	624.3	638.0	699.7	864.276	853.545	888.432	822.056
rocII-4-11	u	900.0	882.0	842.8	824.7	841.542	1058.416	805.381	1043.327
	w	900.0	781.8	900.0	900.0	838.428	873.857	837.505	835.683
	d	900.0	900.0	900.0	900.0	5.727	4.691	5.741	5.395
	q	252.3	400.2	270.3	490.9	126.487	155.780	96.193	160.837
	s	440.6	580.3	594.6	486.4	78.775	98.828	89.099	73.541
	g	257.0	356.0	449.2	308.6	88.776	100.577	139.253	112.417
set1ch	u	372.1	450.6	473.8	482.5	74.244	79.528	60.015	84.076
	w	562.2	611.2	588.8	523.7	42.750	29.076	35.833	37.218
	d	0.5	0.5	0.5	0.5	40.000	40.000	40.000	40.000
	q	0.5	0.5	0.5	0.5	48.000	48.000	48.000	48.000
	s	0.8	0.8	0.8	0.8	61.000	61.000	61.000	61.000
	g	0.5	0.5	0.5	0.5	40.000	40.000	40.000	40.000
seymour	u	0.5	0.5	0.5	0.5	40.000	40.000	40.000	40.000
	w	0.5	0.5	0.5	0.5	40.000	40.000	40.000	40.000
	d	900.0	900.0	900.0	900.0	59.325	58.978	58.642	58.706
	q	900.0	900.0	900.0	900.0	14.105	16.102	15.807	18.239
	s	900.0	900.0	900.0	900.0	32.711	36.036	34.927	37.059
	g	900.0	900.0	900.0	900.0	68.142	75.937	73.866	71.422
sp98ic	u	900.0	900.0	900.0	900.0	73.217	64.057	68.747	60.797
	w	900.0	900.0	900.0	900.0	44.729	46.316	39.723	42.675
	d	900.0	900.0	900.0	900.0	20.548	9.994	16.045	15.640
	q	900.0	900.0	900.0	900.0	18.022	18.858	26.186	19.626

cont. on next page

D. Appendix D

Table D.2: LP pricing results for every tested instance, seed, and pricing method from the experiment in Section 7.2.

Instance	P	time				LPthpt			
		0	1	2	3	0	1	2	3
zib54-UUE	s	900.0	900.0	900.0	900.0	21.944	15.384	20.594	31.573
	g	900.0	900.0	900.0	900.0	31.252	16.221	15.990	17.611
	u	900.0	900.0	900.0	900.0	21.864	22.544	18.539	9.492
	w	900.0	900.0	900.0	900.0	23.056	35.577	21.249	17.683
	d	900.0	900.0	900.0	900.0	149.497	165.691	145.079	164.242
	q	900.0	900.0	900.0	900.0	42.534	44.434	47.953	51.686
	s	900.0	900.0	900.0	900.0	32.047	32.128	30.296	30.923
	g	900.0	900.0	900.0	900.0	168.520	147.731	160.828	159.268
	u	900.0	900.0	900.0	900.0	156.626	154.171	152.450	164.215
	w	900.0	900.0	900.0	900.0	100.997	96.114	93.141	104.656

E

Appendix E

Table E.1: The MIP performance results from Section 9.3 for every instance, tree size estimation, clairvoyant factor ϕ^{clair} , and random seed. The columns **root** and **tree** show the number of root and clairvoyant restarts performed, respectively.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
30n20b8	monotone	10	–	–	–	1	–	–	207.5	158.2	273.9	3981	4292	3752
	monotone	25	–	–	–	–	–	–	169.3	157.6	276.3	4383	4292	3752
	monotone	50	–	–	–	–	–	–	169.1	157.9	273.5	4383	4292	3752
	reg forest	10	–	–	–	1	–	–	182.7	159.9	275.8	6124	4292	3752
	reg forest	25	–	–	–	–	–	–	172.2	160.4	277.0	4383	4292	3752
	reg forest	50	–	–	–	–	–	–	172.0	161.0	277.6	4383	4292	3752
	gap	10	–	–	2	1	–	1	200.5	158.7	313.8	3319	4292	2492
	gap	25	–	–	1	1	–	1	175.2	158.0	307.4	3404	4292	2722
	gap	50	–	–	–	1	–	–	188.1	157.7	273.9	3573	4292	3752
	leaf freq	10	–	–	–	–	–	–	170.8	158.6	273.2	4383	4292	3752
	leaf freq	25	–	–	–	–	–	–	169.3	158.4	275.1	4383	4292	3752
	leaf freq	50	–	–	–	–	–	–	169.5	158.1	272.5	4383	4292	3752
	ssg	10	–	–	3	1	1	1	252.2	166.4	299.8	4112	2207	2249
	ssg	25	–	–	–	1	1	1	261.7	254.3	275.9	5552	3046	3076
	ssg	50	–	–	–	1	1	–	172.4	185.8	273.4	4078	2428	3752
	tree weight	10	–	–	–	1	1	1	281.9	178.1	262.2	3589	1662	2194
	tree weight	25	–	2	–	1	1	1	187.5	183.9	262.1	4182	3071	2194
	tree weight	50	–	–	–	1	–	1	186.4	157.7	262.1	3650	4292	2194
	no clairvoyant	–	–	–	–	–	–	–	170.3	157.3	271.8	4383	4292	3752
	0-restart	–	–	–	–	–	–	–	168.2	157.6	274.3	4383	4292	3752
50v-10	monotone	10	–	–	–	1	–	1	t	t	t	451967	429935	325266
	monotone	25	–	–	–	–	–	1	t	t	t	426182	431462	321602
	monotone	50	–	–	–	–	–	–	t	t	t	426499	433326	493906
	reg forest	10	–	–	–	1	1	1	t	t	t	505907	576227	321705
	reg forest	25	–	–	–	–	–	–	t	t	t	425582	429524	490488
	reg forest	50	–	–	–	–	–	–	t	t	t	426597	429177	491404
	gap	10	–	–	–	1	1	1	t	t	t	356642	404015	357225
	gap	25	–	–	–	–	–	–	t	t	t	426513	431438	498129
	gap	50	–	–	–	–	–	–	t	t	t	426309	432266	491914
	leaf freq	10	–	–	–	1	1	1	t	t	t	417830	424628	398021
	leaf freq	25	–	–	–	1	1	1	t	t	t	408895	427806	454371
	leaf freq	50	–	–	–	1	1	1	t	t	t	406823	450230	407009
	ssg	10	–	–	–	1	1	1	t	t	t	462614	484195	396411
	ssg	25	–	–	–	1	1	1	t	t	t	366108	485447	418361
	ssg	50	–	–	–	1	1	1	t	t	t	440067	485831	392359
	tree weight	10	–	–	–	1	1	1	t	t	t	456994	551393	341917
	tree weight	25	–	–	–	1	1	1	t	t	t	481526	527202	389125
	tree weight	50	–	–	–	1	1	1	t	t	t	528411	445929	458289
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	426245	431252	493149
	0-restart	–	–	–	–	–	–	–	t	t	t	426182	432263	494217

cont. on next page ...

E. Appendix E

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
CMS750_4	monotone	10	–	–	–	1	1	1	387.9	1634.8	1963.3	7035	32262	47928
	monotone	25	–	–	–	1	1	1	388.8	1636.7	1962.4	7035	32262	47928
	monotone	50	–	–	–	1	1	1	390.2	1635.3	1966.5	7035	32262	47928
	reg forest	10	–	–	–	1	1	1	389.5	837.5	t	7035	15815	88116
	reg forest	25	–	–	–	–	–	–	817.0	1374.2	1114.8	12149	42840	18755
	reg forest	50	–	–	–	–	–	–	815.9	1380.0	1104.3	12149	42840	18755
	gap	10	–	–	–	–	–	1	812.1	1375.8	2252.0	12149	42840	60395
	gap	25	–	–	–	–	–	–	812.1	1368.6	1112.7	12149	42840	18755
	gap	50	–	–	–	–	–	–	811.9	1367.4	1115.4	12149	42840	18755
	leaf freq	10	–	–	–	1	1	1	1047.5	1924.2	1963.9	28702	70772	47928
	leaf freq	25	–	–	–	1	1	1	1047.2	1388.0	1965.2	28702	41447	47928
	leaf freq	50	–	–	–	1	–	1	1047.8	1365.1	1971.0	28702	42840	47928
	ssg	10	–	–	–	1	1	1	841.7	1087.2	617.1	11811	24711	19228
	ssg	25	–	–	–	1	1	1	842.7	1112.6	1092.2	11281	32939	30894
	ssg	50	–	–	–	1	1	1	841.4	1437.2	1081.7	12724	47488	12694
	tree weight	10	–	–	–	1	1	1	389.8	1639.8	t	7035	32262	99920
	tree weight	25	–	–	–	1	1	1	630.1	1636.0	t	5836	32262	99747
	tree weight	50	–	–	–	1	1	1	1238.9	1637.3	t	27084	32262	99754
	no clairvoyant	–	–	–	–	–	–	–	813.9	1354.3	1109.2	12149	42840	18755
	0-restart	–	–	–	–	–	–	–	808.5	1368.9	1110.8	12149	42840	18755
academic.small	monotone	10	–	–	–	–	–	–	t	t	t	1457	654	732
	monotone	25	–	–	–	–	–	–	t	t	t	1460	654	734
	monotone	50	–	–	–	–	–	–	t	t	t	1455	650	733
	reg forest	10	–	–	–	–	–	–	t	t	t	1457	649	733
	reg forest	25	–	–	–	–	–	–	t	t	t	1458	651	732
	reg forest	50	–	–	–	–	–	–	t	t	t	1458	652	734
	gap	10	–	–	–	–	–	–	t	t	t	1458	654	730
	gap	25	–	–	–	–	–	–	t	t	t	1457	654	732
	gap	50	–	–	–	–	–	–	t	t	t	1459	652	732
	leaf freq	10	–	–	–	–	–	–	t	t	t	1455	651	732
	leaf freq	25	–	–	–	–	–	–	t	t	t	1457	651	733
	leaf freq	50	–	–	–	–	–	–	t	t	t	1459	650	733
	ssg	10	–	–	–	–	–	–	t	t	t	1457	652	732
	ssg	25	–	–	–	–	–	–	t	t	t	1459	654	732
	ssg	50	–	–	–	–	–	–	t	t	t	1459	656	733
	tree weight	10	–	–	–	–	–	–	t	t	t	1458	654	732
	tree weight	25	–	–	–	–	–	–	t	t	t	1457	651	732
	tree weight	50	–	–	–	–	–	–	t	t	t	1457	654	733
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	1457	654	733
	0-restart	–	–	–	–	–	–	–	t	t	t	1457	654	730
air05	monotone	10	–	–	–	–	–	–	28.0	35.9	36.0	155	410	473
	monotone	25	–	–	–	–	–	–	28.4	36.0	36.1	155	410	473
	monotone	50	–	–	–	–	–	–	28.1	35.8	36.0	155	410	473
	reg forest	10	–	–	–	–	–	–	28.0	36.2	36.3	155	410	473
	reg forest	25	–	–	–	–	–	–	28.3	36.0	36.2	155	410	473
	reg forest	50	–	–	–	–	–	–	28.1	36.2	36.1	155	410	473
	gap	10	–	–	–	–	–	–	29.7	35.9	35.9	155	410	473
	gap	25	–	–	–	–	–	–	28.1	35.9	36.0	155	410	473
	gap	50	–	–	–	–	–	–	28.1	35.8	35.8	155	410	473
	leaf freq	10	–	–	–	–	–	–	28.1	35.7	36.4	155	410	473
	leaf freq	25	–	–	–	–	–	–	28.1	35.9	36.2	155	410	473
	leaf freq	50	–	–	–	–	–	–	28.2	35.8	36.2	155	410	473
	ssg	10	–	–	–	–	–	–	28.2	36.0	35.8	155	410	473
	ssg	25	–	–	–	–	–	–	28.1	35.8	35.9	155	410	473
	ssg	50	–	–	–	–	–	–	28.2	36.0	35.7	155	410	473
	tree weight	10	–	–	–	–	–	–	28.1	35.8	36.0	155	410	473
	tree weight	25	–	–	–	–	–	–	28.2	36.0	36.1	155	410	473
	tree weight	50	–	–	–	–	–	–	28.1	35.9	36.3	155	410	473
	no clairvoyant	–	–	–	–	–	–	–	28.1	35.9	35.7	155	410	473
	0-restart	–	–	–	–	–	–	–	28.0	35.8	35.8	155	410	473
app1-1	monotone	10	–	–	1	–	–	–	3.0	4.7	4.7	20	1	2
	monotone	25	–	–	1	–	–	–	2.9	4.7	4.6	20	1	2
	monotone	50	–	–	1	–	–	–	3.0	4.8	4.7	20	1	2
	reg forest	10	–	–	1	–	–	–	3.1	5.0	4.8	20	1	2
	reg forest	25	–	–	1	–	–	–	3.0	4.9	4.8	20	1	2
	reg forest	50	–	–	1	–	–	–	3.1	4.9	4.8	20	1	2
	gap	10	–	–	1	–	–	–	2.9	4.7	4.8	20	1	2
	gap	25	–	–	1	–	–	–	2.9	4.8	4.7	20	1	2
	gap	50	–	–	1	–	–	–	2.9	4.7	4.7	20	1	2
	leaf freq	10	–	–	1	–	–	–	2.9	4.8	4.7	20	1	2
	leaf freq	25	–	–	1	–	–	–	2.9	4.7	4.8	20	1	2
	leaf freq	50	–	–	1	–	–	–	2.9	4.8	4.7	20	1	2
	ssg	10	–	–	1	–	–	–	3.0	4.7	4.7	20	1	2
	ssg	25	–	–	1	–	–	–	3.0	4.7	4.7	20	1	2
	ssg	50	–	–	1	–	–	–	2.9	4.8	4.7	20	1	2
	tree weight	10	–	–	1	–	–	–	2.9	4.7	4.8	20	1	2

cont. on next page ...

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
app1-2	tree weight	25	–	–	1	–	–	–	3.0	4.8	4.7	20	1	2
	tree weight	50	–	–	1	–	–	–	3.0	4.7	4.7	20	1	2
	no clairvoyant	–	–	–	1	–	–	–	3.0	4.8	4.7	20	1	2
	0-restart	–	–	–	–	–	–	–	2.9	4.8	4.0	20	1	27
	monotone	10	1	1	–	–	–	–	213.0	414.9	262.6	863	7515	3263
	monotone	25	1	1	–	–	–	–	211.7	418.9	263.6	863	7515	3263
	monotone	50	1	1	–	–	–	–	213.3	421.1	262.7	863	7515	3263
	reg forest	10	1	1	–	–	–	–	212.7	423.5	264.5	863	7515	3263
	reg forest	25	1	1	–	–	–	–	213.7	416.1	264.9	863	7515	3263
	reg forest	50	1	1	–	–	–	–	213.2	421.0	264.0	863	7515	3263
	gap	10	1	1	–	–	1	1	227.3	1176.0	302.1	863	4271	3334
	gap	25	1	1	–	–	1	1	211.4	618.3	301.5	863	16625	3334
	gap	50	1	1	–	–	1	–	211.9	1058.0	265.9	863	25803	3263
	leaf freq	10	1	1	–	–	–	–	214.6	419.7	263.1	863	7515	3263
	leaf freq	25	1	1	–	–	–	–	215.7	417.6	262.9	863	7515	3263
	leaf freq	50	1	1	–	–	–	–	213.1	419.3	264.1	863	7515	3263
	ssg	10	1	1	–	–	1	1	211.9	933.4	1077.4	863	20464	23901
	ssg	25	1	1	–	–	1	1	212.6	263.6	373.0	863	2801	3966
	ssg	50	1	1	–	–	1	1	212.4	619.8	299.9	863	16625	3334
	tree weight	10	1	1	–	–	1	–	209.2	503.9	263.5	863	15904	3263
assign1-5-8	tree weight	25	1	1	–	–	–	–	212.3	421.2	267.1	863	7515	3263
	tree weight	50	1	1	–	–	–	–	217.4	417.5	262.2	863	7515	3263
	no clairvoyant	–	1	1	–	–	–	–	211.7	416.2	258.6	863	7515	3263
	0-restart	–	–	–	–	–	–	–	217.8	482.8	262.4	1701	8335	3263
	monotone	10	–	–	–	–	–	–	3241.8	3333.9	2513.2	5093402	5316926	3911052
	monotone	25	–	–	–	–	–	–	3262.7	3336.5	2501.9	5093402	5316926	3911052
	monotone	50	–	–	–	–	–	–	3247.4	3315.2	2512.2	5093402	5316926	3911052
	reg forest	10	–	–	–	–	–	–	3257.9	3341.9	2518.7	5093402	5316926	3911052
	reg forest	25	–	–	–	–	–	–	3238.1	3325.2	2509.8	5093402	5316926	3911052
	reg forest	50	–	–	–	–	–	–	3244.2	3341.4	2509.3	5093402	5316926	3911052
	gap	10	–	–	–	–	–	–	3227.6	3326.9	2494.5	5093402	5316926	3911052
	gap	25	–	–	–	–	–	–	3251.4	3320.5	2515.8	5093402	5316926	3911052
	gap	50	–	–	–	–	–	–	3252.2	3322.4	2499.1	5093402	5316926	3911052
	leaf freq	10	–	–	–	–	–	–	3246.7	3350.7	2501.6	5093402	5316926	3911052
	leaf freq	25	–	–	–	–	–	–	3244.2	3311.2	2516.2	5093402	5316926	3911052
	leaf freq	50	–	–	–	–	–	–	3271.0	3311.9	2504.4	5093402	5316926	3911052
	ssg	10	–	–	–	–	–	–	3244.1	3324.4	2499.8	5093402	5316926	3911052
	ssg	25	–	–	–	–	–	–	3242.6	3329.2	2500.5	5093402	5316926	3911052
	ssg	50	–	–	–	–	–	–	3228.5	3320.9	2506.9	5093402	5316926	3911052
	tree weight	10	–	–	–	–	–	–	3264.6	3362.8	2510.2	5093402	5316926	3911052
atlanta-ip	tree weight	25	–	–	–	–	–	–	3261.9	3329.5	2498.8	5093402	5316926	3911052
	tree weight	50	–	–	–	–	–	–	3249.0	3334.1	2506.6	5093402	5316926	3911052
	no clairvoyant	–	–	–	–	–	–	–	3234.3	3324.4	2499.7	5093402	5316926	3911052
	0-restart	–	–	–	–	–	–	–	3268.5	3321.4	2506.0	5093402	5316926	3911052
	monotone	10	–	–	–	–	–	–	t	t	t	4435	5946	9026
	monotone	25	–	–	–	–	–	–	t	t	t	4433	5948	9018
	monotone	50	–	–	–	–	–	–	t	t	t	4440	5914	8894
	reg forest	10	–	–	–	1	1	1	t	t	t	6298	7398	7608
	reg forest	25	–	–	–	–	–	–	t	t	t	4438	5934	9022
	reg forest	50	–	–	–	–	–	–	t	t	t	4437	5941	8995
	gap	10	–	–	–	1	1	1	t	t	t	6475	6289	5932
	gap	25	–	–	–	1	1	1	t	t	t	4000	5405	7769
	gap	50	–	–	–	–	–	–	t	t	t	4443	5957	9042
	leaf freq	10	–	–	–	–	–	–	t	t	t	4440	6003	9022
	leaf freq	25	–	–	–	–	–	–	t	t	t	4435	5934	9017
	leaf freq	50	–	–	–	–	–	–	t	t	t	4444	5941	9025
	ssg	10	–	–	–	1	1	1	t	t	t	5627	6494	8393
	ssg	25	–	–	–	1	1	1	t	t	t	7478	6188	8089
	ssg	50	–	–	–	1	1	1	t	t	t	7834	6653	9637
bic1s1	tree weight	10	–	–	–	1	1	1	t	t	t	6824	8022	7349
	tree weight	25	–	–	–	1	1	1	t	t	t	9019	7343	6353
	tree weight	50	–	–	–	1	1	1	t	t	t	7362	6436	6323
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	4433	5937	9145
	0-restart	–	–	–	–	–	–	–	t	t	t	4442	5916	9132
	monotone	10	–	–	–	1	1	1	t	t	t	22223	26727	22602
	monotone	25	–	–	–	1	1	1	t	t	t	22084	26682	22295
	monotone	50	–	–	–	–	1	–	t	t	t	27362	26703	23496
	reg forest	10	–	–	–	1	1	1	t	t	t	18788	26798	22324
	reg forest	25	–	–	–	–	1	1	t	t	t	27348	22142	22558
	reg forest	50	–	–	–	–	–	–	t	t	t	27339	29811	23561
	gap	10	–	–	–	1	1	1	t	t	t	22709	22820	25582
	gap	25	–	–	–	1	1	1	t	t	t	33204	22966	24559
	gap	50	–	–	–	1	1	1	t	t	t	27319	23821	20264
	leaf freq	10	–	–	–	1	1	1	t	t	t	24169	20165	19238
	leaf freq	25	–	–	–	1	1	1	t	t	t	25442	27602	19463
	leaf freq	50	–	–	–	–	–	1	t	t	t	27361	29825	19663

cont. on next page ...

E. Appendix E

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
bab2	ssg	10	–	–	–	1	1	1	t	t	t	29578	28133	18169
	ssg	25	–	–	–	1	1	1	t	t	t	25461	28158	18053
	ssg	50	–	–	–	1	1	1	t	t	t	33038	28912	20831
	tree weight	10	–	–	–	1	1	1	t	t	t	23771	20750	22325
	tree weight	25	–	–	–	1	1	1	t	t	t	23856	20754	22403
	tree weight	50	–	–	–	1	1	1	t	t	t	23842	20844	22625
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	27365	29903	23495
	0-restart	–	–	–	–	–	–	–	t	t	t	27348	29915	23463
	monotone	10	–	–	–	–	–	–	t	t	t	386	689	579
	monotone	25	–	–	–	–	–	–	t	t	t	379	629	584
	monotone	50	–	–	–	–	–	–	t	t	t	379	674	584
	reg forest	10	–	–	–	–	–	–	t	t	t	388	681	584
	reg forest	25	–	–	–	–	–	–	t	t	t	379	639	584
	reg forest	50	–	–	–	–	–	–	t	t	t	362	656	585
	gap	10	–	–	–	–	–	–	t	t	t	379	678	579
	gap	25	–	–	–	–	–	–	t	t	t	379	673	584
	gap	50	–	–	–	–	–	–	t	t	t	382	731	584
	leaf freq	10	–	–	–	–	–	–	t	t	t	395	637	584
	leaf freq	25	–	–	–	–	–	–	t	t	t	357	668	579
	leaf freq	50	–	–	–	–	–	–	t	t	t	372	694	579
bab6	ssg	10	–	–	–	–	–	–	t	t	t	379	665	590
	ssg	25	–	–	–	–	–	–	t	t	t	348	621	585
	ssg	50	–	–	–	–	–	–	t	t	t	385	681	590
	tree weight	10	–	–	–	–	–	–	t	t	t	393	606	584
	tree weight	25	–	–	–	–	–	–	t	t	t	388	674	585
	tree weight	50	–	–	–	–	–	–	t	t	t	385	699	584
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	385	694	584
	0-restart	–	–	–	–	–	–	–	t	t	t	368	720	578
	monotone	10	–	–	–	1	–	1	t	t	t	665	687	1464
	monotone	25	–	–	–	–	–	1	t	t	t	653	687	1463
	monotone	50	–	–	–	1	–	1	t	t	t	665	687	1484
	reg forest	10	–	–	–	–	–	1	t	t	t	701	687	1516
	reg forest	25	–	–	–	–	–	–	t	t	t	616	687	1960
	reg forest	50	–	–	–	–	–	–	t	t	t	712	687	1909
	gap	10	–	–	–	–	–	–	t	t	t	729	687	1902
	gap	25	–	–	–	–	–	–	t	t	t	707	687	1858
	gap	50	–	–	–	–	–	–	t	t	t	724	687	1981
	leaf freq	10	–	–	–	–	–	–	t	t	t	692	687	1933
	leaf freq	25	–	–	–	–	–	–	t	t	t	674	687	1910
	leaf freq	50	–	–	–	–	–	–	t	t	t	742	687	1960
beasleyC3	ssg	10	–	–	–	–	–	1	t	t	t	724	687	1484
	ssg	25	–	–	–	–	–	1	t	t	t	729	687	1355
	ssg	50	–	–	–	–	–	1	t	t	t	674	687	1629
	tree weight	10	–	–	–	–	–	–	t	t	t	692	687	1964
	tree weight	25	–	–	–	–	–	–	t	t	t	692	687	1951
	tree weight	50	–	–	–	–	–	–	t	t	t	722	687	1984
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	722	687	1901
	0-restart	–	–	–	–	–	–	–	t	t	t	666	687	1883
	monotone	10	–	–	1	–	–	–	69.0	43.7	25.7	913	126	2
	monotone	25	–	–	1	–	–	–	69.1	43.8	25.7	913	126	2
	monotone	50	–	–	1	–	–	–	69.1	43.7	25.7	913	126	2
	reg forest	10	–	–	1	–	–	–	69.5	43.9	25.9	913	126	2
	reg forest	25	–	–	1	–	–	–	69.4	44.0	25.9	913	126	2
	reg forest	50	–	–	1	–	–	–	69.4	43.9	25.8	913	126	2
	gap	10	–	–	1	–	–	–	68.9	43.7	25.8	913	126	2
	gap	25	–	–	1	–	–	–	69.1	43.8	25.7	913	126	2
	gap	50	–	–	1	–	–	–	69.0	43.8	25.7	913	126	2
	leaf freq	10	–	–	1	–	–	–	68.7	44.1	25.6	913	126	2
	leaf freq	25	–	–	1	–	–	–	69.2	43.7	25.7	913	126	2
	leaf freq	50	–	–	1	–	–	–	69.0	43.6	25.8	913	126	2
binkar10_1	ssg	10	–	–	1	–	–	–	68.9	43.6	25.8	913	126	2
	ssg	25	–	–	1	–	–	–	68.9	43.7	25.7	913	126	2
	ssg	50	–	–	1	–	–	–	69.3	43.8	25.7	913	126	2
	tree weight	10	–	–	1	–	–	–	68.8	43.7	25.7	913	126	2
	tree weight	25	–	–	1	–	–	–	68.9	43.7	25.8	913	126	2
	tree weight	50	–	–	1	–	–	–	68.9	43.8	25.7	913	126	2
	no clairvoyant	–	–	–	1	–	–	–	69.4	43.7	25.7	913	126	2
	0-restart	–	–	–	–	–	–	–	69.2	43.5	38.9	913	126	33
	monotone	10	1	1	1	–	–	–	24.7	23.4	28.7	2294	2055	2103
	monotone	25	1	1	1	–	–	–	24.7	23.5	28.8	2294	2055	2103
	monotone	50	1	1	1	–	–	–	24.8	23.3	28.9	2294	2055	2103
	reg forest	10	1	1	1	–	–	–	25.2	23.7	29.2	2294	2055	2103
	reg forest	25	1	1	1	–	–	–	25.3	23.7	29.2	2294	2055	2103
	reg forest	50	1	1	1	–	–	–	25.2	23.6	29.2	2294	2055	2103
	gap	10	1	1	1	–	–	–	24.9	23.2	28.7	2294	2055	2103
	gap	25	1	1	1	–	–	–	24.7	23.4	28.7	2294	2055	2103

cont. on next page ...

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
blp-ar98	gap	50	1	1	1	–	–	–	24.8	23.3	28.8	2294	2055	2103
	leaf freq	10	1	1	1	–	–	–	24.7	23.4	28.8	2294	2055	2103
	leaf freq	25	1	1	1	–	–	–	24.8	23.4	28.9	2294	2055	2103
	leaf freq	50	1	1	1	–	–	–	24.6	23.5	28.7	2294	2055	2103
	ssg	10	1	1	1	1	–	–	34.8	23.4	28.8	2947	2055	2103
	ssg	25	1	1	1	–	–	–	24.8	23.2	28.7	2294	2055	2103
	ssg	50	1	1	1	–	–	–	24.8	23.4	28.8	2294	2055	2103
	tree weight	10	1	1	1	1	–	–	35.4	23.4	28.7	2819	2055	2103
	tree weight	25	1	1	1	1	–	–	38.7	23.4	28.8	3233	2055	2103
	tree weight	50	1	1	1	–	–	–	24.8	23.3	28.7	2294	2055	2103
	no clairvoyant	–	1	1	1	–	–	–	25.1	23.4	28.8	2294	2055	2103
	0-restart	–	–	–	–	–	–	–	24.5	31.3	24.2	2320	4051	2323
	monotone	10	–	–	–	1	1	1	t	t	t	112163	122729	115961
	monotone	25	–	–	–	1	1	1	t	t	t	112652	121046	117893
	monotone	50	–	–	–	1	1	1	t	t	t	111923	121564	117667
	reg forest	10	–	–	–	1	1	1	t	t	t	112374	121385	117227
	reg forest	25	–	–	–	–	–	–	t	t	t	98922	91100	85821
	reg forest	50	–	–	–	–	–	–	t	t	t	98923	90979	85907
	gap	10	1	–	–	1	1	1	t	t	t	112629	102034	99296
	gap	25	–	–	–	–	–	1	t	t	t	99142	91460	102458
	gap	50	–	–	–	–	–	–	t	t	t	99322	91074	86025
	leaf freq	10	1	1	1	1	1	1	t	t	t	127561	102344	105345
	leaf freq	25	1	–	–	1	–	–	t	t	t	126328	91225	85169
	leaf freq	50	2	–	–	1	–	–	t	t	t	95827	91110	86372
	ssg	10	–	–	–	1	1	1	t	t	t	100309	115970	107326
	ssg	25	–	–	–	1	1	1	t	t	t	98913	116292	99171
	ssg	50	–	–	–	1	1	1	t	t	t	98043	116758	127309
	tree weight	10	–	–	–	1	1	1	t	t	t	100898	122208	118448
	tree weight	25	–	–	–	1	1	1	t	t	t	100302	122268	118012
	tree weight	50	–	–	–	1	1	1	t	t	t	99401	122554	117716
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	98526	91360	84948
	0-restart	–	–	–	–	–	–	–	t	t	t	98928	90985	86532
blp-ic98	monotone	10	–	–	–	1	1	1	t	t	t	120644	132001	112486
	monotone	25	–	–	–	1	1	1	t	t	t	120057	131986	112330
	monotone	50	–	–	–	1	–	1	t	t	t	120288	132996	112617
	reg forest	10	–	–	–	1	1	1	t	t	t	119956	132177	112259
	reg forest	25	–	–	–	1	1	1	t	t	t	121778	131895	111997
	reg forest	50	–	–	–	–	–	–	t	t	t	112511	133405	112930
	gap	10	–	–	–	1	1	1	t	t	t	117304	133047	112183
	gap	25	–	–	–	1	1	1	t	t	t	131561	130518	104037
	gap	50	–	–	–	1	1	1	t	t	t	138085	115529	107380
	leaf freq	10	–	–	–	1	1	1	t	t	t	120975	124979	106670
	leaf freq	25	–	–	–	1	1	1	t	t	t	114351	122650	107579
	leaf freq	50	–	–	–	1	1	–	t	t	t	115190	135827	114694
	ssg	10	–	–	–	1	1	1	t	t	t	119654	128602	111594
	ssg	25	–	–	–	1	1	1	t	t	t	120020	129916	112458
	ssg	50	–	–	–	1	1	1	t	t	t	120009	129272	112634
	tree weight	10	–	–	–	1	1	1	t	t	t	116766	115001	110456
	tree weight	25	–	–	–	1	1	1	t	t	t	120095	114602	110001
	tree weight	50	–	–	–	1	1	1	t	t	t	119253	128061	118654
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	112467	132875	114043
	0-restart	–	–	–	–	–	–	–	t	t	t	112357	133928	114395
bnatt400	monotone	10	–	–	–	1	1	1	214.9	131.6	121.0	6146	5554	4964
	monotone	25	–	–	–	1	–	1	214.3	188.4	120.1	6146	8456	4964
	monotone	50	–	–	–	1	–	1	215.7	188.4	120.0	6146	8456	4964
	reg forest	10	–	–	–	–	–	–	235.4	191.5	189.3	8043	8456	6609
	reg forest	25	–	–	–	–	–	–	237.0	192.9	189.3	8043	8456	6609
	reg forest	50	–	–	–	–	–	–	235.0	192.1	189.8	8043	8456	6609
	gap	10	–	–	–	–	–	–	237.3	188.2	185.8	8043	8456	6609
	gap	25	–	–	–	–	–	–	231.5	188.4	185.8	8043	8456	6609
	gap	50	–	–	–	–	–	–	232.0	187.7	185.7	8043	8456	6609
	leaf freq	10	–	–	–	–	–	–	231.6	188.0	187.0	8043	8456	6609
	leaf freq	25	–	–	–	–	–	–	231.5	188.5	186.2	8043	8456	6609
	leaf freq	50	–	–	–	–	–	–	232.0	188.9	185.9	8043	8456	6609
	ssg	10	–	–	–	1	–	1	203.1	187.5	216.7	5979	8456	7269
	ssg	25	–	–	–	1	–	1	176.6	188.0	257.0	5792	8456	7926
	ssg	50	–	–	–	1	–	–	186.7	189.2	185.1	5532	8456	6609
	tree weight	10	–	–	–	1	1	1	298.7	132.2	120.0	9151	5554	4964
	tree weight	25	–	–	–	1	1	1	161.3	150.1	120.2	5027	6621	4964
	tree weight	50	–	–	–	1	1	1	239.0	184.8	119.9	8492	7822	4964
	no clairvoyant	–	–	–	–	–	–	–	235.2	188.3	185.7	8043	8456	6609
	0-restart	–	–	–	–	–	–	–	231.3	188.3	186.2	8043	8456	6609
bnatt500	monotone	10	–	1	1	1	1	1	670.6	541.3	739.0	23532	20339	24395
	monotone	25	–	1	1	1	1	1	669.3	540.7	738.6	23532	20339	24395
	monotone	50	–	1	1	1	–	–	669.4	554.2	551.1	23532	20244	20619
	reg forest	10	–	1	1	–	–	1	721.2	556.5	560.4	34107	20244	22172

cont. on next page ...

E. Appendix E

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
bppc4-08	reg forest	25	–	1	1	–	–	–	721.8	557.7	552.4	34107	20244	20619
	reg forest	50	–	1	1	–	–	–	720.2	557.4	556.4	34107	20244	20619
	gap	10	–	1	1	–	–	–	718.0	552.2	551.4	34107	20244	20619
	gap	25	–	1	1	–	–	–	716.3	552.9	551.4	34107	20244	20619
	gap	50	–	1	1	–	–	–	714.6	553.5	552.1	34107	20244	20619
	leaf freq	10	–	1	1	1	–	–	811.2	556.3	550.6	30050	20244	20619
	leaf freq	25	–	1	1	–	–	–	718.1	551.4	551.9	34107	20244	20619
	leaf freq	50	–	1	1	–	–	–	716.2	553.0	549.9	34107	20244	20619
	ssg	10	–	1	1	–	–	–	716.9	553.8	552.1	34107	20244	20619
	ssg	25	–	1	1	–	–	–	717.8	552.5	552.2	34107	20244	20619
	ssg	50	–	1	1	–	–	–	719.5	554.6	550.2	34107	20244	20619
	tree weight	10	–	1	1	1	1	1	671.0	542.4	529.7	23532	20339	19411
	tree weight	25	–	1	1	1	1	1	667.2	542.0	561.9	23532	20339	23038
	tree weight	50	–	1	1	1	1	1	463.4	545.0	608.2	22580	21816	20880
	no clairvoyant	–	–	1	1	–	–	–	720.5	552.9	550.4	34107	20244	20619
	0-restart	–	–	–	–	–	–	–	716.3	656.7	608.7	34107	28284	24155
	monotone	10	–	–	–	1	1	–	t	t	t	398086	341168	301078
	monotone	25	–	–	–	–	–	–	t	t	t	377442	370825	301208
	monotone	50	–	–	–	–	–	–	t	t	t	378546	369249	302769
	reg forest	10	–	–	–	1	1	1	t	t	t	397637	339903	316719
	reg forest	25	–	–	–	–	–	–	t	t	t	376104	370183	302496
	reg forest	50	–	–	–	–	–	–	t	t	t	376743	369124	303782
	gap	10	–	–	–	–	1	–	t	t	t	378400	366348	303931
	gap	25	–	–	–	–	1	–	t	t	t	377676	365556	304791
	gap	50	–	–	–	–	–	–	t	t	t	378687	369450	302493
	leaf freq	10	–	–	–	–	1	1	t	t	t	315328	275417	335818
	leaf freq	25	–	–	–	–	1	–	t	t	t	377363	299192	303747
	leaf freq	50	–	–	–	–	1	–	t	t	t	378499	324482	303212
	ssg	10	–	–	–	–	1	1	t	t	t	335896	336777	340571
	ssg	25	–	–	–	–	1	1	t	t	t	264885	327314	325853
	ssg	50	–	–	–	–	1	1	t	t	t	313334	365732	338442
	tree weight	10	–	–	–	–	1	1	t	t	t	307475	385488	357071
	tree weight	25	–	–	–	–	1	1	t	t	t	319008	281127	325542
	tree weight	50	–	–	–	–	1	1	t	t	t	306703	340332	339522
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	377836	371475	303059
	0-restart	–	–	–	–	–	–	–	t	t	t	378379	369284	300576
brazil3	monotone	10	–	–	–	–	–	–	t	t	t	1303	568	1328
	monotone	25	–	–	–	–	–	–	t	t	t	1297	997	1326
	monotone	50	–	–	–	–	–	–	t	t	t	1286	994	1443
	reg forest	10	–	–	–	–	–	–	t	t	t	1398	1119	1147
	reg forest	25	–	–	–	–	–	–	t	t	t	1394	1119	1147
	reg forest	50	–	–	–	–	–	–	t	t	t	1404	1122	1150
	gap	10	–	–	–	–	–	–	t	t	t	1404	1122	1145
	gap	25	–	–	–	–	–	–	t	t	t	1410	1122	1146
	gap	50	–	–	–	–	–	–	t	t	t	1412	1122	1147
	leaf freq	10	–	–	–	–	–	–	t	t	t	1407	1122	1140
	leaf freq	25	–	–	–	–	–	–	t	t	t	1417	1122	1145
	leaf freq	50	–	–	–	–	–	–	t	t	t	1410	1122	1138
	ssg	10	–	–	–	–	–	–	t	t	t	1396	1122	1151
	ssg	25	–	–	–	–	–	–	t	t	t	1410	1120	1138
	ssg	50	–	–	–	–	–	–	t	t	t	1410	1123	1145
	tree weight	10	–	–	–	–	–	–	t	t	t	1417	1122	1141
	tree weight	25	–	–	–	–	–	–	t	t	t	1413	1123	1146
	tree weight	50	–	–	–	–	–	–	t	t	t	1410	1120	1146
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	1404	1120	1143
	0-restart	–	–	–	–	–	–	–	t	t	t	1410	1122	1150
buildingenergy	monotone	10	–	–	–	–	–	–	t	t	t	23	1	1
	monotone	25	–	–	–	–	–	–	t	t	t	22	1	1
	monotone	50	–	–	–	–	–	–	t	t	t	23	1	1
	reg forest	10	–	–	–	–	–	–	t	t	t	23	1	1
	reg forest	25	–	–	–	–	–	–	t	t	t	22	1	1
	reg forest	50	–	–	–	–	–	–	t	t	t	23	1	1
	gap	10	–	–	–	–	–	–	t	t	t	22	1	1
	gap	25	–	–	–	–	–	–	t	t	t	22	1	1
	gap	50	–	–	–	–	–	–	t	t	t	23	1	1
	leaf freq	10	–	–	–	–	–	–	t	t	t	22	1	1
	leaf freq	25	–	–	–	–	–	–	t	t	t	24	1	1
	leaf freq	50	–	–	–	–	–	–	t	t	t	23	1	1
	ssg	10	–	–	–	–	–	–	t	t	t	23	1	1
	ssg	25	–	–	–	–	–	–	t	t	t	22	1	1
	ssg	50	–	–	–	–	–	–	t	t	t	23	1	1
	tree weight	10	–	–	–	–	–	–	t	t	t	23	1	1
	tree weight	25	–	–	–	–	–	–	t	t	t	23	1	1
	tree weight	50	–	–	–	–	–	–	t	t	t	22	1	1
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	22	1	1
	0-restart	–	–	–	–	–	–	–	t	t	t	23	1	1

cont. on next page ...

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
cbs-cta	monotone	10	–	–	–	–	–	–	4.6	40.7	551.1	1	1	41
	monotone	25	–	–	–	–	–	–	4.5	40.6	553.1	1	1	41
	monotone	50	–	–	–	–	–	–	4.6	40.7	547.1	1	1	41
	reg forest	10	–	–	–	–	–	–	4.6	40.7	553.1	1	1	41
	reg forest	25	–	–	–	–	–	–	4.7	40.8	551.8	1	1	41
	reg forest	50	–	–	–	–	–	–	4.6	40.7	552.7	1	1	41
	gap	10	–	–	–	–	–	–	4.6	40.7	551.4	1	1	41
	gap	25	–	–	–	–	–	–	4.6	40.6	552.3	1	1	41
	gap	50	–	–	–	–	–	–	4.6	40.6	553.8	1	1	41
	leaf freq	10	–	–	–	–	–	–	4.6	40.6	553.5	1	1	41
	leaf freq	25	–	–	–	–	–	–	4.6	40.6	551.1	1	1	41
	leaf freq	50	–	–	–	–	–	–	4.6	40.7	552.5	1	1	41
	ssg	10	–	–	–	–	–	–	4.6	40.7	552.9	1	1	41
	ssg	25	–	–	–	–	–	–	4.6	40.6	553.1	1	1	41
	ssg	50	–	–	–	–	–	–	4.7	40.6	552.2	1	1	41
	tree weight	10	–	–	–	–	–	–	4.6	40.7	552.6	1	1	41
	tree weight	25	–	–	–	–	–	–	4.6	40.6	552.7	1	1	41
	tree weight	50	–	–	–	–	–	–	4.5	40.7	551.3	1	1	41
	no clairvoyant	–	–	–	–	–	–	–	4.6	40.5	547.7	1	1	41
	0-restart	–	–	–	–	–	–	–	4.6	40.6	552.4	1	1	41
chromatic.1024-7	monotone	10	–	–	–	–	–	–	t	t	t	171	229	306
	monotone	25	–	–	–	–	–	–	t	t	t	171	229	325
	monotone	50	–	–	–	–	–	–	t	t	t	150	229	321
	reg forest	10	–	–	–	–	–	–	t	t	t	171	229	321
	reg forest	25	–	–	–	–	–	–	t	t	t	167	214	328
	reg forest	50	–	–	–	–	–	–	t	t	t	171	229	332
	gap	10	–	–	–	–	–	–	t	t	t	152	229	321
	gap	25	–	–	–	–	–	–	t	t	t	171	229	321
	gap	50	–	–	–	–	–	–	t	t	t	152	229	311
	leaf freq	10	–	–	–	–	–	–	t	t	t	164	229	332
	leaf freq	25	–	–	–	–	–	–	t	t	t	166	229	331
	leaf freq	50	–	–	–	–	–	–	t	t	t	150	229	312
	ssg	10	–	–	–	–	–	–	t	t	t	171	229	325
	ssg	25	–	–	–	–	–	–	t	t	t	157	229	321
	ssg	50	–	–	–	–	–	–	t	t	t	162	229	321
	tree weight	10	–	–	–	–	–	–	t	t	t	157	229	322
	tree weight	25	–	–	–	–	–	–	t	t	t	171	229	325
	tree weight	50	–	–	–	–	–	–	t	t	t	164	229	321
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	171	229	306
	0-restart	–	–	–	–	–	–	–	t	t	t	161	229	317
chromatic.512-7	monotone	10	–	–	–	–	–	–	1450.7	3491.5	t	3781	26980	19230
	monotone	25	–	–	–	–	–	–	1451.3	3484.7	t	3781	26980	19226
	monotone	50	–	–	–	–	–	–	1450.3	3492.1	t	3781	26980	19256
	reg forest	10	–	–	–	–	–	–	1451.9	3485.0	t	3781	26980	19230
	reg forest	25	–	–	–	–	–	–	1448.8	3504.5	t	3781	26980	19230
	reg forest	50	–	–	–	–	–	–	1449.7	3483.6	t	3781	26980	19237
	gap	10	–	–	–	–	–	–	1469.8	3489.5	t	3781	26980	19141
	gap	25	–	–	–	–	–	–	1458.2	3490.8	t	3781	26980	19230
	gap	50	–	–	–	–	–	–	1453.3	3490.0	t	3781	26980	19011
	leaf freq	10	–	–	–	–	–	–	1451.0	3485.3	t	3781	26980	19216
	leaf freq	25	–	–	–	–	–	–	1459.5	3506.9	t	3781	26980	19229
	leaf freq	50	–	–	–	–	–	–	1456.6	3494.6	t	3781	26980	19229
	ssg	10	–	–	–	–	–	–	1456.1	3483.7	t	3781	26980	19236
	ssg	25	–	–	–	–	–	–	1456.2	3485.8	t	3781	26980	19230
	ssg	50	–	–	–	–	–	–	1451.6	3484.6	t	3781	26980	19230
	tree weight	10	–	–	–	–	–	–	1455.4	3494.9	t	3781	26980	19230
	tree weight	25	–	–	–	–	–	–	1456.2	3446.5	t	3781	26980	19292
	tree weight	50	–	–	–	–	–	–	1437.8	3480.6	t	3781	26980	19342
	no clairvoyant	–	–	–	–	–	–	–	1437.8	3506.2	t	3781	26980	19229
	0-restart	–	–	–	–	–	–	–	1448.0	3467.2	t	3781	26980	19079
cmflsp50-24-8-8	monotone	10	–	–	–	1	1	1	t	t	t	25818	28007	27289
	monotone	25	–	–	–	–	1	1	t	t	t	26994	28329	27327
	monotone	50	–	–	–	–	1	1	t	t	t	26950	27976	27411
	reg forest	10	–	–	–	1	1	1	t	t	t	24368	28132	27329
	reg forest	25	–	–	–	–	1	–	t	t	t	27162	22051	24383
	reg forest	50	–	–	–	–	–	–	t	t	t	27188	23981	24381
	gap	10	–	–	–	1	1	1	t	t	t	22752	18917	24938
	gap	25	–	–	–	1	1	1	t	t	t	22620	10205	29392
	gap	50	–	–	–	1	1	1	t	t	t	20898	21006	29031
	leaf freq	10	–	–	–	1	1	1	t	t	t	24503	22370	32183
	leaf freq	25	–	–	–	–	1	–	t	t	t	27007	22404	24211
	leaf freq	50	–	–	–	–	–	–	t	t	t	27011	23888	24357
	ssg	10	–	–	–	1	1	1	t	t	t	29009	24396	27364
	ssg	25	–	–	–	1	1	1	t	t	t	25869	24176	27404
	ssg	50	–	–	–	1	1	1	t	t	t	22774	24244	22461
	tree weight	10	–	–	–	1	1	1	t	t	t	22889	28178	27509

cont. on next page ...

E. Appendix E

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
co-100	tree weight	25	–	–	–	1	1	1	t	t	t	21875	28405	27622
	tree weight	50	–	–	–	1	1	1	t	t	t	27534	27973	27524
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	27005	24007	24386
	0-restart	–	–	–	–	–	–	–	t	t	t	27010	23993	24385
	monotone	10	–	–	–	–	–	–	t	t	t	1475	2585	183
	monotone	25	–	–	–	–	–	–	t	t	t	1482	2674	182
	monotone	50	–	–	–	–	–	–	t	t	t	1482	2596	181
	reg forest	10	–	–	–	–	–	–	t	t	t	1482	2585	181
	reg forest	25	–	–	–	–	–	–	t	t	t	1446	2631	182
	reg forest	50	–	–	–	–	–	–	t	t	t	1475	2579	167
	gap	10	–	–	–	–	–	–	t	t	t	1482	2624	182
	gap	25	–	–	–	–	–	–	t	t	t	1482	2619	182
	gap	50	–	–	–	–	–	–	t	t	t	1482	2624	182
	leaf freq	10	–	–	–	–	–	–	t	t	t	1482	2480	182
	leaf freq	25	–	–	–	–	–	–	t	t	t	1482	2683	181
	leaf freq	50	–	–	–	–	–	–	t	t	t	1475	2673	182
	ssg	10	–	–	–	–	–	–	t	t	t	1482	2686	182
	ssg	25	–	–	–	–	–	–	t	t	t	1467	2616	200
	ssg	50	–	–	–	–	–	–	t	t	t	1461	2585	181
	tree weight	10	–	–	–	–	1	–	t	t	t	1482	1706	181
cod105	tree weight	25	–	–	–	–	1	–	t	t	t	1482	1687	181
	tree weight	50	–	–	–	–	1	–	t	t	t	1467	1257	181
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	1461	2586	181
	0-restart	–	–	–	–	–	–	–	t	t	t	1482	2631	181
	monotone	10	–	–	–	–	–	–	615.4	447.0	523.0	795	355	429
	monotone	25	–	–	–	–	–	–	609.8	444.1	526.8	795	355	429
	monotone	50	–	–	–	–	–	–	608.5	441.7	524.0	795	355	429
	reg forest	10	–	–	–	–	–	–	615.6	442.5	524.3	795	355	429
	reg forest	25	–	–	–	–	–	–	609.9	444.3	525.0	795	355	429
	reg forest	50	–	–	–	–	–	–	608.7	444.6	523.1	795	355	429
	gap	10	–	–	–	–	–	–	611.7	445.2	523.3	795	355	429
	gap	25	–	–	–	–	–	–	609.0	443.4	522.8	795	355	429
	gap	50	–	–	–	–	–	–	610.6	444.1	523.5	795	355	429
	leaf freq	10	–	–	–	–	–	–	609.7	444.1	522.7	795	355	429
	leaf freq	25	–	–	–	–	–	–	613.5	443.9	526.4	795	355	429
	leaf freq	50	–	–	–	–	–	–	608.9	450.7	523.6	795	355	429
	ssg	10	–	–	–	–	–	–	610.6	448.3	524.9	795	355	429
	ssg	25	–	–	–	–	–	–	608.2	446.4	524.0	795	355	429
	ssg	50	–	–	–	–	–	–	613.8	449.3	520.1	795	355	429
	tree weight	10	–	–	–	–	–	–	605.0	444.0	524.2	795	355	429
comp07-2idx	tree weight	25	–	–	–	–	–	–	609.2	444.6	523.5	795	355	429
	tree weight	50	–	–	–	–	–	–	611.4	445.1	524.5	795	355	429
	no clairvoyant	–	–	–	–	–	–	–	609.0	444.4	534.5	795	355	429
	0-restart	–	–	–	–	–	–	–	612.7	443.8	524.8	795	355	429
	monotone	10	–	–	–	1	–	–	t	t	t	927	687	343
	monotone	25	–	–	–	1	–	–	t	t	t	927	730	343
	monotone	50	–	–	–	1	–	–	t	t	t	926	720	340
	reg forest	10	–	–	–	–	–	–	t	t	t	1030	719	340
	reg forest	25	–	–	–	–	–	–	t	t	t	1032	720	340
	reg forest	50	–	–	–	–	–	–	t	t	t	1030	717	341
	gap	10	–	–	–	–	–	–	t	t	t	1030	728	336
	gap	25	–	–	–	–	–	–	t	t	t	1029	738	340
	gap	50	–	–	–	–	–	–	t	t	t	1031	691	347
	leaf freq	10	–	–	–	–	–	–	t	t	t	1032	704	346
	leaf freq	25	–	–	–	–	–	–	t	t	t	1029	717	346
	leaf freq	50	–	–	–	–	–	–	t	t	t	1030	737	340
	ssg	10	–	–	–	–	–	–	t	t	t	1029	738	343
	ssg	25	–	–	–	–	–	–	t	t	t	1029	753	343
	ssg	50	–	–	–	–	–	–	t	t	t	1030	735	343
	tree weight	10	–	–	–	–	–	–	t	t	t	1032	695	345
comp21-2idx	tree weight	25	–	–	–	–	–	–	t	t	t	1031	728	343
	tree weight	50	–	–	–	–	–	–	t	t	t	1030	737	346
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	1030	754	343
	0-restart	–	–	–	–	–	–	–	t	t	t	1039	735	340
	monotone	10	–	–	–	1	1	1	t	t	t	2530	2989	2259
	monotone	25	–	–	–	1	1	1	t	t	t	2506	2993	2260
	monotone	50	–	–	–	1	1	1	t	t	t	2530	2940	2256
	reg forest	10	–	–	–	1	1	1	t	t	t	3062	1685	2730
	reg forest	25	–	–	–	–	–	–	t	t	t	3379	4734	2343
	reg forest	50	–	–	–	–	–	–	t	t	t	3393	4737	2340
	gap	10	–	–	–	1	1	1	t	t	t	2643	2494	2891
	gap	25	–	–	–	1	1	1	t	t	t	1748	1804	2887
	gap	50	–	–	–	1	1	1	t	t	t	3544	3218	2889
	leaf freq	10	–	–	–	–	–	–	t	t	t	3380	4735	2345
	leaf freq	25	–	–	–	–	–	–	t	t	t	3382	4735	2343
	leaf freq	50	–	–	–	–	–	–	t	t	t	3391	4733	2349

cont. on next page ...

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
cost266-UUE	ssg	10	–	–	–	1	1	1	t	t	t	4005	2988	2498
	ssg	25	–	–	–	1	1	1	t	t	t	3992	2920	2489
	ssg	50	–	–	–	1	1	1	t	t	t	4000	2063	2482
	tree weight	10	–	–	–	–	–	–	t	t	t	3375	4739	2334
	tree weight	25	–	–	–	–	–	–	t	t	t	3381	4734	2339
	tree weight	50	–	–	–	–	–	–	t	t	t	3387	4747	2343
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	3382	4734	2347
	0-restart	–	–	–	–	–	–	–	t	t	t	3391	4735	2348
	monotone	10	–	–	–	1	1	1	3123.6	2286.9	2948.2	224838	164378	183209
	monotone	25	–	–	–	1	–	–	3129.5	t	3534.3	224838	289806	255070
	monotone	50	–	–	–	–	–	–	3419.8	t	3553.1	244413	288455	255070
	reg forest	10	–	–	–	1	1	1	3134.1	2307.6	t	224838	164378	257109
	reg forest	25	–	–	–	1	1	–	3131.2	3578.3	3573.5	224838	306236	255070
	reg forest	50	–	–	–	–	–	–	3438.4	t	3555.6	244413	287261	255070
	gap	10	–	–	–	1	1	1	3310.4	t	t	240574	282927	251654
	gap	25	–	–	–	1	1	1	2722.6	2812.4	3515.7	191547	220365	269137
	gap	50	–	–	–	–	–	–	3409.6	t	3546.6	244413	289469	255070
	leaf freq	10	–	–	–	1	1	1	3122.5	3400.6	3261.7	229137	256065	236816
	leaf freq	25	–	–	–	1	1	1	3464.7	3103.8	3310.8	270883	241489	248575
	leaf freq	50	–	–	–	1	1	1	3363.2	3227.6	t	225752	242847	280190
	ssg	10	–	–	–	1	1	1	2892.4	2911.1	2542.0	187388	214682	178272
	ssg	25	–	–	–	1	1	1	3001.5	2810.9	3286.9	190716	239858	234394
	ssg	50	–	–	–	1	1	1	3268.7	3194.7	3051.0	234188	260329	211906
	tree weight	10	–	–	–	1	1	1	2690.7	2303.8	3303.3	183069	164378	248916
	tree weight	25	–	–	–	1	1	1	2816.6	3129.1	3133.6	214702	221473	226814
	tree weight	50	–	–	–	1	1	1	2358.2	3283.4	3480.6	178535	259271	275448
cryptanalysis.14	no clairvoyant	–	–	–	–	–	–	–	3422.2	t	3557.1	244413	288665	255070
	0-restart	–	–	–	–	–	–	–	3402.3	t	3537.0	244413	290241	255070
	monotone	10	–	–	–	–	–	–	t	t	t	47	53	34
	monotone	25	–	–	–	–	–	–	t	t	t	47	53	34
	monotone	50	–	–	–	–	–	–	t	t	t	47	53	35
	reg forest	10	–	–	–	–	–	–	t	t	t	47	53	33
	reg forest	25	–	–	–	–	–	–	t	t	t	47	53	35
	reg forest	50	–	–	–	–	–	–	t	t	t	47	53	34
	gap	10	–	–	–	–	–	–	t	t	t	47	53	33
	gap	25	–	–	–	–	–	–	t	t	t	47	53	34
	gap	50	–	–	–	–	–	–	t	t	t	47	53	34
	leaf freq	10	–	–	–	–	–	–	t	t	t	47	53	34
	leaf freq	25	–	–	–	–	–	–	t	t	t	47	53	34
	leaf freq	50	–	–	–	–	–	–	t	t	t	47	53	34
	ssg	10	–	–	–	–	–	–	t	t	t	47	53	33
	ssg	25	–	–	–	–	–	–	t	t	t	47	53	34
	ssg	50	–	–	–	–	–	–	t	t	t	47	53	34
	tree weight	10	–	–	–	–	–	–	t	t	t	47	57	34
	tree weight	25	–	–	–	–	–	–	t	t	t	47	58	34
	tree weight	50	–	–	–	–	–	–	t	t	t	47	53	34
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	48	53	34
	0-restart	–	–	–	–	–	–	–	t	t	t	47	53	34
cryptanalysis.16	monotone	10	–	–	–	–	–	–	3035.7	t	t	77	41	38
	monotone	25	–	–	–	–	–	–	3041.2	t	t	77	41	38
	monotone	50	–	–	–	–	–	–	3053.5	t	t	77	41	38
	reg forest	10	–	–	–	–	–	–	3037.0	t	t	77	42	38
	reg forest	25	–	–	–	–	–	–	3042.2	t	t	77	42	38
	reg forest	50	–	–	–	–	–	–	3026.5	t	t	77	41	38
	gap	10	–	–	–	–	–	–	3047.7	t	t	77	41	38
	gap	25	–	–	–	–	–	–	3036.4	t	t	77	41	38
	gap	50	–	–	–	–	–	–	3034.8	t	t	77	41	38
	leaf freq	10	–	–	–	–	–	–	3039.3	t	t	77	41	38
	leaf freq	25	–	–	–	–	–	–	3033.8	t	t	77	41	38
	leaf freq	50	–	–	–	–	–	–	3029.9	t	t	77	41	38
	ssg	10	–	–	–	–	–	–	3022.9	t	t	77	41	38
	ssg	25	–	–	–	–	–	–	3042.8	t	t	77	41	38
	ssg	50	–	–	–	–	–	–	3033.7	t	t	77	41	38
	tree weight	10	–	–	–	–	–	–	3026.3	t	t	77	41	38
	tree weight	25	–	–	–	–	–	–	3044.9	t	t	77	41	38
	tree weight	50	–	–	–	–	–	–	3028.4	t	t	77	41	38
	no clairvoyant	–	–	–	–	–	–	–	3027.4	t	t	77	42	38
	0-restart	–	–	–	–	–	–	–	3027.4	t	t	77	41	38
csched007	monotone	10	–	–	–	1	1	1	2194.8	2201.5	1538.4	182694	182009	124123
	monotone	25	–	–	–	–	1	1	2507.9	2199.7	1541.9	198097	182009	124123
	monotone	50	–	–	–	–	1	1	2502.2	2195.6	1540.4	198097	182009	124123
	reg forest	10	–	–	–	1	1	1	2203.8	2205.5	1120.2	182694	182009	78854
	reg forest	25	–	–	–	1	1	–	2204.6	3411.8	1667.1	182694	266205	134735
	reg forest	50	–	–	–	–	–	–	2522.3	3471.5	1666.6	198097	282058	134735
	gap	10	–	–	–	1	1	1	1736.9	2188.7	1696.4	127247	167164	123238
	gap	25	–	–	–	1	1	1	1948.2	3143.4	1711.8	125234	226331	124355

cont. on next page ...

E. Appendix E

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
csched008	gap	50	–	–	–	1	1	1	2627.6	2536.6	1175.1	176304	174127	86791
	leaf freq	10	–	–	–	1	1	1	1548.7	2641.1	1356.1	104081	191284	99795
	leaf freq	25	–	–	–	1	1	1	1844.8	2267.2	1575.1	130181	164909	109900
	leaf freq	50	–	–	–	1	1	–	1886.2	2852.5	1668.7	138764	225052	134735
	ssg	10	–	–	–	1	1	1	1481.8	2688.3	1346.3	85846	204199	90561
	ssg	25	–	–	–	1	1	1	1578.7	2689.1	1352.1	109792	204199	90561
	ssg	50	–	–	–	1	1	1	1460.0	2696.0	1351.3	111912	204199	90561
	tree weight	10	–	–	–	1	1	1	1293.7	2195.0	1236.0	81846	182009	90235
	tree weight	25	–	–	–	1	1	1	1583.8	2192.4	1124.0	93920	182009	84978
	tree weight	50	–	–	–	1	1	1	1375.1	3136.9	1370.0	89036	230824	111805
	no clairvoyant	–	–	–	–	–	–	–	2504.0	3462.8	1658.7	198097	282058	134735
	0-restart	–	–	–	–	–	–	–	2489.5	3459.2	1667.9	198097	282058	134735
	monotone	10	–	–	–	1	–	1	965.7	1061.4	446.3	81204	99564	39272
	monotone	25	–	–	–	1	–	1	968.6	1061.6	446.5	81204	99564	39272
	monotone	50	–	–	–	–	–	–	1461.8	1060.6	681.5	143604	99564	67391
	reg forest	10	–	–	–	1	–	1	982.2	1075.5	451.3	81204	99564	39272
	reg forest	25	–	–	–	–	–	1	1473.0	1075.8	522.9	143604	99564	39751
	reg forest	50	–	–	–	–	–	–	1474.5	1078.6	688.1	143604	99564	67391
	gap	10	–	–	–	1	1	1	689.8	1019.0	645.9	62404	98797	44404
	gap	25	–	–	–	–	1	–	1459.0	1020.5	680.3	143604	98797	67391
	gap	50	–	–	–	–	–	–	1461.1	1063.8	680.2	143604	99564	67391
	leaf freq	10	–	–	–	1	1	1	582.6	580.6	676.2	41815	49352	44156
	leaf freq	25	–	–	–	1	1	–	776.7	759.8	681.4	62578	46439	67391
	leaf freq	50	–	–	–	–	1	–	1455.3	761.4	680.4	143604	46439	67391
	ssg	10	–	–	–	1	1	1	967.5	1017.7	745.0	81204	98797	50450
	ssg	25	–	–	–	1	1	1	966.4	1014.9	712.5	81204	98797	47877
	ssg	50	–	–	–	1	1	1	968.8	1016.9	710.6	81204	98797	47877
	tree weight	10	–	–	–	1	1	1	578.4	525.0	370.9	51232	45984	26983
	tree weight	25	–	–	–	1	1	1	578.5	450.0	528.7	47080	37443	40368
	tree weight	50	–	–	–	1	1	1	538.1	631.5	840.5	41972	54949	105419
	no clairvoyant	–	–	–	–	–	–	–	1475.5	1064.0	681.3	143604	99564	67391
	0-restart	–	–	–	–	–	–	–	1458.4	1063.1	680.7	143604	99564	67391
cvs16r128-89	monotone	10	–	–	–	–	–	–	t	t	t	19211	17524	28463
	monotone	25	–	–	–	–	–	–	t	t	t	19194	17503	28580
	monotone	50	–	–	–	–	–	–	t	t	t	19178	17502	28803
	reg forest	10	–	–	–	–	–	–	t	t	t	19192	17504	28840
	reg forest	25	–	–	–	–	–	–	t	t	t	19192	17455	28587
	reg forest	50	–	–	–	–	–	–	t	t	t	19092	17522	28573
	gap	10	–	–	–	–	–	–	t	t	t	19203	17504	28524
	gap	25	–	–	–	–	–	–	t	t	t	19192	17609	28572
	gap	50	–	–	–	–	–	–	t	t	t	19200	17503	28620
	leaf freq	10	–	–	–	–	–	–	t	t	t	19150	17501	28638
	leaf freq	25	–	–	–	–	–	–	t	t	t	19191	17620	28714
	leaf freq	50	–	–	–	–	–	–	t	t	t	19298	17512	28673
	ssg	10	–	–	–	–	–	–	t	t	t	19199	17471	28603
	ssg	25	–	–	–	–	–	–	t	t	t	19213	17500	28680
	ssg	50	–	–	–	–	–	–	t	t	t	19211	17512	28844
	tree weight	10	–	–	–	–	–	–	t	t	t	19132	17492	28636
	tree weight	25	–	–	–	–	–	–	t	t	t	19192	17524	28574
	tree weight	50	–	–	–	–	–	–	t	t	t	19203	17512	28535
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	19192	17460	28454
	0-restart	–	–	–	–	–	–	–	t	t	t	19213	17486	28554
dano3_3	monotone	10	–	–	–	–	–	–	112.9	121.3	87.9	27	17	18
	monotone	25	–	–	–	–	–	–	112.6	121.5	88.3	27	17	18
	monotone	50	–	–	–	–	–	–	112.8	120.7	88.4	27	17	18
	reg forest	10	–	–	–	–	–	–	112.8	121.7	88.7	27	17	18
	reg forest	25	–	–	–	–	–	–	113.0	121.7	88.6	27	17	18
	reg forest	50	–	–	–	–	–	–	112.9	121.5	88.8	27	17	18
	gap	10	–	–	–	–	–	–	112.7	121.3	88.3	27	17	18
	gap	25	–	–	–	–	–	–	113.0	121.5	88.4	27	17	18
	gap	50	–	–	–	–	–	–	112.9	121.4	88.5	27	17	18
	leaf freq	10	–	–	–	–	–	–	112.2	121.5	87.8	27	17	18
	leaf freq	25	–	–	–	–	–	–	113.1	121.4	88.2	27	17	18
	leaf freq	50	–	–	–	–	–	–	112.8	121.4	88.3	27	17	18
	ssg	10	–	–	–	–	–	–	112.3	121.4	88.4	27	17	18
	ssg	25	–	–	–	–	–	–	112.8	121.7	88.4	27	17	18
	ssg	50	–	–	–	–	–	–	112.2	121.4	88.5	27	17	18
	tree weight	10	–	–	–	–	–	–	112.7	121.3	87.9	27	17	18
	tree weight	25	–	–	–	–	–	–	112.7	121.4	88.8	27	17	18
	tree weight	50	–	–	–	–	–	–	113.1	121.2	88.3	27	17	18
	no clairvoyant	–	–	–	–	–	–	–	112.7	121.5	88.3	27	17	18
	0-restart	–	–	–	–	–	–	–	112.7	121.2	88.4	27	17	18
dano3_5	monotone	10	–	–	–	–	–	–	274.2	344.1	422.7	199	273	243
	monotone	25	–	–	–	–	–	–	273.4	342.2	420.7	199	273	243
	monotone	50	–	–	–	–	–	–	273.9	345.0	422.7	199	273	243
	reg forest	10	–	–	–	–	–	–	273.8	343.6	423.9	199	273	243

cont. on next page ...

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
decomp2	reg forest	25	–	–	–	–	–	–	273.4	343.8	422.5	199	273	243
	reg forest	50	–	–	–	–	–	–	273.2	343.5	422.7	199	273	243
	gap	10	–	–	–	–	–	–	271.9	343.7	423.2	199	273	243
	gap	25	–	–	–	–	–	–	273.2	343.7	423.1	199	273	243
	gap	50	–	–	–	–	–	–	273.1	343.8	423.8	199	273	243
	leaf freq	10	–	–	–	–	–	–	273.5	343.9	420.6	199	273	243
	leaf freq	25	–	–	–	–	–	–	273.7	343.5	422.4	199	273	243
	leaf freq	50	–	–	–	–	–	–	273.3	345.4	420.3	199	273	243
	ssg	10	–	–	–	–	–	–	273.3	343.3	423.5	199	273	243
	ssg	25	–	–	–	–	–	–	273.2	343.4	423.5	199	273	243
	ssg	50	–	–	–	–	–	–	273.1	343.1	422.8	199	273	243
	tree weight	10	–	–	–	–	–	–	273.4	343.7	422.2	199	273	243
	tree weight	25	–	–	–	–	–	–	273.2	342.8	422.0	199	273	243
	tree weight	50	–	–	–	–	–	–	275.3	343.6	422.6	199	273	243
	no clairvoyant	–	–	–	–	–	–	–	273.2	344.7	422.3	199	273	243
	0-restart	–	–	–	–	–	–	–	273.4	343.5	426.1	199	273	243
	monotone	10	–	–	–	–	–	–	2.0	2.0	1.9	1	1	1
	monotone	25	–	–	–	–	–	–	2.0	1.9	1.9	1	1	1
	monotone	50	–	–	–	–	–	–	1.9	1.9	1.9	1	1	1
	reg forest	10	–	–	–	–	–	–	1.9	2.1	1.9	1	1	1
	reg forest	25	–	–	–	–	–	–	1.9	2.0	2.0	1	1	1
	reg forest	50	–	–	–	–	–	–	2.0	2.0	2.0	1	1	1
	gap	10	–	–	–	–	–	–	1.9	2.0	1.9	1	1	1
	gap	25	–	–	–	–	–	–	1.9	2.0	1.9	1	1	1
	gap	50	–	–	–	–	–	–	1.9	2.0	1.9	1	1	1
	leaf freq	10	–	–	–	–	–	–	2.0	2.0	1.9	1	1	1
	leaf freq	25	–	–	–	–	–	–	1.9	2.0	1.9	1	1	1
	leaf freq	50	–	–	–	–	–	–	1.9	2.0	1.9	1	1	1
	ssg	10	–	–	–	–	–	–	2.0	2.0	1.9	1	1	1
	ssg	25	–	–	–	–	–	–	1.9	2.0	1.9	1	1	1
	ssg	50	–	–	–	–	–	–	2.0	2.0	1.9	1	1	1
	tree weight	10	–	–	–	–	–	–	2.0	2.0	1.9	1	1	1
	tree weight	25	–	–	–	–	–	–	1.9	2.0	2.0	1	1	1
	tree weight	50	–	–	–	–	–	–	1.9	2.0	2.0	1	1	1
	no clairvoyant	–	–	–	–	–	–	–	2.0	2.0	1.9	1	1	1
	0-restart	–	–	–	–	–	–	–	2.0	1.9	1.9	1	1	1
drayage-100-23	monotone	10	–	–	–	–	–	–	9.6	14.4	8.8	13	508	27
	monotone	25	–	–	–	–	–	–	9.7	14.4	8.9	13	508	27
	monotone	50	–	–	–	–	–	–	9.7	14.3	8.9	13	508	27
	reg forest	10	–	–	–	–	–	–	9.8	15.0	9.0	13	508	27
	reg forest	25	–	–	–	–	–	–	9.8	14.9	9.1	13	508	27
	reg forest	50	–	–	–	–	–	–	9.8	15.0	9.2	13	508	27
	gap	10	–	–	–	–	–	–	9.6	14.5	8.9	13	508	27
	gap	25	–	–	–	–	–	–	9.7	14.3	8.9	13	508	27
	gap	50	–	–	–	–	–	–	9.7	14.4	8.9	13	508	27
	leaf freq	10	–	–	–	–	–	–	9.7	14.4	8.9	13	508	27
	leaf freq	25	–	–	–	–	–	–	9.7	14.5	8.9	13	508	27
	leaf freq	50	–	–	–	–	–	–	9.7	14.4	8.9	13	508	27
	ssg	10	–	–	–	–	–	–	9.7	14.5	8.8	13	508	27
	ssg	25	–	–	–	–	–	–	9.7	14.5	8.8	13	508	27
	ssg	50	–	–	–	–	–	–	9.7	14.5	8.9	13	508	27
	tree weight	10	–	–	–	–	–	–	9.7	14.4	8.8	13	508	27
	tree weight	25	–	–	–	–	–	–	9.7	14.5	8.9	13	508	27
	tree weight	50	–	–	–	–	–	–	9.7	14.4	8.8	13	508	27
	no clairvoyant	–	–	–	–	–	–	–	9.8	14.4	8.9	13	508	27
	0-restart	–	–	–	–	–	–	–	9.6	14.3	8.9	13	508	27
drayage-25-23	monotone	10	–	–	–	–	–	–	t	t	t	356760	489746	360735
	monotone	25	–	–	–	–	–	–	t	t	t	358448	492928	359715
	monotone	50	–	–	–	–	–	–	t	t	t	357767	496083	360014
	reg forest	10	–	–	–	–	–	–	t	t	t	358091	489208	358140
	reg forest	25	–	–	–	–	–	–	t	t	t	357894	486820	357322
	reg forest	50	–	–	–	–	–	–	t	t	t	357030	491564	356535
	gap	10	–	–	–	–	–	–	t	t	t	357323	490720	359559
	gap	25	–	–	–	–	–	–	t	t	t	356838	491564	360053
	gap	50	–	–	–	–	–	–	t	t	t	357856	490993	358733
	leaf freq	10	–	–	–	–	–	–	t	t	t	356377	490037	359952
	leaf freq	25	–	–	–	–	–	–	t	t	t	357068	492117	359329
	leaf freq	50	–	–	–	–	–	–	t	t	t	357501	492572	359237
	ssg	10	–	–	–	–	–	–	t	t	t	357394	491908	362185
	ssg	25	–	–	–	–	–	–	t	t	t	357281	493263	359137
	ssg	50	–	–	–	–	–	–	t	t	t	356617	491536	361150
	tree weight	10	–	–	–	–	–	–	t	t	t	357107	492184	360468
	tree weight	25	–	–	–	–	–	–	t	t	t	357251	492089	357715
	tree weight	50	–	–	–	–	–	–	t	t	t	358099	493480	360790
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	358117	494358	360149
	0-restart	–	–	–	–	–	–	–	t	t	t	357689	494513	360647

cont. on next page ...

E. Appendix E

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
dws008-01	monotone	10	–	–	–	1	1	1	t	t	t	172779	176613	222024
	monotone	25	–	–	–	1	1	1	t	t	t	175577	177095	218209
	monotone	50	–	–	–	–	1	–	t	t	t	175652	178070	194687
	reg forest	10	–	–	–	1	1	1	t	t	t	172845	177298	219945
	reg forest	25	–	–	–	–	1	–	t	t	t	175118	207596	189843
	reg forest	50	–	–	–	–	–	–	t	t	t	175484	170798	192555
	gap	10	–	–	–	1	1	1	t	t	t	215677	199495	244919
	gap	25	–	–	–	1	1	1	t	t	t	215403	203269	226631
	gap	50	–	–	–	1	1	1	t	t	t	234687	212591	227559
	leaf freq	10	–	–	–	1	1	1	t	t	t	203489	224568	204431
	leaf freq	25	–	–	–	1	1	1	t	t	t	202644	226071	175181
	leaf freq	50	–	–	–	1	1	1	t	t	t	196163	202570	174473
	ssg	10	–	–	–	1	1	1	t	t	t	227643	184768	206177
	ssg	25	–	–	–	1	1	1	t	t	t	250750	184846	211580
	ssg	50	–	–	–	1	1	1	t	t	t	198221	165607	209594
	tree weight	10	–	–	–	1	1	1	t	t	t	188885	175099	217672
	tree weight	25	–	–	–	1	1	1	t	t	t	190877	206125	192873
	tree weight	50	–	–	–	1	1	1	t	t	t	221414	188358	220965
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	175011	172152	195273
	0-restart	–	–	–	–	–	–	–	t	t	t	176384	173322	196056
eil33-2	monotone	10	–	–	–	–	–	–	78.0	77.5	65.7	683	679	521
	monotone	25	–	–	–	–	–	–	78.4	77.3	65.5	683	679	521
	monotone	50	–	–	–	–	–	–	78.0	77.6	65.7	683	679	521
	reg forest	10	–	–	–	–	–	–	78.5	77.6	66.0	683	679	521
	reg forest	25	–	–	–	–	–	–	78.3	77.3	65.4	683	679	521
	reg forest	50	–	–	–	–	–	–	78.8	77.9	66.1	683	679	521
	gap	10	–	–	–	–	–	–	78.2	77.6	65.7	683	679	521
	gap	25	–	–	–	–	–	–	77.8	77.0	66.5	683	679	521
	gap	50	–	–	–	–	–	–	78.3	77.6	65.6	683	679	521
	leaf freq	10	–	–	–	–	–	–	78.2	77.6	65.5	683	679	521
	leaf freq	25	–	–	–	–	–	–	78.2	77.4	65.0	683	679	521
	leaf freq	50	–	–	–	–	–	–	77.9	77.5	66.2	683	679	521
	ssg	10	–	–	–	–	–	–	77.5	77.4	65.7	683	679	521
	ssg	25	–	–	–	–	–	–	77.8	77.8	65.8	683	679	521
	ssg	50	–	–	–	–	–	–	77.9	77.4	66.2	683	679	521
	tree weight	10	–	–	–	–	–	–	77.8	77.5	65.6	683	679	521
	tree weight	25	–	–	–	–	–	–	77.9	77.6	66.1	683	679	521
	tree weight	50	–	–	–	–	–	–	77.7	77.7	65.7	683	679	521
	no clairvoyant	–	–	–	–	–	–	–	78.3	77.4	65.7	683	679	521
	0-restart	–	–	–	–	–	–	–	77.8	77.3	66.0	683	679	521
eilA101-2	monotone	10	–	–	–	–	–	–	t	t	t	3626	3500	2956
	monotone	25	–	–	–	–	–	–	t	t	t	3641	3489	3180
	monotone	50	–	–	–	–	–	–	t	t	t	3626	3505	3191
	reg forest	10	–	–	–	–	1	–	t	t	t	3639	2934	3085
	reg forest	25	–	–	–	–	–	–	t	t	t	3627	3428	2959
	reg forest	50	–	–	–	–	–	–	t	t	t	3720	3459	3179
	gap	10	–	–	–	1	1	1	t	t	t	2716	3444	2273
	gap	25	–	–	–	–	–	1	t	t	t	3639	3505	3160
	gap	50	–	–	–	–	–	–	t	t	t	3639	3428	3191
	leaf freq	10	–	–	–	–	–	–	t	t	t	3639	3469	3128
	leaf freq	25	–	–	–	–	–	–	t	t	t	3662	3522	3094
	leaf freq	50	–	–	–	–	–	–	t	t	t	3561	3502	3004
	ssg	10	–	–	–	–	–	–	t	t	t	3639	3519	3178
	ssg	25	–	–	–	–	–	–	t	t	t	3545	3504	3170
	ssg	50	–	–	–	–	–	–	t	t	t	3679	3565	3234
	tree weight	10	–	–	–	1	–	1	t	t	t	2804	3649	3135
	tree weight	25	–	–	–	1	–	1	t	t	t	2854	3501	3135
	tree weight	50	–	–	–	1	–	1	t	t	t	2719	3521	3153
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	3749	3531	3114
	0-restart	–	–	–	–	–	–	–	t	t	t	3721	3576	3177
enlight_hard	monotone	10	–	–	–	–	–	–	0.5	0.5	0.5	1	1	1
	monotone	25	–	–	–	–	–	–	0.5	0.5	0.5	1	1	1
	monotone	50	–	–	–	–	–	–	0.5	0.5	0.5	1	1	1
	reg forest	10	–	–	–	–	–	–	0.5	0.5	0.5	1	1	1
	reg forest	25	–	–	–	–	–	–	0.5	0.5	0.5	1	1	1
	reg forest	50	–	–	–	–	–	–	0.5	0.5	0.5	1	1	1
	gap	10	–	–	–	–	–	–	0.5	0.5	0.5	1	1	1
	gap	25	–	–	–	–	–	–	0.5	0.5	0.5	1	1	1
	gap	50	–	–	–	–	–	–	0.5	0.5	0.5	1	1	1
	leaf freq	10	–	–	–	–	–	–	0.5	0.5	0.5	1	1	1
	leaf freq	25	–	–	–	–	–	–	0.5	0.5	0.5	1	1	1
	leaf freq	50	–	–	–	–	–	–	0.5	0.5	0.5	1	1	1
	ssg	10	–	–	–	–	–	–	0.5	0.5	0.5	1	1	1
	ssg	25	–	–	–	–	–	–	0.5	0.5	0.5	1	1	1
	ssg	50	–	–	–	–	–	–	0.5	0.5	0.5	1	1	1
	tree weight	10	–	–	–	–	–	–	0.5	0.5	0.5	1	1	1

cont. on next page ...

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
ex10	tree weight	25	–	–	–	–	–	–	0.5	0.5	0.5	1	1	1
	tree weight	50	–	–	–	–	–	–	0.5	0.5	0.5	1	1	1
	no clairvoyant	–	–	–	–	–	–	–	0.5	0.5	0.5	1	1	1
	0-restart	–	–	–	–	–	–	–	0.5	0.5	0.5	1	1	1
	monotone	10	–	–	–	–	–	–	119.7	120.6	128.1	1	1	1
	monotone	25	–	–	–	–	–	–	119.8	119.5	121.4	1	1	1
	monotone	50	–	–	–	–	–	–	119.4	119.6	120.4	1	1	1
	reg forest	10	–	–	–	–	–	–	118.0	120.3	120.7	1	1	1
	reg forest	25	–	–	–	–	–	–	120.3	119.5	122.1	1	1	1
	reg forest	50	–	–	–	–	–	–	119.5	119.2	120.3	1	1	1
	gap	10	–	–	–	–	–	–	118.2	119.6	122.2	1	1	1
	gap	25	–	–	–	–	–	–	119.4	120.2	122.5	1	1	1
	gap	50	–	–	–	–	–	–	119.2	119.2	121.3	1	1	1
	leaf freq	10	–	–	–	–	–	–	119.9	119.2	120.6	1	1	1
	leaf freq	25	–	–	–	–	–	–	120.6	119.5	120.0	1	1	1
	leaf freq	50	–	–	–	–	–	–	119.1	119.9	121.1	1	1	1
	ssg	10	–	–	–	–	–	–	117.2	119.5	120.8	1	1	1
	ssg	25	–	–	–	–	–	–	119.3	119.1	121.2	1	1	1
	ssg	50	–	–	–	–	–	–	120.0	118.8	121.6	1	1	1
	tree weight	10	–	–	–	–	–	–	120.1	119.9	122.1	1	1	1
ex9	tree weight	25	–	–	–	–	–	–	119.9	119.2	121.4	1	1	1
	tree weight	50	–	–	–	–	–	–	119.0	119.3	121.0	1	1	1
	no clairvoyant	–	–	–	–	–	–	–	119.8	118.4	121.5	1	1	1
	0-restart	–	–	–	–	–	–	–	119.4	118.5	122.5	1	1	1
	monotone	10	–	–	–	–	–	–	11.6	10.8	11.3	1	1	1
	monotone	25	–	–	–	–	–	–	11.6	10.8	11.3	1	1	1
	monotone	50	–	–	–	–	–	–	11.8	10.8	11.4	1	1	1
	reg forest	10	–	–	–	–	–	–	11.6	10.7	11.4	1	1	1
	reg forest	25	–	–	–	–	–	–	11.5	10.9	11.2	1	1	1
	reg forest	50	–	–	–	–	–	–	11.8	10.7	11.4	1	1	1
	gap	10	–	–	–	–	–	–	11.6	10.9	11.3	1	1	1
	gap	25	–	–	–	–	–	–	11.8	10.7	11.3	1	1	1
	gap	50	–	–	–	–	–	–	11.7	10.7	11.3	1	1	1
	leaf freq	10	–	–	–	–	–	–	11.7	10.8	11.3	1	1	1
	leaf freq	25	–	–	–	–	–	–	11.5	10.9	11.3	1	1	1
	leaf freq	50	–	–	–	–	–	–	11.7	10.8	11.3	1	1	1
	ssg	10	–	–	–	–	–	–	11.5	10.6	11.2	1	1	1
	ssg	25	–	–	–	–	–	–	11.5	10.9	11.5	1	1	1
	ssg	50	–	–	–	–	–	–	11.7	10.8	11.2	1	1	1
	tree weight	10	–	–	–	–	–	–	11.6	10.7	11.3	1	1	1
exp-1-500-5-5	tree weight	25	–	–	–	–	–	–	11.2	10.9	11.3	1	1	1
	tree weight	50	–	–	–	–	–	–	11.7	10.8	11.3	1	1	1
	no clairvoyant	–	–	–	–	–	–	–	11.7	10.5	11.3	1	1	1
	0-restart	–	–	–	–	–	–	–	11.5	10.9	11.4	1	1	1
	monotone	10	–	–	–	–	–	–	2.9	2.9	2.7	1	1	1
	monotone	25	–	–	–	–	–	–	2.8	2.9	2.7	1	1	1
	monotone	50	–	–	–	–	–	–	2.8	2.9	2.7	1	1	1
	reg forest	10	–	–	–	–	–	–	2.9	2.9	2.8	1	1	1
	reg forest	25	–	–	–	–	–	–	2.9	2.9	2.8	1	1	1
	reg forest	50	–	–	–	–	–	–	3.0	3.0	2.8	1	1	1
	gap	10	–	–	–	–	–	–	2.8	2.9	2.7	1	1	1
	gap	25	–	–	–	–	–	–	2.8	2.9	2.7	1	1	1
	gap	50	–	–	–	–	–	–	2.8	2.9	2.7	1	1	1
	leaf freq	10	–	–	–	–	–	–	2.8	2.9	2.6	1	1	1
	leaf freq	25	–	–	–	–	–	–	2.8	2.9	2.7	1	1	1
	leaf freq	50	–	–	–	–	–	–	2.8	2.9	2.7	1	1	1
	ssg	10	–	–	–	–	–	–	2.8	2.9	2.7	1	1	1
	ssg	25	–	–	–	–	–	–	2.8	2.9	2.7	1	1	1
	ssg	50	–	–	–	–	–	–	2.8	2.9	2.7	1	1	1
	tree weight	10	–	–	–	–	–	–	2.8	2.9	2.7	1	1	1
fast0507	tree weight	25	–	–	–	–	–	–	2.8	2.9	2.7	1	1	1
	tree weight	50	–	–	–	–	–	–	2.9	2.9	2.7	1	1	1
	no clairvoyant	–	–	–	–	–	–	–	2.9	2.9	2.7	1	1	1
	0-restart	–	–	–	–	–	–	–	2.8	2.9	2.7	1	1	1
	monotone	10	–	–	–	–	–	–	217.6	135.2	123.8	830	602	496
	monotone	25	–	–	–	–	–	–	218.8	135.8	123.3	830	602	496
	monotone	50	–	–	–	–	–	–	218.2	135.4	123.2	830	602	496
	reg forest	10	–	–	–	–	–	–	217.3	136.3	124.5	830	602	496
	reg forest	25	–	–	–	–	–	–	217.8	136.2	123.5	830	602	496
	reg forest	50	–	–	–	–	–	–	218.9	135.8	124.0	830	602	496
	gap	10	–	–	–	–	–	–	217.7	136.4	123.8	830	602	496
	gap	25	–	–	–	–	–	–	218.8	135.8	123.9	830	602	496
	gap	50	–	–	–	–	–	–	217.4	135.8	123.8	830	602	496
	leaf freq	10	–	–	–	–	–	–	217.3	135.5	123.5	830	602	496
	leaf freq	25	–	–	–	–	–	–	218.1	135.8	123.9	830	602	496
	leaf freq	50	–	–	–	–	–	–	217.5	136.3	124.2	830	602	496

cont. on next page ...

E. Appendix E

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
fastxgemm-n2.	ssg	10	–	–	–	–	–	–	217.2	135.1	123.9	830	602	496
	ssg	25	–	–	–	–	–	–	217.2	137.0	123.5	830	602	496
	ssg	50	–	–	–	–	–	–	218.8	135.8	123.5	830	602	496
	tree weight	10	–	–	–	–	–	–	216.7	135.3	122.4	830	602	496
	tree weight	25	–	–	–	–	–	–	215.7	135.8	123.9	830	602	496
	tree weight	50	–	–	–	–	–	–	218.5	135.7	124.1	830	602	496
	no clairvoyant	–	–	–	–	–	–	–	217.3	136.0	123.8	830	602	496
	0-restart	–	–	–	–	–	–	–	216.7	135.8	123.5	830	602	496
	monotone	10	–	–	–	–	–	–	420.7	469.0	767.2	62903	84400	143836
	monotone	25	–	–	–	–	–	–	421.4	467.1	768.1	62903	84400	143836
	monotone	50	–	–	–	–	–	–	420.2	466.8	767.0	62903	84400	143836
	reg forest	10	–	–	–	–	–	–	421.4	467.6	770.3	62903	84400	143836
	reg forest	25	–	–	–	–	–	–	419.0	469.0	770.2	62903	84400	143836
	reg forest	50	–	–	–	–	–	–	420.6	470.3	769.0	62903	84400	143836
	gap	10	–	–	–	–	–	–	420.4	466.8	768.1	62903	84400	143836
	gap	25	–	–	–	–	–	–	420.2	468.1	767.8	62903	84400	143836
	gap	50	–	–	–	–	–	–	420.6	468.1	765.9	62903	84400	143836
	leaf freq	10	–	–	–	–	–	–	420.5	466.8	767.7	62903	84400	143836
	leaf freq	25	–	–	–	–	–	–	420.6	464.9	768.3	62903	84400	143836
	leaf freq	50	–	–	–	–	–	–	419.1	468.2	767.5	62903	84400	143836
	ssg	10	–	–	–	–	–	–	420.4	466.1	769.3	62903	84400	143836
	ssg	25	–	–	–	–	–	–	421.8	467.2	770.5	62903	84400	143836
	ssg	50	–	–	–	–	–	–	420.5	468.0	766.0	62903	84400	143836
	tree weight	10	–	–	–	–	–	–	419.7	466.6	768.6	62903	84400	143836
	tree weight	25	–	–	–	–	–	–	420.4	467.1	767.0	62903	84400	143836
	tree weight	50	–	–	–	–	–	–	418.4	466.4	766.5	62903	84400	143836
	no clairvoyant	–	–	–	–	–	–	–	420.0	467.2	767.4	62903	84400	143836
	0-restart	–	–	–	–	–	–	–	420.2	467.9	768.2	62903	84400	143836
fhnw-binpack4-4	monotone	10	–	–	–	–	–	–	t	t	t	6243623	6282879	5406245
	monotone	25	–	–	–	–	–	–	t	t	t	6250478	6262326	5409306
	monotone	50	–	–	–	–	–	–	t	t	t	6254291	6268018	5404104
	reg forest	10	–	–	–	–	–	–	t	t	t	6258309	6283028	5404104
	reg forest	25	–	–	–	–	–	–	t	t	t	6252032	6251808	5392934
	reg forest	50	–	–	–	–	–	–	t	t	t	6237981	6277601	5418045
	gap	10	–	–	–	–	–	–	t	t	t	6252311	6270369	5411184
	gap	25	–	–	–	–	–	–	t	t	t	6246810	6253860	5412706
	gap	50	–	–	–	–	–	–	t	t	t	6241007	6255068	5407221
	leaf freq	10	–	–	–	–	–	–	t	t	t	6224769	6285341	5399459
	leaf freq	25	–	–	–	–	–	–	t	t	t	6220939	6258647	5440275
	leaf freq	50	–	–	–	–	–	–	t	t	t	6246172	6267470	5437873
	ssg	10	–	–	–	–	–	–	t	t	t	6232747	6279769	5401995
	ssg	25	–	–	–	–	–	–	t	t	t	6256504	6282249	5419058
	ssg	50	–	–	–	–	–	–	t	t	t	6296999	6267178	5412121
	tree weight	10	–	–	–	–	–	–	t	t	t	6250931	6279698	5407618
	tree weight	25	–	–	–	–	–	–	t	t	t	6287091	6266622	5412121
	tree weight	50	–	–	–	–	–	–	t	t	t	6234525	6250111	5399607
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	6253073	6277037	5412418
	0-restart	–	–	–	–	–	–	–	t	t	t	6234074	6294595	5392178
fhnw-binpack4-48	monotone	10	–	–	–	1	1	1	t	t	t	2275616	2277178	2445216
	monotone	25	–	–	–	1	1	1	t	t	t	2284956	2277559	2476865
	monotone	50	–	–	–	1	1	1	t	t	t	2269827	2274133	2453101
	reg forest	10	–	–	–	–	–	–	t	t	t	2689876	2602183	2027941
	reg forest	25	–	–	–	–	–	–	t	t	t	2701143	2593989	2048715
	reg forest	50	–	–	–	–	–	–	t	t	t	2697389	2609735	2029807
	gap	10	–	–	–	–	–	–	t	t	t	2737061	2639298	2046253
	gap	25	–	–	–	–	–	–	t	t	t	2723300	2648962	2037415
	gap	50	–	–	–	–	–	–	t	t	t	2723990	2640850	2055948
	leaf freq	10	–	–	–	1	1	1	t	t	t	2655323	2493667	2302298
	leaf freq	25	–	–	–	–	–	–	t	t	t	2716315	2636914	2053971
	leaf freq	50	–	–	–	–	–	–	t	t	t	2733288	2632067	2050114
	ssg	10	–	–	–	–	–	–	t	t	t	2719646	2640365	2056165
	ssg	25	–	–	–	–	–	–	t	t	t	2735828	2628089	2053840
	ssg	50	–	–	–	–	–	–	t	t	t	2725127	2642433	2057528
	tree weight	10	–	–	–	–	–	–	t	t	t	2724742	2630352	2051490
	tree weight	25	–	–	–	–	–	–	t	t	t	2716274	2643142	2055141
	tree weight	50	–	–	–	–	–	–	t	t	t	2722098	2648329	2063627
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	2721765	2638444	2058448
	0-restart	–	–	–	–	–	–	–	t	t	t	2728994	2642014	2056416
fiball	monotone	10	–	–	–	1	1	1	217.7	429.8	486.0	3764	10841	13373
	monotone	25	–	–	–	1	1	–	217.5	430.9	385.3	3764	10841	10437
	monotone	50	–	–	–	1	1	–	218.0	432.0	384.8	3764	10841	10437
	reg forest	10	–	–	–	1	1	1	215.5	434.7	485.0	3764	10841	13373
	reg forest	25	–	–	–	–	–	–	190.2	498.4	387.3	3605	16051	10437
	reg forest	50	–	–	–	–	–	–	190.2	497.4	385.9	3605	16051	10437
	gap	10	–	–	–	–	–	–	187.0	495.9	385.4	3605	16051	10437
	gap	25	–	–	–	–	–	–	190.3	494.3	381.6	3605	16051	10437

cont. on next page ...

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
gen-ip002	gap	50	–	–	–	–	–	–	189.4	493.3	383.1	3605	16051	10437
	leaf freq	10	–	–	–	1	1	1	312.1	470.1	468.2	8225	11896	11726
	leaf freq	25	–	–	–	–	1	–	190.7	473.1	383.7	3605	11896	10437
	leaf freq	50	–	–	–	–	–	–	189.0	495.0	382.3	3605	16051	10437
	ssg	10	–	–	–	1	1	1	218.1	431.6	482.9	3764	10841	13373
	ssg	25	–	–	–	1	1	1	217.3	289.7	483.2	3764	4377	13373
	ssg	50	–	–	–	1	1	1	216.7	388.5	481.6	3764	7408	13373
	tree weight	10	–	–	–	1	1	1	217.1	434.1	478.8	3764	10841	13373
	tree weight	25	–	–	–	1	1	1	217.0	400.4	479.8	3764	11040	13373
	tree weight	50	–	–	–	1	1	1	216.8	404.5	483.7	3764	8193	13373
	no clairvoyant	–	–	–	–	–	–	–	190.3	495.7	383.6	3605	16051	10437
	0-restart	–	–	–	–	–	–	–	189.4	494.2	381.0	3605	16051	10437
	monotone	10	–	–	–	1	1	1	1494.4	1477.5	1428.5	5304760	5110974	5117244
	monotone	25	–	–	–	1	1	1	1496.6	1471.3	1438.8	5304760	5110974	5117244
	monotone	50	–	–	–	1	1	1	1504.5	1472.5	1438.0	5304760	5110974	5117244
	reg forest	10	–	–	–	1	1	1	1507.1	1499.8	1441.5	5304760	5110974	5117244
	reg forest	25	–	–	–	–	–	–	1854.6	1897.9	1666.1	6014978	5957779	5489120
	reg forest	50	–	–	–	–	–	–	1851.4	1910.5	1668.9	6014978	5957779	5489120
	gap	10	–	–	–	–	–	–	1682.8	1753.3	1509.0	6014978	5957779	5489120
	gap	25	–	–	–	–	–	–	1691.0	1744.5	1515.7	6014978	5957779	5489120
	gap	50	–	–	–	–	–	–	1682.1	1746.0	1519.0	6014978	5957779	5489120
	leaf freq	10	–	–	–	1	1	1	1536.4	1390.1	1550.4	5197947	4907681	5522119
	leaf freq	25	–	–	–	1	1	1	1328.3	1382.9	1450.5	4620335	4907681	5165450
	leaf freq	50	–	–	–	1	1	1	1466.7	1412.3	1584.8	4958468	4943804	5224880
	ssg	10	–	–	–	1	1	1	1328.1	1510.5	1426.7	4599760	5310018	5107698
	ssg	25	–	–	–	1	1	1	1344.9	1363.7	1425.8	4826759	4916885	5086373
	ssg	50	–	–	–	1	1	1	1478.5	1359.5	1346.8	5083939	4890126	4826386
	tree weight	10	–	–	–	1	1	1	1491.1	1468.2	1607.9	5304760	5110974	5718642
	tree weight	25	–	–	–	1	1	1	1500.8	1473.3	1400.5	5304760	5110974	5071121
	tree weight	50	–	–	–	1	1	1	1490.3	1470.6	1342.3	5304760	5110974	4875727
	no clairvoyant	–	–	–	–	–	–	–	1699.4	1748.6	1513.1	6014978	5957779	5489120
	0-restart	–	–	–	–	–	–	–	1692.4	1740.8	1516.1	6014978	5957779	5489120
gen-ip054	monotone	10	–	–	–	1	1	1	1396.1	892.2	1766.3	5558155	3517217	7000324
	monotone	25	–	–	–	1	1	1	1408.2	894.8	1765.5	5558155	3517217	7000324
	monotone	50	–	–	–	1	1	1	1410.6	892.5	1762.3	5558155	3517217	7000324
	reg forest	10	–	–	–	1	1	1	1401.5	1096.9	1773.8	5558155	4393243	7000324
	reg forest	25	–	–	–	1	1	–	1437.0	1105.5	2520.6	4505940	3403496	8978468
	reg forest	50	–	–	–	–	–	–	t	1432.6	2536.5	12572935	5140274	8978468
	gap	10	–	–	–	1	–	1	1411.5	1297.7	1782.6	5558155	5140274	7000324
	gap	25	–	–	–	1	–	1	1397.1	1295.1	1758.4	5558155	5140274	7000324
	gap	50	–	–	–	1	–	1	1401.5	1299.3	1764.7	5558155	5140274	7000324
	leaf freq	10	–	–	–	1	1	1	1404.6	840.0	1760.4	5558155	3394610	7000324
	leaf freq	25	–	–	–	1	1	1	1400.1	1243.9	1753.8	5558155	4612063	7000324
	leaf freq	50	–	–	–	1	1	1	1393.3	1734.5	1755.0	5558155	5184114	7000324
	ssg	10	–	–	–	1	1	1	1405.2	1212.9	1758.6	5558155	4726000	7000324
	ssg	25	–	–	–	1	1	1	1405.4	737.6	1768.4	5558155	3001275	7000324
	ssg	50	–	–	–	1	1	1	1402.2	888.9	1772.7	5558155	3589939	7000324
	tree weight	10	–	–	–	1	1	1	1392.6	892.6	1755.2	5558155	3517217	7000324
	tree weight	25	–	–	–	1	1	1	1407.5	893.8	1771.0	5558155	3517217	7000324
	tree weight	50	–	–	–	1	1	1	1411.6	896.3	1777.6	5558155	3517217	7000324
	no clairvoyant	–	–	–	–	–	–	–	3529.5	1297.4	2298.9	13474791	5140274	8978468
	0-restart	–	–	–	–	–	–	–	3516.5	1294.9	2312.1	13474791	5140274	8978468
germanrr	monotone	10	–	–	–	1	–	1	t	t	t	1207	1024	1175
	monotone	25	–	–	–	1	–	1	t	t	t	1225	1024	1175
	monotone	50	–	–	–	1	–	1	t	t	t	1226	1028	1175
	reg forest	10	–	–	–	–	–	1	t	t	t	1263	1024	1182
	reg forest	25	–	–	–	–	–	–	t	t	t	1263	1024	1173
	reg forest	50	–	–	–	–	–	–	t	t	t	1257	1025	1173
	gap	10	–	–	–	–	–	–	t	t	t	1270	1024	1173
	gap	25	–	–	–	–	–	–	t	t	t	1273	1027	1168
	gap	50	–	–	–	–	–	–	t	t	t	1271	1028	1170
	leaf freq	10	–	–	–	–	–	–	t	t	t	1271	1024	1178
	leaf freq	25	–	–	–	–	–	–	t	t	t	1268	1024	1182
	leaf freq	50	–	–	–	–	–	–	t	t	t	1270	1024	1174
	ssg	10	–	–	–	–	–	–	t	t	t	1270	1024	1170
	ssg	25	–	–	–	–	–	–	t	t	t	1270	1024	1170
	ssg	50	–	–	–	–	–	–	t	t	t	1270	1024	1168
	tree weight	10	–	–	–	1	–	–	t	t	t	1219	1033	1173
	tree weight	25	–	–	–	1	–	–	t	t	t	1230	1024	1170
	tree weight	50	–	–	–	1	–	–	t	t	t	1207	1027	1175
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	1272	1026	1175
	0-restart	–	–	–	–	–	–	–	t	t	t	1273	1024	1170
gfd-schedulen18.	monotone	10	–	–	–	–	–	–	t	t	t	1	1	1
	monotone	25	–	–	–	–	–	–	t	t	t	1	1	1
	monotone	50	–	–	–	–	–	–	t	t	t	1	1	1
	reg forest	10	–	–	–	–	–	–	t	t	t	1	1	1

cont. on next page ...

E. Appendix E

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
glass-sc	reg forest	25	–	–	–	–	–	–	t	t	t	1	1	1
	reg forest	50	–	–	–	–	–	–	t	t	t	1	1	1
	gap	10	–	–	–	–	–	–	t	t	t	1	1	1
	gap	25	–	–	–	–	–	–	t	t	t	1	1	1
	gap	50	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	10	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	25	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	50	–	–	–	–	–	–	t	t	t	1	1	1
	ssg	10	–	–	–	–	–	–	t	t	t	1	1	1
	ssg	25	–	–	–	–	–	–	t	t	t	1	1	1
	ssg	50	–	–	–	–	–	–	t	t	t	1	1	1
	tree weight	10	–	–	–	–	–	–	t	t	t	1	1	1
	tree weight	25	–	–	–	–	–	–	t	t	t	1	1	1
	tree weight	50	–	–	–	–	–	–	t	t	t	1	1	1
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	1	1	1
	0-restart	–	–	–	–	–	–	–	t	t	t	1	1	1
	monotone	10	–	–	–	–	–	–	t	t	3047.7	288128	273760	248382
	monotone	25	–	–	–	–	–	–	t	t	3041.9	286305	272083	248382
	monotone	50	–	–	–	–	–	–	t	t	3039.3	289391	272781	248382
	reg forest	10	–	–	–	1	1	1	t	3393.1	3104.2	284307	270103	247228
	reg forest	25	–	–	–	–	–	–	t	t	3073.7	288518	271749	248382
	reg forest	50	–	–	–	–	–	–	t	t	3062.9	285932	271281	248382
	gap	10	–	–	–	1	1	1	t	t	3124.3	275001	277210	242297
	gap	25	–	–	–	1	1	1	t	t	3143.8	267619	263188	241542
	gap	50	–	–	–	–	–	–	t	t	3055.0	287388	271282	248382
	leaf freq	10	–	–	–	1	1	1	t	t	3501.5	287831	272986	260031
	leaf freq	25	–	–	–	1	1	1	t	t	3570.0	260897	236325	261887
	leaf freq	50	–	–	–	–	1	1	t	t	t	288971	232554	258519
	ssg	10	–	–	–	1	1	1	t	t	3122.1	281445	273745	247228
	ssg	25	–	–	–	1	1	1	t	t	3135.7	278260	273067	243863
	ssg	50	–	–	–	1	1	1	t	t	3346.8	286169	266784	251918
	tree weight	10	–	–	–	1	1	1	t	t	3209.7	288151	284136	257764
	tree weight	25	–	–	–	1	1	1	t	t	3201.4	286487	284012	257764
	tree weight	50	–	–	–	1	1	1	t	t	3192.9	289234	284711	257764
	no clairvoyant	–	–	–	–	–	–	–	t	t	3038.6	288879	273109	248382
	0-restart	–	–	–	–	–	–	–	t	t	3048.6	287351	272891	248382
glass4	monotone	10	–	–	–	1	1	1	1507.6	881.0	298.0	1667231	1052626	319641
	monotone	25	–	–	–	1	1	1	1506.4	885.9	298.6	1667231	1052626	319641
	monotone	50	–	–	–	1	1	–	1494.8	884.8	1471.9	1667231	1052626	1238074
	reg forest	10	–	–	–	1	1	1	1499.1	896.4	303.4	1667231	1052626	319641
	reg forest	25	–	–	–	1	–	1	209.3	1044.8	2865.0	238260	860747	1910450
	reg forest	50	–	–	–	–	–	–	t	1050.5	1529.1	2891512	860747	1238074
	gap	10	–	–	–	1	1	1	812.9	470.4	298.2	663718	473085	319641
	gap	25	–	–	–	1	1	1	552.1	142.8	950.1	702057	135695	996884
	gap	50	–	–	–	1	1	1	327.6	97.2	761.8	324833	102699	675509
	leaf freq	10	–	–	–	1	1	1	201.6	125.9	99.8	207587	134142	77448
	leaf freq	25	–	–	–	1	1	1	178.7	126.4	99.6	178408	121287	77448
	leaf freq	50	–	–	–	1	1	1	159.6	127.0	154.0	143914	121287	138448
	ssg	10	–	–	–	1	1	1	2049.9	150.5	299.0	2560444	140046	319641
	ssg	25	–	–	–	1	1	1	2048.3	150.2	296.7	2560444	140046	319641
	ssg	50	–	–	–	1	1	1	2041.3	150.0	298.1	2560444	140046	319641
	tree weight	10	–	–	–	1	1	1	1510.0	327.8	1429.7	1667231	326944	1348284
	tree weight	25	–	–	–	1	1	1	562.9	328.9	960.0	548156	326944	996884
	tree weight	50	–	–	–	1	1	1	326.3	327.3	764.7	324833	326944	675509
	no clairvoyant	–	–	–	–	–	–	–	t	994.5	1463.8	2987288	860747	1238074
	0-restart	–	–	–	–	–	–	–	t	1003.5	1462.7	3017781	860747	1238074
gmu-35-40	monotone	10	1	1	2	1	1	1	t	t	t	2721498	1441413	2215662
	monotone	25	1	1	2	1	1	1	t	t	t	2723798	1440608	2218096
	monotone	50	1	1	2	1	1	1	t	t	t	2719677	1442039	2218801
	reg forest	10	1	1	2	–	1	1	t	t	t	1161669	1442156	2216517
	reg forest	25	1	1	2	–	–	–	t	t	t	1155232	1913242	1896168
	reg forest	50	1	1	2	–	–	–	t	t	t	1160395	1906566	1898175
	gap	10	1	1	2	–	–	–	t	t	t	1170500	1942751	1917879
	gap	25	1	1	2	–	–	–	t	t	t	1169802	1935687	1921600
	gap	50	1	1	2	–	–	–	t	t	t	1170360	1928665	1926627
	leaf freq	10	1	1	2	1	1	1	t	t	t	1498975	2239871	2230463
	leaf freq	25	1	1	2	1	1	1	t	t	t	1905388	1516544	2231113
	leaf freq	50	1	1	2	1	1	1	t	t	t	1917687	1125229	2217542
	ssg	10	1	1	2	1	1	1	t	t	t	2722781	1400190	2221456
	ssg	25	1	1	2	1	1	1	t	t	t	2720419	1406276	2226743
	ssg	50	1	1	2	1	1	1	t	t	t	2723355	1407947	2220843
	tree weight	10	1	1	2	1	1	1	t	t	t	2150378	1515108	2178451
	tree weight	25	1	1	2	1	1	1	t	t	t	1899417	1517369	2187844
	tree weight	50	1	1	2	1	1	1	t	t	t	2444465	1517588	2178199
	no clairvoyant	–	1	1	2	–	–	–	t	t	t	1169900	1938453	1927386
	0-restart	–	–	–	–	–	–	–	t	t	t	957490	1591349	1603377

cont. on next page ...

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
gmu-35-50	monotone	10	1	1	1	1	1	1	t	t	t	1250153	897585	734239
	monotone	25	1	1	1	1	1	1	t	t	t	1250075	901716	734225
	monotone	50	1	1	1	1	1	1	t	t	t	1250282	902948	737249
	reg forest	10	1	1	1	1	-	-	t	t	t	831983	857478	1049171
	reg forest	25	1	1	1	-	-	-	t	t	t	1153134	858289	1049592
	reg forest	50	1	1	1	-	-	-	t	t	t	1151038	857927	1050412
	gap	10	1	1	1	-	-	-	t	t	t	1159330	865427	1059243
	gap	25	1	1	1	-	-	-	t	t	t	1161319	865427	1061784
	gap	50	1	1	1	-	-	-	t	t	t	1167679	867292	1061366
	leaf freq	10	1	1	1	1	1	1	t	t	t	791438	1137797	757751
	leaf freq	25	1	1	1	1	1	1	t	t	t	726430	926581	894221
	leaf freq	50	1	1	1	1	1	1	t	t	t	725524	856948	966145
	ssg	10	1	1	1	1	1	1	t	t	t	791756	1044159	810440
	ssg	25	1	1	1	1	1	1	t	t	t	1036353	951270	1792751
	ssg	50	1	1	1	1	1	1	t	t	t	806040	823965	927843
	tree weight	10	1	1	1	1	1	1	t	t	t	1246957	900756	735057
	tree weight	25	1	1	1	1	1	1	t	t	t	1247152	900756	732692
	tree weight	50	1	1	1	1	1	1	t	t	t	1255570	901167	733334
	no clairvoyant	-	1	1	1	-	-	-	t	t	t	1162516	864560	1057550
	0-restart	-	-	-	-	-	-	-	t	t	t	838559	737428	959637
graph20-20-1rand	monotone	10	-	-	-	-	-	-	5.2	7.8	7.2	151	285	302
	monotone	25	-	-	-	-	-	-	5.2	7.8	7.2	151	285	302
	monotone	50	-	-	-	-	-	-	5.1	7.8	7.3	151	285	302
	reg forest	10	-	-	-	-	-	-	5.6	8.3	8.3	151	285	302
	reg forest	25	-	-	-	-	-	-	5.6	8.3	8.3	151	285	302
	reg forest	50	-	-	-	-	-	-	5.7	8.3	8.3	151	285	302
	gap	10	-	-	-	-	-	-	5.1	7.8	7.2	151	285	302
	gap	25	-	-	-	-	-	-	5.2	7.8	7.3	151	285	302
	gap	50	-	-	-	-	-	-	5.2	7.8	7.3	151	285	302
	leaf freq	10	-	-	-	-	-	-	5.1	7.8	7.2	151	285	302
	leaf freq	25	-	-	-	-	-	-	5.1	7.7	7.2	151	285	302
	leaf freq	50	-	-	-	-	-	-	5.2	7.8	7.2	151	285	302
	ssg	10	-	-	-	-	-	-	5.1	7.8	7.2	151	285	302
	ssg	25	-	-	-	-	-	-	5.1	7.8	7.3	151	285	302
	ssg	50	-	-	-	-	-	-	5.2	7.8	7.2	151	285	302
	tree weight	10	-	-	-	-	-	-	5.1	7.8	7.2	151	285	302
	tree weight	25	-	-	-	-	-	-	5.2	7.8	7.2	151	285	302
	tree weight	50	-	-	-	-	-	-	5.2	7.8	7.2	151	285	302
	no clairvoyant	-	-	-	-	-	-	-	5.5	7.9	7.3	151	285	302
	0-restart	-	-	-	-	-	-	-	5.1	7.8	7.2	151	285	302
graphdraw-domain	monotone	10	-	-	-	1	1	1	1229.3	1092.2	550.9	2102031	1827774	873534
	monotone	25	-	-	-	1	1	1	1226.8	1090.0	551.5	2102031	1827774	873534
	monotone	50	-	-	-	1	1	1	1528.3	1094.3	551.1	2769061	1827774	873534
	reg forest	10	-	-	-	1	1	1	1234.8	1108.7	503.9	2102031	1827774	872345
	reg forest	25	-	-	-	1	1	1	593.2	892.2	860.4	926303	1471964	1415795
	reg forest	50	-	-	-	-	-	-	1612.1	1581.7	1145.7	2769061	2573912	1964436
	gap	10	-	-	-	1	1	1	1175.5	886.0	696.0	2091516	1577020	1125620
	gap	25	-	-	-	1	1	1	1297.3	877.5	693.0	2088290	1625324	1125620
	gap	50	-	-	-	1	1	1	1353.0	962.2	692.7	2390835	1513548	1125620
	leaf freq	10	-	-	-	1	1	1	460.2	440.8	473.2	784144	719696	746408
	leaf freq	25	-	-	-	1	1	1	683.2	536.3	450.9	1059764	831793	717122
	leaf freq	50	-	-	-	1	1	1	679.1	618.8	644.3	1059764	1055329	1059793
	ssg	10	-	-	-	1	1	1	1488.6	929.5	553.4	2685161	1487875	884593
	ssg	25	-	-	-	1	1	1	967.6	535.6	553.1	1651330	862815	884593
	ssg	50	-	-	-	1	1	1	1494.7	485.3	514.1	2516238	894820	917842
	tree weight	10	-	-	-	1	1	1	1221.1	1099.4	550.6	2102031	1827774	873534
	tree weight	25	-	-	-	1	1	1	1225.4	1093.7	552.0	2102031	1827774	873534
	tree weight	50	-	-	-	1	1	1	1223.3	1095.7	1093.6	2102031	1827774	1811145
	no clairvoyant	-	-	-	-	-	-	-	1520.9	1489.5	1072.2	2769061	2573912	1964436
	0-restart	-	-	-	-	-	-	-	1516.7	1483.8	1070.2	2769061	2573912	1964436
h80x6320d	monotone	10	3	4	3	-	-	-	75.2	83.2	67.1	4	7	8
	monotone	25	3	4	3	-	-	-	75.3	83.5	67.3	4	7	8
	monotone	50	3	4	3	-	-	-	75.2	83.4	67.2	4	7	8
	reg forest	10	3	4	3	-	-	-	75.5	82.4	67.2	4	7	8
	reg forest	25	3	4	3	-	-	-	75.8	83.3	66.8	4	7	8
	reg forest	50	3	4	3	-	-	-	75.6	83.2	67.0	4	7	8
	gap	10	3	4	3	-	-	-	75.4	83.6	67.2	4	7	8
	gap	25	3	4	3	-	-	-	75.2	83.1	67.4	4	7	8
	gap	50	3	4	3	-	-	-	75.0	83.2	67.4	4	7	8
	leaf freq	10	3	4	3	-	-	-	75.2	83.3	67.3	4	7	8
	leaf freq	25	3	4	3	-	-	-	75.2	83.2	67.6	4	7	8
	leaf freq	50	3	4	3	-	-	-	75.5	83.0	67.1	4	7	8
	ssg	10	3	4	3	-	-	-	75.3	83.2	67.3	4	7	8
	ssg	25	3	4	3	-	-	-	75.1	83.3	67.1	4	7	8
	ssg	50	3	4	3	-	-	-	75.5	83.2	67.1	4	7	8
	tree weight	10	3	4	3	-	-	-	75.5	83.5	67.4	4	7	8

cont. on next page ...

E. Appendix E

Table E.1 cont.

Table E.1 cont.														
mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
highschool1.	tree weight	25	3	4	3	—	—	—	75.1	82.8	66.5	4	7	8
	tree weight	50	3	4	3	—	—	—	75.1	83.0	67.5	4	7	8
	no clairvoyant	—	3	4	3	—	—	—	75.9	82.8	67.2	4	7	8
	0-restart	—	—	—	—	—	—	—	72.2	57.7	72.7	3	5	7
	monotone	10	—	—	—	—	—	—	t	t	t	1	1	1
	monotone	25	—	—	—	—	—	—	t	t	t	1	1	1
	monotone	50	—	—	—	—	—	—	t	t	t	1	1	1
	reg forest	10	—	—	—	—	—	—	t	t	t	1	1	1
	reg forest	25	—	—	—	—	—	—	t	t	t	1	1	1
	reg forest	50	—	—	—	—	—	—	t	t	t	1	1	1
	gap	10	—	—	—	—	—	—	t	t	t	1	1	1
	gap	25	—	—	—	—	—	—	t	t	t	1	1	1
	gap	50	—	—	—	—	—	—	t	t	t	1	1	1
	leaf freq	10	—	—	—	—	—	—	t	t	t	1	1	1
	leaf freq	25	—	—	—	—	—	—	t	t	t	1	1	1
	leaf freq	50	—	—	—	—	—	—	t	t	t	1	1	1
	ssg	10	—	—	—	—	—	—	t	t	t	1	1	1
	ssg	25	—	—	—	—	—	—	t	t	t	1	1	1
	ssg	50	—	—	—	—	—	—	t	t	t	1	1	1
	hypothyroid-k1	tree weight	10	—	—	—	—	—	—	t	t	t	1	1
tree weight		25	—	—	—	—	—	—	t	t	t	1	1	1
tree weight		50	—	—	—	—	—	—	t	t	t	1	1	1
no clairvoyant		—	—	—	—	—	—	—	t	t	t	1	1	1
0-restart		—	—	—	—	—	—	—	t	t	t	1	1	1
monotone		10	—	—	—	—	—	—	20.7	20.7	20.7	1	1	1
monotone		25	—	—	—	—	—	—	20.8	20.9	20.9	1	1	1
monotone		50	—	—	—	—	—	—	20.6	20.6	20.8	1	1	1
reg forest		10	—	—	—	—	—	—	20.5	20.6	20.5	1	1	1
reg forest		25	—	—	—	—	—	—	20.8	20.8	20.5	1	1	1
reg forest		50	—	—	—	—	—	—	20.9	20.8	20.7	1	1	1
gap		10	—	—	—	—	—	—	20.8	20.9	20.8	1	1	1
gap		25	—	—	—	—	—	—	20.8	20.9	20.8	1	1	1
gap		50	—	—	—	—	—	—	20.9	20.7	20.8	1	1	1
leaf freq		10	—	—	—	—	—	—	20.9	20.5	20.7	1	1	1
leaf freq		25	—	—	—	—	—	—	20.7	20.6	20.8	1	1	1
leaf freq		50	—	—	—	—	—	—	20.7	20.7	20.8	1	1	1
ssg		10	—	—	—	—	—	—	20.6	20.8	20.8	1	1	1
ssg		25	—	—	—	—	—	—	20.9	20.5	20.6	1	1	1
ssg		50	—	—	—	—	—	—	20.8	20.7	20.7	1	1	1
ic97_potential	tree weight	10	—	—	—	—	—	—	20.8	21.0	20.7	1	1	1
	tree weight	25	—	—	—	—	—	—	20.7	20.7	20.7	1	1	1
	tree weight	50	—	—	—	—	—	—	20.6	20.8	20.6	1	1	1
	no clairvoyant	—	—	—	—	—	—	—	20.7	20.8	20.9	1	1	1
	0-restart	—	—	—	—	—	—	—	20.4	20.6	20.7	1	1	1
	monotone	10	—	—	—	1	1	1	t	t	t	3433303	3171182	3225410
	monotone	25	—	—	—	1	1	1	t	t	t	3422070	3165613	3221872
	monotone	50	—	—	—	—	1	1	t	t	t	3319278	3156003	3213291
	reg forest	10	—	—	—	1	1	1	t	t	t	3435162	3147873	3203878
	reg forest	25	—	—	—	1	1	—	t	t	t	3010700	3112526	3183985
	reg forest	50	—	—	—	—	—	—	t	t	t	3243809	3182798	3151051
	gap	10	—	—	—	1	1	1	t	t	t	2906905	3119903	3082395
	gap	25	—	—	—	1	1	—	t	t	t	3299785	3203349	3249793
	gap	50	—	—	—	—	—	—	t	t	t	3325908	3271050	3249743
	leaf freq	10	—	—	—	1	1	1	t	t	t	3157832	3466159	3076816
	leaf freq	25	—	—	—	1	1	1	t	t	t	3345313	3406697	3098132
	leaf freq	50	—	—	—	1	1	1	t	t	t	3190775	3341568	2851216
	ssg	10	—	—	—	1	1	1	t	t	t	3428555	3774863	3217438
	ssg	25	—	—	—	1	1	1	t	t	t	2882502	3450109	2738989
	ssg	50	—	—	—	1	1	1	t	t	t	3048618	3692789	3056003
icir97_tension	tree weight	10	—	—	—	1	1	1	t	t	t	2837304	3221074	3217733
	tree weight	25	—	—	—	1	1	1	t	t	t	2847808	3537254	2955166
	tree weight	50	—	—	—	1	1	1	t	t	t	2838411	3552497	2783410
	no clairvoyant	—	—	—	—	—	—	—	t	t	t	3313862	3270992	3231918
	0-restart	—	—	—	—	—	—	—	t	t	t	3309497	3282448	3234427
	monotone	10	—	—	—	—	—	—	t	t	3514.7	1730050	1852104	1294688
	monotone	25	—	—	—	—	—	—	t	t	3515.6	1734836	1848164	1294688
	monotone	50	—	—	—	—	—	—	t	t	3522.2	1733754	1860154	1294688
	reg forest	10	—	—	—	—	—	—	t	t	3530.5	1716081	1859688	1294688
	reg forest	25	—	—	—	—	—	—	t	t	3519.6	1718566	1850338	1294688
	reg forest	50	—	—	—	—	—	—	t	t	3524.9	1724272	1845381	1294688
	gap	10	—	—	—	—	—	—	t	t	3512.3	1730639	1841019	1294688
	gap	25	—	—	—	—	—	—	t	t	3520.6	1728609	1852978	1294688
	gap	50	—	—	—	—	—	—	t	t	3530.2	1724527	1852650	1294688
	leaf freq	10	—	—	—	—	—	—	t	t	3512.9	1741791	1856435	1294688
	leaf freq	25	—	—	—	—	—	—	t	t	3523.5	1734683	1857681	1294688
	leaf freq	50	—	—	—	—	—	—	t	t	3525.6	1733223	1856681	1294688

cont. on next page ...

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
irish-elec.	ssg	10	–	–	–	–	–	–	t	t	3525.4	1737921	1858978	1294688
	ssg	25	–	–	–	–	–	–	t	t	3519.6	1723635	1851830	1294688
	ssg	50	–	–	–	–	–	–	t	t	3531.5	1725015	1858622	1294688
	tree weight	10	–	–	–	–	–	–	t	t	3526.1	1732857	1849446	1294688
	tree weight	25	–	–	–	–	–	–	t	t	3531.2	1725711	1855054	1294688
	tree weight	50	–	–	–	–	–	–	t	t	3519.6	1734572	1850605	1294688
	no clairvoyant	–	–	–	–	–	–	–	t	t	3512.8	1730655	1854105	1294688
	0-restart	–	–	–	–	–	–	–	t	t	3523.9	1745951	1857676	1294688
	monotone	10	–	–	–	–	–	–	t	t	t	2185	1971	1801
	monotone	25	–	–	–	–	–	–	t	t	t	2185	1965	1801
	monotone	50	–	–	–	–	–	–	t	t	t	2185	1971	1797
	reg forest	10	–	–	–	–	–	–	t	t	t	2185	1969	1805
	reg forest	25	–	–	–	–	–	–	t	t	t	2181	1971	1788
	reg forest	50	–	–	–	–	–	–	t	t	t	2181	1964	1805
	gap	10	–	–	–	–	–	–	t	t	t	2185	1971	1813
	gap	25	–	–	–	–	–	–	t	t	t	2185	1965	1801
	gap	50	–	–	–	–	–	–	t	t	t	2182	1958	1814
	leaf freq	10	–	–	–	–	–	–	t	t	t	2185	1971	1801
	leaf freq	25	–	–	–	–	–	–	t	t	t	2197	1971	1792
	leaf freq	50	–	–	–	–	–	–	t	t	t	2175	1964	1797
	ssg	10	–	–	–	–	–	–	t	t	t	2181	1976	1801
	ssg	25	–	–	–	–	–	–	t	t	t	2185	1965	1809
	ssg	50	–	–	–	–	–	–	t	t	t	2185	1964	1814
	tree weight	10	–	–	–	–	–	–	t	t	t	2182	1971	1797
	tree weight	25	–	–	–	–	–	–	t	t	t	2181	1971	1801
	tree weight	50	–	–	–	–	–	–	t	t	t	2188	1971	1801
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	2181	1969	1801
	0-restart	–	–	–	–	–	–	–	t	t	t	2191	1971	1787
irp	monotone	10	6	5	7	–	–	–	16.4	14.7	15.5	7	6	8
	monotone	25	6	5	7	–	–	–	16.5	14.6	15.4	7	6	8
	monotone	50	6	5	7	–	–	–	16.3	14.6	15.4	7	6	8
	reg forest	10	6	5	7	–	–	–	16.5	14.9	15.7	7	6	8
	reg forest	25	6	5	7	–	–	–	16.6	15.0	15.7	7	6	8
	reg forest	50	6	5	7	–	–	–	16.7	14.9	15.7	7	6	8
	gap	10	6	5	7	–	–	–	16.3	14.7	15.5	7	6	8
	gap	25	6	5	7	–	–	–	16.4	14.7	15.3	7	6	8
	gap	50	6	5	7	–	–	–	16.4	14.6	15.5	7	6	8
	leaf freq	10	6	5	7	–	–	–	16.4	14.7	15.5	7	6	8
	leaf freq	25	6	5	7	–	–	–	16.5	14.6	15.3	7	6	8
	leaf freq	50	6	5	7	–	–	–	16.3	14.7	15.4	7	6	8
	ssg	10	6	5	7	–	–	–	16.3	14.7	15.4	7	6	8
	ssg	25	6	5	7	–	–	–	16.3	14.6	15.5	7	6	8
	ssg	50	6	5	7	–	–	–	16.4	14.6	15.4	7	6	8
	tree weight	10	6	5	7	–	–	–	16.3	14.7	15.4	7	6	8
	tree weight	25	6	5	7	–	–	–	16.4	14.7	15.4	7	6	8
	tree weight	50	6	5	7	–	–	–	16.3	14.7	15.4	7	6	8
	no clairvoyant	–	6	5	7	–	–	–	16.6	14.7	15.4	7	6	8
	0-restart	–	–	–	–	–	–	–	10.7	13.0	11.8	3	5	3
istanbul.	monotone	10	–	–	–	–	–	–	100.2	95.3	103.2	271	271	321
	monotone	25	–	–	–	–	–	–	100.5	95.7	103.6	271	271	321
	monotone	50	–	–	–	–	–	–	100.0	95.3	103.3	271	271	321
	reg forest	10	–	–	–	–	–	–	99.9	95.6	103.9	271	271	321
	reg forest	25	–	–	–	–	–	–	100.2	95.6	103.4	271	271	321
	reg forest	50	–	–	–	–	–	–	100.4	95.2	103.5	271	271	321
	gap	10	–	–	–	–	–	–	100.3	95.6	103.2	271	271	321
	gap	25	–	–	–	–	–	–	100.2	95.2	103.3	271	271	321
	gap	50	–	–	–	–	–	–	100.2	95.2	103.4	271	271	321
	leaf freq	10	–	–	–	–	–	–	100.1	95.3	104.1	271	271	321
	leaf freq	25	–	–	–	–	–	–	100.4	95.5	103.2	271	271	321
	leaf freq	50	–	–	–	–	–	–	100.0	95.3	103.6	271	271	321
	ssg	10	–	–	–	–	–	–	99.4	95.0	103.6	271	271	321
	ssg	25	–	–	–	–	–	–	100.4	94.8	103.3	271	271	321
	ssg	50	–	–	–	–	–	–	100.2	95.3	103.2	271	271	321
	tree weight	10	–	–	–	–	–	–	100.5	95.0	103.3	271	271	321
	tree weight	25	–	–	–	–	–	–	100.5	95.2	103.4	271	271	321
	tree weight	50	–	–	–	–	–	–	100.1	95.0	103.3	271	271	321
	no clairvoyant	–	–	–	–	–	–	–	100.4	95.6	103.4	271	271	321
	0-restart	–	–	–	–	–	–	–	99.8	94.8	103.6	271	271	321
kimushroom	monotone	10	–	–	–	–	–	–	2505.9	1426.8	1907.6	10	1	21
	monotone	25	–	–	–	–	–	–	2548.2	1439.7	1980.1	10	1	21
	monotone	50	–	–	–	–	–	–	2492.4	1431.7	1962.5	10	1	21
	reg forest	10	–	–	–	–	–	–	2489.1	1400.8	1955.0	10	1	21
	reg forest	25	–	–	–	–	–	–	2496.8	1424.7	1977.2	10	1	21
	reg forest	50	–	–	–	–	–	–	2525.4	1436.7	1968.7	10	1	21
	gap	10	–	–	–	–	–	–	2500.1	1418.8	1968.6	10	1	21
	gap	25	–	–	–	–	–	–	2497.8	1430.8	1973.5	10	1	21

cont. on next page ...

E. Appendix E

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
lectsched-5-obj	gap	50	–	–	–	–	–	–	2520.3	1428.5	1965.3	10	1	21
	leaf freq	10	–	–	–	–	–	–	2536.9	1425.8	1965.6	10	1	21
	leaf freq	25	–	–	–	–	–	–	2526.0	1426.6	1947.1	10	1	21
	leaf freq	50	–	–	–	–	–	–	2503.9	1434.8	1942.1	10	1	21
	ssg	10	–	–	–	–	–	–	2497.1	1432.1	1958.9	10	1	21
	ssg	25	–	–	–	–	–	–	2491.6	1419.0	1986.5	10	1	21
	ssg	50	–	–	–	–	–	–	2488.5	1421.8	1964.5	10	1	21
	tree weight	10	–	–	–	–	–	–	2492.9	1418.2	1969.6	10	1	21
	tree weight	25	–	–	–	–	–	–	2456.7	1410.6	1961.8	10	1	21
	tree weight	50	–	–	–	–	–	–	2490.9	1410.3	1954.6	10	1	21
	no clairvoyant	–	–	–	–	–	–	–	2480.2	1405.2	1945.6	10	1	21
	0-restart	–	–	–	–	–	–	–	2483.8	1416.6	1954.7	10	1	21
	monotone	10	–	–	–	1	1	1	t	t	t	39771	76746	300609
	monotone	25	–	–	–	–	1	1	t	t	t	285807	76754	300490
	monotone	50	–	–	–	–	1	1	t	t	t	285712	76765	300048
	reg forest	10	–	–	–	1	1	1	t	t	t	86905	76696	298541
	reg forest	25	–	–	–	–	–	1	t	t	t	280972	149247	120627
	reg forest	50	–	–	–	–	–	–	t	t	t	282406	149240	274749
	gap	10	–	–	–	1	1	1	t	t	t	139303	73622	172579
	gap	25	–	–	–	1	1	1	t	t	t	333098	208621	206955
	gap	50	–	–	–	1	1	1	t	t	t	85575	134127	68390
	leaf freq	10	–	–	–	1	1	1	t	t	t	60297	198676	133806
	leaf freq	25	–	–	–	1	1	1	t	t	t	282934	196792	92449
	leaf freq	50	–	–	–	1	–	1	t	t	t	284882	150001	92523
	ssg	10	–	–	–	1	1	1	t	t	t	312853	268068	73524
	ssg	25	–	–	–	1	1	1	t	t	t	271000	48481	313992
	ssg	50	–	–	–	1	1	1	t	t	t	272106	147351	143869
	tree weight	10	–	–	–	1	1	1	t	t	t	255024	76916	300072
	tree weight	25	–	–	–	1	1	1	t	t	t	310370	76718	124486
	tree weight	50	–	–	–	1	1	1	t	t	t	175005	76744	73639
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	285853	149620	275766
	0-restart	–	–	–	–	–	–	–	t	t	t	285050	149866	277078
leo1	monotone	10	–	–	–	1	1	1	t	t	t	99721	92105	117094
	monotone	25	–	–	–	1	1	1	t	t	t	102001	91633	116014
	monotone	50	–	–	–	–	1	1	t	t	t	110160	92108	116762
	reg forest	10	–	–	–	1	1	1	t	t	t	102044	91992	116341
	reg forest	25	–	–	–	–	1	–	t	t	t	108586	92546	93432
	reg forest	50	–	–	–	–	–	–	t	t	t	109488	85821	94424
	gap	10	–	–	–	1	1	1	t	t	t	103333	90368	103731
	gap	25	–	–	–	–	–	–	t	t	t	109708	86163	91037
	gap	50	–	–	–	–	–	–	t	t	t	108983	86226	95052
	leaf freq	10	–	–	–	1	1	1	t	t	t	105037	92047	91141
	leaf freq	25	–	–	–	1	1	1	t	t	t	104058	92289	90023
	leaf freq	50	–	–	–	1	1	1	t	t	t	104678	92616	81915
	ssg	10	–	–	–	1	1	1	t	t	t	90587	94192	92771
	ssg	25	–	–	–	1	1	1	t	t	t	96936	98999	92650
	ssg	50	–	–	–	1	1	1	t	t	t	109322	85983	91840
	tree weight	10	–	–	–	1	1	1	t	t	t	100713	103478	116828
	tree weight	25	–	–	–	1	1	1	t	t	t	101094	94766	116981
	tree weight	50	–	–	–	1	1	1	t	t	t	101278	95593	116927
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	108205	86685	95058
	0-restart	–	–	–	–	–	–	–	t	t	t	109478	85935	95145
leo2	monotone	10	–	–	–	–	–	–	t	t	t	47596	40599	50827
	monotone	25	–	–	–	–	–	–	t	t	t	47582	40678	50967
	monotone	50	–	–	–	–	–	–	t	t	t	47882	40627	50718
	reg forest	10	–	–	–	1	–	1	t	t	t	41794	40616	52440
	reg forest	25	–	–	–	–	–	–	t	t	t	47440	40634	50618
	reg forest	50	–	–	–	–	–	–	t	t	t	47238	40182	50957
	gap	10	–	–	–	1	1	1	t	t	t	41884	41737	56545
	gap	25	–	–	–	1	1	1	t	t	t	48736	47009	41813
	gap	50	–	–	–	1	1	–	t	t	t	41567	33907	50908
	leaf freq	10	–	–	–	1	1	1	t	t	t	44182	42744	39916
	leaf freq	25	–	–	–	1	–	1	t	t	t	36719	40626	36191
	leaf freq	50	–	–	–	1	–	1	t	t	t	45109	40622	42137
	ssg	10	–	–	–	1	1	1	t	t	t	42331	32922	44435
	ssg	25	–	–	–	1	1	1	t	t	t	45791	37997	44450
	ssg	50	–	–	–	1	1	1	t	t	t	43098	33333	46399
	tree weight	10	–	–	–	1	1	1	t	t	t	42860	56943	54843
	tree weight	25	–	–	–	1	1	1	t	t	t	27350	43983	65928
	tree weight	50	–	–	–	1	1	1	t	t	t	44231	44071	37812
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	47959	40662	51176
	0-restart	–	–	–	–	–	–	–	t	t	t	47764	40371	50908
lotsize	monotone	10	–	–	–	1	1	1	t	t	t	13241	16364	13619
	monotone	25	–	–	–	1	1	1	t	t	t	13132	16563	13616
	monotone	50	–	–	–	1	–	1	t	t	t	13133	12241	13604
	reg forest	10	–	–	–	1	1	1	t	t	t	12694	20264	13604

cont. on next page ...

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
mad	reg forest	25	–	–	–	–	–	–	t	t	t	7341	12235	4533
	reg forest	50	–	–	–	–	–	–	t	t	t	7330	12358	4533
	gap	10	–	–	–	–	–	–	t	t	t	7270	12203	4533
	gap	25	–	–	–	–	–	–	t	t	t	7333	12204	4533
	gap	50	–	–	–	–	–	–	t	t	t	7331	12208	4577
	leaf freq	10	–	–	–	–	–	–	t	t	t	7331	12204	4533
	leaf freq	25	–	–	–	–	–	–	t	t	t	7349	12204	4533
	leaf freq	50	–	–	–	–	–	–	t	t	t	7331	12241	4533
	ssg	10	–	–	–	1	1	1	t	t	t	13133	16852	15091
	ssg	25	–	–	–	1	1	1	t	t	t	13133	17383	11870
	ssg	50	–	–	–	1	1	1	t	t	t	12534	15199	10907
	tree weight	10	–	–	–	1	1	1	t	t	t	17244	15439	13389
	tree weight	25	–	–	–	1	1	1	t	t	t	13851	14088	8580
	tree weight	50	–	–	–	1	1	1	t	t	t	15440	11930	9787
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	7331	12241	4533
	0-restart	–	–	–	–	–	–	–	t	t	t	7318	12241	4533
	monotone	10	–	–	–	–	–	1	t	t	t	5855351	5372200	6354635
	monotone	25	–	–	–	–	–	–	t	t	t	5850435	5389974	6890397
	monotone	50	–	–	–	–	–	–	t	t	t	5876364	5413243	6856670
	reg forest	10	–	–	–	–	–	1	t	t	t	5652013	5261473	6319175
	reg forest	25	–	–	–	–	–	–	t	t	t	5631086	5253804	6661820
	reg forest	50	–	–	–	–	–	–	t	t	t	5601545	5255630	6693340
	gap	10	–	–	–	–	–	–	t	t	t	5869589	5389029	6885701
	gap	25	–	–	–	–	–	–	t	t	t	5895738	5389850	6871377
	gap	50	–	–	–	–	–	–	t	t	t	5836420	5391137	6875583
	leaf freq	10	–	–	–	1	1	1	t	t	t	5794830	6309336	5317825
	leaf freq	25	–	–	–	1	1	1	t	t	t	5308072	5721650	4862852
	leaf freq	50	–	–	–	1	1	1	t	t	t	5634478	5383560	6867526
	ssg	10	–	–	–	1	1	1	t	t	t	5265061	6862870	5796151
	ssg	25	–	–	–	1	1	1	t	t	t	5241567	6186980	5343308
	ssg	50	–	–	–	1	1	1	t	t	t	5253554	5746715	6272448
	tree weight	10	–	–	–	1	1	1	t	t	t	6638454	6123531	5690773
	tree weight	25	–	–	–	1	1	1	t	t	t	6430624	5768319	6291416
	tree weight	50	–	–	–	1	1	1	t	t	t	5655595	5583973	5646004
map10	no clairvoyant	–	–	–	–	–	–	–	t	t	t	5868011	5390909	6869889
	0-restart	–	–	–	–	–	–	–	t	t	t	5865739	5353510	6876047
	monotone	10	–	–	–	–	–	–	1025.0	979.8	860.5	1982	1395	1252
	monotone	25	–	–	–	–	–	–	1034.5	980.4	856.8	1982	1395	1252
	monotone	50	–	–	–	–	–	–	1025.5	983.0	866.4	1982	1395	1252
	reg forest	10	–	–	–	–	–	–	1035.5	985.7	861.8	1982	1395	1252
	reg forest	25	–	–	–	–	–	–	1033.4	982.8	862.8	1982	1395	1252
	reg forest	50	–	–	–	–	–	–	1031.3	982.3	858.7	1982	1395	1252
	gap	10	–	–	–	–	–	–	1035.4	980.3	869.5	1982	1395	1252
	gap	25	–	–	–	–	–	–	1028.3	981.1	865.2	1982	1395	1252
	gap	50	–	–	–	–	–	–	1036.7	980.8	863.3	1982	1395	1252
	leaf freq	10	–	–	–	–	–	–	1034.2	980.1	864.1	1982	1395	1252
	leaf freq	25	–	–	–	–	–	–	1035.5	987.4	866.5	1982	1395	1252
	leaf freq	50	–	–	–	–	–	–	1027.1	983.3	861.9	1982	1395	1252
	ssg	10	–	–	–	–	1	–	1035.5	1147.8	861.7	1982	1612	1252
	ssg	25	–	–	–	–	–	–	1029.6	977.7	864.2	1982	1395	1252
	ssg	50	–	–	–	–	–	–	1034.3	984.7	865.7	1982	1395	1252
	tree weight	10	–	–	–	1	1	–	1228.3	1223.0	864.3	2036	1998	1252
	tree weight	25	–	–	–	–	1	–	1035.6	1252.1	866.5	1982	2037	1252
	tree weight	50	–	–	–	–	1	–	1031.1	1234.1	864.5	1982	1891	1252
	no clairvoyant	–	–	–	–	–	–	–	1036.7	977.7	867.5	1982	1395	1252
	0-restart	–	–	–	–	–	–	–	1026.6	984.3	865.0	1982	1395	1252
	monotone	10	–	–	–	–	–	–	2002.9	1964.2	1758.0	2115	2089	2141
	monotone	25	–	–	–	–	–	–	2008.8	1982.4	1761.1	2115	2089	2141
	monotone	50	–	–	–	–	–	–	1998.0	1983.6	1756.9	2115	2089	2141
	reg forest	10	–	–	–	–	–	–	2009.9	1979.7	1764.3	2115	2089	2141
	reg forest	25	–	–	–	–	–	–	2005.6	1992.5	1759.4	2115	2089	2141
	reg forest	50	–	–	–	–	–	–	1981.5	1983.8	1757.9	2115	2089	2141
	gap	10	–	–	–	–	1	1	2008.4	2745.3	2971.1	2115	2801	3485
	gap	25	–	–	–	–	–	–	1995.9	1981.4	1756.6	2115	2089	2141
	gap	50	–	–	–	–	–	–	2002.0	1971.3	1756.4	2115	2089	2141
	leaf freq	10	–	–	–	–	–	–	2003.9	1993.6	1757.4	2115	2089	2141
	leaf freq	25	–	–	–	–	–	–	1998.3	1979.8	1762.5	2115	2089	2141
	leaf freq	50	–	–	–	–	–	–	2002.3	1988.8	1755.2	2115	2089	2141
	ssg	10	–	–	–	1	–	–	2777.3	1967.4	1763.0	3042	2089	2141
	ssg	25	–	–	–	–	–	–	2008.0	1979.9	1757.8	2115	2089	2141
	ssg	50	–	–	–	–	–	–	2003.0	1967.9	1758.7	2115	2089	2141
	tree weight	10	–	–	–	1	–	–	2786.8	1985.6	1757.9	3042	2089	2141
	tree weight	25	–	–	–	–	–	–	2010.4	1978.4	1755.7	2115	2089	2141
	tree weight	50	–	–	–	–	–	–	2009.6	1991.3	1756.2	2115	2089	2141
	no clairvoyant	–	–	–	–	–	–	–	2006.9	1976.4	1763.5	2115	2089	2141
	0-restart	–	–	–	–	–	–	–	1997.1	1983.4	1742.0	2115	2089	2141

cont. on next page ...

E. Appendix E

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
markshare2	monotone	10	–	–	–	1	1	1	t	t	t	4812871	3183250	3030315
	monotone	25	–	–	–	1	1	1	t	t	t	4841606	3169273	3019755
	monotone	50	–	–	–	1	1	1	t	t	t	4829938	3176029	3035726
	reg forest	10	–	–	–	1	1	1	t	t	t	3147569	3160234	3105893
	reg forest	25	–	–	–	–	–	–	t	t	t	3071355	3594063	4018434
	reg forest	50	–	–	–	–	–	–	t	t	t	3051796	3600138	4028837
	gap	10	–	–	–	–	–	–	t	t	t	3092755	3649264	4101685
	gap	25	–	–	–	–	–	–	t	t	t	3080841	3656012	4103957
	gap	50	–	–	–	–	–	–	t	t	t	3099210	3665073	4110502
	leaf freq	10	–	–	–	1	1	1	t	t	t	3070811	2892404	3880313
	leaf freq	25	–	–	–	1	1	1	t	t	t	3940351	2892811	3129753
	leaf freq	50	–	–	–	1	1	1	t	t	t	4693981	2888700	4115417
	ssg	10	–	–	–	1	1	1	t	t	t	3564154	3968314	4854122
	ssg	25	–	–	–	1	1	1	t	t	t	5117359	3328678	3660932
	ssg	50	–	–	–	1	1	1	t	t	t	6139660	5382820	3028015
	tree weight	10	–	–	–	–	–	–	t	t	t	3093222	3666607	4103957
	tree weight	25	–	–	–	–	–	–	t	t	t	3093222	3666607	4099346
	tree weight	50	–	–	–	–	–	–	t	t	t	3075000	3640981	4099346
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	3074199	3662683	4104598
	0-restart	–	–	–	–	–	–	–	t	t	t	3096563	3652423	4069088
markshare_4_0	monotone	10	–	–	–	1	1	1	281.3	99.5	232.9	2082015	764166	1777139
	monotone	25	–	–	–	1	1	1	281.4	100.6	234.4	2082015	764166	1777139
	monotone	50	–	–	–	1	1	1	282.4	100.3	234.6	2082015	764166	1777139
	reg forest	10	–	–	–	1	1	1	286.2	104.3	239.9	2082015	764166	1777139
	reg forest	25	–	–	–	–	–	–	191.5	267.6	297.5	1391562	1914160	2051707
	reg forest	50	–	–	–	–	–	–	192.6	267.4	298.6	1391562	1914160	2051707
	gap	10	–	–	–	–	–	–	179.4	248.2	276.8	1391562	1914160	2051707
	gap	25	–	–	–	–	–	–	179.3	247.7	277.2	1391562	1914160	2051707
	gap	50	–	–	–	–	–	–	177.9	248.3	277.4	1391562	1914160	2051707
	leaf freq	10	–	–	–	1	1	1	210.6	266.6	287.6	1563826	2023996	2155323
	leaf freq	25	–	–	–	1	1	1	158.2	199.8	271.1	1231148	1561801	1992936
	leaf freq	50	–	–	–	–	–	–	179.1	249.2	277.4	1391562	1914160	2051707
	ssg	10	–	–	–	1	1	1	192.2	260.2	234.5	1451327	1936031	1777139
	ssg	25	–	–	–	1	1	1	231.4	276.3	233.4	1750594	2116381	1777139
	ssg	50	–	–	–	1	1	1	204.2	276.4	235.8	1551382	2116381	1777139
	tree weight	10	–	–	–	1	1	1	280.7	183.9	290.6	2082015	1371575	2155323
	tree weight	25	–	–	–	1	1	1	279.5	181.7	289.8	2082015	1371575	2155323
	tree weight	50	–	–	–	1	1	1	281.1	184.7	289.8	2082015	1371575	2155323
	no clairvoyant	–	–	–	–	–	–	–	184.4	247.3	276.1	1391562	1914160	2051707
	0-restart	–	–	–	–	–	–	–	178.7	247.3	275.2	1391562	1914160	2051707
mas74	monotone	10	2	2	1	1	1	1	2333.8	2139.5	2396.3	6291932	5578706	6415730
	monotone	25	2	2	1	1	1	1	2320.5	2125.2	2392.1	6291932	5578706	6415730
	monotone	50	2	2	1	–	–	–	1643.9	2007.9	2398.6	3587962	4508789	6415730
	reg forest	10	2	2	1	1	1	1	2352.6	2145.8	2395.0	6291932	5578706	6415730
	reg forest	25	2	2	1	–	1	1	1791.9	1586.0	1487.1	3587962	4526646	4260336
	reg forest	50	2	2	1	–	–	–	1801.5	2185.2	1848.9	3587962	4508789	3915677
	gap	10	2	2	1	1	1	1	1642.9	1399.6	2395.3	4456148	4373226	6415730
	gap	25	2	2	1	1	–	1	1631.5	1996.8	2401.9	4288968	4508789	6415730
	gap	50	2	2	1	–	–	1	1637.3	1996.3	2388.7	3587962	4508789	6415730
	leaf freq	10	2	2	1	1	1	1	1574.0	2203.6	2390.7	4527560	5991172	6415730
	leaf freq	25	2	2	1	1	1	1	1518.2	2225.8	2388.4	4294913	6015777	6415730
	leaf freq	50	2	2	1	1	1	1	1611.5	1537.8	2392.3	4281561	4576852	6415730
	ssg	10	2	2	1	1	1	1	1677.0	2114.2	2390.6	4502548	5726547	6415730
	ssg	25	2	2	1	1	1	1	1571.5	1675.7	2387.5	4508839	4608643	6415730
	ssg	50	2	2	1	1	1	1	1930.5	1659.1	2392.8	5328134	4810878	6415730
	tree weight	10	2	2	1	1	1	1	2319.0	2131.5	2407.1	6291932	5578706	6415730
	tree weight	25	2	2	1	1	1	1	1734.4	2183.3	2386.6	4855547	5896010	6415730
	tree weight	50	2	2	1	1	1	1	1922.7	2078.1	2394.1	5021762	5633840	6415730
	no clairvoyant	–	2	2	1	–	–	–	1648.7	1996.7	1687.6	3587962	4508789	3915677
	0-restart	–	–	–	–	–	–	–	1380.9	1356.4	1335.8	3571847	3488848	3467284
mas76	monotone	10	2	1	2	–	1	1	108.0	136.2	117.3	246169	336245	251311
	monotone	25	2	1	2	–	–	1	108.6	129.1	116.4	246169	276468	251311
	monotone	50	2	1	2	–	–	–	108.6	129.2	116.3	246169	276468	185206
	reg forest	10	2	1	2	1	1	1	116.3	137.5	119.2	267099	336245	251311
	reg forest	25	2	1	2	–	–	–	118.3	139.8	123.8	246169	276468	185206
	reg forest	50	2	1	2	–	–	–	118.4	139.3	124.2	246169	276468	185206
	gap	10	2	1	2	1	1	1	133.4	143.3	174.3	313950	370229	438236
	gap	25	2	1	2	–	–	–	108.6	128.7	116.0	246169	276468	185206
	gap	50	2	1	2	–	–	–	108.3	127.7	116.0	246169	276468	185206
	leaf freq	10	2	1	2	1	1	1	124.6	140.6	140.9	269438	344289	330907
	leaf freq	25	2	1	2	–	–	1	108.6	129.4	174.9	246169	276468	436908
	leaf freq	50	2	1	2	–	–	1	107.5	128.7	175.1	246169	276468	436908
	ssg	10	2	1	2	1	1	1	125.3	122.6	125.3	299339	291965	249513
	ssg	25	2	1	2	1	1	1	128.5	116.4	131.6	327843	239234	256384
	ssg	50	2	1	2	1	1	1	121.3	136.6	145.2	278898	295220	311787
	tree weight	10	2	1	2	1	1	1	111.0	133.9	126.2	253247	302620	245653

cont. on next page ...

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
mc11	tree weight	25	2	1	2	1	1	1	132.7	157.6	106.7	307170	398181	200995
	tree weight	50	2	1	2	1	1	–	115.8	119.4	115.5	243128	250619	185206
	no clairvoyant	–	2	1	2	–	–	–	110.0	129.0	115.2	246169	276468	185206
	0-restart	–	–	–	–	–	–	–	174.7	205.5	157.3	379920	463778	322187
	monotone	10	–	1	–	–	–	1	278.5	101.1	140.6	6768	1332	2956
	monotone	25	–	1	–	–	–	1	278.3	101.4	140.8	6768	1332	2956
	monotone	50	–	1	–	–	–	–	278.4	101.3	317.1	6768	1332	10120
	reg forest	10	–	1	–	–	–	1	280.3	102.0	110.9	6768	1332	1792
	reg forest	25	–	1	–	–	–	–	279.3	101.8	318.2	6768	1332	10120
	reg forest	50	–	1	–	–	–	–	280.1	101.9	318.2	6768	1332	10120
	gap	10	–	1	–	–	–	–	279.5	100.8	316.5	6768	1332	10120
	gap	25	–	1	–	–	–	–	277.8	101.1	316.2	6768	1332	10120
	gap	50	–	1	–	–	–	–	279.1	101.3	318.4	6768	1332	10120
	leaf freq	10	–	1	–	–	–	–	279.3	101.2	316.4	6768	1332	10120
	leaf freq	25	–	1	–	–	–	–	278.9	101.1	316.2	6768	1332	10120
	leaf freq	50	–	1	–	–	–	–	278.5	101.2	315.5	6768	1332	10120
	ssg	10	–	1	–	1	–	1	122.8	101.2	110.3	1508	1332	1743
	ssg	25	–	1	–	1	–	1	127.2	101.4	178.5	1668	1332	3389
	ssg	50	–	1	–	1	–	1	152.9	101.3	182.1	2312	1332	3648
	tree weight	10	–	1	–	1	–	1	141.7	101.4	140.6	1964	1332	2956
	tree weight	25	–	1	–	1	–	1	155.6	101.1	141.2	2305	1332	2956
	tree weight	50	–	1	–	–	–	1	278.4	101.5	137.6	6768	1332	1869
	no clairvoyant	–	–	1	–	–	–	–	279.6	101.1	316.4	6768	1332	10120
	0-restart	–	–	–	–	–	–	–	278.5	482.0	318.2	6768	20145	10120
mcsched	monotone	10	–	–	–	–	–	–	264.0	286.0	230.6	14546	13237	9529
	monotone	25	–	–	–	–	–	–	263.7	285.0	230.7	14546	13237	9529
	monotone	50	–	–	–	–	–	–	263.2	286.1	231.6	14546	13237	9529
	reg forest	10	–	–	–	–	–	–	265.0	286.9	231.6	14546	13237	9529
	reg forest	25	–	–	–	–	–	–	265.4	287.3	232.2	14546	13237	9529
	reg forest	50	–	–	–	–	–	–	266.3	287.1	231.9	14546	13237	9529
	gap	10	–	–	–	1	1	1	220.5	250.8	253.0	10013	9926	9659
	gap	25	–	–	–	1	1	1	224.2	258.2	256.6	10352	10217	8696
	gap	50	–	–	–	–	–	–	264.4	286.0	231.0	14546	13237	9529
	leaf freq	10	–	–	–	–	–	–	263.9	285.4	230.8	14546	13237	9529
	leaf freq	25	–	–	–	–	–	–	264.0	286.2	231.9	14546	13237	9529
	leaf freq	50	–	–	–	–	–	–	264.1	285.7	230.7	14546	13237	9529
	ssg	10	–	–	–	1	1	1	286.3	324.3	210.1	11459	14856	7056
	ssg	25	–	–	–	1	1	1	243.2	249.4	212.0	11270	10012	7824
	ssg	50	–	–	–	1	1	1	239.6	258.4	244.3	11325	11376	9452
	tree weight	10	–	–	–	1	1	1	289.0	323.6	229.3	15984	14856	8410
	tree weight	25	–	–	–	1	1	1	288.5	324.3	233.7	15984	14856	8808
	tree weight	50	–	–	–	1	1	1	287.9	283.6	233.2	15984	10840	8808
	no clairvoyant	–	–	–	–	–	–	–	264.5	284.4	230.7	14546	13237	9529
	0-restart	–	–	–	–	–	–	–	263.4	285.3	230.7	14546	13237	9529
mik-250-20-75-4	monotone	10	1	1	1	–	–	–	33.0	31.5	54.2	16251	14375	23230
	monotone	25	1	1	1	–	–	–	33.0	31.4	54.3	16251	14375	23230
	monotone	50	1	1	1	–	–	–	33.1	31.4	54.2	16251	14375	23230
	reg forest	10	1	1	1	–	–	1	34.4	32.7	56.9	16251	14375	19018
	reg forest	25	1	1	1	–	–	–	34.7	32.5	55.7	16251	14375	23230
	reg forest	50	1	1	1	–	–	–	34.7	32.6	55.8	16251	14375	23230
	gap	10	1	1	1	–	1	–	33.0	43.4	54.2	16251	19099	23230
	gap	25	1	1	1	–	–	–	33.1	31.4	54.2	16251	14375	23230
	gap	50	1	1	1	–	–	–	33.0	31.4	54.2	16251	14375	23230
	leaf freq	10	1	1	1	–	–	–	32.9	31.5	54.2	16251	14375	23230
	leaf freq	25	1	1	1	–	–	–	32.9	31.2	54.1	16251	14375	23230
	leaf freq	50	1	1	1	–	–	–	33.0	31.2	54.4	16251	14375	23230
	ssg	10	1	1	1	–	1	1	33.1	41.1	52.2	16251	15736	19193
	ssg	25	1	1	1	–	1	1	33.0	47.2	39.5	16251	16157	12562
	ssg	50	1	1	1	–	1	1	33.1	40.3	56.9	16251	12832	18715
	tree weight	10	1	1	1	1	1	1	33.6	53.6	41.5	12227	18942	12877
	tree weight	25	1	1	1	1	1	1	40.5	31.9	51.9	15993	11791	16767
	tree weight	50	1	1	1	1	1	1	37.9	37.3	50.2	15465	14597	14343
	no clairvoyant	–	1	1	1	–	–	–	34.0	31.4	54.0	16251	14375	23230
	0-restart	–	–	–	–	–	–	–	49.8	31.0	30.2	24894	19780	16100
milo-v12.	monotone	10	–	–	–	–	–	–	t	t	t	114845	289080	220466
	monotone	25	–	–	–	–	–	–	t	t	t	114689	288408	220273
	monotone	50	–	–	–	–	–	–	t	t	t	115169	288136	220533
	reg forest	10	–	–	–	–	–	–	t	t	t	115349	287442	221118
	reg forest	25	–	–	–	–	–	–	t	t	t	114640	288388	220792
	reg forest	50	–	–	–	–	–	–	t	t	t	115359	288928	219401
	gap	10	–	–	–	–	–	–	t	t	t	115085	288840	219666
	gap	25	–	–	–	–	–	–	t	t	t	115645	288699	221659
	gap	50	–	–	–	–	–	–	t	t	t	115496	288303	221080
	leaf freq	10	–	–	–	–	–	–	t	t	t	115045	290093	221023
	leaf freq	25	–	–	–	–	–	–	t	t	t	115251	289080	220774
	leaf freq	50	–	–	–	–	–	–	t	t	t	114758	289551	220618

cont. on next page ...

E. Appendix E

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
momentum1	ssg	10	–	–	–	–	–	–	t	t	t	114495	288894	220573
	ssg	25	–	–	–	–	–	–	t	t	t	114898	289022	220948
	ssg	50	–	–	–	–	–	–	t	t	t	115078	289243	220452
	tree weight	10	–	–	–	–	–	–	t	t	t	115112	288107	220771
	tree weight	25	–	–	–	–	–	–	t	t	t	115474	289243	221255
	tree weight	50	–	–	–	–	–	–	t	t	t	114621	288529	220808
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	114410	288501	220452
	0-restart	–	–	–	–	–	–	–	t	t	t	115093	288378	220664
	monotone	10	–	–	–	1	1	1	t	t	t	11635	4340	18272
	monotone	25	–	–	–	1	–	1	t	t	t	11638	2143	18292
	monotone	50	–	–	–	1	–	–	t	t	t	11640	2130	4770
	reg forest	10	–	–	–	1	1	1	t	t	t	11660	4345	18266
	reg forest	25	–	–	–	1	1	1	t	t	t	21886	8149	18260
	reg forest	50	–	–	–	–	–	–	t	t	t	32768	2143	4770
	gap	10	–	–	–	1	1	1	t	t	t	17074	3118	7156
	gap	25	–	–	–	1	1	1	t	t	t	11378	2319	8872
	gap	50	–	–	–	1	1	–	t	t	t	21533	1882	4774
	leaf freq	10	–	–	–	1	1	1	t	t	t	24929	6205	7426
	leaf freq	25	–	–	–	1	–	1	t	t	t	22201	2127	7540
	leaf freq	50	–	–	–	–	–	–	t	t	t	32566	2143	4779
mushroom-best	ssg	10	–	–	–	1	1	1	t	t	t	4551	4347	10012
	ssg	25	–	–	–	1	1	1	t	t	t	30627	3846	9418
	ssg	50	–	–	–	1	1	1	t	t	t	24579	3425	9976
	tree weight	10	–	–	–	1	1	1	t	t	t	23305	4347	8608
	tree weight	25	–	–	–	1	1	1	t	t	t	23273	4347	21086
	tree weight	50	–	–	–	1	1	1	t	t	t	23274	4347	6785
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	32858	2149	4781
	0-restart	–	–	–	–	–	–	–	t	t	t	32809	2152	4781
	monotone	10	–	–	–	1	1	1	t	3421.0	t	154134	226092	165666
	monotone	25	–	–	–	1	1	1	t	3430.2	t	151849	226092	164891
	monotone	50	–	–	–	1	1	–	t	3417.4	t	153033	226092	39650
	reg forest	10	–	–	–	1	1	1	t	t	t	113207	195597	163771
	reg forest	25	–	–	–	1	1	–	t	t	t	116217	152011	39693
	reg forest	50	–	–	–	–	–	–	t	t	t	36407	38193	39655
	gap	10	–	–	–	1	1	1	t	t	t	136929	203775	144578
	gap	25	–	–	–	1	1	1	t	t	t	122240	169219	135538
	gap	50	–	–	–	1	1	1	t	t	t	116750	217572	123674
	leaf freq	10	–	–	–	1	1	1	t	t	t	143930	87754	129824
	leaf freq	25	–	–	–	–	–	–	t	t	t	36487	78203	39966
mzzv11	leaf freq	50	–	–	–	–	–	–	t	t	t	36741	44318	39765
	ssg	10	–	–	–	1	1	1	t	3032.3	t	126952	174275	143403
	ssg	25	–	–	–	1	1	1	t	t	t	116597	152849	146483
	ssg	50	–	–	–	1	1	1	t	t	t	118752	166450	98186
	tree weight	10	–	–	–	1	1	1	t	3436.0	t	153396	226092	157465
	tree weight	25	–	–	–	1	1	1	t	3417.6	t	109303	226092	114201
	tree weight	50	–	–	–	1	1	1	t	3435.1	t	132681	226092	135982
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	36456	38233	39933
	0-restart	–	–	–	–	–	–	–	t	t	t	36490	38195	39777
	monotone	10	–	–	–	–	–	–	323.5	312.0	530.1	1904	2505	3777
	monotone	25	–	–	–	–	–	–	324.2	311.0	535.5	1904	2505	3777
	monotone	50	–	–	–	–	–	–	322.9	311.0	530.9	1904	2505	3777
	reg forest	10	–	–	–	–	–	–	324.2	312.3	531.4	1904	2505	3777
	reg forest	25	–	–	–	–	–	–	323.1	312.2	531.5	1904	2505	3777
	reg forest	50	–	–	–	–	–	–	322.9	312.0	528.6	1904	2505	3777
	gap	10	–	–	–	–	–	–	320.6	311.6	529.5	1904	2505	3777
	gap	25	–	–	–	–	–	–	323.7	312.8	529.8	1904	2505	3777
	gap	50	–	–	–	–	–	–	323.1	308.6	530.2	1904	2505	3777
	leaf freq	10	–	–	–	–	–	–	324.9	312.5	529.8	1904	2505	3777
mzzv42z	leaf freq	25	–	–	–	–	–	–	323.4	311.6	531.0	1904	2505	3777
	leaf freq	50	–	–	–	–	–	–	322.5	311.2	530.7	1904	2505	3777
	ssg	10	7	6	1	1	1	1	446.7	351.9	480.6	1333	1247	2231
	ssg	25	7	–	1	1	–	1	447.9	311.0	576.7	1333	2505	3121
	ssg	50	7	–	3	1	–	1	446.9	311.0	609.2	1333	2505	2227
	tree weight	10	–	6	–	–	1	1	323.0	364.1	516.8	1904	1030	3100
	tree weight	25	–	6	3	–	1	1	323.3	364.9	627.9	1904	1030	2421
	tree weight	50	–	5	–	–	1	–	323.4	362.8	533.7	1904	1688	3777
	no clairvoyant	–	–	–	–	–	–	–	324.5	311.1	531.4	1904	2505	3777
	0-restart	–	–	–	–	–	–	–	322.5	310.2	531.6	1904	2505	3777
	monotone	10	–	–	–	–	–	–	185.7	329.2	250.8	987	547	900
	monotone	25	–	–	–	–	–	–	186.2	329.0	249.1	987	547	900
	monotone	50	–	–	–	–	–	–	186.2	329.4	250.6	987	547	900
	reg forest	10	–	–	–	–	–	–	186.8	329.6	250.6	987	547	900
	reg forest	25	–	–	–	–	–	–	186.3	327.8	250.9	987	547	900
	reg forest	50	–	–	–	–	–	–	186.9	329.0	251.0	987	547	900
	gap	10	–	–	–	–	–	–	186.8	329.0	251.2	987	547	900
	gap	25	–	–	–	–	–	–	186.9	328.1	251.4	987	547	900

cont. on next page ...

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
n2seq36q	gap	50	–	–	–	–	–	–	186.2	329.1	250.8	987	547	900
	leaf freq	10	–	–	–	–	–	–	186.4	329.7	250.7	987	547	900
	leaf freq	25	–	–	–	–	–	–	186.2	328.4	250.7	987	547	900
	leaf freq	50	–	–	–	–	–	–	184.9	328.6	250.9	987	547	900
	ssg	10	–	–	–	–	–	–	186.2	328.3	250.9	987	547	900
	ssg	25	–	–	–	–	–	–	186.4	329.0	250.5	987	547	900
	ssg	50	–	–	–	–	–	–	186.3	328.7	251.4	987	547	900
	tree weight	10	3	–	–	1	–	–	238.1	329.2	250.8	931	547	900
	tree weight	25	3	–	–	1	–	–	237.5	328.3	251.5	931	547	900
	tree weight	50	3	–	–	1	–	–	238.7	328.1	251.7	931	547	900
	no clairvoyant	–	–	–	–	–	–	–	186.5	329.2	250.6	987	547	900
	0-restart	–	–	–	–	–	–	–	185.8	328.2	250.5	987	547	900
	monotone	10	–	1	1	1	1	1	701.1	1272.7	373.9	3314	7694	1364
	monotone	25	–	1	–	1	1	–	698.5	1271.6	284.4	3314	7694	1836
	monotone	50	–	1	–	1	1	–	700.3	1271.5	283.2	3314	7694	1836
	reg forest	10	–	1	–	1	1	–	698.6	1272.4	285.3	3314	7694	1836
	reg forest	25	–	1	–	–	–	–	494.2	752.2	284.7	2900	4707	1836
	reg forest	50	–	1	–	–	–	–	495.4	751.7	284.5	2900	4707	1836
	gap	10	–	1	–	–	–	–	491.8	748.2	283.9	2900	4707	1836
	gap	25	–	1	–	–	–	–	491.7	749.7	284.4	2900	4707	1836
	gap	50	–	1	–	–	–	–	494.9	749.2	284.1	2900	4707	1836
	leaf freq	10	–	1	–	–	–	–	494.4	750.9	284.0	2900	4707	1836
	leaf freq	25	–	1	–	–	–	–	495.6	747.8	284.2	2900	4707	1836
	leaf freq	50	–	1	–	–	–	–	490.5	748.3	283.9	2900	4707	1836
	ssg	10	–	1	–	1	1	1	545.1	835.4	344.7	3163	3213	3005
	ssg	25	–	1	–	1	1	–	543.6	835.1	283.6	3163	3213	1836
	ssg	50	–	1	–	1	1	–	544.8	1161.8	284.1	3163	7440	1836
	tree weight	10	1	1	–	1	1	–	583.7	835.0	284.7	3082	3213	1836
	tree weight	25	1	1	–	1	1	–	629.0	836.7	283.6	2327	3213	1836
	tree weight	50	2	1	–	1	1	–	906.1	834.8	282.6	9259	3213	1836
	no clairvoyant	–	–	1	–	–	–	–	493.5	747.5	283.5	2900	4707	1836
	0-restart	–	–	–	–	–	–	–	493.1	262.5	286.5	2900	1647	1836
n3div36	monotone	10	1	1	1	–	1	–	t	t	t	85942	98022	76979
	monotone	25	1	1	1	–	–	–	t	t	t	86879	78531	76615
	monotone	50	1	1	1	–	–	–	t	t	t	86279	78290	76204
	reg forest	10	1	1	1	1	1	–	t	t	t	116570	96212	76257
	reg forest	25	1	1	1	–	–	–	t	t	t	86384	78278	75829
	reg forest	50	1	1	1	–	–	–	t	t	t	86733	79212	75931
	gap	10	1	1	1	1	1	1	t	t	t	110881	96831	101895
	gap	25	1	1	1	1	1	1	t	t	t	102947	99895	94720
	gap	50	1	1	1	–	–	–	t	t	t	85742	78570	76214
	leaf freq	10	1	1	1	1	1	1	t	t	t	97494	102035	94176
	leaf freq	25	1	1	1	1	1	1	t	t	t	95873	93915	88224
	leaf freq	50	1	1	1	1	1	1	t	t	t	77581	84593	76942
	ssg	10	1	1	1	1	1	1	t	t	t	117770	95641	100508
	ssg	25	1	1	1	1	1	1	t	t	t	119354	97065	99270
	ssg	50	1	1	1	1	1	1	t	t	t	115029	98245	93856
	tree weight	10	1	1	1	1	1	1	t	t	t	115842	97767	97137
	tree weight	25	1	1	1	1	1	1	t	t	t	111021	98120	96067
	tree weight	50	1	1	1	1	1	1	t	t	t	114055	99599	98087
	no clairvoyant	–	1	1	1	–	–	–	t	t	t	85904	78267	76091
	0-restart	–	–	–	–	–	–	–	t	t	t	67112	64264	63996
n5-3	monotone	10	–	–	–	–	–	–	34.3	43.4	33.6	1032	1492	786
	monotone	25	–	–	–	–	–	–	34.3	43.3	33.3	1032	1492	786
	monotone	50	–	–	–	–	–	–	34.4	43.6	33.3	1032	1492	786
	reg forest	10	–	–	–	–	–	–	34.9	44.0	33.9	1032	1492	786
	reg forest	25	–	–	–	–	–	–	35.0	44.2	33.6	1032	1492	786
	reg forest	50	–	–	–	–	–	–	34.8	44.0	34.1	1032	1492	786
	gap	10	–	–	–	–	–	–	34.3	43.4	33.4	1032	1492	786
	gap	25	–	–	–	–	–	–	34.2	43.6	33.3	1032	1492	786
	gap	50	–	–	–	–	–	–	34.4	43.4	33.4	1032	1492	786
	leaf freq	10	–	–	–	–	–	–	34.3	43.3	33.4	1032	1492	786
	leaf freq	25	–	–	–	–	–	–	34.3	43.4	33.2	1032	1492	786
	leaf freq	50	–	–	–	–	–	–	34.5	43.4	33.2	1032	1492	786
	ssg	10	–	–	–	–	–	–	34.3	43.5	33.2	1032	1492	786
	ssg	25	–	–	–	–	–	–	34.3	43.4	33.6	1032	1492	786
	ssg	50	–	–	–	–	–	–	34.4	43.4	33.4	1032	1492	786
	tree weight	10	–	–	–	–	–	–	34.1	43.2	33.2	1032	1492	786
	tree weight	25	–	–	–	–	–	–	34.3	43.3	33.3	1032	1492	786
	tree weight	50	–	–	–	–	–	–	34.3	43.4	33.3	1032	1492	786
	no clairvoyant	–	–	–	–	–	–	–	34.8	43.4	33.4	1032	1492	786
	0-restart	–	–	–	–	–	–	–	34.2	43.5	33.4	1032	1492	786
neos-1122047	monotone	10	–	–	–	–	–	–	10.2	10.4	10.3	1	1	1
	monotone	25	–	–	–	–	–	–	10.3	10.3	10.1	1	1	1
	monotone	50	–	–	–	–	–	–	10.4	10.4	10.6	1	1	1
	reg forest	10	–	–	–	–	–	–	10.4	10.4	10.3	1	1	1

cont. on next page ...

E. Appendix E

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
neos-1171448	reg forest	25	–	–	–	–	–	–	10.4	10.5	10.3	1	1	1
	reg forest	50	–	–	–	–	–	–	10.3	10.2	10.3	1	1	1
	gap	10	–	–	–	–	–	–	10.5	10.2	10.6	1	1	1
	gap	25	–	–	–	–	–	–	10.5	10.2	10.3	1	1	1
	gap	50	–	–	–	–	–	–	10.2	10.4	10.3	1	1	1
	leaf freq	10	–	–	–	–	–	–	10.2	10.3	10.4	1	1	1
	leaf freq	25	–	–	–	–	–	–	10.3	10.3	10.2	1	1	1
	leaf freq	50	–	–	–	–	–	–	10.6	10.3	10.4	1	1	1
	ssg	10	–	–	–	–	–	–	10.4	10.4	10.4	1	1	1
	ssg	25	–	–	–	–	–	–	10.4	10.6	10.4	1	1	1
	ssg	50	–	–	–	–	–	–	10.3	10.4	10.2	1	1	1
	tree weight	10	–	–	–	–	–	–	10.4	10.2	10.2	1	1	1
	tree weight	25	–	–	–	–	–	–	10.4	10.3	9.8	1	1	1
	tree weight	50	–	–	–	–	–	–	10.4	10.2	10.1	1	1	1
	no clairvoyant	–	–	–	–	–	–	–	10.7	10.1	10.3	1	1	1
	0-restart	–	–	–	–	–	–	–	10.3	10.2	10.1	1	1	1
	monotone	10	–	–	–	–	–	–	6.1	8.3	8.0	1	1	1
	monotone	25	–	–	–	–	–	–	6.1	8.4	7.8	1	1	1
	monotone	50	–	–	–	–	–	–	6.1	8.4	7.9	1	1	1
	reg forest	10	–	–	–	–	–	–	6.2	8.4	8.0	1	1	1
	reg forest	25	–	–	–	–	–	–	6.1	8.4	8.0	1	1	1
	reg forest	50	–	–	–	–	–	–	6.2	8.3	8.0	1	1	1
	gap	10	–	–	–	–	–	–	6.2	8.3	7.9	1	1	1
	gap	25	–	–	–	–	–	–	6.0	8.3	7.9	1	1	1
	gap	50	–	–	–	–	–	–	6.1	8.3	7.9	1	1	1
	leaf freq	10	–	–	–	–	–	–	6.1	8.2	7.9	1	1	1
	leaf freq	25	–	–	–	–	–	–	6.2	8.2	7.9	1	1	1
	leaf freq	50	–	–	–	–	–	–	6.1	8.3	7.9	1	1	1
	ssg	10	–	–	–	–	–	–	6.2	8.3	7.8	1	1	1
	ssg	25	–	–	–	–	–	–	6.1	8.3	7.9	1	1	1
	ssg	50	–	–	–	–	–	–	6.2	8.3	7.8	1	1	1
	tree weight	10	–	–	–	–	–	–	6.1	8.3	7.8	1	1	1
	tree weight	25	–	–	–	–	–	–	6.1	8.3	7.9	1	1	1
	tree weight	50	–	–	–	–	–	–	6.1	8.2	7.9	1	1	1
	no clairvoyant	–	–	–	–	–	–	–	6.2	8.3	7.9	1	1	1
	0-restart	–	–	–	–	–	–	–	6.1	8.2	7.9	1	1	1
neos-1171737	monotone	10	–	–	–	–	–	–	t	t	t	2471	2386	1687
	monotone	25	–	–	–	–	–	–	t	t	t	2471	2394	1705
	monotone	50	–	–	–	–	–	–	t	t	t	2476	2376	1686
	reg forest	10	–	–	–	–	–	–	t	t	t	2471	2382	1685
	reg forest	25	–	–	–	–	–	–	t	t	t	2477	2397	1686
	reg forest	50	–	–	–	–	–	–	t	t	t	2471	2390	1683
	gap	10	–	–	–	–	–	–	t	t	t	2474	2382	1685
	gap	25	–	–	–	–	–	–	t	t	t	2468	2387	1685
	gap	50	–	–	–	–	–	–	t	t	t	2471	2382	1680
	leaf freq	10	–	–	–	–	–	–	t	t	t	2471	2387	1686
	leaf freq	25	–	–	–	–	–	–	t	t	t	2476	2382	1686
	leaf freq	50	–	–	–	–	–	–	t	t	t	2476	2385	1685
	ssg	10	–	–	–	–	–	–	t	t	t	2467	2386	1692
	ssg	25	–	–	–	–	–	–	t	t	t	2471	2388	1685
	ssg	50	–	–	–	–	–	–	t	t	t	2471	2389	1685
	tree weight	10	–	–	–	–	–	–	t	t	t	2492	2386	1687
	tree weight	25	–	–	–	–	–	–	t	t	t	2472	2377	1689
	tree weight	50	–	–	–	–	–	–	t	t	t	2471	2382	1685
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	2471	2388	1689
	0-restart	–	–	–	–	–	–	–	t	t	t	2479	2386	1686
neos-1354092	monotone	10	–	–	–	–	–	–	t	t	t	79	179	122
	monotone	25	–	–	–	–	–	–	t	t	t	79	179	122
	monotone	50	–	–	–	–	–	–	t	t	t	79	179	122
	reg forest	10	–	–	–	–	–	–	t	t	t	79	179	122
	reg forest	25	–	–	–	–	–	–	t	t	t	79	179	122
	reg forest	50	–	–	–	–	–	–	t	t	t	79	179	122
	gap	10	–	–	–	–	–	–	t	t	t	79	179	122
	gap	25	–	–	–	–	–	–	t	t	t	79	179	122
	gap	50	–	–	–	–	–	–	t	t	t	79	179	122
	leaf freq	10	–	–	–	–	–	–	t	t	t	79	179	122
	leaf freq	25	–	–	–	–	–	–	t	t	t	79	179	122
	leaf freq	50	–	–	–	–	–	–	t	t	t	79	179	122
	ssg	10	–	–	–	–	–	–	t	t	t	79	179	122
	ssg	25	–	–	–	–	–	–	t	t	t	79	179	122
	ssg	50	–	–	–	–	–	–	t	t	t	79	179	122
	tree weight	10	–	–	–	–	–	–	t	t	t	79	179	122
	tree weight	25	–	–	–	–	–	–	t	t	t	79	179	122
	tree weight	50	–	–	–	–	–	–	t	t	t	79	179	122
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	79	179	122
	0-restart	–	–	–	–	–	–	–	t	t	t	79	179	122

cont. on next page ...

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
neos-1445765	monotone	10	–	–	–	–	–	–	30.9	36.2	34.6	117	151	107
	monotone	25	–	–	–	–	–	–	31.2	36.0	34.5	117	151	107
	monotone	50	–	–	–	–	–	–	30.9	35.8	34.3	117	151	107
	reg forest	10	–	–	–	–	–	–	31.0	36.1	34.5	117	151	107
	reg forest	25	–	–	–	–	–	–	31.0	36.3	34.6	117	151	107
	reg forest	50	–	–	–	–	–	–	30.9	36.1	34.9	117	151	107
	gap	10	–	–	–	–	–	–	30.8	36.0	34.6	117	151	107
	gap	25	–	–	–	–	–	–	31.0	35.8	34.3	117	151	107
	gap	50	–	–	–	–	–	–	31.0	36.1	34.4	117	151	107
	leaf freq	10	–	–	–	–	–	–	30.9	36.0	34.4	117	151	107
	leaf freq	25	–	–	–	–	–	–	30.9	35.9	34.5	117	151	107
	leaf freq	50	–	–	–	–	–	–	30.8	36.0	34.2	117	151	107
	ssg	10	–	–	–	–	–	–	31.0	36.0	34.8	117	151	107
	ssg	25	–	–	–	–	–	–	30.7	35.9	34.4	117	151	107
	ssg	50	–	–	–	–	–	–	30.9	36.0	34.4	117	151	107
	tree weight	10	–	–	–	–	–	–	30.8	36.0	34.5	117	151	107
	tree weight	25	–	–	–	–	–	–	31.0	36.0	34.4	117	151	107
	tree weight	50	–	–	–	–	–	–	30.9	35.9	34.6	117	151	107
	no clairvoyant	–	–	–	–	–	–	–	31.1	36.1	34.3	117	151	107
	0-restart	–	–	–	–	–	–	–	30.8	35.8	34.4	117	151	107
neos-1456979	monotone	10	–	–	–	1	1	1	t	t	t	14654	17149	11370
	monotone	25	–	–	–	1	1	–	t	t	t	14891	17129	11878
	monotone	50	–	–	–	–	–	–	t	t	t	16052	19372	11932
	reg forest	10	–	–	–	1	1	1	t	t	t	14652	17094	11508
	reg forest	25	–	–	–	1	1	1	t	t	t	14189	17633	11443
	reg forest	50	–	–	–	–	–	–	t	t	t	16070	19245	11877
	gap	10	–	–	–	1	1	1	t	t	t	13969	24790	10172
	gap	25	–	–	–	1	1	–	t	t	t	12691	24786	11982
	gap	50	–	–	–	1	1	–	t	t	t	13663	18601	11877
	leaf freq	10	–	–	–	1	1	–	t	t	t	13077	19543	11889
	leaf freq	25	–	–	–	–	–	–	t	t	t	16057	19398	11976
	leaf freq	50	–	–	–	–	–	–	t	t	t	16094	19318	11972
	ssg	10	–	–	–	1	1	1	t	t	t	17234	17038	10420
	ssg	25	–	–	–	1	1	1	t	t	t	17478	17009	10380
	ssg	50	–	–	–	1	1	1	2787.5	t	t	41174	17093	10405
	tree weight	10	–	–	–	1	1	1	t	t	t	14531	19796	13688
	tree weight	25	–	–	–	1	1	1	t	t	t	18552	20105	12253
	tree weight	50	–	–	–	1	1	1	t	t	t	16790	19809	9639
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	16116	19423	11876
	0-restart	–	–	–	–	–	–	–	t	t	t	16126	19358	11769
neos-1582420	monotone	10	–	–	1	–	–	–	33.9	17.5	21.8	436	77	61
	monotone	25	–	–	1	–	–	–	34.0	17.6	21.9	436	77	61
	monotone	50	–	–	1	–	–	–	34.0	17.5	21.9	436	77	61
	reg forest	10	–	–	1	–	–	–	34.3	17.7	22.0	436	77	61
	reg forest	25	–	–	1	–	–	–	34.3	17.7	22.1	436	77	61
	reg forest	50	–	–	1	–	–	–	34.3	17.7	22.1	436	77	61
	gap	10	–	–	1	–	–	–	34.0	17.5	22.0	436	77	61
	gap	25	–	–	1	–	–	–	33.9	17.5	22.1	436	77	61
	gap	50	–	–	1	–	–	–	33.8	17.5	21.9	436	77	61
	leaf freq	10	–	–	1	–	–	–	34.0	17.5	21.8	436	77	61
	leaf freq	25	–	–	1	–	–	–	33.8	17.6	21.9	436	77	61
	leaf freq	50	–	–	1	–	–	–	34.0	17.6	21.9	436	77	61
	ssg	10	–	–	1	–	–	–	34.0	17.5	21.9	436	77	61
	ssg	25	–	–	1	–	–	–	33.9	17.5	21.9	436	77	61
	ssg	50	–	–	1	–	–	–	34.0	17.4	21.8	436	77	61
	tree weight	10	–	–	1	–	–	–	33.8	17.5	21.9	436	77	61
	tree weight	25	–	–	1	–	–	–	33.9	17.5	22.0	436	77	61
	tree weight	50	–	–	1	–	–	–	33.9	17.7	21.9	436	77	61
	no clairvoyant	–	–	–	1	–	–	–	34.5	17.5	21.9	436	77	61
	0-restart	–	–	–	–	–	–	–	33.8	17.5	20.4	436	77	87
neos-.temuka	monotone	10	–	–	–	–	–	–	t	t	t	1	1	1
	monotone	25	–	–	–	–	–	–	t	t	t	1	1	1
	monotone	50	–	–	–	–	–	–	t	t	t	1	1	1
	reg forest	10	–	–	–	–	–	–	t	t	t	1	1	1
	reg forest	25	–	–	–	–	–	–	t	t	t	1	1	1
	reg forest	50	–	–	–	–	–	–	t	t	t	1	1	1
	gap	10	–	–	–	–	–	–	t	t	t	1	1	1
	gap	25	–	–	–	–	–	–	t	t	t	1	1	1
	gap	50	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	10	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	25	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	50	–	–	–	–	–	–	t	t	t	1	1	1
	ssg	10	–	–	–	–	–	–	t	t	t	1	1	1
	ssg	25	–	–	–	–	–	–	t	t	t	1	1	1
	ssg	50	–	–	–	–	–	–	t	t	t	1	1	1
	tree weight	10	–	–	–	–	–	–	t	t	t	1	1	1

cont. on next page ...

E. Appendix E

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
neos-.-crna	tree weight	25	–	–	–	–	–	–	t	t	t	1	1	1
	tree weight	50	–	–	–	–	–	–	t	t	t	1	1	1
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	1	1	1
	0-restart	–	–	–	–	–	–	–	t	t	t	1	1	1
	monotone	10	–	–	–	1	1	1	t	t	t	1764394	3428097	1141288
	monotone	25	–	–	–	1	1	1	t	t	t	1765898	3428730	1137588
	monotone	50	–	–	–	1	1	1	t	t	t	1752129	3434123	1139974
	reg forest	10	–	–	–	1	1	1	t	t	t	1748731	2648223	2400033
	reg forest	25	–	–	–	–	–	–	t	t	t	2397863	913926	1606853
	reg forest	50	–	–	–	–	–	–	t	t	t	2396324	912981	1603796
	gap	10	–	–	–	–	–	–	t	t	t	2414477	935365	1634555
	gap	25	–	–	–	–	–	–	t	t	t	2429722	930132	1628793
	gap	50	–	–	–	–	–	–	t	t	t	2425709	928592	1636464
	leaf freq	10	–	–	–	1	1	1	t	t	t	2961886	3008592	2393439
	leaf freq	25	–	–	–	1	1	1	t	t	t	4025633	3376019	4013418
	leaf freq	50	–	–	–	1	1	1	t	t	t	4026599	2980911	3165709
	ssg	10	–	–	–	1	1	1	t	t	t	2019598	3528786	2806427
	ssg	25	–	–	–	1	1	1	t	t	t	2263545	3431312	2612018
	ssg	50	–	–	–	1	1	1	t	t	t	1151669	2280886	3649233
neos-.-doon	tree weight	10	–	–	–	1	1	1	t	t	t	2089698	2986524	1905703
	tree weight	25	–	–	–	1	1	1	t	t	t	2088337	4078582	1907660
	tree weight	50	–	–	–	1	1	1	t	t	t	2091630	2714061	982177
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	2413861	928297	1636537
	0-restart	–	–	–	–	–	–	–	t	t	t	2426843	927255	1638732
	monotone	10	–	–	1	–	–	–	t	t	t	23943	4928	49645
	monotone	25	–	–	1	–	–	–	t	t	t	24246	4928	49881
	monotone	50	–	–	1	–	–	–	t	t	t	24339	4944	49647
	reg forest	10	–	–	1	–	–	–	t	t	t	24235	4926	49401
	reg forest	25	–	–	1	–	–	–	t	t	t	23978	4937	49669
	reg forest	50	–	–	1	–	–	–	t	t	t	24214	4942	50055
	gap	10	–	–	1	–	–	–	t	t	t	24223	4944	49645
	gap	25	–	–	1	–	–	–	t	t	t	24198	4919	49808
	gap	50	–	–	1	–	–	–	t	t	t	24251	4928	49645
	leaf freq	10	–	–	1	–	–	–	t	t	t	24200	4936	49655
	leaf freq	25	–	–	1	–	–	–	t	t	t	24530	4918	49394
	leaf freq	50	–	–	1	–	–	–	t	t	t	24272	4929	49326
	ssg	10	–	–	1	–	–	–	t	t	t	24231	4928	49330
	ssg	25	–	–	1	–	–	–	t	t	t	23965	4936	49716
	ssg	50	–	–	1	–	–	–	t	t	t	23965	4928	49680
neos-.-inde	tree weight	10	–	–	1	–	–	–	t	t	t	24229	4928	49645
	tree weight	25	–	–	1	–	–	–	t	t	t	24201	4935	49598
	tree weight	50	–	–	1	–	–	–	t	t	t	23961	4919	50085
	no clairvoyant	–	–	–	1	–	–	–	t	t	t	23991	4936	49661
	0-restart	–	–	–	–	–	–	–	t	t	t	24023	4936	31326
	monotone	10	1	1	–	1	–	–	t	t	3001.6	240430	234757	223895
	monotone	25	1	1	–	1	–	–	t	t	2991.1	240480	235010	223895
	monotone	50	1	1	–	1	–	–	t	t	3006.7	240030	235196	223895
	reg forest	10	1	1	–	1	–	–	t	t	2992.3	239867	232995	223895
	reg forest	25	1	1	–	–	–	–	t	t	3006.0	262976	232114	223895
	reg forest	50	1	1	–	–	–	–	t	t	3003.6	261829	233763	223895
	gap	10	1	1	–	–	–	–	t	t	3001.0	261504	234879	223895
	gap	25	1	1	–	–	–	–	t	t	3002.7	264066	234424	223895
	gap	50	1	1	–	–	–	–	t	t	3009.1	263073	235611	223895
	leaf freq	10	1	1	–	1	1	–	t	t	2982.3	208765	196963	223895
	leaf freq	25	1	1	–	1	1	–	t	t	2994.9	227732	205306	223895
	leaf freq	50	1	1	–	1	1	–	t	t	2998.7	236713	205965	223895
	ssg	10	1	1	–	1	1	–	t	t	3002.4	247998	173848	223895
	ssg	25	1	1	–	1	1	–	t	t	2999.3	256642	201188	223895
	ssg	50	1	1	–	1	1	–	t	t	3008.2	255799	210409	223895
neos-.-joes	tree weight	10	1	1	–	1	1	–	t	t	3009.3	252019	293358	223895
	tree weight	25	1	1	–	1	1	–	t	t	3003.7	245510	236403	223895
	tree weight	50	1	1	–	1	1	–	t	t	2980.2	266145	257670	223895
	no clairvoyant	–	1	1	–	–	–	–	t	t	3004.2	264358	236540	223895
	0-restart	–	–	–	–	–	–	–	t	1848.8	3007.0	279140	104553	223895
	monotone	10	–	–	–	–	–	–	18.1	16.4	15.9	1	1	1
	monotone	25	–	–	–	–	–	–	18.1	16.4	16.0	1	1	1
	monotone	50	–	–	–	–	–	–	18.1	16.4	15.8	1	1	1
	reg forest	10	–	–	–	–	–	–	18.2	16.4	16.2	1	1	1
	reg forest	25	–	–	–	–	–	–	18.3	17.1	16.1	1	1	1
	reg forest	50	–	–	–	–	–	–	17.9	16.5	16.1	1	1	1
	gap	10	–	–	–	–	–	–	18.1	16.4	16.0	1	1	1
	gap	25	–	–	–	–	–	–	18.2	16.4	15.8	1	1	1
	gap	50	–	–	–	–	–	–	18.1	16.3	16.0	1	1	1
	leaf freq	10	–	–	–	–	–	–	18.2	16.3	16.0	1	1	1
	leaf freq	25	–	–	–	–	–	–	18.1	16.4	15.9	1	1	1
	leaf freq	50	–	–	–	–	–	–	18.2	16.4	16.0	1	1	1

cont. on next page ...

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
neos-.-krka	ssg	10	-	-	-	-	-	-	18.1	16.4	16.1	1	1	1
	ssg	25	-	-	-	-	-	-	18.2	16.4	15.8	1	1	1
	ssg	50	-	-	-	-	-	-	18.1	16.4	15.9	1	1	1
	tree weight	10	-	-	-	-	-	-	18.1	16.5	16.0	1	1	1
	tree weight	25	-	-	-	-	-	-	18.1	16.3	15.8	1	1	1
	tree weight	50	-	-	-	-	-	-	18.1	16.3	16.0	1	1	1
	no clairvoyant	-	-	-	-	-	-	-	18.1	16.4	16.0	1	1	1
	0-restart	-	-	-	-	-	-	-	18.0	16.2	15.9	1	1	1
	monotone	10	-	-	-	1	1	1	237.3	112.4	106.4	6552	4696	4990
	monotone	25	-	-	-	1	1	1	238.2	111.9	105.5	6552	4696	4990
	monotone	50	-	-	-	1	1	1	239.4	112.8	105.1	6552	4696	4990
	reg forest	10	-	-	-	1	1	1	238.5	112.6	105.1	6552	4696	4990
	reg forest	25	-	-	-	-	-	-	71.0	632.1	220.8	4893	44465	12084
	reg forest	50	-	-	-	-	-	-	70.9	627.2	221.5	4893	44465	12084
	gap	10	-	-	-	1	-	1	238.1	623.5	105.5	6552	44465	4990
	gap	25	-	-	-	1	-	1	237.9	626.1	104.8	6552	44465	4990
	gap	50	-	-	-	1	-	1	238.4	620.5	104.9	6552	44465	4990
	leaf freq	10	-	-	-	1	1	1	239.0	513.9	105.7	6552	21251	4990
	leaf freq	25	-	-	-	1	1	1	238.7	730.0	105.6	6552	44487	4990
	leaf freq	50	-	-	-	1	-	1	238.9	623.5	105.3	6552	44465	4990
	ssg	10	-	-	-	1	1	1	238.8	305.3	105.6	6552	12556	4990
	ssg	25	-	-	-	1	1	1	237.9	1157.0	105.0	6552	160879	4990
	ssg	50	-	-	-	1	1	1	237.9	677.0	104.3	6552	28673	4990
	tree weight	10	-	-	-	1	-	1	235.2	621.1	105.1	6552	44465	4990
	tree weight	25	-	-	-	1	-	1	237.8	622.9	104.8	6552	44465	4990
	tree weight	50	-	-	-	1	-	1	237.2	621.6	104.9	6552	44465	4990
neos-.-loue	no clairvoyant	-	-	-	-	-	-	-	70.7	621.9	221.1	4893	44465	12084
	0-restart	-	-	-	-	-	-	-	69.8	619.0	220.6	4893	44465	12084
	monotone	10	-	-	-	1	1	1	1196.2	1414.8	t	62701	83579	220156
	monotone	25	-	-	-	1	1	1	1185.2	1418.2	t	62701	83579	219649
	monotone	50	-	-	-	1	1	1	1194.2	1419.4	t	62701	83579	219744
	reg forest	10	-	-	-	1	1	1	1196.2	1424.8	t	62701	83579	220190
	reg forest	25	-	-	-	-	-	-	t	3255.7	t	314794	228746	355747
	reg forest	50	-	-	-	-	-	-	t	3258.7	t	316069	228746	355695
	gap	10	-	-	-	1	1	1	1195.9	1416.2	t	62701	83579	219400
	gap	25	-	-	-	1	1	1	1195.6	1425.6	t	62701	83579	220166
	gap	50	-	-	-	1	1	1	1197.1	1404.5	t	62701	83579	220014
	leaf freq	10	-	-	-	1	1	1	1194.6	1424.0	t	62701	83579	219157
	leaf freq	25	-	-	-	1	1	1	1185.1	1413.2	t	62701	83579	219626
	leaf freq	50	-	-	-	1	1	1	1198.8	1414.6	t	62701	83579	217695
	ssg	10	-	-	-	1	1	1	1196.0	1420.0	t	62701	83579	220228
	ssg	25	-	-	-	1	1	1	1198.1	1414.7	t	62701	83579	219664
	ssg	50	-	-	-	1	1	1	1194.3	1422.4	t	62701	83579	220516
	tree weight	10	-	-	-	1	1	1	1192.8	1421.9	t	62701	83579	219337
	tree weight	25	-	-	-	1	1	1	1197.2	1422.3	t	62701	83579	219629
	tree weight	50	-	-	-	1	1	1	1196.4	1421.7	t	62701	83579	219873
	no clairvoyant	-	-	-	-	-	-	-	t	3252.1	t	314174	228746	361702
	0-restart	-	-	-	-	-	-	-	t	3218.1	t	317801	228746	361003
neos-.-murg	monotone	10	-	-	-	1	1	1	t	t	t	4264891	3396342	3853846
	monotone	25	-	-	-	1	1	1	t	t	t	4263454	3408427	3827582
	monotone	50	-	-	-	1	1	1	t	t	t	4281216	3406507	3842307
	reg forest	10	-	-	-	1	1	1	t	t	t	3865009	3395778	3775878
	reg forest	25	-	-	-	1	1	1	t	t	t	3787612	3973859	4077428
	reg forest	50	-	-	-	-	-	-	t	t	t	3692402	3918051	3638220
	gap	10	-	-	-	1	1	1	t	t	t	3400886	4227465	4199577
	gap	25	-	-	-	1	1	1	t	t	t	4049956	3854966	3645339
	gap	50	-	-	-	1	1	1	t	t	t	4553089	4214825	3998798
	leaf freq	10	-	-	-	1	1	1	t	t	t	3722465	3928991	3815978
	leaf freq	25	-	-	-	1	1	1	t	t	t	3709729	4140723	4139126
	leaf freq	50	-	-	-	1	1	1	t	t	t	3706251	4101766	3922475
	ssg	10	-	-	-	1	1	1	t	t	t	3720854	3901715	4191758
	ssg	25	-	-	-	1	1	1	t	t	t	4457707	4289354	4116587
	ssg	50	-	-	-	1	1	1	t	t	t	3876557	4062996	3823897
	tree weight	10	-	-	-	1	1	1	t	t	t	3596910	3738533	3888858
	tree weight	25	-	-	-	1	1	1	t	t	t	4127635	4397799	3434185
	tree weight	50	-	-	-	1	1	1	t	t	t	4104728	3534590	3719614
	no clairvoyant	-	-	-	-	-	-	-	t	t	t	3788238	4069349	3772605
	0-restart	-	-	-	-	-	-	-	t	t	t	3812977	4066288	3748342
neos-.-nubu	monotone	10	1	-	2	1	-	1	21.4	15.4	19.5	3106	1771	2168
	monotone	25	-	-	2	-	-	1	15.2	15.5	19.5	1865	1771	2168
	monotone	50	-	-	-	-	-	-	15.1	15.6	54.8	1865	1771	11285
	reg forest	10	-	-	-	-	-	-	15.8	16.0	56.6	1865	1771	11285
	reg forest	25	-	-	-	-	-	-	15.8	16.0	56.6	1865	1771	11285
	reg forest	50	-	-	-	-	-	-	15.8	16.0	56.6	1865	1771	11285
	gap	10	-	-	-	-	-	-	15.1	15.5	54.8	1865	1771	11285
	gap	25	-	-	-	-	-	-	15.3	15.6	54.7	1865	1771	11285

cont. on next page ...

E. Appendix E

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
neos--puriri	gap	50	–	–	–	–	–	–	15.1	15.5	54.9	1865	1771	11285
	leaf freq	10	–	–	–	–	–	–	15.2	15.5	55.0	1865	1771	11285
	leaf freq	25	–	–	–	–	–	–	15.2	15.6	55.0	1865	1771	11285
	leaf freq	50	–	–	–	–	–	–	15.2	15.5	54.9	1865	1771	11285
	ssg	10	1	–	2	1	–	1	21.4	15.6	19.3	3107	1771	2168
	ssg	25	3	–	2	1	–	1	17.9	15.5	26.0	1422	1771	4499
	ssg	50	–	–	1	–	–	1	15.2	15.5	17.8	1865	1771	1772
	tree weight	10	1	–	2	1	–	1	21.2	15.5	18.1	3106	1771	1822
	tree weight	25	1	–	2	1	–	1	21.3	15.4	20.3	3106	1771	2525
	tree weight	50	3	–	3	1	–	1	17.9	15.5	43.7	1422	1771	7079
	no clairvoyant	–	–	–	–	–	–	–	15.7	15.6	54.7	1865	1771	11285
	0-restart	–	–	–	–	–	–	–	15.2	15.4	54.6	1865	1771	11285
	monotone	10	–	–	–	1	1	1	t	t	t	1637	2177	1854
	monotone	25	–	–	–	1	1	1	t	t	t	1637	2162	1859
	monotone	50	–	–	–	1	–	–	t	t	t	1637	2035	2011
	reg forest	10	–	–	–	1	1	1	t	t	t	1626	2163	1865
	reg forest	25	–	–	–	1	1	1	t	t	t	1712	1910	1675
	reg forest	50	–	–	–	–	–	–	t	t	t	2155	2058	2050
	gap	10	–	–	–	1	1	1	t	t	t	1522	1631	1718
	gap	25	–	–	–	1	1	1	t	t	t	1522	1622	1765
	gap	50	–	–	–	1	–	1	t	t	t	1784	2079	1874
	leaf freq	10	–	–	–	–	–	–	t	t	t	2147	2054	2015
	leaf freq	25	–	–	–	–	–	–	t	t	t	2155	2062	2011
	leaf freq	50	–	–	–	–	–	–	t	t	t	2148	2058	2017
	ssg	10	–	–	–	1	1	1	t	t	t	1637	1624	1840
	ssg	25	–	–	–	1	1	1	t	t	t	1776	1622	1952
	ssg	50	–	–	–	1	1	1	t	t	t	1898	1623	1996
neos--awhea	tree weight	10	–	–	–	1	1	1	t	t	t	1636	2107	1864
	tree weight	25	–	–	–	1	1	1	t	t	t	1637	1900	1828
	tree weight	50	–	–	–	1	1	1	t	t	t	1786	1909	1849
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	2157	2057	2018
	0-restart	–	–	–	–	–	–	–	t	t	t	2148	2084	2013
	monotone	10	–	–	–	–	–	–	1.1	1.1	1.1	1	1	1
	monotone	25	–	–	–	–	–	–	1.1	1.1	1.1	1	1	1
	monotone	50	–	–	–	–	–	–	1.1	1.1	1.1	1	1	1
	reg forest	10	–	–	–	–	–	–	1.2	1.1	1.2	1	1	1
	reg forest	25	–	–	–	–	–	–	1.2	1.1	1.2	1	1	1
	reg forest	50	–	–	–	–	–	–	1.2	1.1	1.2	1	1	1
	gap	10	–	–	–	–	–	–	1.1	1.1	1.1	1	1	1
	gap	25	–	–	–	–	–	–	1.1	1.1	1.1	1	1	1
	gap	50	–	–	–	–	–	–	1.1	1.1	1.1	1	1	1
	leaf freq	10	–	–	–	–	–	–	1.1	1.1	1.1	1	1	1
	leaf freq	25	–	–	–	–	–	–	1.1	1.1	1.1	1	1	1
	leaf freq	50	–	–	–	–	–	–	1.1	1.1	1.1	1	1	1
	ssg	10	–	–	–	–	–	–	1.1	1.1	1.1	1	1	1
	ssg	25	–	–	–	–	–	–	1.1	1.1	1.1	1	1	1
	ssg	50	–	–	–	–	–	–	1.1	1.1	1.1	1	1	1
	tree weight	10	–	–	–	–	–	–	1.1	1.1	1.1	1	1	1
	tree weight	25	–	–	–	–	–	–	1.1	1.1	1.1	1	1	1
	tree weight	50	–	–	–	–	–	–	1.1	1.1	1.1	1	1	1
	no clairvoyant	–	–	–	–	–	–	–	1.2	1.1	1.1	1	1	1
	0-restart	–	–	–	–	–	–	–	1.1	1.1	1.1	1	1	1
neos--bobin	monotone	10	–	–	–	1	1	1	2551.8	3455.3	2582.2	12596	23911	14310
	monotone	25	–	–	–	–	–	1	1186.3	895.8	2575.1	4938	4848	14310
	monotone	50	–	–	–	–	–	–	1182.0	896.6	887.9	4938	4848	6257
	reg forest	10	–	–	–	–	–	1	1186.8	898.1	2593.3	4938	4848	14310
	reg forest	25	–	–	–	–	–	–	1189.8	895.8	888.6	4938	4848	6257
	reg forest	50	–	–	–	–	–	–	1183.3	899.5	888.9	4938	4848	6257
	gap	10	–	–	–	–	–	–	1177.7	896.8	891.6	4938	4848	6257
	gap	25	–	–	–	–	–	–	1172.4	894.4	887.1	4938	4848	6257
	gap	50	–	–	–	–	–	–	1172.7	895.1	891.8	4938	4848	6257
	leaf freq	10	–	–	–	–	–	–	1180.0	897.7	886.0	4938	4848	6257
	leaf freq	25	–	–	–	–	–	–	1184.0	902.5	885.5	4938	4848	6257
	leaf freq	50	–	–	–	–	–	–	1180.1	895.4	893.4	4938	4848	6257
	ssg	10	–	–	–	1	1	1	1707.8	t	2554.4	5577	26141	14310
	ssg	25	–	–	–	1	1	1	2336.7	t	2573.9	11055	35659	14310
	ssg	50	–	–	–	–	–	1	1178.1	906.2	3011.8	4938	4848	16089
	tree weight	10	–	–	–	1	1	1	t	2752.7	2221.3	79326	11448	17076
	tree weight	25	–	–	–	1	1	1	t	2761.3	t	79107	11448	43581
	tree weight	50	–	–	–	1	1	1	t	2745.0	t	79342	11448	47689
	no clairvoyant	–	–	–	–	–	–	–	1185.3	895.8	890.9	4938	4848	6257
	0-restart	–	–	–	–	–	–	–	1182.8	899.1	890.5	4938	4848	6257
neos--bohle	monotone	10	–	–	–	–	–	–	t	t	t	1	1	1
	monotone	25	–	–	–	–	–	–	t	t	t	1	1	1
	monotone	50	–	–	–	–	–	–	t	t	t	1	1	1
	reg forest	10	–	–	–	–	–	–	t	t	t	1	1	1

cont. on next page ...

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
neor--turama	reg forest	25	—	—	—	—	—	—	t	t	t	1	1	1
	reg forest	50	—	—	—	—	—	—	t	t	t	1	1	1
	gap	10	—	—	—	—	—	—	t	t	t	1	1	1
	gap	25	—	—	—	—	—	—	t	t	t	1	1	1
	gap	50	—	—	—	—	—	—	t	t	t	1	1	1
	leaf freq	10	—	—	—	—	—	—	t	t	t	1	1	1
	leaf freq	25	—	—	—	—	—	—	t	t	t	1	1	1
	leaf freq	50	—	—	—	—	—	—	t	t	t	1	1	1
	ssg	10	—	—	—	—	—	—	t	t	t	1	1	1
	ssg	25	—	—	—	—	—	—	t	t	t	1	1	1
	ssg	50	—	—	—	—	—	—	t	t	t	1	1	1
	tree weight	10	—	—	—	—	—	—	t	t	t	1	1	1
	tree weight	25	—	—	—	—	—	—	t	t	t	1	1	1
	tree weight	50	—	—	—	—	—	—	t	t	t	1	1	1
	no clairvoyant	—	—	—	—	—	—	—	t	t	t	1	1	1
	0-restart	—	—	—	—	—	—	—	t	t	t	1	1	1
	monotone	10	—	—	—	—	—	—	t	t	t	72	1073	10
	monotone	25	—	—	—	—	—	—	t	t	t	72	1073	10
	monotone	50	—	—	—	—	—	—	t	t	t	74	1073	10
	reg forest	10	—	—	—	—	—	—	t	t	t	72	1072	10
	reg forest	25	—	—	—	—	—	—	t	t	t	71	1073	10
	reg forest	50	—	—	—	—	—	—	t	t	t	72	1073	10
	gap	10	—	—	—	—	—	—	t	t	t	75	1073	10
	gap	25	—	—	—	—	—	—	t	t	t	71	1073	10
	gap	50	—	—	—	—	—	—	t	t	t	72	1076	10
	leaf freq	10	—	—	—	—	—	—	t	t	t	72	1073	10
	leaf freq	25	—	—	—	—	—	—	t	t	t	72	1073	10
	leaf freq	50	—	—	—	—	—	—	t	t	t	72	1073	10
	ssg	10	—	—	—	—	—	—	t	t	t	71	1073	10
	ssg	25	—	—	—	—	—	—	t	t	t	72	1073	10
	ssg	50	—	—	—	—	—	—	t	t	t	71	1073	10
	tree weight	10	—	—	—	—	—	—	t	t	t	74	1073	10
	tree weight	25	—	—	—	—	—	—	t	t	t	71	1071	10
	tree weight	50	—	—	—	—	—	—	t	t	t	72	1073	10
	no clairvoyant	—	—	—	—	—	—	—	t	t	t	72	1072	10
	0-restart	—	—	—	—	—	—	—	t	t	t	74	1073	10
neor--kasai	monotone	10	—	—	—	—	—	—	t	t	t	2168241	1621393	2321593
	monotone	25	—	—	—	—	—	—	t	t	t	2154283	1611565	2329633
	monotone	50	—	—	—	—	—	—	t	t	t	2158281	1621491	2335202
	reg forest	10	—	—	—	—	—	—	t	t	t	2147237	1613121	2315975
	reg forest	25	—	—	—	—	—	—	t	t	t	2148252	1618551	2328994
	reg forest	50	—	—	—	—	—	—	t	t	t	2153470	1617020	2320795
	gap	10	—	—	—	—	—	—	t	t	t	2150824	1622098	2308819
	gap	25	—	—	—	—	—	—	t	t	t	2166200	1623893	2319081
	gap	50	—	—	—	—	—	—	t	t	t	2152493	1615315	2329171
	leaf freq	10	—	—	—	—	—	—	t	t	t	2156610	1622330	2330401
	leaf freq	25	—	—	—	—	—	—	t	t	t	2148501	1619410	2330937
	leaf freq	50	—	—	—	—	—	—	t	t	t	2174963	1622551	2329722
	ssg	10	—	—	—	—	—	—	t	t	t	2154892	1619552	2327766
	ssg	25	—	—	—	—	—	—	t	t	t	2161322	1619761	2328740
	ssg	50	—	—	—	—	—	—	t	t	t	2156605	1622826	2334793
	tree weight	10	—	—	—	—	—	—	t	t	t	2155294	1621260	2321227
	tree weight	25	—	—	—	—	—	—	t	t	t	2145471	1619190	2334662
	tree weight	50	—	—	—	—	—	—	t	t	t	2161041	1621173	2325326
	no clairvoyant	—	—	—	—	—	—	—	t	t	t	2148745	1620862	2323783
	0-restart	—	—	—	—	—	—	—	t	t	t	2157682	1620139	2332963
neor--kumeu	monotone	10	—	—	—	—	—	—	t	t	t	68	40	1
	monotone	25	—	—	—	—	—	—	t	t	t	68	40	1
	monotone	50	—	—	—	—	—	—	t	t	t	67	40	1
	reg forest	10	—	—	—	—	—	—	t	t	t	68	40	1
	reg forest	25	—	—	—	—	—	—	t	t	t	69	40	1
	reg forest	50	—	—	—	—	—	—	t	t	t	68	40	1
	gap	10	—	—	—	—	—	—	t	t	t	69	40	1
	gap	25	—	—	—	—	—	—	t	t	t	68	40	1
	gap	50	—	—	—	—	—	—	t	t	t	67	39	1
	leaf freq	10	—	—	—	—	—	—	t	t	t	68	39	1
	leaf freq	25	—	—	—	—	—	—	t	t	t	68	39	1
	leaf freq	50	—	—	—	—	—	—	t	t	t	68	39	1
	ssg	10	—	—	—	—	—	—	t	t	t	68	40	1
	ssg	25	—	—	—	—	—	—	t	t	t	68	43	1
	ssg	50	—	—	—	—	—	—	t	t	t	68	40	1
	tree weight	10	—	—	—	—	—	—	t	t	t	67	43	1
	tree weight	25	—	—	—	—	—	—	t	t	t	68	40	1
	tree weight	50	—	—	—	—	—	—	t	t	t	68	40	1
	no clairvoyant	—	—	—	—	—	—	—	t	t	t	68	40	1
	0-restart	—	—	—	—	—	—	—	t	t	t	68	40	1

cont. on next page ...

E. Appendix E

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
neos--nidda	monotone	10	–	–	–	1	1	1	t	t	t	7350460	8055303	8006770
	monotone	25	–	–	–	1	1	1	t	t	t	7325497	8134786	8000601
	monotone	50	–	–	–	1	1	–	t	t	t	7335271	8068909	8465995
	reg forest	10	–	–	–	1	1	1	t	t	t	7389396	8363124	8157817
	reg forest	25	–	–	–	–	–	–	t	t	t	7152712	7491463	7816851
	reg forest	50	–	–	–	–	–	–	t	t	t	7229017	7491903	7806345
	gap	10	–	–	–	–	–	–	t	t	t	7677894	8016603	8331077
	gap	25	–	–	–	–	–	–	t	t	t	7685585	8043071	8412625
	gap	50	–	–	–	–	–	–	t	t	t	7685156	7996311	8371675
	leaf freq	10	–	–	–	1	1	1	t	t	t	7735673	8496604	8835359
	leaf freq	25	–	–	–	1	1	1	t	t	t	7616538	8753882	8178000
	leaf freq	50	–	–	–	1	1	1	t	t	t	7571923	8747486	7952207
	ssg	10	–	–	–	1	1	1	t	t	t	7143552	7973159	8032514
	ssg	25	–	–	–	1	1	1	t	t	t	7124035	7597790	8156112
	ssg	50	–	–	–	1	1	1	t	t	t	7085588	7617103	8144908
	tree weight	10	–	–	–	1	1	1	t	t	t	7603982	8059060	8328594
	tree weight	25	–	–	–	1	1	1	t	t	t	7545734	8094328	8337399
	tree weight	50	–	–	–	1	1	1	t	t	t	7532719	8233678	8350748
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	7664833	8043420	8458031
	0-restart	–	–	–	–	–	–	–	t	t	t	7769814	8001812	8485887
neos--wolgan	monotone	10	–	–	–	1	–	1	t	t	t	1232	151	1194
	monotone	25	–	–	–	–	–	1	t	t	t	15656	151	1190
	monotone	50	–	–	–	–	–	1	t	t	t	15661	151	1190
	reg forest	10	–	–	–	1	–	1	t	t	t	1230	150	1194
	reg forest	25	–	–	–	–	–	–	t	t	t	15662	153	2963
	reg forest	50	–	–	–	–	–	–	t	t	t	15661	150	2982
	gap	10	–	–	–	–	–	–	t	t	t	15654	151	2978
	gap	25	–	–	–	–	–	–	t	t	t	15673	151	2959
	gap	50	–	–	–	–	–	–	t	t	t	15661	150	3031
	leaf freq	10	–	–	–	–	–	–	t	t	t	15662	151	3013
	leaf freq	25	–	–	–	–	–	–	t	t	t	15662	151	3032
	leaf freq	50	–	–	–	–	–	–	t	t	t	15661	150	2957
	ssg	10	–	–	–	–	–	1	t	t	t	15656	151	1194
	ssg	25	–	–	–	–	–	1	t	t	t	15654	151	1198
	ssg	50	–	–	–	–	–	1	t	t	t	15661	150	1158
	tree weight	10	–	–	–	1	–	1	t	t	t	15585	151	1194
	tree weight	25	–	–	–	1	–	1	t	t	t	15585	151	1190
	tree weight	50	–	–	–	1	–	1	t	t	t	15585	151	1194
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	15662	151	2957
	0-restart	–	–	–	–	–	–	–	t	t	t	15661	151	2979
neos--rahue	monotone	10	–	–	–	1	1	1	t	t	t	1516	1359	1516
	monotone	25	–	–	–	1	–	–	t	t	t	1515	1871	1889
	monotone	50	–	–	–	–	–	–	t	t	t	1969	1866	1882
	reg forest	10	–	–	–	1	1	1	t	t	t	1687	1360	1623
	reg forest	25	–	–	–	1	–	1	t	t	t	1770	1875	1686
	reg forest	50	–	–	–	–	–	–	t	t	t	1935	1902	1886
	gap	10	–	–	–	1	1	1	t	t	t	1598	1359	1671
	gap	25	–	–	–	1	1	1	t	t	t	1586	1762	1695
	gap	50	–	–	–	1	1	–	t	t	t	1627	1767	1886
	leaf freq	10	–	–	–	–	–	–	t	t	t	1963	1862	1881
	leaf freq	25	–	–	–	–	–	–	t	t	t	1965	1871	1886
	leaf freq	50	–	–	–	–	–	–	t	t	t	1971	1889	1881
	ssg	10	–	–	–	1	1	–	t	t	t	1609	1486	1884
	ssg	25	–	–	–	1	–	–	t	t	t	1505	1868	1873
	ssg	50	–	–	–	1	–	–	t	t	t	1639	1871	1873
	tree weight	10	–	–	–	1	1	1	t	t	t	1583	1366	1516
	tree weight	25	–	–	–	1	1	1	t	t	t	1663	1360	1516
	tree weight	50	–	–	–	1	1	1	t	t	t	1756	1640	1766
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	1966	1867	1889
	0-restart	–	–	–	–	–	–	–	t	t	t	1965	1874	1880
neos--snowy	monotone	10	–	–	–	1	1	1	t	t	t	4496858	2779691	2675417
	monotone	25	–	–	–	1	1	1	t	t	t	4498092	2779586	2651255
	monotone	50	–	–	–	1	1	1	t	t	t	4503691	2782259	2653818
	reg forest	10	–	–	–	1	–	1	t	t	t	2563209	2752310	2643951
	reg forest	25	–	–	–	1	–	1	t	t	t	2778455	2723657	2741638
	reg forest	50	–	–	–	–	–	–	t	t	t	2921407	2725321	2697615
	gap	10	–	–	–	1	1	1	t	t	t	2754077	2734617	2693242
	gap	25	–	–	–	1	1	1	t	t	t	2604790	2740522	2687097
	gap	50	–	–	–	1	1	1	t	t	t	2642076	2745932	2418428
	leaf freq	10	–	–	–	1	1	1	t	t	t	3013192	2801394	2632909
	leaf freq	25	–	–	–	1	1	1	t	t	t	2841349	2791322	2731582
	leaf freq	50	–	–	–	1	1	1	t	t	t	3028161	2819390	3090440
	ssg	10	–	–	–	1	1	1	t	t	t	2743076	2957005	2874432
	ssg	25	–	–	–	1	1	1	t	t	t	2907915	2953891	2921768
	ssg	50	–	–	–	1	1	1	t	t	t	3132795	2960073	2817325
	tree weight	10	–	–	–	1	1	1	t	t	t	2598289	2638365	2614214

cont. on next page ...

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
neos-.-tavua	tree weight	25	–	–	–	1	1	1	t	t	t	2608120	2635310	2625329
	tree weight	50	–	–	–	1	1	1	t	t	t	2615846	2638697	2626046
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	2991115	2777817	2755180
	0-restart	–	–	–	–	–	–	–	t	t	t	2989726	2765393	2791112
	monotone	10	–	–	–	1	–	1	t	t	t	8992	8938	12428
	monotone	25	–	–	–	–	–	1	t	t	t	10708	8927	12456
	monotone	50	–	–	–	–	–	–	t	t	t	10650	8935	9348
	reg forest	10	–	–	–	1	1	1	t	t	t	8977	7741	7701
	reg forest	25	–	–	–	–	–	1	t	t	t	10692	8901	7677
	reg forest	50	–	–	–	–	–	–	t	t	t	10679	8932	9143
	gap	10	–	–	–	1	1	1	t	t	t	10290	8197	9576
	gap	25	–	–	–	1	–	1	t	t	t	9442	9057	7855
	gap	50	–	–	–	–	–	–	t	t	t	10730	8950	9271
	leaf freq	10	–	–	–	1	1	1	t	t	t	10310	9997	12129
	leaf freq	25	–	–	–	1	–	–	t	t	t	10279	9052	9262
	leaf freq	50	–	–	–	–	–	–	t	t	t	10985	8957	9203
	ssg	10	–	–	–	1	1	1	t	t	t	10425	8098	11285
	ssg	25	–	–	–	1	–	1	t	t	t	9329	8907	15877
	ssg	50	–	–	–	–	–	–	t	t	t	10784	8922	9267
	tree weight	10	–	–	–	1	1	1	t	t	t	10655	9007	12439
neos-.-turia	tree weight	25	–	–	–	1	1	1	t	t	t	10796	9006	11752
	tree weight	50	–	–	–	1	1	1	t	t	t	13284	7567	9663
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	10574	8877	9262
	0-restart	–	–	–	–	–	–	–	t	t	t	10786	8925	9275
	monotone	10	3	2	2	–	–	–	433.3	382.6	491.1	4	3	3
	monotone	25	3	2	2	–	–	–	439.2	378.9	489.4	4	3	3
	monotone	50	3	2	2	–	–	–	441.5	379.5	499.2	4	3	3
	reg forest	10	3	2	2	–	–	–	441.0	381.0	497.7	4	3	3
	reg forest	25	3	2	2	–	–	–	436.2	383.4	503.4	4	3	3
	reg forest	50	3	2	2	–	–	–	432.7	379.6	491.8	4	3	3
	gap	10	3	2	2	–	–	–	445.0	395.9	487.2	4	3	3
	gap	25	3	2	2	–	–	–	436.2	381.9	487.6	4	3	3
	gap	50	3	2	2	–	–	–	434.3	380.1	487.7	4	3	3
	leaf freq	10	3	2	2	–	–	–	440.8	380.3	485.8	4	3	3
	leaf freq	25	3	2	2	–	–	–	433.4	381.1	489.6	4	3	3
	leaf freq	50	3	2	2	–	–	–	443.1	381.3	489.7	4	3	3
	ssg	10	3	2	2	–	–	–	449.8	389.0	486.3	4	3	3
	ssg	25	3	2	2	–	–	–	434.2	380.1	488.0	4	3	3
	ssg	50	3	2	2	–	–	–	436.1	380.3	491.9	4	3	3
	tree weight	10	3	2	2	–	–	–	435.6	390.5	486.7	4	3	3
neos-.-waihi	tree weight	25	3	2	2	–	–	–	432.8	381.1	487.1	4	3	3
	tree weight	50	3	2	2	–	–	–	432.9	378.9	485.5	4	3	3
	no clairvoyant	–	3	2	2	–	–	–	433.9	377.9	488.8	4	3	3
	0-restart	–	–	–	–	–	–	–	859.5	991.6	714.2	972	1159	452
	monotone	10	–	–	–	–	–	–	t	t	t	1	1	1
	monotone	25	–	–	–	–	–	–	t	t	t	1	1	1
	monotone	50	–	–	–	–	–	–	t	t	t	1	1	1
	reg forest	10	–	–	–	–	–	–	t	t	t	1	1	1
	reg forest	25	–	–	–	–	–	–	t	t	t	1	1	1
	reg forest	50	–	–	–	–	–	–	t	t	t	1	1	1
	gap	10	–	–	–	–	–	–	t	t	t	1	1	1
	gap	25	–	–	–	–	–	–	t	t	t	1	1	1
	gap	50	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	10	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	25	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	50	–	–	–	–	–	–	t	t	t	1	1	1
	ssg	10	–	–	–	–	–	–	t	t	t	1	1	1
	ssg	25	–	–	–	–	–	–	t	t	t	1	1	1
	ssg	50	–	–	–	–	–	–	t	t	t	1	1	1
	tree weight	10	–	–	–	–	–	–	t	t	t	1	1	1
neos-.-tutaki	tree weight	25	–	–	–	–	–	–	t	t	t	1	1	1
	tree weight	50	–	–	–	–	–	–	t	t	t	1	1	1
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	1	1	1
	0-restart	–	–	–	–	–	–	–	t	t	t	1	1	1
	monotone	10	2	2	2	–	–	–	t	t	t	1953	2757	2699
	monotone	25	2	2	2	–	–	–	t	t	t	1953	2741	2748
	monotone	50	2	2	2	–	–	–	t	t	t	1953	2741	2674
	reg forest	10	2	2	2	–	–	–	t	t	t	1937	2741	2739
	reg forest	25	2	2	2	–	–	–	t	t	t	1926	2741	2712
	reg forest	50	2	2	2	–	–	–	t	t	t	1953	2760	2748
	gap	10	2	2	2	–	–	–	t	t	t	1946	2741	2705
	gap	25	2	2	2	–	–	–	t	t	t	1953	2781	2748
	gap	50	2	2	2	–	–	–	t	t	t	1953	2741	2748
	leaf freq	10	2	2	2	–	–	–	t	t	t	1905	2753	2748
	leaf freq	25	2	2	2	–	–	–	t	t	t	1953	2760	2748
	leaf freq	50	2	2	2	–	–	–	t	t	t	1918	2741	2748

cont. on next page ...

E. Appendix E

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
neos-.-widden	ssg	10	2	2	2	–	–	–	t	t	t	1953	2753	2745
	ssg	25	2	2	2	–	–	–	t	t	t	1953	2717	2748
	ssg	50	2	2	2	–	–	–	t	t	t	1936	2773	2816
	tree weight	10	2	2	2	–	–	–	t	t	t	1953	2741	2688
	tree weight	25	2	2	2	–	–	–	t	t	t	1953	2735	2748
	tree weight	50	2	2	2	–	–	–	t	t	t	1953	2741	2748
	no clairvoyant	–	2	2	2	–	–	–	t	t	t	1953	2732	2744
	0-restart	–	–	–	–	–	–	–	t	t	t	8227	8191	10523
	monotone	10	–	–	–	–	–	–	422.0	397.1	510.6	1211	1878	1013
	monotone	25	–	–	–	–	–	–	422.1	396.3	510.0	1211	1878	1013
	monotone	50	–	–	–	–	–	–	418.2	396.2	509.5	1211	1878	1013
	reg forest	10	–	–	–	–	–	–	422.9	395.9	515.9	1211	1878	1013
	reg forest	25	–	–	–	–	–	–	421.4	397.2	511.0	1211	1878	1013
	reg forest	50	–	–	–	–	–	–	419.6	399.1	514.7	1211	1878	1013
	gap	10	–	–	–	–	–	–	422.2	400.8	510.4	1211	1878	1013
	gap	25	–	–	–	–	–	–	421.2	398.9	510.3	1211	1878	1013
	gap	50	–	–	–	–	–	–	422.8	397.3	514.9	1211	1878	1013
	leaf freq	10	–	–	–	–	–	–	415.6	399.3	512.0	1211	1878	1013
	leaf freq	25	–	–	–	–	–	–	418.8	396.7	510.9	1211	1878	1013
	leaf freq	50	–	–	–	–	–	–	421.2	396.6	510.9	1211	1878	1013
	ssg	10	–	–	–	–	–	–	421.1	396.6	508.0	1211	1878	1013
	ssg	25	–	–	–	–	–	–	422.7	397.1	511.8	1211	1878	1013
	ssg	50	–	–	–	–	–	–	421.4	395.6	510.3	1211	1878	1013
	tree weight	10	–	–	–	–	–	–	421.8	397.7	515.4	1211	1878	1013
	tree weight	25	–	–	–	–	–	–	421.6	398.3	510.7	1211	1878	1013
	tree weight	50	–	–	–	–	–	–	424.3	395.4	511.2	1211	1878	1013
neos-.-atrato	no clairvoyant	–	–	–	–	–	–	–	423.4	399.8	510.0	1211	1878	1013
	0-restart	–	–	–	–	–	–	–	418.2	399.1	520.4	1211	1878	1013
	monotone	10	–	–	–	1	–	–	754.9	1299.4	694.5	74499	167590	79322
	monotone	25	–	–	–	–	–	–	757.1	631.5	707.7	74499	59085	79322
	monotone	50	–	–	–	–	–	–	755.6	632.4	698.4	74499	59085	79322
	reg forest	10	1	–	–	1	1	–	718.9	1322.4	707.5	77111	167590	79322
	reg forest	25	–	–	–	–	–	–	769.2	641.3	711.7	74499	59085	79322
	reg forest	50	–	–	–	–	–	–	762.8	640.1	710.8	74499	59085	79322
	gap	10	–	–	–	–	–	–	758.0	634.4	700.2	74499	59085	79322
	gap	25	–	–	–	–	–	–	760.6	631.7	704.5	74499	59085	79322
	gap	50	–	–	–	–	–	–	756.4	629.8	700.1	74499	59085	79322
	leaf freq	10	1	–	1	1	1	1	752.4	861.1	400.2	98980	118538	31408
	leaf freq	25	2	1	1	1	1	1	554.3	3518.6	403.9	58928	390011	31408
	leaf freq	50	1	1	2	1	1	1	605.5	3514.0	520.6	57273	390011	45227
	ssg	10	–	–	–	1	1	1	804.4	504.4	255.4	88927	47567	19240
	ssg	25	–	1	–	1	1	1	409.4	491.2	263.2	45834	46792	23582
	ssg	50	1	1	–	1	1	1	844.8	604.7	387.7	110006	65292	37479
	tree weight	10	–	–	1	1	1	1	461.6	650.6	488.4	45724	72981	45735
	tree weight	25	–	–	1	1	1	1	461.4	950.8	365.8	45724	110250	36533
	tree weight	50	–	–	1	1	1	1	462.1	1277.7	335.6	45724	149039	32583
	no clairvoyant	–	–	–	–	–	–	–	764.2	628.7	700.7	74499	59085	79322
	0-restart	–	–	–	–	–	–	–	758.4	631.6	700.0	74499	59085	79322
neos-.-toguru	monotone	10	–	–	–	1	–	–	t	t	t	7194	6470	5268
	monotone	25	–	–	–	–	–	–	t	t	t	4864	6497	5213
	monotone	50	–	–	–	–	–	–	t	t	t	4732	6477	5194
	reg forest	10	–	–	–	1	–	–	t	t	t	6131	6477	5184
	reg forest	25	–	–	–	–	–	–	t	t	t	4727	6489	5234
	reg forest	50	–	–	–	–	–	–	t	t	t	4842	6497	5177
	gap	10	–	–	–	1	1	1	t	t	t	3657	8535	4158
	gap	25	–	–	–	1	1	1	t	t	t	4362	8203	2960
	gap	50	–	–	–	–	1	–	t	t	t	4727	5604	5272
	leaf freq	10	–	–	–	–	–	–	t	t	t	4756	6489	5194
	leaf freq	25	–	–	–	–	–	–	t	t	t	4727	6445	5240
	leaf freq	50	–	–	–	–	–	–	t	t	t	4727	6470	5156
	ssg	10	–	–	–	1	1	1	t	t	t	6853	3754	922
	ssg	25	–	–	–	1	1	1	t	t	t	5838	3885	1701
	ssg	50	–	–	–	1	1	1	t	t	t	6122	3943	2580
	tree weight	10	–	–	–	1	1	–	t	t	t	5899	8003	5234
	tree weight	25	–	–	–	1	–	–	t	t	t	4755	6530	5249
	tree weight	50	–	–	–	1	–	–	t	t	t	7119	6501	5280
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	4811	6522	5300
	0-restart	–	–	–	–	–	–	–	t	t	t	4835	6497	5155
neos-.-berkel	monotone	10	–	–	–	–	–	–	t	t	t	534123	526887	377443
	monotone	25	–	–	–	–	–	–	t	t	t	535924	525639	375294
	monotone	50	–	–	–	–	–	–	t	t	t	537601	526971	376760
	reg forest	10	–	–	–	–	–	–	t	t	t	535591	525736	376582
	reg forest	25	–	–	–	–	–	–	t	t	t	535031	525227	375652
	reg forest	50	–	–	–	–	–	–	t	t	t	532953	524730	374768
	gap	10	–	–	–	–	–	–	t	t	t	536140	524998	374263
	gap	25	–	–	–	–	–	–	t	t	t	538639	527235	374919

cont. on next page ...

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
neos-. -cuanza	gap	50	-	-	-	-	-	-	t	t	t	538303	526622	376877
	leaf freq	10	-	-	-	-	-	-	t	t	t	535942	526478	377397
	leaf freq	25	-	-	-	-	-	-	t	t	t	536104	526275	377018
	leaf freq	50	-	-	-	-	-	-	t	t	t	537383	525961	374948
	ssg	10	-	-	-	-	-	-	t	t	t	536064	527988	375709
	ssg	25	-	-	-	-	-	-	t	t	t	536257	525550	375664
	ssg	50	-	-	-	-	-	-	t	t	t	536868	526760	375453
	tree weight	10	-	-	-	-	-	-	t	t	t	537202	528282	377296
	tree weight	25	-	-	-	-	-	-	t	t	t	538539	524608	375848
	tree weight	50	-	-	-	-	-	-	t	t	t	536689	525509	375218
	no clairvoyant	-	-	-	-	-	-	-	t	t	t	534352	526828	376220
	0-restart	-	-	-	-	-	-	-	t	t	t	535412	526474	375921
	monotone	10	-	-	-	-	-	-	t	t	t	6	7	7
	monotone	25	-	-	-	-	-	-	t	t	t	6	7	7
	monotone	50	-	-	-	-	-	-	t	t	t	6	7	7
	reg forest	10	-	-	-	-	-	-	t	t	t	6	7	7
	reg forest	25	-	-	-	-	-	-	t	t	t	6	7	7
	reg forest	50	-	-	-	-	-	-	t	t	t	6	7	7
	gap	10	-	-	-	-	-	-	t	t	t	6	7	7
	gap	25	-	-	-	-	-	-	t	t	t	6	7	7
	gap	50	-	-	-	-	-	-	t	t	t	6	7	7
	leaf freq	10	-	-	-	-	-	-	t	t	t	6	7	7
	leaf freq	25	-	-	-	-	-	-	t	t	t	6	7	7
	leaf freq	50	-	-	-	-	-	-	t	t	t	6	7	7
	ssg	10	-	-	-	-	-	-	t	t	t	6	7	7
	ssg	25	-	-	-	-	-	-	t	t	t	6	7	7
	ssg	50	-	-	-	-	-	-	t	t	t	6	7	7
	tree weight	10	-	-	-	-	-	-	t	t	t	6	7	7
	tree weight	25	-	-	-	-	-	-	t	t	t	6	7	7
	tree weight	50	-	-	-	-	-	-	t	t	t	6	7	7
neos-. -cygnet	no clairvoyant	-	-	-	-	-	-	-	t	t	t	6	7	7
	0-restart	-	-	-	-	-	-	-	t	t	t	6	7	7
	monotone	10	-	-	-	-	-	-	t	t	t	58	46	45
	monotone	25	-	-	-	-	-	-	t	t	t	58	46	44
	monotone	50	-	-	-	-	-	-	t	t	t	58	46	45
	reg forest	10	-	-	-	-	-	-	t	t	t	58	45	45
	reg forest	25	-	-	-	-	-	-	t	t	t	58	46	46
	reg forest	50	-	-	-	-	-	-	t	t	t	58	46	45
	gap	10	-	-	-	-	-	-	t	t	t	58	45	46
	gap	25	-	-	-	-	-	-	t	t	t	58	46	45
	gap	50	-	-	-	-	-	-	t	t	t	58	48	44
	leaf freq	10	-	-	-	-	-	-	t	t	t	58	46	45
	leaf freq	25	-	-	-	-	-	-	t	t	t	58	46	44
	leaf freq	50	-	-	-	-	-	-	t	t	t	58	46	45
	ssg	10	-	-	-	-	-	-	t	t	t	58	46	44
	ssg	25	-	-	-	-	-	-	t	t	t	58	45	44
	ssg	50	-	-	-	-	-	-	t	t	t	58	46	45
	tree weight	10	-	-	-	-	-	-	t	t	t	58	45	44
	tree weight	25	-	-	-	-	-	-	t	t	t	58	46	45
	tree weight	50	-	-	-	-	-	-	t	t	t	58	46	45
	no clairvoyant	-	-	-	-	-	-	-	t	t	t	58	46	45
	0-restart	-	-	-	-	-	-	-	t	t	t	58	46	45
neos-. -huahum	monotone	10	-	-	-	1	-	1	t	t	t	18292	22272	16912
	monotone	25	-	-	-	-	-	-	t	t	t	20235	22272	21572
	monotone	50	-	-	-	-	-	-	t	t	t	20251	22273	21292
	reg forest	10	-	-	-	1	1	1	t	t	t	18316	23842	16952
	reg forest	25	-	-	-	-	-	-	t	t	t	20471	22271	21311
	reg forest	50	-	-	-	-	-	-	t	t	t	20133	22108	21233
	gap	10	-	-	-	1	1	1	t	t	t	17790	22973	16856
	gap	25	-	-	-	1	1	1	t	t	t	16969	21342	19952
	gap	50	-	-	-	1	1	1	t	t	t	17457	19924	16962
	leaf freq	10	-	-	-	1	1	1	t	t	t	17600	18408	16863
	leaf freq	25	-	-	-	-	1	1	t	t	t	20160	22129	17579
	leaf freq	50	-	-	-	-	-	-	t	t	t	20061	22245	21251
	ssg	10	-	-	-	1	1	1	t	t	t	17021	21441	17267
	ssg	25	-	-	-	1	1	1	t	t	t	17369	21651	22743
	ssg	50	-	-	-	1	1	1	t	t	t	18117	21028	18084
	tree weight	10	-	-	-	1	1	1	t	t	t	18379	22937	21053
	tree weight	25	-	-	-	1	1	1	t	t	t	18375	22992	20936
	tree weight	50	-	-	-	1	1	1	t	t	t	18532	23017	21042
	no clairvoyant	-	-	-	-	-	-	-	t	t	t	20396	22242	21217
	0-restart	-	-	-	-	-	-	-	t	t	t	20101	22053	21424
neos-. -jarama	monotone	10	-	-	-	-	-	-	t	t	t	1	1	1
	monotone	25	-	-	-	-	-	-	t	t	t	1	1	1
	monotone	50	-	-	-	-	-	-	t	t	t	1	1	1
	reg forest	10	-	-	-	-	-	-	t	t	t	1	1	1

cont. on next page ...

E. Appendix E

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
neos--kakapo	reg forest	25	–	–	–	–	–	–	t	t	t	1	1	1
	reg forest	50	–	–	–	–	–	–	t	t	t	1	1	1
	gap	10	–	–	–	–	–	–	t	t	t	1	1	1
	gap	25	–	–	–	–	–	–	t	t	t	1	1	1
	gap	50	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	10	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	25	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	50	–	–	–	–	–	–	t	t	t	1	1	1
	ssg	10	–	–	–	–	–	–	t	t	t	1	1	1
	ssg	25	–	–	–	–	–	–	t	t	t	1	1	1
	ssg	50	–	–	–	–	–	–	t	t	t	1	1	1
	tree weight	10	–	–	–	–	–	–	t	t	t	1	1	1
	tree weight	25	–	–	–	–	–	–	t	t	t	1	1	1
	tree weight	50	–	–	–	–	–	–	t	t	t	1	1	1
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	1	1	1
	0-restart	–	–	–	–	–	–	–	t	t	t	1	1	1
	monotone	10	–	–	–	1	1	1	t	t	t	837053	817287	862695
	monotone	25	–	–	–	–	–	1	t	t	t	858562	851304	845844
	monotone	50	–	–	–	–	–	–	t	t	t	862131	838772	820490
	reg forest	10	–	–	–	1	1	1	t	t	t	864384	812459	862257
	reg forest	25	–	–	–	–	–	1	t	t	t	856195	830700	909029
	reg forest	50	–	–	–	–	–	–	t	t	t	849452	830814	820467
	gap	10	–	–	–	1	1	1	t	t	t	835998	815546	900470
	gap	25	–	–	–	1	1	1	t	t	t	851420	810964	834960
	gap	50	–	–	–	1	1	1	t	t	t	847036	813097	801290
	leaf freq	10	–	–	–	1	1	1	t	t	t	991712	925514	930624
	leaf freq	25	–	–	–	1	1	1	t	t	t	994030	969967	886205
	leaf freq	50	–	–	–	1	1	1	t	t	t	882086	993420	942384
	ssg	10	–	–	–	1	1	1	t	t	t	929497	1107849	862166
	ssg	25	–	–	–	1	1	1	t	t	t	976368	1104258	864841
	ssg	50	–	–	–	1	1	1	t	t	t	924940	1397872	874707
	tree weight	10	–	–	–	1	1	1	t	t	t	875611	783922	1505372
	tree weight	25	–	–	–	1	1	1	t	t	t	876644	779440	1544891
	tree weight	50	–	–	–	1	1	1	t	t	t	842798	783725	1524859
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	862999	839897	820258
	0-restart	–	–	–	–	–	–	–	t	t	t	865812	839021	818032
neos--kasavu	monotone	10	–	–	–	–	–	–	t	t	t	1	1	1
	monotone	25	–	–	–	–	–	–	t	t	t	1	1	1
	monotone	50	–	–	–	–	–	–	t	t	t	1	1	1
	reg forest	10	–	–	–	–	–	–	t	t	t	1	1	1
	reg forest	25	–	–	–	–	–	–	t	t	t	1	1	1
	reg forest	50	–	–	–	–	–	–	t	t	t	1	1	1
	gap	10	–	–	–	–	–	–	t	t	t	1	1	1
	gap	25	–	–	–	–	–	–	t	t	t	1	1	1
	gap	50	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	10	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	25	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	50	–	–	–	–	–	–	t	t	t	1	1	1
	ssg	10	–	–	–	–	–	–	t	t	t	1	1	1
	ssg	25	–	–	–	–	–	–	t	t	t	1	1	1
	ssg	50	–	–	–	–	–	–	t	t	t	1	1	1
	tree weight	10	–	–	–	–	–	–	t	t	t	1	1	1
	tree weight	25	–	–	–	–	–	–	t	t	t	1	1	1
	tree weight	50	–	–	–	–	–	–	t	t	t	1	1	1
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	1	1	1
	0-restart	–	–	–	–	–	–	–	t	t	t	1	1	1
neos--nattai	monotone	10	–	–	–	–	–	1	1371.3	2412.0	1712.1	12547	28132	10732
	monotone	25	–	–	–	–	–	–	1361.0	2419.7	1442.3	12547	28132	12159
	monotone	50	–	–	–	–	–	–	1358.0	2413.4	1446.9	12547	28132	12159
	reg forest	10	–	–	–	1	1	1	1640.4	1754.5	1705.9	10737	16229	10732
	reg forest	25	–	–	–	–	–	–	1369.1	2420.7	1442.6	12547	28132	12159
	reg forest	50	–	–	–	–	–	–	1363.1	2408.3	1442.1	12547	28132	12159
	gap	10	–	–	–	1	1	1	1876.4	2731.5	1955.4	13835	19871	13473
	gap	25	–	–	–	1	1	1	2087.6	2713.1	2074.8	14304	19687	13636
	gap	50	–	–	–	1	1	1	2237.1	2622.1	2077.5	14909	19918	13506
	leaf freq	10	–	–	–	–	–	1	1362.7	3267.1	1445.7	12547	29548	12159
	leaf freq	25	–	–	–	–	–	–	1369.0	2387.9	1438.8	12547	28132	12159
	leaf freq	50	–	–	–	–	–	–	1364.1	2413.9	1433.2	12547	28132	12159
	ssg	10	–	–	–	1	1	1	1605.9	2284.5	2082.0	11545	20463	14367
	ssg	25	–	–	–	1	1	1	1799.5	2248.8	2131.5	13281	19236	13810
	ssg	50	–	–	–	1	1	1	2043.0	2148.0	2106.1	15675	19640	13858
	tree weight	10	–	–	–	1	1	1	1230.1	2539.7	1710.6	10225	25681	10732
	tree weight	25	–	–	–	1	1	1	1224.8	2539.7	1607.3	10225	25681	10830
	tree weight	50	–	–	–	1	1	–	1344.1	2553.3	1446.9	11208	25681	12159
	no clairvoyant	–	–	–	–	–	–	–	1360.8	2399.2	1447.1	12547	28132	12159
	0-restart	–	–	–	–	–	–	–	1365.4	2403.5	1447.5	12547	28132	12159

cont. on next page ...

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
neos-. -niemur	monotone	10	-	-	-	1	1	1	1948.4	1947.3	3493.7	87512	79553	174039
	monotone	25	-	-	-	-	1	1	2916.3	1955.5	3499.6	187896	79553	174039
	monotone	50	-	-	-	-	-	1	2953.6	2319.0	3484.5	187896	144421	174039
	reg forest	10	-	-	-	-	-	-	2967.8	2332.5	2989.5	187896	144421	143497
	reg forest	25	-	-	-	-	-	-	2957.4	2333.5	2985.9	187896	144421	143497
	reg forest	50	-	-	-	-	-	-	2958.4	2338.8	2977.3	187896	144421	143497
	gap	10	-	-	-	1	1	1	2147.8	2149.0	2973.4	148783	105271	101242
	gap	25	-	-	-	1	1	1	1759.8	2139.6	2535.7	92346	105271	79412
	gap	50	-	-	-	1	1	1	1419.2	2137.0	2176.9	56205	105271	63130
	leaf freq	10	-	-	-	1	1	1	1886.3	2121.7	3019.3	72362	93506	97134
	leaf freq	25	-	-	-	1	-	1	1743.8	2322.9	3244.6	66021	144421	182875
	leaf freq	50	-	-	-	-	-	-	2941.9	2337.4	2966.3	187896	144421	143497
	ssg	10	-	-	-	1	1	1	1631.7	2303.4	1499.1	65522	99147	83353
	ssg	25	-	-	-	1	1	1	1623.1	2244.8	2837.2	65522	102676	138780
	ssg	50	-	-	-	1	1	1	1510.5	2526.8	2840.4	53810	121002	138780
	tree weight	10	-	-	-	1	1	1	2336.2	1892.6	1522.9	131242	95199	59280
	tree weight	25	-	-	-	1	1	1	1422.7	1892.3	1536.9	58860	95199	59280
	tree weight	50	-	-	-	1	1	1	1407.5	1958.2	1521.2	48102	87693	59280
	no clairvoyant	-	-	-	-	-	-	-	2942.2	2320.5	2967.0	187896	144421	143497
	0-restart	-	-	-	-	-	-	-	2944.2	2323.3	2965.9	187896	144421	143497
neos-631710	monotone	10	-	-	-	-	-	-	t	t	t	1	1	1
	monotone	25	-	-	-	-	-	-	t	t	t	1	1	1
	monotone	50	-	-	-	-	-	-	t	t	t	1	1	1
	reg forest	10	-	-	-	-	-	-	t	t	t	1	1	1
	reg forest	25	-	-	-	-	-	-	t	t	t	1	1	1
	reg forest	50	-	-	-	-	-	-	t	t	t	1	1	1
	gap	10	-	-	-	-	-	-	t	t	t	1	1	1
	gap	25	-	-	-	-	-	-	t	t	t	1	1	1
	gap	50	-	-	-	-	-	-	t	t	t	1	1	1
	leaf freq	10	-	-	-	-	-	-	t	t	t	1	1	1
	leaf freq	25	-	-	-	-	-	-	t	t	t	1	1	1
	leaf freq	50	-	-	-	-	-	-	t	t	t	1	1	1
	ssg	10	-	-	-	-	-	-	t	t	t	1	1	1
	ssg	25	-	-	-	-	-	-	t	t	t	1	1	1
	ssg	50	-	-	-	-	-	-	t	t	t	1	1	1
	tree weight	10	-	-	-	-	-	-	t	t	t	1	1	1
	tree weight	25	-	-	-	-	-	-	t	t	t	1	1	1
	tree weight	50	-	-	-	-	-	-	t	t	t	1	1	1
	no clairvoyant	-	-	-	-	-	-	-	t	t	t	1	1	1
	0-restart	-	-	-	-	-	-	-	t	t	t	1	1	1
neos-662469	monotone	10	-	-	-	1	1	1	t	t	t	28757	25768	24735
	monotone	25	-	-	-	1	1	1	t	t	t	28904	25629	24735
	monotone	50	-	-	-	1	1	1	t	t	t	28904	25822	24735
	reg forest	10	-	-	-	-	1	1	t	t	t	12493	25689	24735
	reg forest	25	-	-	-	-	-	-	t	t	t	12532	16225	21233
	reg forest	50	-	-	-	-	-	-	t	t	t	12610	16088	21295
	gap	10	-	-	-	-	1	1	t	t	t	12610	25688	24735
	gap	25	-	-	-	-	1	1	t	t	t	12505	25736	24735
	gap	50	-	-	-	-	1	1	t	t	t	12572	25662	24735
	leaf freq	10	-	-	-	-	1	1	t	t	t	12490	25698	24735
	leaf freq	25	-	-	-	-	1	1	t	t	t	12520	25768	24735
	leaf freq	50	-	-	-	-	1	1	t	t	t	12468	25787	24735
	ssg	10	-	-	-	1	1	1	t	t	t	28756	25744	24735
	ssg	25	-	-	-	1	1	1	t	t	t	30354	25647	24735
	ssg	50	-	-	-	1	1	1	t	t	t	30078	25750	24735
	tree weight	10	-	-	-	1	1	1	t	t	t	11891	25825	24735
	tree weight	25	-	-	-	1	1	1	t	t	t	23173	25810	24735
	tree weight	50	-	-	-	1	1	1	t	t	t	31567	25821	24735
	no clairvoyant	-	-	-	-	-	-	-	t	t	t	12520	16208	21740
	0-restart	-	-	-	-	-	-	-	t	t	t	12520	16268	21412
neos-787933	monotone	10	-	-	-	-	-	-	1.9	1.9	1.9	1	1	1
	monotone	25	-	-	-	-	-	-	2.0	2.0	1.9	1	1	1
	monotone	50	-	-	-	-	-	-	1.9	1.9	1.9	1	1	1
	reg forest	10	-	-	-	-	-	-	2.0	1.9	1.9	1	1	1
	reg forest	25	-	-	-	-	-	-	1.9	1.9	1.9	1	1	1
	reg forest	50	-	-	-	-	-	-	2.0	2.0	1.9	1	1	1
	gap	10	-	-	-	-	-	-	1.9	2.0	1.9	1	1	1
	gap	25	-	-	-	-	-	-	1.9	1.9	1.9	1	1	1
	gap	50	-	-	-	-	-	-	1.9	1.9	1.9	1	1	1
	leaf freq	10	-	-	-	-	-	-	1.9	1.9	1.9	1	1	1
	leaf freq	25	-	-	-	-	-	-	1.9	1.9	1.9	1	1	1
	leaf freq	50	-	-	-	-	-	-	1.9	1.9	1.9	1	1	1
	ssg	10	-	-	-	-	-	-	1.9	1.9	1.9	1	1	1
	ssg	25	-	-	-	-	-	-	1.9	1.9	1.9	1	1	1
	ssg	50	-	-	-	-	-	-	1.9	1.9	1.9	1	1	1
	tree weight	10	-	-	-	-	-	-	1.9	1.9	1.9	1	1	1

cont. on next page ...

E. Appendix E

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
neos-827175	tree weight	25	–	–	–	–	–	–	1.9	1.9	1.9	1	1	1
	tree weight	50	–	–	–	–	–	–	1.9	1.9	1.9	1	1	1
	no clairvoyant	–	–	–	–	–	–	–	1.9	1.9	1.9	1	1	1
	0-restart	–	–	–	–	–	–	–	1.9	1.9	1.9	1	1	1
	monotone	10	–	–	–	–	–	–	7.3	7.4	7.3	1	1	1
	monotone	25	–	–	–	–	–	–	7.3	7.4	7.3	1	1	1
	monotone	50	–	–	–	–	–	–	7.3	7.4	7.3	1	1	1
	reg forest	10	–	–	–	–	–	–	7.4	7.4	7.4	1	1	1
	reg forest	25	–	–	–	–	–	–	7.4	7.4	7.3	1	1	1
	reg forest	50	–	–	–	–	–	–	7.4	7.3	7.3	1	1	1
	gap	10	–	–	–	–	–	–	7.4	7.3	7.3	1	1	1
	gap	25	–	–	–	–	–	–	7.4	7.3	7.2	1	1	1
	gap	50	–	–	–	–	–	–	7.4	7.3	7.2	1	1	1
	leaf freq	10	–	–	–	–	–	–	7.3	7.3	7.3	1	1	1
	leaf freq	25	–	–	–	–	–	–	7.5	7.3	7.4	1	1	1
	leaf freq	50	–	–	–	–	–	–	7.4	7.3	7.4	1	1	1
	ssg	10	–	–	–	–	–	–	7.4	7.3	7.2	1	1	1
	ssg	25	–	–	–	–	–	–	7.4	7.3	7.3	1	1	1
	ssg	50	–	–	–	–	–	–	7.4	7.4	7.4	1	1	1
	tree weight	10	–	–	–	–	–	–	7.4	7.3	7.3	1	1	1
neos-848589	tree weight	25	–	–	–	–	–	–	7.3	7.3	7.2	1	1	1
	tree weight	50	–	–	–	–	–	–	7.3	7.3	7.3	1	1	1
	no clairvoyant	–	–	–	–	–	–	–	7.5	7.3	7.3	1	1	1
	0-restart	–	–	–	–	–	–	–	7.3	7.4	7.3	1	1	1
	monotone	10	–	–	–	–	–	2111.5	t	t	t	70	145	133
	monotone	25	–	–	–	–	–	2114.6	t	t	t	70	144	133
	monotone	50	–	–	–	–	–	2127.7	t	t	t	70	150	133
	reg forest	10	–	–	–	–	–	2103.7	t	t	t	70	145	133
	reg forest	25	–	–	–	–	–	2132.6	t	t	t	70	146	133
	reg forest	50	–	–	–	–	–	2132.3	t	t	t	70	144	133
	gap	10	–	–	–	–	–	2131.2	t	t	t	70	145	133
	gap	25	–	–	–	–	–	2119.8	t	t	t	70	146	133
	gap	50	–	–	–	–	–	2117.8	t	t	t	70	145	133
	leaf freq	10	–	–	–	–	–	2132.8	t	t	t	70	145	133
	leaf freq	25	–	–	–	–	–	2109.3	t	t	t	70	145	133
	leaf freq	50	–	–	–	–	–	2136.6	t	t	t	70	144	133
	ssg	10	–	–	–	–	–	2123.1	t	t	t	70	145	133
	ssg	25	–	–	–	–	–	2140.2	t	t	t	70	145	133
	ssg	50	–	–	–	–	–	2117.9	t	t	t	70	145	133
	tree weight	10	–	–	–	–	–	2131.3	t	t	t	70	145	133
neos-860300	tree weight	25	–	–	–	–	–	2126.5	t	t	t	70	150	133
	tree weight	50	–	–	–	–	–	2121.9	t	t	t	70	145	133
	no clairvoyant	–	–	–	–	–	–	2113.9	t	t	t	70	145	133
	0-restart	–	–	–	–	–	–	2124.1	t	t	t	70	144	133
	monotone	10	1	1	1	–	–	19.6	19.5	16.9	2	2	2	2
	monotone	25	1	1	1	–	–	19.7	19.3	16.9	2	2	2	2
	monotone	50	1	1	1	–	–	19.7	19.3	16.9	2	2	2	2
	reg forest	10	1	1	1	–	–	19.7	19.5	17.1	2	2	2	2
	reg forest	25	1	1	1	–	–	19.9	19.3	16.9	2	2	2	2
	reg forest	50	1	1	1	–	–	19.7	19.6	16.9	2	2	2	2
	gap	10	1	1	1	–	–	19.7	19.4	16.6	2	2	2	2
	gap	25	1	1	1	–	–	19.5	19.5	16.8	2	2	2	2
	gap	50	1	1	1	–	–	19.6	19.2	16.8	2	2	2	2
	leaf freq	10	1	1	1	–	–	19.7	19.2	16.8	2	2	2	2
	leaf freq	25	1	1	1	–	–	19.9	19.5	16.9	2	2	2	2
	leaf freq	50	1	1	1	–	–	19.6	19.3	17.0	2	2	2	2
	ssg	10	1	1	1	–	–	19.5	19.6	17.0	2	2	2	2
	ssg	25	1	1	1	–	–	19.6	19.2	16.8	2	2	2	2
	ssg	50	1	1	1	–	–	19.7	19.5	16.8	2	2	2	2
	tree weight	10	1	1	1	–	–	19.6	19.3	16.8	2	2	2	2
neos-873061	tree weight	25	1	1	1	–	–	19.6	19.5	16.9	2	2	2	2
	tree weight	50	1	1	1	–	–	19.7	19.3	16.8	2	2	2	2
	no clairvoyant	–	1	1	1	–	–	19.6	19.1	16.9	2	2	2	2
	0-restart	–	–	–	–	–	–	17.0	15.7	15.6	1	1	1	1
	monotone	10	1	1	1	–	–	t	t	t	2	2	2	2
	monotone	25	1	1	1	–	–	t	t	t	2	2	2	2
	monotone	50	1	1	1	–	–	t	t	t	2	2	2	2
	reg forest	10	1	1	1	–	–	t	t	t	2	2	2	2
	reg forest	25	1	1	1	–	–	t	t	t	2	2	2	2
	reg forest	50	1	1	1	–	–	t	t	t	2	2	2	2
	gap	10	1	1	1	–	–	t	t	t	2	2	2	2
	gap	25	1	1	1	–	–	t	t	t	2	2	2	2
	gap	50	1	1	1	–	–	t	t	t	2	2	2	2
	leaf freq	10	1	1	1	–	–	t	t	t	2	2	2	2
	leaf freq	25	1	1	1	–	–	t	t	t	2	2	2	2
	leaf freq	50	1	1	1	–	–	t	t	t	2	2	2	2

cont. on next page ...

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
neos-911970	ssg	10	1	1	1	-	-	-	t	t	t	2	2	2
	ssg	25	1	1	1	-	-	-	t	t	t	2	2	2
	ssg	50	1	1	1	-	-	-	t	t	t	2	2	2
	tree weight	10	1	1	1	-	-	-	t	t	t	2	2	2
	tree weight	25	1	1	1	-	-	-	t	t	t	2	2	2
	tree weight	50	1	1	1	-	-	-	t	t	t	2	2	2
	no clairvoyant	-	1	1	1	-	-	-	t	t	t	2	2	2
	0-restart	-	-	-	-	-	-	-	t	t	t	5	63	2
	monotone	10	1	1	2	-	-	1	t	t	t	5498861	550179	5188077
	monotone	25	1	1	2	-	-	1	t	t	t	5513981	550164	5185413
	monotone	50	1	1	1	-	-	-	t	t	t	5519600	550169	5090979
	reg forest	10	1	1	2	-	-	1	t	t	t	5358537	550146	5165139
	reg forest	25	1	1	1	-	-	-	t	t	t	5346057	550157	4961827
	reg forest	50	1	1	1	-	-	-	t	t	t	5378107	550159	4982958
	gap	10	1	1	1	-	1	-	t	t	t	5521586	2001571	5119722
	gap	25	1	1	1	-	-	-	t	t	t	5513188	550163	5124296
	gap	50	1	1	1	-	-	-	t	t	t	5518630	550161	5178187
	leaf freq	10	4	1	2	1	1	1	408.4	1869.4	t	271968	1548790	5450851
	leaf freq	25	3	1	2	1	1	1	t	1950.0	437.2	2621052	1359537	412500
	leaf freq	50	2	1	2	1	-	1	t	t	t	5807794	550142	4234660
	ssg	10	1	1	2	1	1	1	t	t	t	5624978	830054	5045872
	ssg	25	2	1	2	1	1	1	t	2256.8	t	3620152	199307	3897312
	ssg	50	1	1	2	1	1	1	t	139.3	t	4754471	38958	4594140
	tree weight	10	1	1	1	1	1	1	t	t	t	5171893	568052	4922071
	tree weight	25	3	1	2	1	1	1	t	t	t	3011830	361186	2756209
	tree weight	50	1	1	1	-	1	1	t	t	t	5501323	2939473	687074
	no clairvoyant	-	1	1	1	-	-	-	t	t	t	5510641	550158	5102108
	0-restart	-	-	-	-	-	-	-	t	t	t	575063	2690636	3610813
neos-933966	monotone	10	-	-	-	-	-	-	2253.0	t	3498.6	279	528	438
	monotone	25	-	-	-	-	-	-	2247.2	t	3500.3	279	526	438
	monotone	50	-	-	-	-	-	-	2253.0	t	3508.1	279	528	438
	reg forest	10	-	-	-	-	-	-	2240.8	t	3507.6	269	527	438
	reg forest	25	-	-	-	-	-	-	2236.1	t	3473.0	269	527	438
	reg forest	50	-	-	-	-	-	-	2226.0	t	3500.0	269	527	438
	gap	10	-	-	-	-	-	-	2231.2	t	3483.9	269	524	438
	gap	25	-	-	-	-	-	-	2225.4	t	3501.0	269	527	438
	gap	50	-	-	-	-	-	-	2229.7	t	3487.7	269	527	438
	leaf freq	10	-	-	-	-	-	-	2227.9	t	3500.3	269	527	438
	leaf freq	25	-	-	-	-	-	-	2213.0	t	3512.3	269	527	438
	leaf freq	50	-	-	-	-	-	-	2225.6	t	3501.3	269	527	438
	ssg	10	-	-	-	-	-	-	2224.5	t	3493.2	269	528	438
	ssg	25	-	-	-	-	-	-	2238.0	t	3494.9	269	528	438
	ssg	50	-	-	-	-	-	-	2232.6	t	3497.0	269	528	438
	tree weight	10	-	-	-	-	-	-	2231.0	t	3501.0	269	527	438
	tree weight	25	-	-	-	-	-	-	2228.7	t	3507.4	269	528	438
	tree weight	50	-	-	-	-	-	-	2226.8	t	3481.2	269	528	438
	no clairvoyant	-	-	-	-	-	-	-	2214.2	t	3493.7	269	527	438
	0-restart	-	-	-	-	-	-	-	2229.4	t	3492.5	269	527	438
neos-950242	monotone	10	-	-	-	-	-	-	153.3	315.4	t	94	224	4648
	monotone	25	-	-	-	-	-	-	153.1	317.1	t	94	224	4648
	monotone	50	-	-	-	-	-	-	153.0	318.5	t	94	224	4672
	reg forest	10	-	-	-	-	-	-	152.1	319.1	t	94	224	4677
	reg forest	25	-	-	-	-	-	-	153.2	317.8	t	94	224	4636
	reg forest	50	-	-	-	-	-	-	153.0	317.2	t	94	224	4639
	gap	10	-	-	-	-	-	-	153.0	319.2	t	94	224	4690
	gap	25	-	-	-	-	-	-	153.2	317.2	t	94	224	4701
	gap	50	-	-	-	-	-	-	153.2	318.4	t	94	224	4688
	leaf freq	10	-	-	-	-	-	-	152.6	317.1	t	94	224	4673
	leaf freq	25	-	-	-	-	-	-	153.1	319.4	t	94	224	4693
	leaf freq	50	-	-	-	-	-	-	153.3	316.1	t	94	224	4648
	ssg	10	-	-	-	-	-	-	153.4	319.4	t	94	224	4677
	ssg	25	-	-	-	-	-	-	152.9	316.9	t	94	224	4677
	ssg	50	-	-	-	-	-	-	153.6	316.7	t	94	224	4636
	tree weight	10	-	-	-	-	-	-	153.3	316.6	t	94	224	4643
	tree weight	25	-	-	-	-	-	-	152.8	317.8	t	94	224	4656
	tree weight	50	-	-	-	-	-	-	152.7	316.8	t	94	224	4691
	no clairvoyant	-	-	-	-	-	-	-	153.2	314.1	t	94	224	4672
	0-restart	-	-	-	-	-	-	-	153.6	316.7	t	94	224	4675
neos-957323	monotone	10	-	-	-	-	-	-	193.6	105.7	421.9	48	7	144
	monotone	25	-	-	-	-	-	-	194.9	106.6	419.7	48	7	144
	monotone	50	-	-	-	-	-	-	194.2	106.3	418.8	48	7	144
	reg forest	10	-	-	-	-	-	-	194.4	106.7	421.6	48	7	144
	reg forest	25	-	-	-	-	-	-	194.6	105.4	422.0	48	7	144
	reg forest	50	-	-	-	-	-	-	195.0	106.4	422.5	48	7	144
	gap	10	-	-	-	-	-	-	195.3	106.4	419.0	48	7	144
	gap	25	-	-	-	-	-	-	194.0	105.9	421.1	48	7	144

cont. on next page ...

E. Appendix E

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
neos-960392	gap	50	–	–	–	–	–	–	195.3	106.3	420.9	48	7	144
	leaf freq	10	–	–	–	–	–	–	193.8	105.9	422.4	48	7	144
	leaf freq	25	–	–	–	–	–	–	194.5	106.4	420.1	48	7	144
	leaf freq	50	–	–	–	–	–	–	194.1	105.6	421.2	48	7	144
	ssg	10	–	–	–	–	–	–	193.9	105.7	421.0	48	7	144
	ssg	25	–	–	–	–	–	–	194.8	106.4	421.5	48	7	144
	ssg	50	–	–	–	–	–	–	195.1	106.3	423.3	48	7	144
	tree weight	10	–	–	–	–	–	–	194.5	106.3	421.0	48	7	144
	tree weight	25	–	–	–	–	–	–	192.8	106.7	420.1	48	7	144
	tree weight	50	–	–	–	–	–	–	192.8	106.1	419.8	48	7	144
	no clairvoyant	–	–	–	–	–	–	–	195.4	105.8	421.4	48	7	144
	0-restart	–	–	–	–	–	–	–	194.2	106.3	420.8	48	7	144
	monotone	10	–	–	–	–	–	–	623.8	1770.8	289.8	18	176	1
	monotone	25	–	–	–	–	–	–	618.5	1766.8	291.0	18	176	1
	monotone	50	–	–	–	–	–	–	619.3	1764.2	289.2	18	176	1
	reg forest	10	–	–	–	–	–	–	621.5	1768.6	291.0	18	176	1
	reg forest	25	–	–	–	–	–	–	623.6	1773.5	288.8	18	176	1
	reg forest	50	–	–	–	–	–	–	622.8	1780.8	289.4	18	176	1
	gap	10	–	–	–	–	–	–	620.3	1763.6	288.8	18	176	1
	gap	25	–	–	–	–	–	–	622.0	1761.9	289.5	18	176	1
	gap	50	–	–	–	–	–	–	623.5	1773.3	291.4	18	176	1
	leaf freq	10	–	–	–	–	–	–	621.6	1769.2	290.2	18	176	1
	leaf freq	25	–	–	–	–	–	–	620.3	1760.5	289.7	18	176	1
	leaf freq	50	–	–	–	–	–	–	622.1	1762.1	289.6	18	176	1
	ssg	10	–	–	–	–	–	–	622.8	1762.8	289.3	18	176	1
	ssg	25	–	–	–	–	–	–	621.7	1752.3	289.1	18	176	1
	ssg	50	–	–	–	–	–	–	621.8	1764.7	289.2	18	176	1
	tree weight	10	–	–	–	–	–	–	622.9	1774.2	289.9	18	176	1
	tree weight	25	–	–	–	–	–	–	621.5	1768.1	289.6	18	176	1
	tree weight	50	–	–	–	–	–	–	621.2	1771.5	289.6	18	176	1
	no clairvoyant	–	–	–	–	–	–	–	620.3	1776.7	289.5	18	176	1
	0-restart	–	–	–	–	–	–	–	622.6	1763.4	290.9	18	176	1
neos17	monotone	10	–	–	–	–	–	–	13.4	7.3	8.2	8971	3490	4150
	monotone	25	–	–	–	–	–	–	13.5	7.4	8.4	8971	3490	4150
	monotone	50	–	–	–	–	–	–	13.5	7.4	8.3	8971	3490	4150
	reg forest	10	–	–	–	–	–	–	14.5	8.0	9.1	8971	3490	4150
	reg forest	25	–	–	–	–	–	–	14.3	8.1	9.1	8971	3490	4150
	reg forest	50	–	–	–	–	–	–	14.4	8.0	9.2	8971	3490	4150
	gap	10	–	–	–	–	–	–	13.4	7.4	8.2	8971	3490	4150
	gap	25	–	–	–	–	–	–	13.4	7.4	8.3	8971	3490	4150
	gap	50	–	–	–	–	–	–	13.4	7.3	8.4	8971	3490	4150
	leaf freq	10	–	–	–	–	–	–	13.4	7.3	8.4	8971	3490	4150
	leaf freq	25	–	–	–	–	–	–	13.3	7.4	8.3	8971	3490	4150
	leaf freq	50	–	–	–	–	–	–	13.4	7.4	8.2	8971	3490	4150
	ssg	10	–	–	–	–	–	–	13.4	7.4	8.3	8971	3490	4150
	ssg	25	–	–	–	–	–	–	13.5	7.4	8.4	8971	3490	4150
	ssg	50	–	–	–	–	–	–	13.4	7.2	8.4	8971	3490	4150
	tree weight	10	–	–	–	–	–	–	13.3	7.4	8.3	8971	3490	4150
	tree weight	25	–	–	–	–	–	–	13.3	7.4	8.3	8971	3490	4150
	tree weight	50	–	–	–	–	–	–	13.4	7.3	8.3	8971	3490	4150
	no clairvoyant	–	–	–	–	–	–	–	14.2	7.4	8.3	8971	3490	4150
	0-restart	–	–	–	–	–	–	–	13.2	7.4	8.4	8971	3490	4150
neos5	monotone	10	–	–	–	–	–	–	181.7	119.4	185.0	428990	284757	430282
	monotone	25	–	–	–	–	–	–	183.9	120.0	185.8	428990	284757	430282
	monotone	50	–	–	–	–	–	–	183.0	119.8	185.1	428990	284757	430282
	reg forest	10	–	–	–	–	–	–	187.2	122.9	188.4	428990	284757	430282
	reg forest	25	–	–	–	–	–	–	185.1	123.0	188.3	428990	284757	430282
	reg forest	50	–	–	–	–	–	–	186.0	123.0	189.4	428990	284757	430282
	gap	10	–	–	–	–	–	–	182.0	120.4	183.6	428990	284757	430282
	gap	25	–	–	–	–	–	–	181.8	120.3	185.0	428990	284757	430282
	gap	50	–	–	–	–	–	–	182.5	120.1	184.3	428990	284757	430282
	leaf freq	10	–	–	–	–	–	–	183.1	120.5	185.3	428990	284757	430282
	leaf freq	25	–	–	–	–	–	–	183.0	119.7	186.2	428990	284757	430282
	leaf freq	50	–	–	–	–	–	–	181.8	120.4	185.2	428990	284757	430282
	ssg	10	–	–	–	–	–	–	183.0	120.0	184.7	428990	284757	430282
	ssg	25	–	–	–	–	–	–	182.6	119.7	184.4	428990	284757	430282
	ssg	50	–	–	–	–	–	–	182.8	119.8	184.9	428990	284757	430282
	tree weight	10	–	–	–	–	–	–	182.3	120.4	184.8	428990	284757	430282
	tree weight	25	–	–	–	–	–	–	182.9	120.0	184.9	428990	284757	430282
	tree weight	50	–	–	–	–	–	–	182.5	120.0	185.2	428990	284757	430282
	no clairvoyant	–	–	–	–	–	–	–	184.5	119.6	185.1	428990	284757	430282
	0-restart	–	–	–	–	–	–	–	183.2	120.6	185.0	428990	284757	430282
neos8	monotone	10	–	–	–	–	–	–	7.4	5.2	5.4	1	1	1
	monotone	25	–	–	–	–	–	–	7.3	5.2	5.5	1	1	1
	monotone	50	–	–	–	–	–	–	7.3	5.2	5.4	1	1	1
	reg forest	10	–	–	–	–	–	–	7.3	5.2	5.4	1	1	1

cont. on next page ...

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
neos859080	reg forest	25	–	–	–	–	–	–	7.4	5.2	5.3	1	1	1
	reg forest	50	–	–	–	–	–	–	7.4	5.2	5.5	1	1	1
	gap	10	–	–	–	–	–	–	7.2	5.2	5.4	1	1	1
	gap	25	–	–	–	–	–	–	7.3	5.2	5.4	1	1	1
	gap	50	–	–	–	–	–	–	7.3	5.2	5.5	1	1	1
	leaf freq	10	–	–	–	–	–	–	7.4	5.2	5.4	1	1	1
	leaf freq	25	–	–	–	–	–	–	7.4	5.2	5.5	1	1	1
	leaf freq	50	–	–	–	–	–	–	7.5	5.2	5.4	1	1	1
	ssg	10	–	–	–	–	–	–	7.5	5.2	5.4	1	1	1
	ssg	25	–	–	–	–	–	–	7.4	5.2	5.4	1	1	1
	ssg	50	–	–	–	–	–	–	7.3	5.2	5.3	1	1	1
	tree weight	10	–	–	–	–	–	–	7.3	5.1	5.4	1	1	1
	tree weight	25	–	–	–	–	–	–	7.3	5.1	5.3	1	1	1
	tree weight	50	–	–	–	–	–	–	7.3	5.2	5.4	1	1	1
	no clairvoyant	–	–	–	–	–	–	–	7.3	5.2	5.3	1	1	1
	0-restart	–	–	–	–	–	–	–	7.1	5.2	5.3	1	1	1
	monotone	10	1	–	–	1	–	1	2.2	2.0	4.3	1046	703	2059
	monotone	25	1	–	–	1	–	1	2.2	2.0	4.3	1046	703	2059
	monotone	50	1	–	–	1	–	1	2.2	1.9	4.3	1046	703	2059
	reg forest	10	1	–	–	1	–	1	5.3	3.8	8.0	1208	703	2059
	reg forest	25	–	–	–	–	–	–	5.7	3.7	10.1	1551	703	6773
	reg forest	50	–	–	–	–	–	–	5.6	3.7	10.1	1551	703	6773
	gap	10	–	–	–	–	–	–	2.5	2.1	6.5	1551	703	6773
	gap	25	–	–	–	–	–	–	2.4	1.9	6.2	1551	703	6773
	gap	50	–	–	–	–	–	–	2.6	1.9	6.3	1551	703	6773
	leaf freq	10	–	–	1	–	–	1	2.5	2.0	4.6	1551	703	2846
	leaf freq	25	–	–	–	–	–	1	2.6	2.0	4.7	1551	703	3274
	leaf freq	50	–	–	–	–	–	–	2.5	1.9	6.4	1551	703	6773
	ssg	10	–	–	–	–	–	–	2.5	2.0	6.3	1551	703	6773
	ssg	25	–	–	–	–	–	–	2.5	2.0	6.4	1551	703	6773
	ssg	50	–	–	–	–	–	–	2.6	2.0	6.4	1551	703	6773
	tree weight	10	–	–	–	–	–	1	2.5	1.9	4.3	1551	703	2059
	tree weight	25	–	–	–	–	–	1	2.6	2.0	4.3	1551	703	2059
	tree weight	50	–	–	–	–	–	1	2.5	1.9	4.6	1551	703	1806
	no clairvoyant	–	–	–	–	–	–	–	5.6	1.9	6.4	1551	703	6773
	0-restart	–	–	–	–	–	–	–	2.5	2.0	6.4	1551	703	6773
net12	monotone	10	–	–	–	–	–	–	365.9	688.2	659.0	1615	1779	2393
	monotone	25	–	–	–	–	–	–	364.6	688.8	657.5	1615	1779	2393
	monotone	50	–	–	–	–	–	–	365.3	689.9	662.2	1615	1779	2393
	reg forest	10	–	–	–	–	–	–	365.5	690.5	658.7	1615	1779	2393
	reg forest	25	–	–	–	–	–	–	365.3	688.9	659.4	1615	1779	2393
	reg forest	50	–	–	–	–	–	–	367.4	684.4	657.9	1615	1779	2393
	gap	10	–	–	–	–	1	1	365.9	583.9	411.4	1615	1653	1985
	gap	25	–	–	–	–	–	1	365.6	690.3	364.8	1615	1779	1765
	gap	50	–	–	–	–	–	–	366.1	688.6	656.9	1615	1779	2393
	leaf freq	10	–	–	–	–	–	–	362.9	687.8	657.6	1615	1779	2393
	leaf freq	25	–	–	–	–	–	–	365.9	689.9	655.6	1615	1779	2393
	leaf freq	50	–	–	–	–	–	–	365.0	690.8	656.6	1615	1779	2393
	ssg	10	–	–	–	1	–	–	442.3	686.8	656.6	1962	1779	2393
	ssg	25	–	–	–	–	–	–	364.5	688.5	657.3	1615	1779	2393
	ssg	50	–	–	–	–	–	–	365.1	689.9	660.0	1615	1779	2393
	tree weight	10	–	–	–	–	–	1	365.6	684.4	504.4	1615	1779	2134
	tree weight	25	–	–	–	–	–	–	362.3	690.3	655.8	1615	1779	2393
	tree weight	50	–	–	–	–	–	–	365.6	689.4	657.5	1615	1779	2393
	no clairvoyant	–	–	–	–	–	–	–	365.6	689.4	656.4	1615	1779	2393
	0-restart	–	–	–	–	–	–	–	367.1	693.2	657.3	1615	1779	2393
netdiversion	monotone	10	–	–	–	–	–	–	639.0	493.1	978.6	23	5	37
	monotone	25	–	–	–	–	–	–	641.2	497.6	973.8	23	5	37
	monotone	50	–	–	–	–	–	–	632.2	496.1	975.1	23	5	37
	reg forest	10	–	–	–	–	–	–	637.8	498.3	983.1	23	5	37
	reg forest	25	–	–	–	–	–	–	638.0	491.0	973.0	23	5	37
	reg forest	50	–	–	–	–	–	–	639.3	497.0	974.0	23	5	37
	gap	10	–	–	–	–	–	–	644.5	497.3	977.5	23	5	37
	gap	25	–	–	–	–	–	–	639.4	496.1	980.5	23	5	37
	gap	50	–	–	–	–	–	–	644.8	497.4	969.8	23	5	37
	leaf freq	10	–	–	–	–	–	–	638.3	494.8	975.9	23	5	37
	leaf freq	25	–	–	–	–	–	–	641.3	499.8	979.6	23	5	37
	leaf freq	50	–	–	–	–	–	–	638.6	497.2	976.7	23	5	37
	ssg	10	–	–	–	–	–	–	634.7	496.8	976.3	23	5	37
	ssg	25	–	–	–	–	–	–	643.3	495.7	974.6	23	5	37
	ssg	50	–	–	–	–	–	–	638.8	496.9	978.7	23	5	37
	tree weight	10	–	–	–	–	–	–	637.5	496.6	976.2	23	5	37
	tree weight	25	–	–	–	–	–	–	640.1	493.8	973.1	23	5	37
	tree weight	50	–	–	–	–	–	–	646.1	500.2	973.5	23	5	37
	no clairvoyant	–	–	–	–	–	–	–	642.4	496.0	981.1	23	5	37
	0-restart	–	–	–	–	–	–	–	642.0	487.8	977.3	23	5	37

cont. on next page ...

E. Appendix E

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
nexp-150-20-8-5	monotone	10	–	1	–	–	–	–	2661.0	596.2	t	3605	2	4656
	monotone	25	–	1	–	–	–	–	2690.5	599.1	t	3605	2	4677
	monotone	50	–	1	–	–	–	–	2695.0	599.9	t	3605	2	4708
	reg forest	10	–	1	6	–	–	1	2663.6	600.0	1942.3	3605	2	1118
	reg forest	25	–	1	–	–	–	–	2676.0	602.0	t	3605	2	4656
	reg forest	50	–	1	–	–	–	–	2678.6	599.9	t	3605	2	4681
	gap	10	–	1	–	–	–	–	2688.1	600.5	t	3605	2	4688
	gap	25	–	1	–	–	–	–	2680.6	597.7	t	3605	2	4714
	gap	50	–	1	–	–	–	–	2683.0	599.9	t	3605	2	4668
	leaf freq	10	–	1	–	–	–	–	2674.3	598.8	t	3605	2	4654
	leaf freq	25	–	1	–	–	–	–	2685.1	602.6	t	3605	2	4674
	leaf freq	50	–	1	–	–	–	–	2688.7	600.3	t	3605	2	4714
	ssg	10	–	1	6	1	–	1	2200.3	597.1	1930.4	2133	2	1118
	ssg	25	–	1	6	1	–	1	2594.5	599.8	1934.2	3178	2	1118
	ssg	50	–	1	–	1	–	1	2692.7	598.8	1920.5	3484	2	988
	tree weight	10	–	1	3	1	–	1	2280.7	599.5	2016.0	2393	2	1052
	tree weight	25	–	1	–	1	–	1	2689.8	599.7	2266.7	3497	2	1299
	tree weight	50	–	1	–	–	–	1	2662.1	599.4	2268.2	3605	2	1339
	no clairvoyant	–	–	1	–	–	–	–	2681.2	596.2	t	3605	2	4684
	0-restart	–	–	–	–	–	–	–	2699.8	3036.8	t	3605	3158	4593
ns1116954	monotone	10	–	–	–	–	–	–	t	t	t	6	1	7
	monotone	25	–	–	–	–	–	–	t	t	t	6	1	7
	monotone	50	–	–	–	–	–	–	t	t	t	6	1	7
	reg forest	10	–	–	–	–	–	–	t	t	t	6	1	5
	reg forest	25	–	–	–	–	–	–	t	t	t	6	1	7
	reg forest	50	–	–	–	–	–	–	t	t	t	6	1	7
	gap	10	–	–	–	–	–	–	t	t	t	6	1	7
	gap	25	–	–	–	–	–	–	t	t	t	6	1	7
	gap	50	–	–	–	–	–	–	t	t	t	6	1	7
	leaf freq	10	–	–	–	–	–	–	t	t	t	6	1	7
	leaf freq	25	–	–	–	–	–	–	t	t	t	6	1	7
	leaf freq	50	–	–	–	–	–	–	t	t	t	6	1	7
	ssg	10	–	–	–	–	–	–	t	t	t	6	1	7
	ssg	25	–	–	–	–	–	–	t	t	t	6	1	7
	ssg	50	–	–	–	–	–	–	t	t	t	6	1	7
	tree weight	10	–	–	–	–	–	–	t	t	t	6	1	7
	tree weight	25	–	–	–	–	–	–	t	t	t	6	1	7
	tree weight	50	–	–	–	–	–	–	t	t	t	6	1	7
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	6	1	7
	0-restart	–	–	–	–	–	–	–	t	t	t	6	1	7
ns1208400	monotone	10	–	–	–	–	–	–	1819.2	3204.3	1373.9	6826	12565	4139
	monotone	25	–	–	–	–	–	–	1822.9	3209.7	1374.0	6826	12565	4139
	monotone	50	–	–	–	–	–	–	1818.2	3215.7	1377.3	6826	12565	4139
	reg forest	10	–	–	–	–	–	–	1824.7	3211.8	1372.7	6826	12565	4139
	reg forest	25	–	–	–	–	–	–	1822.9	3208.4	1373.2	6826	12565	4139
	reg forest	50	–	–	–	–	–	–	1827.1	3230.4	1373.9	6826	12565	4139
	gap	10	–	–	–	–	–	–	1821.6	3217.6	1373.4	6826	12565	4139
	gap	25	–	–	–	–	–	–	1825.3	3214.3	1372.5	6826	12565	4139
	gap	50	–	–	–	–	–	–	1820.7	3218.4	1370.8	6826	12565	4139
	leaf freq	10	–	–	–	–	–	–	1820.6	3224.0	1371.0	6826	12565	4139
	leaf freq	25	–	–	–	–	–	–	1818.1	3223.9	1371.2	6826	12565	4139
	leaf freq	50	–	–	–	–	–	–	1825.7	3217.6	1369.0	6826	12565	4139
	ssg	10	–	–	–	–	–	–	1823.9	3225.6	1377.1	6826	12565	4139
	ssg	25	–	–	–	–	–	–	1817.0	3224.2	1377.6	6826	12565	4139
	ssg	50	–	–	–	–	–	–	1821.1	3225.7	1370.2	6826	12565	4139
	tree weight	10	–	–	–	–	–	–	1821.8	3219.9	1367.0	6826	12565	4139
	tree weight	25	–	–	–	–	–	–	1817.0	3220.2	1370.7	6826	12565	4139
	tree weight	50	–	–	–	–	–	–	1818.1	3217.5	1374.0	6826	12565	4139
	no clairvoyant	–	–	–	–	–	–	–	1825.3	3212.4	1370.0	6826	12565	4139
	0-restart	–	–	–	–	–	–	–	1823.5	3213.9	1369.2	6826	12565	4139
ns1644855	monotone	10	–	–	–	–	–	–	t	464.8	t	44	1	23
	monotone	25	–	–	–	–	–	–	t	463.9	t	44	1	24
	monotone	50	–	–	–	–	–	–	t	470.9	t	44	1	23
	reg forest	10	–	–	–	–	–	–	t	463.7	t	44	1	23
	reg forest	25	–	–	–	–	–	–	t	459.6	t	44	1	24
	reg forest	50	–	–	–	–	–	–	t	455.8	t	44	1	23
	gap	10	–	–	–	–	–	–	t	458.2	t	44	1	24
	gap	25	–	–	–	–	–	–	t	463.2	t	44	1	24
	gap	50	–	–	–	–	–	–	t	457.9	t	44	1	24
	leaf freq	10	–	–	–	–	–	–	t	460.8	t	44	1	24
	leaf freq	25	–	–	–	–	–	–	t	465.4	t	44	1	24
	leaf freq	50	–	–	–	–	–	–	t	464.0	t	44	1	23
	ssg	10	–	–	–	–	–	–	t	457.7	t	44	1	24
	ssg	25	–	–	–	–	–	–	t	462.1	t	44	1	24
	ssg	50	–	–	–	–	–	–	t	461.7	t	44	1	24
	tree weight	10	–	–	–	–	–	–	t	465.1	t	44	1	24

cont. on next page ...

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
ns1760995	tree weight	25	–	–	–	–	–	–	t	463.6	t	44	1	24
	tree weight	50	–	–	–	–	–	–	t	460.9	t	44	1	23
	no clairvoyant	–	–	–	–	–	–	–	t	458.6	t	44	1	23
	0-restart	–	–	–	–	–	–	–	t	460.4	t	44	1	24
	monotone	10	–	–	–	–	–	–	t	t	t	1	1	1
	monotone	25	–	–	–	–	–	–	t	t	t	1	1	1
	monotone	50	–	–	–	–	–	–	t	t	t	1	1	1
	reg forest	10	–	–	–	–	–	–	t	t	t	1	1	1
	reg forest	25	–	–	–	–	–	–	t	t	t	1	1	1
	reg forest	50	–	–	–	–	–	–	t	t	t	1	1	1
	gap	10	–	–	–	–	–	–	t	t	t	1	1	1
	gap	25	–	–	–	–	–	–	t	t	t	1	1	1
	gap	50	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	10	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	25	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	50	–	–	–	–	–	–	t	t	t	1	1	1
	ssg	10	–	–	–	–	–	–	t	t	t	1	1	1
	ssg	25	–	–	–	–	–	–	t	t	t	1	1	1
	ssg	50	–	–	–	–	–	–	t	t	t	1	1	1
	tree weight	10	–	–	–	–	–	–	t	t	t	1	1	1
ns1830653	tree weight	25	–	–	–	–	–	–	t	t	t	1	1	1
	tree weight	50	–	–	–	–	–	–	t	t	t	1	1	1
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	1	1	1
	0-restart	–	–	–	–	–	–	–	t	t	t	1	1	1
	monotone	10	–	–	–	–	–	–	112.7	163.4	126.6	4993	7194	7298
	monotone	25	–	–	–	–	–	–	113.2	163.8	126.4	4993	7194	7298
	monotone	50	–	–	–	–	–	–	112.7	164.6	126.0	4993	7194	7298
	reg forest	10	–	–	–	1	1	–	213.0	200.1	128.6	9889	10553	7298
	reg forest	25	–	–	–	–	–	–	113.2	164.6	127.8	4993	7194	7298
	reg forest	50	–	–	–	–	–	–	112.9	164.1	127.5	4993	7194	7298
	gap	10	–	–	–	1	1	1	137.4	155.8	141.1	5130	8279	6863
	gap	25	–	–	–	1	1	1	141.6	172.3	148.9	5121	6354	6725
	gap	50	–	1	–	–	1	1	113.1	197.2	151.9	4993	8588	6089
	leaf freq	10	–	–	–	–	–	–	113.1	163.7	126.7	4993	7194	7298
	leaf freq	25	–	–	–	–	–	–	112.5	163.4	126.5	4993	7194	7298
	leaf freq	50	–	–	–	–	–	–	112.9	163.2	126.7	4993	7194	7298
	ssg	10	–	–	–	–	1	1	112.5	170.4	129.7	4993	8064	5235
	ssg	25	–	–	–	–	1	1	112.7	161.4	142.3	4993	6954	5955
	ssg	50	–	–	–	–	–	1	112.7	163.4	160.8	4993	7194	7861
ns1952667	tree weight	10	–	–	–	1	1	1	140.6	162.2	123.8	5478	9030	5287
	tree weight	25	–	–	–	1	1	1	136.1	170.5	150.2	5358	7663	6947
	tree weight	50	–	–	–	–	1	1	112.9	166.1	150.2	4993	7773	7943
	no clairvoyant	–	–	–	–	–	–	–	112.7	163.7	126.1	4993	7194	7298
	0-restart	–	–	–	–	–	–	–	112.5	163.3	126.9	4993	7194	7298
	monotone	10	–	–	–	1	1	1	641.2	754.8	t	7960	6499	12123
	monotone	25	–	–	–	1	1	1	655.4	754.6	t	7960	6499	11998
	monotone	50	–	–	–	1	1	1	654.8	754.0	t	7960	6499	12123
	reg forest	10	–	–	–	1	1	1	655.3	498.0	t	7960	4484	12111
	reg forest	25	–	–	–	–	–	–	t	2008.1	1935.6	11840	7003	16039
	reg forest	50	–	–	–	–	–	–	t	1997.9	1940.5	11832	7003	16039
	gap	10	–	–	–	–	–	–	t	2003.4	1931.5	11787	7003	16039
	gap	25	–	–	–	–	–	–	t	2008.0	1936.5	11840	7003	16039
	gap	50	–	–	–	–	–	–	t	2008.7	1942.2	11755	7003	16039
	leaf freq	10	–	–	–	–	–	1	t	1994.0	3402.9	11859	7003	24117
	leaf freq	25	–	–	–	–	–	–	t	1992.7	1930.4	11844	7003	16039
	leaf freq	50	–	–	–	–	–	–	t	1919.7	1881.8	11791	7003	16039
	ssg	10	–	–	–	–	1	–	t	1063.1	1938.6	11787	5610	16039
	ssg	25	–	–	–	–	1	–	t	t	1924.7	11791	38108	16039
	ssg	50	–	–	–	–	1	–	t	461.5	1937.9	11842	3347	16039
nu25-pr12	tree weight	10	–	–	–	–	–	–	t	1972.1	1941.9	11809	7003	16039
	tree weight	25	–	–	–	–	–	–	t	1995.9	1935.5	11859	7003	16039
	tree weight	50	–	–	–	–	–	–	t	1995.6	1932.3	11777	7003	16039
	no clairvoyant	–	–	–	–	–	–	–	t	1994.6	1938.6	11842	7003	16039
	0-restart	–	–	–	–	–	–	–	t	1999.7	1935.5	11821	7003	16039
	monotone	10	–	–	2	–	–	–	5.8	5.6	6.2	86	92	107
	monotone	25	–	–	2	–	–	–	5.8	5.6	6.2	86	92	107
	monotone	50	–	–	2	–	–	–	5.8	5.6	6.2	86	92	107
	reg forest	10	–	–	2	–	–	–	5.9	5.7	6.3	86	92	107
	reg forest	25	–	–	2	–	–	–	5.9	5.7	6.5	86	92	107
	reg forest	50	–	–	2	–	–	–	5.9	5.7	6.4	86	92	107
	gap	10	–	–	2	–	–	–	5.8	5.6	6.2	86	92	107
	gap	25	–	–	2	–	–	–	5.8	5.6	6.2	86	92	107
	gap	50	–	–	2	–	–	–	5.8	5.6	6.2	86	92	107
	leaf freq	10	–	–	2	–	–	–	5.8	5.6	6.3	86	92	107
	leaf freq	25	–	–	2	–	–	–	5.8	5.6	6.2	86	92	107
	leaf freq	50	–	–	2	–	–	–	5.7	5.6	6.3	86	92	107

cont. on next page ...

E. Appendix E

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
nursesched-m.	ssg	10	–	–	2	–	–	–	5.8	5.5	6.3	86	92	107
	ssg	25	–	–	2	–	–	–	5.8	5.6	6.2	86	92	107
	ssg	50	–	–	2	–	–	–	5.8	5.6	6.2	86	92	107
	tree weight	10	–	–	2	–	–	–	5.8	5.6	6.2	86	92	107
	tree weight	25	–	–	2	–	–	–	5.8	5.6	6.2	86	92	107
	tree weight	50	–	–	2	–	–	–	5.8	5.6	6.2	86	92	107
	no clairvoyant	–	–	–	2	–	–	–	5.9	5.6	6.3	86	92	107
	0-restart	–	–	–	–	–	–	–	5.7	5.6	3.7	86	92	50
	monotone	10	–	–	–	–	–	–	t	t	t	562	708	487
	monotone	25	–	–	–	–	–	–	t	t	t	548	712	479
	monotone	50	–	–	–	–	–	–	t	t	t	553	717	490
	reg forest	10	–	–	–	–	–	–	t	t	t	567	708	493
	reg forest	25	–	–	–	–	–	–	t	t	t	567	708	473
	reg forest	50	–	–	–	–	–	–	t	t	t	567	708	481
	gap	10	–	–	–	–	–	–	t	t	t	553	710	475
	gap	25	–	–	–	–	–	–	t	t	t	567	708	481
	gap	50	–	–	–	–	–	–	t	t	t	539	692	476
	leaf freq	10	–	–	–	–	–	–	t	t	t	567	708	462
	leaf freq	25	–	–	–	–	–	–	t	t	t	567	712	480
	leaf freq	50	–	–	–	–	–	–	t	t	t	583	696	487
	ssg	10	–	–	–	–	–	–	t	t	t	584	708	473
	ssg	25	–	–	–	–	–	–	t	t	t	517	708	483
	ssg	50	–	–	–	–	–	–	t	t	t	594	708	480
	tree weight	10	–	–	–	–	–	–	t	t	t	603	712	480
	tree weight	25	–	–	–	–	–	–	t	t	t	553	708	473
	tree weight	50	–	–	–	–	–	–	t	t	t	589	708	487
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	596	708	480
	0-restart	–	–	–	–	–	–	–	t	t	t	609	717	487
nursesched-s.	monotone	10	–	1	–	–	–	–	108.3	85.2	103.5	272	303	204
	monotone	25	–	1	–	–	–	–	108.2	85.3	103.7	272	303	204
	monotone	50	–	1	–	–	–	–	107.5	85.4	103.7	272	303	204
	reg forest	10	–	1	–	–	–	–	108.2	85.2	104.0	272	303	204
	reg forest	25	–	1	–	–	–	–	108.1	85.2	104.0	272	303	204
	reg forest	50	–	1	–	–	–	–	108.0	85.3	104.1	272	303	204
	gap	10	–	1	–	–	–	–	108.4	85.4	103.8	272	303	204
	gap	25	–	1	–	–	–	–	108.1	85.1	103.8	272	303	204
	gap	50	–	1	–	–	–	–	108.6	84.9	103.5	272	303	204
	leaf freq	10	–	1	–	–	–	–	108.4	84.8	103.3	272	303	204
	leaf freq	25	–	1	–	–	–	–	108.4	85.5	103.5	272	303	204
	leaf freq	50	–	1	–	–	–	–	108.6	85.1	104.0	272	303	204
	ssg	10	–	1	–	–	–	–	107.3	85.2	104.2	272	303	204
	ssg	25	–	1	–	–	–	–	108.2	85.1	103.7	272	303	204
	ssg	50	–	1	–	–	–	–	108.2	85.1	103.8	272	303	204
	tree weight	10	–	1	–	–	–	–	108.1	86.0	103.7	272	303	204
	tree weight	25	–	1	–	–	–	–	108.3	85.4	103.7	272	303	204
	tree weight	50	–	1	–	–	–	–	108.5	85.3	104.0	272	303	204
	no clairvoyant	–	–	1	–	–	–	–	108.4	84.3	104.7	272	303	204
	0-restart	–	–	–	–	–	–	–	109.1	75.2	103.8	272	63	204
nw04	monotone	10	–	5	7	–	–	–	37.1	33.3	31.6	11	6	8
	monotone	25	–	5	7	–	–	–	36.9	33.2	31.3	11	6	8
	monotone	50	–	5	7	–	–	–	37.1	33.2	31.5	11	6	8
	reg forest	10	–	5	7	–	–	–	37.1	33.5	31.8	11	6	8
	reg forest	25	–	5	7	–	–	–	37.1	33.5	31.6	11	6	8
	reg forest	50	–	5	7	–	–	–	37.0	33.4	31.7	11	6	8
	gap	10	–	5	7	–	–	–	36.6	33.2	31.2	11	6	8
	gap	25	–	5	7	–	–	–	37.1	33.4	31.4	11	6	8
	gap	50	–	5	7	–	–	–	36.7	33.3	31.5	11	6	8
	leaf freq	10	–	5	7	–	–	–	37.0	33.2	31.2	11	6	8
	leaf freq	25	–	5	7	–	–	–	36.9	33.2	31.5	11	6	8
	leaf freq	50	–	5	7	–	–	–	36.6	33.1	31.3	11	6	8
	ssg	10	–	5	7	–	–	–	36.6	33.2	31.3	11	6	8
	ssg	25	–	5	7	–	–	–	36.8	33.1	31.5	11	6	8
	ssg	50	–	5	7	–	–	–	36.6	33.1	31.5	11	6	8
	tree weight	10	–	5	7	–	–	–	37.0	33.2	31.5	11	6	8
	tree weight	25	–	5	7	–	–	–	36.9	33.3	31.2	11	6	8
	tree weight	50	–	5	7	–	–	–	36.9	33.2	31.2	11	6	8
	no clairvoyant	–	–	5	7	–	–	–	37.2	33.1	31.3	11	6	8
	0-restart	–	–	–	–	–	–	–	50.7	33.9	34.0	1	1	1
opm2-z10-s4	monotone	10	–	–	–	–	–	–	t	t	t	752	388	446
	monotone	25	–	–	–	–	–	–	t	t	t	754	389	446
	monotone	50	–	–	–	–	–	–	t	t	t	754	389	448
	reg forest	10	–	–	–	–	–	–	t	t	t	754	389	446
	reg forest	25	–	–	–	–	–	–	t	t	t	752	388	446
	reg forest	50	–	–	–	–	–	–	t	t	t	754	388	446
	gap	10	–	–	–	–	–	–	t	t	t	752	388	446
	gap	25	–	–	–	–	–	–	t	t	t	754	388	446

cont. on next page ...

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
p200x1188c	gap	50	–	–	–	–	–	–	t	t	t	754	389	446
	leaf freq	10	–	–	–	–	–	–	t	t	t	752	388	446
	leaf freq	25	–	–	–	–	–	–	t	t	t	754	389	446
	leaf freq	50	–	–	–	–	–	–	t	t	t	754	388	446
	ssg	10	–	–	–	–	–	–	t	t	t	751	388	446
	ssg	25	–	–	–	–	–	–	t	t	t	751	389	446
	ssg	50	–	–	–	–	–	–	t	t	t	754	388	446
	tree weight	10	–	–	–	–	–	–	t	t	t	754	389	447
	tree weight	25	–	–	–	–	–	–	t	t	t	754	387	446
	tree weight	50	–	–	–	–	–	–	t	t	t	754	388	446
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	754	388	446
	0-restart	–	–	–	–	–	–	–	t	t	t	754	388	443
	monotone	10	–	–	–	–	–	–	2.7	2.7	2.7	1	1	1
	monotone	25	–	–	–	–	–	–	2.7	2.8	2.8	1	1	1
	monotone	50	–	–	–	–	–	–	2.7	2.8	2.8	1	1	1
	reg forest	10	–	–	–	–	–	–	2.8	2.8	2.8	1	1	1
	reg forest	25	–	–	–	–	–	–	2.8	2.8	2.8	1	1	1
	reg forest	50	–	–	–	–	–	–	2.8	2.8	2.8	1	1	1
	gap	10	–	–	–	–	–	–	2.8	2.8	2.8	1	1	1
	gap	25	–	–	–	–	–	–	2.8	2.8	2.8	1	1	1
	gap	50	–	–	–	–	–	–	2.7	2.8	2.8	1	1	1
	leaf freq	10	–	–	–	–	–	–	2.8	2.8	2.8	1	1	1
	leaf freq	25	–	–	–	–	–	–	2.8	2.8	2.8	1	1	1
	leaf freq	50	–	–	–	–	–	–	2.7	2.8	2.8	1	1	1
	ssg	10	–	–	–	–	–	–	2.7	2.7	2.8	1	1	1
	ssg	25	–	–	–	–	–	–	2.8	2.8	2.8	1	1	1
	ssg	50	–	–	–	–	–	–	2.7	2.7	2.8	1	1	1
	tree weight	10	–	–	–	–	–	–	2.7	2.7	2.8	1	1	1
	tree weight	25	–	–	–	–	–	–	2.8	2.8	2.7	1	1	1
	tree weight	50	–	–	–	–	–	–	2.8	2.7	2.8	1	1	1
	no clairvoyant	–	–	–	–	–	–	–	2.8	2.7	2.7	1	1	1
	0-restart	–	–	–	–	–	–	–	2.8	2.8	2.8	1	1	1
peg-solitaire-a3	monotone	10	–	–	–	1	1	1	t	t	t	1925	1987	1831
	monotone	25	–	–	–	1	1	1	t	t	t	1952	1954	1828
	monotone	50	–	–	–	1	1	1	t	t	t	1935	1961	1844
	reg forest	10	–	–	–	–	–	–	t	t	t	2048	1901	1750
	reg forest	25	–	–	–	–	–	–	t	t	t	2048	1897	1749
	reg forest	50	–	–	–	–	–	–	t	t	t	2049	1909	1747
	gap	10	–	–	–	–	–	–	t	t	t	2052	1900	1749
	gap	25	–	–	–	–	–	–	t	t	t	2049	1902	1748
	gap	50	–	–	–	–	–	–	t	t	t	2053	1906	1750
	leaf freq	10	–	–	–	–	–	–	t	t	t	2049	1903	1757
	leaf freq	25	–	–	–	–	–	–	t	t	t	2049	1898	1750
	leaf freq	50	–	–	–	–	–	–	t	t	t	2049	1903	1754
	ssg	10	–	–	–	–	–	–	t	t	t	2053	1901	1750
	ssg	25	–	–	–	–	–	–	t	t	t	2049	1901	1749
	ssg	50	–	–	–	–	–	–	t	t	t	2050	1907	1748
	tree weight	10	–	–	–	1	–	1	t	t	t	1688	1890	1781
	tree weight	25	–	–	–	1	–	1	t	t	t	1692	1901	1780
	tree weight	50	–	–	–	1	–	1	t	t	t	1705	1901	1777
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	2041	1904	1749
	0-restart	–	–	–	–	–	–	–	t	t	t	2049	1901	1746
pg	monotone	10	1	1	1	–	–	–	18.4	14.6	16.4	655	141	607
	monotone	25	1	1	1	–	–	–	18.4	14.6	16.5	655	141	607
	monotone	50	1	1	1	–	–	–	18.4	14.7	16.6	655	141	607
	reg forest	10	1	1	1	–	–	–	18.6	14.8	16.8	655	141	607
	reg forest	25	1	1	1	–	–	–	18.6	14.8	16.7	655	141	607
	reg forest	50	1	1	1	–	–	–	18.6	14.8	16.8	655	141	607
	gap	10	1	1	1	–	–	–	18.4	14.7	16.4	655	141	607
	gap	25	1	1	1	–	–	–	18.4	14.7	16.4	655	141	607
	gap	50	1	1	1	–	–	–	18.4	14.7	16.4	655	141	607
	leaf freq	10	1	1	1	–	–	–	18.4	14.6	16.4	655	141	607
	leaf freq	25	1	1	1	–	–	–	18.4	14.7	16.4	655	141	607
	leaf freq	50	1	1	1	–	–	–	18.4	14.7	16.5	655	141	607
	ssg	10	1	1	1	–	–	–	18.4	14.7	16.4	655	141	607
	ssg	25	1	1	1	–	–	–	18.4	14.7	16.6	655	141	607
	ssg	50	1	1	1	–	–	–	18.3	14.7	16.5	655	141	607
	tree weight	10	1	1	1	–	–	–	18.5	14.6	16.4	655	141	607
	tree weight	25	1	1	1	–	–	–	18.4	14.7	16.5	655	141	607
	tree weight	50	1	1	1	–	–	–	18.4	14.7	16.4	655	141	607
	no clairvoyant	–	1	1	1	–	–	–	18.6	14.6	16.5	655	141	607
	0-restart	–	–	–	–	–	–	–	16.3	13.8	19.3	547	142	623
pg5_34	monotone	10	1	1	1	1	–	1	3172.7	1445.5	1540.8	196994	241832	263584
	monotone	25	1	1	1	1	–	–	3175.7	1450.9	t	196994	241832	215322
	monotone	50	1	1	1	–	–	–	t	1447.5	t	197764	241832	215293
	reg forest	10	1	1	1	1	1	1	3188.7	1263.3	1556.5	196994	189870	263584

cont. on next page ...

E. Appendix E

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
p.sched3-3	reg forest	25	1	1	1	–	–	–	t	1456.7	t	197625	241832	215285
	reg forest	50	1	1	1	–	–	–	t	1457.4	t	197640	241832	215280
	gap	10	1	1	1	–	–	–	t	1443.9	t	197709	241832	215351
	gap	25	1	1	1	–	–	–	t	1445.9	t	197682	241832	215398
	gap	50	1	1	1	–	–	–	t	1448.0	t	197709	241832	215378
	leaf freq	10	1	1	1	1	1	1	t	1726.7	1300.2	207236	276644	216502
	leaf freq	25	1	1	1	1	1	1	2834.8	1439.5	1220.0	197256	218451	185540
	leaf freq	50	1	1	1	1	–	1	1686.4	1445.6	1080.7	222085	241832	169295
	ssg	10	1	1	1	1	1	1	2515.1	1263.5	1661.2	204747	189870	248093
	ssg	25	1	1	1	1	1	1	3271.3	1204.6	1439.3	198455	214799	232211
	ssg	50	1	1	1	1	1	1	1735.9	1527.3	1466.7	198421	254894	221332
	tree weight	10	1	1	1	1	1	1	3173.0	1263.0	1421.4	196994	189870	216173
	tree weight	25	1	1	1	1	1	1	3171.6	1180.9	1229.9	196994	196218	189916
	tree weight	50	1	1	1	1	1	1	3162.1	1279.6	1530.2	217639	212468	229739
	no clairvoyant	–	1	1	1	–	–	–	t	1432.7	t	197620	241832	215357
	0-restart	–	–	–	–	–	–	–	1451.8	2268.6	1310.3	250620	169636	200618
	monotone	10	–	–	–	–	–	–	t	t	t	115	78	40
	monotone	25	–	–	–	–	–	–	t	t	t	112	78	40
	monotone	50	–	–	–	–	–	–	t	t	t	115	78	39
	reg forest	10	–	–	–	–	–	–	t	t	t	115	78	40
	reg forest	25	–	–	–	–	–	–	t	t	t	115	78	40
	reg forest	50	–	–	–	–	–	–	t	t	t	115	78	40
	gap	10	–	–	–	–	–	–	t	t	t	115	78	40
	gap	25	–	–	–	–	–	–	t	t	t	114	78	40
	gap	50	–	–	–	–	–	–	t	t	t	115	78	40
	leaf freq	10	–	–	–	–	–	–	t	t	t	115	78	40
	leaf freq	25	–	–	–	–	–	–	t	t	t	115	78	40
	leaf freq	50	–	–	–	–	–	–	t	t	t	115	78	40
	ssg	10	–	–	–	–	–	–	t	t	t	114	78	40
	ssg	25	–	–	–	–	–	–	t	t	t	115	78	40
	ssg	50	–	–	–	–	–	–	t	t	t	117	78	40
	tree weight	10	–	–	–	–	–	–	t	t	t	115	78	40
	tree weight	25	–	–	–	–	–	–	t	t	t	113	78	40
	tree weight	50	–	–	–	–	–	–	t	t	t	114	78	40
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	114	78	40
	0-restart	–	–	–	–	–	–	–	t	t	t	112	78	39
p.sched6-2	monotone	10	1	–	–	–	–	–	74.4	865.0	65.9	2	2528	1
	monotone	25	1	–	–	–	–	–	74.2	871.4	65.4	2	2528	1
	monotone	50	1	–	–	–	–	–	74.1	864.5	66.1	2	2528	1
	reg forest	10	1	–	–	–	–	–	74.2	868.2	66.3	2	2528	1
	reg forest	25	1	–	–	–	–	–	74.3	861.4	66.4	2	2528	1
	reg forest	50	1	–	–	–	–	–	74.5	863.3	66.0	2	2528	1
	gap	10	1	–	–	–	–	–	74.2	869.8	66.4	2	2528	1
	gap	25	1	–	–	–	–	–	73.8	863.6	65.8	2	2528	1
	gap	50	1	–	–	–	–	–	74.3	864.0	66.1	2	2528	1
	leaf freq	10	1	–	–	–	–	–	74.0	860.8	65.9	2	2528	1
	leaf freq	25	1	–	–	–	–	–	74.1	865.3	66.0	2	2528	1
	leaf freq	50	1	–	–	–	–	–	73.3	867.3	66.0	2	2528	1
	ssg	10	1	–	–	–	1	–	74.2	695.3	65.9	2	1333	1
	ssg	25	1	–	–	–	–	–	74.3	862.8	66.2	2	2528	1
	ssg	50	1	–	–	–	–	–	73.8	868.5	66.0	2	2528	1
	tree weight	10	1	–	–	–	1	–	74.3	677.1	66.2	2	1172	1
	tree weight	25	1	–	–	–	–	–	74.2	868.0	66.0	2	2528	1
	tree weight	50	1	–	–	–	–	–	74.2	863.6	65.4	2	2528	1
	no clairvoyant	–	1	–	–	–	–	–	74.1	865.2	66.2	2	2528	1
	0-restart	–	–	–	–	–	–	–	115.2	863.9	66.3	53	2528	1
	monotone	10	–	–	–	–	–	–	42.9	22.0	34.1	1257	220	228
	monotone	25	–	–	–	–	–	–	43.3	22.1	33.3	1257	220	228
	monotone	50	–	–	–	–	–	–	43.3	22.2	33.8	1257	220	228
	reg forest	10	–	–	–	–	–	–	44.3	22.6	34.8	1257	220	228
	reg forest	25	–	–	–	–	–	–	44.2	22.8	34.7	1257	220	228
	reg forest	50	–	–	–	–	–	–	44.6	22.5	34.7	1257	220	228
	gap	10	1	–	–	1	–	–	62.9	22.0	33.6	1197	220	228
	gap	25	1	–	–	1	–	–	62.8	21.7	33.8	1197	220	228
	gap	50	–	–	–	–	–	–	43.2	22.0	34.1	1257	220	228
	leaf freq	10	–	–	–	–	–	–	43.2	21.9	34.1	1257	220	228
	leaf freq	25	–	–	–	–	–	–	43.0	22.0	34.0	1257	220	228
	leaf freq	50	–	–	–	–	–	–	43.3	22.2	33.8	1257	220	228
	ssg	10	–	–	–	–	–	–	42.6	22.1	33.6	1257	220	228
	ssg	25	–	–	–	–	–	–	43.2	22.0	33.8	1257	220	228
	ssg	50	–	–	–	–	–	–	43.1	22.0	33.7	1257	220	228
	tree weight	10	1	–	–	1	–	–	62.1	22.2	33.8	1197	220	228
	tree weight	25	–	–	–	–	–	–	43.2	22.0	33.9	1257	220	228
	tree weight	50	–	–	–	–	–	–	43.2	22.0	33.8	1257	220	228
	no clairvoyant	–	–	–	–	–	–	–	44.3	22.2	34.1	1257	220	228
	0-restart	–	–	–	–	–	–	–	43.1	22.0	33.8	1257	220	228
piperout-08	monotone	10	–	–	–	–	–	–	42.9	22.0	34.1	1257	220	228
	monotone	25	–	–	–	–	–	–	43.3	22.1	33.3	1257	220	228
	monotone	50	–	–	–	–	–	–	43.3	22.2	33.8	1257	220	228
	reg forest	10	–	–	–	–	–	–	44.3	22.6	34.8	1257	220	228
	reg forest	25	–	–	–	–	–	–	44.2	22.8	34.7	1257	220	228
	reg forest	50	–	–	–	–	–	–	44.6	22.5	34.7	1257	220	228
	gap	10	1	–	–	1	–	–	62.9	22.0	33.6	1197	220	228
	gap	25	1	–	–	1	–	–	62.8	21.7	33.8	1197	220	228
	gap	50	–	–	–	–	–	–	43.2	22.0	34.1	1257	220	228
	leaf freq	10	–	–	–	–	–	–	43.2	21.9	34.1	1257	220	228
	leaf freq	25	–	–	–	–	–	–	43.0	22.0	34.0	1257	220	228
	leaf freq	50	–	–	–	–	–	–	43.3	22.2	33.8	1257	220	228
	ssg	10	–	–	–	–	–	–	42.6	22.1	33.6	1257	220	228
	ssg	25	–	–	–	–	–	–	43.2	22.0	33.8	1257	220	228
	ssg	50	–	–	–	–	–	–	43.1	22.0	33.7	1257	220	228
	tree weight	10	1	–	–	1	–	–	62.1	22.2	33.8	1197	220	228
	tree weight	25	–	–	–	–	–	–	43.2	22.0	33.9	1257	220	228
	tree weight	50	–	–	–	–	–	–	43.2	22.0	33.8	1257	220	228
	no clairvoyant	–	–	–	–	–	–	–	44.3	22.2	34.1	1257	220	228
	0-restart	–	–	–	–	–	–	–	43.1	22.0	33.8	1257	220	228

cont. on next page ...

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
piperout-27	monotone	10	–	1	1	–	–	–	35.9	43.2	13.5	64	77	2
	monotone	25	–	1	1	–	–	–	35.7	43.0	13.2	64	77	2
	monotone	50	–	1	1	–	–	–	35.8	43.5	13.4	64	77	2
	reg forest	10	–	1	1	–	–	–	36.2	43.7	13.9	64	77	2
	reg forest	25	–	1	1	–	–	–	36.1	43.4	14.0	64	77	2
	reg forest	50	–	1	1	–	–	–	36.4	43.5	13.8	64	77	2
	gap	10	–	1	1	–	–	–	35.7	43.0	13.4	64	77	2
	gap	25	–	1	1	–	–	–	36.1	42.6	13.6	64	77	2
	gap	50	–	1	1	–	–	–	36.0	43.3	13.6	64	77	2
	leaf freq	10	–	1	1	–	–	–	35.7	43.0	13.4	64	77	2
	leaf freq	25	–	1	1	–	–	–	35.9	43.1	13.4	64	77	2
	leaf freq	50	–	1	1	–	–	–	35.7	43.2	13.4	64	77	2
	ssg	10	–	1	1	–	–	–	35.9	42.8	13.4	64	77	2
	ssg	25	–	1	1	–	–	–	35.7	43.0	13.5	64	77	2
	ssg	50	–	1	1	–	–	–	36.0	43.4	13.5	64	77	2
	tree weight	10	–	1	1	–	–	–	35.7	43.2	13.5	64	77	2
	tree weight	25	–	1	1	–	–	–	35.7	43.1	13.5	64	77	2
	tree weight	50	–	1	1	–	–	–	35.6	43.0	13.5	64	77	2
	no clairvoyant	–	–	1	1	–	–	–	36.1	42.9	13.6	64	77	2
	0-restart	–	–	–	–	–	–	–	35.6	23.9	14.0	64	23	8
pk1	monotone	10	–	–	–	1	1	1	118.1	131.2	122.6	301271	341496	323572
	monotone	25	–	–	–	1	1	1	117.3	130.1	122.2	301271	341496	323572
	monotone	50	–	–	–	–	–	1	146.3	148.3	122.6	345884	339401	323572
	reg forest	10	–	–	–	1	1	1	121.2	133.0	125.4	301271	341496	323572
	reg forest	25	–	–	–	1	1	1	127.7	125.5	142.8	308886	299317	349304
	reg forest	50	–	–	–	–	–	–	161.5	164.1	178.3	345884	339401	395948
	gap	10	–	–	–	1	1	1	121.2	121.9	119.6	305818	310020	306624
	gap	25	–	–	–	1	–	1	143.1	148.1	128.4	370429	339401	330723
	gap	50	–	–	–	1	–	–	129.8	147.9	162.9	332294	339401	395948
	leaf freq	10	–	–	–	1	1	1	130.0	117.4	134.0	304900	302324	331171
	leaf freq	25	–	–	–	1	1	1	129.4	128.1	131.2	304900	321168	308403
	leaf freq	50	–	–	–	1	1	1	123.9	138.0	138.3	308415	339893	338146
	ssg	10	–	–	–	1	1	1	132.9	124.9	124.5	336044	301691	316284
	ssg	25	–	–	–	1	1	1	133.9	122.6	125.2	336898	306037	316284
	ssg	50	–	–	–	1	–	–	137.2	148.0	161.6	336129	339401	395948
	tree weight	10	–	–	–	1	1	1	124.4	130.5	121.3	309731	341496	310695
	tree weight	25	–	–	–	1	1	1	126.4	130.7	128.8	309731	341496	348275
	tree weight	50	–	–	–	1	1	1	125.0	130.8	142.2	309731	341496	371134
	no clairvoyant	–	–	–	–	–	–	–	148.3	147.1	161.8	345884	339401	395948
	0-restart	–	–	–	–	–	–	–	146.9	146.8	160.2	345884	339401	395948
prot.121hz512p9	monotone	10	–	–	–	–	–	–	t	t	t	12827	24772	15473
	monotone	25	–	–	–	–	–	–	t	t	t	12978	24141	15672
	monotone	50	–	–	–	–	–	–	t	t	t	12903	24723	15326
	reg forest	10	–	–	–	–	–	–	t	t	t	12912	24520	15421
	reg forest	25	–	–	–	–	–	–	t	t	t	12579	24251	15363
	reg forest	50	–	–	–	–	–	–	t	t	t	12957	24597	15392
	gap	10	–	–	–	1	1	1	t	t	t	5656	12474	5753
	gap	25	–	–	–	–	1	–	t	t	t	12788	17462	15525
	gap	50	–	–	–	–	–	–	t	t	t	12801	24960	15478
	leaf freq	10	–	–	–	1	1	1	t	t	t	12115	18916	10839
	leaf freq	25	–	–	–	–	–	–	t	t	t	13046	24132	15423
	leaf freq	50	–	–	–	–	–	–	t	t	t	12808	24152	15293
	ssg	10	–	–	–	1	1	1	t	t	t	2713	1082	1528
	ssg	25	–	–	–	1	1	1	t	t	t	2713	1082	1528
	ssg	50	–	–	–	1	1	1	t	t	t	2713	1082	1528
	tree weight	10	–	–	–	1	–	1	t	t	t	2734	24132	3980
	tree weight	25	–	–	–	1	–	1	t	t	t	3416	24624	14126
	tree weight	50	–	–	–	–	–	1	t	t	t	12859	24710	14136
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	12970	24987	15593
	0-restart	–	–	–	–	–	–	–	t	t	t	12849	24766	15352
prot.122trx1p8	monotone	10	–	–	–	–	–	–	t	t	t	23204	32043	33316
	monotone	25	–	–	–	–	–	–	t	t	t	23305	32334	33606
	monotone	50	–	–	–	–	–	–	t	t	t	23346	32315	33621
	reg forest	10	–	–	–	–	–	–	t	t	t	23262	32368	33346
	reg forest	25	–	–	–	–	–	–	t	t	t	23763	32041	33814
	reg forest	50	–	–	–	–	–	–	t	t	t	23095	32242	33814
	gap	10	–	–	–	1	1	1	t	t	t	19075	17914	17864
	gap	25	–	–	–	–	–	–	t	t	t	23095	32170	33405
	gap	50	–	–	–	–	–	–	t	t	t	23152	32273	33785
	leaf freq	10	–	–	–	1	1	1	t	t	t	12077	9286	22961
	leaf freq	25	–	–	–	1	–	–	t	t	t	18830	32231	33264
	leaf freq	50	–	–	–	–	–	–	t	t	t	23248	32112	33604
	ssg	10	–	–	–	1	1	1	t	t	t	2886	15768	1503
	ssg	25	–	–	–	1	1	1	t	t	t	2905	15761	21170
	ssg	50	–	–	–	1	1	1	t	t	t	5286	2317	20978
	tree weight	10	–	–	–	1	1	1	t	t	t	5646	6028	8678

cont. on next page ...

E. Appendix E

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
qap10	tree weight	25	–	–	–	1	1	1	t	t	t	9016	6028	8729
	tree weight	50	–	–	–	1	1	1	t	t	t	4632	6028	8706
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	23260	32331	33649
	0-restart	–	–	–	–	–	–	–	t	t	t	23244	32595	33540
	monotone	10	1	–	–	–	–	–	43.7	73.3	78.2	1	2	2
	monotone	25	1	–	–	–	–	–	43.8	73.2	78.2	1	2	2
	monotone	50	1	–	–	–	–	–	43.7	73.2	77.8	1	2	2
	reg forest	10	1	–	–	–	–	–	43.8	73.3	78.5	1	2	2
	reg forest	25	1	–	–	–	–	–	43.9	73.3	78.4	1	2	2
	reg forest	50	1	–	–	–	–	–	43.8	73.1	78.3	1	2	2
	gap	10	1	–	–	–	–	–	43.6	73.2	78.3	1	2	2
	gap	25	1	–	–	–	–	–	43.5	72.8	78.2	1	2	2
	gap	50	1	–	–	–	–	–	43.7	73.2	78.4	1	2	2
	leaf freq	10	1	–	–	–	–	–	43.7	73.2	78.5	1	2	2
	leaf freq	25	1	–	–	–	–	–	43.6	73.3	78.3	1	2	2
	leaf freq	50	1	–	–	–	–	–	43.6	73.3	78.2	1	2	2
	ssg	10	1	–	–	–	–	–	43.8	73.7	78.3	1	2	2
	ssg	25	1	–	–	–	–	–	43.5	73.2	78.3	1	2	2
	ssg	50	1	–	–	–	–	–	43.7	73.3	78.2	1	2	2
	tree weight	10	1	–	–	–	–	–	43.7	73.2	78.3	1	2	2
rad.m18-12-05	tree weight	25	1	–	–	–	–	–	43.6	73.1	78.3	1	2	2
	tree weight	50	1	–	–	–	–	–	43.7	73.1	78.3	1	2	2
	no clairvoyant	–	1	–	–	–	–	–	43.8	73.4	78.3	1	2	2
	0-restart	–	–	–	–	–	–	–	52.5	73.4	78.7	1	2	2
	monotone	10	–	–	–	1	1	1	t	t	t	458330	345840	425910
	monotone	25	–	–	–	1	1	1	t	t	t	456752	345013	428401
	monotone	50	–	–	–	1	1	1	t	t	t	457737	345642	424044
	reg forest	10	–	–	–	1	1	1	t	t	t	513787	344202	425624
	reg forest	25	–	–	–	–	–	–	t	t	t	486274	555509	383711
	reg forest	50	–	–	–	–	–	–	t	t	t	487857	557207	381772
	gap	10	–	–	–	1	1	1	t	t	t	422424	638102	425600
	gap	25	–	–	–	1	1	1	t	t	t	577147	636252	426321
	gap	50	–	–	–	1	1	1	t	t	t	437836	636733	426527
	leaf freq	10	–	–	–	1	1	1	t	t	t	526818	783571	425786
	leaf freq	25	–	–	–	1	1	1	t	t	t	523804	581069	425415
	leaf freq	50	–	–	–	1	1	1	t	t	t	512100	462632	425420
	ssg	10	–	–	–	1	1	1	t	t	t	513425	681518	425499
	ssg	25	–	–	–	1	1	1	t	t	t	510421	681434	426871
	ssg	50	–	–	–	1	1	1	t	t	t	480678	681985	426990
rad.m40-10-02	tree weight	10	–	–	–	1	1	1	t	t	t	458152	635462	426195
	tree weight	25	–	–	–	1	1	1	t	t	t	458724	637120	426300
	tree weight	50	–	–	–	1	1	1	t	t	t	457412	635882	426891
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	489205	566302	383479
	0-restart	–	–	–	–	–	–	–	t	t	t	491282	566876	384993
	monotone	10	–	–	–	1	1	1	t	t	t	60139	61855	86771
	monotone	25	–	–	–	1	1	1	t	t	t	60726	62091	86405
	monotone	50	–	–	–	1	1	1	t	t	t	60982	61563	86077
	reg forest	10	–	–	–	1	1	1	t	t	t	81062	61798	86405
	reg forest	25	–	–	–	1	–	1	t	t	t	61480	57189	86484
	reg forest	50	–	–	–	–	–	–	t	t	t	74629	57722	67427
	gap	10	–	–	–	1	1	1	t	t	t	67078	50351	29167
	gap	25	–	–	–	1	1	1	t	t	t	50841	50416	89649
	gap	50	–	–	–	1	1	1	t	t	t	77092	50473	81957
	leaf freq	10	–	–	–	1	1	1	t	t	t	83741	62652	57307
	leaf freq	25	–	–	–	1	1	1	t	t	t	84484	56537	68294
	leaf freq	50	–	–	–	–	1	1	t	t	t	76164	52699	68429
	ssg	10	–	–	–	1	1	1	t	t	t	54924	98634	92978
	ssg	25	–	–	–	1	1	1	t	t	t	54841	98492	86189
	ssg	50	–	–	–	1	1	1	t	t	t	54684	58672	89577
rail01	tree weight	10	–	–	–	1	1	1	t	t	t	80079	62125	53426
	tree weight	25	–	–	–	1	1	1	t	t	t	68479	61820	60990
	tree weight	50	–	–	–	1	1	1	t	t	t	73307	61731	51460
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	74861	56703	67954
	0-restart	–	–	–	–	–	–	–	t	t	t	74026	58636	68133
	monotone	10	–	–	–	–	–	–	t	t	t	1	1	1
	monotone	25	–	–	–	–	–	–	t	t	t	1	1	1
	monotone	50	–	–	–	–	–	–	t	t	t	1	1	1
	reg forest	10	–	–	–	–	–	–	t	t	t	1	1	1
	reg forest	25	–	–	–	–	–	–	t	t	t	1	1	1
	reg forest	50	–	–	–	–	–	–	t	t	t	1	1	1
	gap	10	–	–	–	–	–	–	t	t	t	1	1	1
	gap	25	–	–	–	–	–	–	t	t	t	1	1	1
	gap	50	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	10	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	25	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	50	–	–	–	–	–	–	t	t	t	1	1	1

cont. on next page ...

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
rail02	ssg	10	–	–	–	–	–	–	t	t	t	1	1	1
	ssg	25	–	–	–	–	–	–	t	t	t	1	1	1
	ssg	50	–	–	–	–	–	–	t	t	t	1	1	1
	tree weight	10	–	–	–	–	–	–	t	t	t	1	1	1
	tree weight	25	–	–	–	–	–	–	t	t	t	1	1	1
	tree weight	50	–	–	–	–	–	–	t	t	t	1	1	1
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	1	1	1
	0-restart	–	–	–	–	–	–	–	t	t	t	1	1	1
	monotone	10	–	–	–	–	–	–	t	t	t	1	1	1
	monotone	25	–	–	–	–	–	–	t	t	t	1	1	1
	monotone	50	–	–	–	–	–	–	t	t	t	1	1	1
	reg forest	10	–	–	–	–	–	–	t	t	t	1	1	1
	reg forest	25	–	–	–	–	–	–	t	t	t	1	1	1
	reg forest	50	–	–	–	–	–	–	t	t	t	1	1	1
	gap	10	–	–	–	–	–	–	t	t	t	1	1	1
	gap	25	–	–	–	–	–	–	t	t	t	1	1	1
	gap	50	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	10	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	25	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	50	–	–	–	–	–	–	t	t	t	1	1	1
	ssg	10	–	–	–	–	–	–	t	t	t	1	1	1
	ssg	25	–	–	–	–	–	–	t	t	t	1	1	1
	ssg	50	–	–	–	–	–	–	t	t	t	1	1	1
	tree weight	10	–	–	–	–	–	–	t	t	t	1	1	1
	tree weight	25	–	–	–	–	–	–	t	t	t	1	1	1
	tree weight	50	–	–	–	–	–	–	t	t	t	1	1	1
rail507	no clairvoyant	–	–	–	–	–	–	–	t	t	t	1	1	1
	0-restart	–	–	–	–	–	–	–	t	t	t	1	1	1
	monotone	10	–	–	–	–	–	–	249.4	201.2	190.0	990	856	856
	monotone	25	–	–	–	–	–	–	247.8	201.6	191.9	990	856	856
	monotone	50	–	–	–	–	–	–	250.4	202.5	191.9	990	856	856
	reg forest	10	–	–	–	–	–	–	251.0	202.3	190.3	990	856	856
	reg forest	25	–	–	–	–	–	–	248.7	203.0	191.6	990	856	856
	reg forest	50	–	–	–	–	–	–	249.3	203.8	190.5	990	856	856
	gap	10	–	–	–	–	–	–	249.2	200.2	189.9	990	856	856
	gap	25	–	–	–	–	–	–	248.7	201.4	191.1	990	856	856
	gap	50	–	–	–	–	–	–	251.2	202.2	190.4	990	856	856
	leaf freq	10	–	–	–	–	–	–	250.1	202.1	191.3	990	856	856
	leaf freq	25	–	–	–	–	–	–	248.3	202.8	191.7	990	856	856
	leaf freq	50	–	–	–	–	–	–	249.6	202.8	193.1	990	856	856
	ssg	10	–	–	–	–	–	–	250.6	201.1	190.1	990	856	856
	ssg	25	–	–	–	–	–	–	251.2	202.6	191.5	990	856	856
	ssg	50	–	–	–	–	–	–	250.1	202.5	191.2	990	856	856
	tree weight	10	–	–	–	–	–	–	248.6	202.0	190.2	990	856	856
	tree weight	25	–	–	–	–	–	–	249.1	201.8	191.4	990	856	856
	tree weight	50	–	–	–	–	–	–	248.8	201.2	191.2	990	856	856
	no clairvoyant	–	–	–	–	–	–	–	249.9	201.4	189.9	990	856	856
	0-restart	–	–	–	–	–	–	–	249.4	201.3	190.4	990	856	856
ran14x18-disj-8	monotone	10	–	–	–	1	1	1	1665.5	2478.0	1922.9	319747	555824	364603
	monotone	25	–	–	–	1	1	1	1665.3	2473.8	1921.5	319747	555824	364603
	monotone	50	–	–	–	–	–	1	2501.3	3088.3	1916.2	393538	535859	364603
	reg forest	10	–	–	–	1	1	1	1667.9	2476.2	1919.3	319747	555824	364603
	reg forest	25	–	–	–	1	1	1	1463.0	1787.6	1638.2	347564	328489	362276
	reg forest	50	–	–	–	–	–	–	2520.9	3121.8	2146.9	393538	535859	569911
	gap	10	–	–	–	1	1	1	1705.2	2012.6	1650.9	320096	470806	327943
	gap	25	–	–	–	–	1	1	2506.8	1928.9	1713.3	393538	452141	383500
	gap	50	–	–	–	–	–	–	2497.5	3088.7	2113.9	393538	535859	569911
	leaf freq	10	–	–	–	1	1	1	1746.2	1750.7	1761.4	354662	403484	354029
	leaf freq	25	–	–	–	1	1	1	2357.0	1631.5	1661.2	482726	361937	311648
	leaf freq	50	–	–	–	1	1	1	2348.9	1975.6	1657.9	482726	429218	311648
	ssg	10	–	–	–	1	1	1	1860.7	2215.0	1466.4	382365	359725	302741
	ssg	25	–	–	–	1	1	1	2086.1	1467.0	2115.9	505312	282049	401257
	ssg	50	–	–	–	1	1	1	2233.4	1766.2	1409.7	383814	362703	309121
	tree weight	10	–	–	–	1	1	1	1660.4	2295.3	1913.6	319747	473775	364603
	tree weight	25	–	–	–	1	1	1	1662.4	2303.9	1916.2	319747	473775	364603
	tree weight	50	–	–	–	1	1	1	1840.5	2254.8	1920.5	443604	420432	364603
	no clairvoyant	–	–	–	–	–	–	–	2500.5	3091.9	2116.1	393538	535859	569911
	0-restart	–	–	–	–	–	–	–	2495.6	3086.6	2115.9	393538	535859	569911
rd-rplusc-21	monotone	10	–	–	–	–	–	–	t	t	t	315525	181500	371633
	monotone	25	–	–	–	–	–	–	t	t	t	311945	181602	371827
	monotone	50	–	–	–	–	–	–	t	t	t	311649	180309	372714
	reg forest	10	–	–	–	–	–	–	t	t	t	310162	179139	371269
	reg forest	25	–	–	–	–	–	–	t	t	t	311460	181514	371065
	reg forest	50	–	–	–	–	–	–	t	t	t	313086	180352	369439
	gap	10	–	–	–	–	–	–	t	t	t	317711	181610	372086
	gap	25	–	–	–	–	–	–	t	t	t	314377	181584	368617

cont. on next page ...

E. Appendix E

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
reblock115	gap	50	–	–	–	–	–	–	t	t	t	313268	181685	372410
	leaf freq	10	–	–	–	1	–	1	t	t	t	444255	182865	339159
	leaf freq	25	–	–	–	–	–	–	t	t	t	312741	181014	368416
	leaf freq	50	–	–	–	–	–	–	t	t	t	313571	181663	372491
	ssg	10	–	–	–	–	–	–	t	t	t	314409	181628	371729
	ssg	25	–	–	–	–	–	–	t	t	t	311765	180314	372513
	ssg	50	–	–	–	–	–	–	t	t	t	314311	181610	371553
	tree weight	10	–	–	–	1	1	1	t	t	t	312775	527410	556128
	tree weight	25	–	–	–	1	1	1	t	t	t	229755	509832	344993
	tree weight	50	–	–	–	1	1	1	t	t	t	313387	492812	414183
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	310138	181526	369197
	0-restart	–	–	–	–	–	–	–	t	t	t	315653	181510	371757
	monotone	10	–	–	–	–	–	–	t	t	t	533833	494204	490784
	monotone	25	–	–	–	–	–	–	t	t	t	533672	494430	490101
	monotone	50	–	–	–	–	–	–	t	t	t	535409	494056	490646
	reg forest	10	–	–	–	1	–	–	t	t	t	416195	492273	489508
	reg forest	25	–	–	–	1	–	–	t	t	t	574858	491875	488948
	reg forest	50	–	–	–	–	–	–	t	t	t	531496	491329	489005
	gap	10	–	–	–	1	–	1	t	t	t	489991	494471	458829
	gap	25	–	–	–	–	–	–	t	t	t	531853	494931	491971
	gap	50	–	–	–	–	–	–	t	t	t	533383	493791	491003
	leaf freq	10	–	–	–	1	1	1	t	t	t	446398	443247	453771
	leaf freq	25	–	–	–	1	1	1	t	t	t	446927	467232	428173
	leaf freq	50	–	–	–	1	1	1	t	t	t	495462	474352	427441
	ssg	10	–	–	–	1	1	1	t	t	t	416924	490503	588718
	ssg	25	–	–	–	1	1	1	t	t	t	415903	601013	539553
	ssg	50	–	–	–	1	1	1	t	t	t	416208	498435	623486
	tree weight	10	–	–	–	1	1	1	t	t	t	537695	478144	509164
	tree weight	25	–	–	–	1	1	1	t	t	t	546557	493330	475815
	tree weight	50	–	–	–	1	1	1	t	t	t	488127	399629	590383
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	532121	494893	492795
	0-restart	–	–	–	–	–	–	–	t	t	t	533383	493992	490923
rmatr100-p10	monotone	10	–	–	–	–	–	–	147.8	146.0	142.6	796	828	737
	monotone	25	–	–	–	–	–	–	147.3	145.7	142.9	796	828	737
	monotone	50	–	–	–	–	–	–	147.9	146.1	142.6	796	828	737
	reg forest	10	–	–	–	–	–	–	147.5	146.0	143.0	796	828	737
	reg forest	25	–	–	–	–	–	–	147.2	145.8	142.5	796	828	737
	reg forest	50	–	–	–	–	–	–	147.4	145.9	142.4	796	828	737
	gap	10	–	–	–	–	–	–	147.1	146.2	142.7	796	828	737
	gap	25	–	–	–	–	–	–	147.7	145.6	142.6	796	828	737
	gap	50	–	–	–	–	–	–	146.4	145.4	142.7	796	828	737
	leaf freq	10	–	–	–	–	–	–	147.4	145.7	142.6	796	828	737
	leaf freq	25	–	–	–	–	–	–	146.4	145.9	142.8	796	828	737
	leaf freq	50	–	–	–	–	–	–	147.1	146.0	142.5	796	828	737
	ssg	10	–	–	–	–	–	–	147.6	146.0	142.7	796	828	737
	ssg	25	–	–	–	–	–	–	147.4	145.8	143.2	796	828	737
	ssg	50	–	–	–	–	–	–	148.2	145.7	143.0	796	828	737
	tree weight	10	–	–	–	–	–	–	147.3	145.8	142.7	796	828	737
	tree weight	25	–	–	–	–	–	–	147.1	145.6	142.9	796	828	737
	tree weight	50	–	–	–	–	–	–	147.2	145.8	142.4	796	828	737
	no clairvoyant	–	–	–	–	–	–	–	147.4	145.5	142.6	796	828	737
	0-restart	–	–	–	–	–	–	–	147.8	145.2	141.6	796	828	737
rmatr200-p5	monotone	10	–	–	–	–	–	–	t	t	t	93	34	102
	monotone	25	–	–	–	–	–	–	t	t	t	94	34	100
	monotone	50	–	–	–	–	–	–	t	t	t	92	34	102
	reg forest	10	–	–	–	–	–	–	t	t	t	91	33	101
	reg forest	25	–	–	–	–	–	–	t	t	t	94	33	101
	reg forest	50	–	–	–	–	–	–	t	t	t	92	34	101
	gap	10	–	–	–	–	–	–	t	t	t	93	36	101
	gap	25	–	–	–	–	–	–	t	t	t	94	34	101
	gap	50	–	–	–	–	–	–	t	t	t	94	34	101
	leaf freq	10	–	–	–	–	–	–	t	t	t	91	34	101
	leaf freq	25	–	–	–	–	–	–	t	t	t	93	33	101
	leaf freq	50	–	–	–	–	–	–	t	t	t	92	33	99
	ssg	10	–	–	–	–	–	–	t	t	t	91	34	101
	ssg	25	–	–	–	–	–	–	t	t	t	92	34	101
	ssg	50	–	–	–	–	–	–	t	t	t	93	34	101
	tree weight	10	–	–	–	–	–	–	t	t	t	91	33	101
	tree weight	25	–	–	–	–	–	–	t	t	t	91	34	101
	tree weight	50	–	–	–	–	–	–	t	t	t	93	34	101
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	91	34	102
	0-restart	–	–	–	–	–	–	–	t	t	t	93	34	102
rocI-4-11	monotone	10	–	–	–	–	–	–	53.6	53.9	52.4	13437	13378	13128
	monotone	25	–	–	–	–	–	–	53.8	53.8	52.5	13437	13378	13128
	monotone	50	–	–	–	–	–	–	53.7	53.8	52.5	13437	13378	13128
	reg forest	10	–	–	–	–	–	–	54.8	55.1	53.5	13437	13378	13128

cont. on next page ...

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
rocII-5-11	reg forest	25	–	–	–	–	–	–	54.7	54.7	53.6	13437	13378	13128
	reg forest	50	–	–	–	–	–	–	54.7	54.9	53.3	13437	13378	13128
	gap	10	–	–	–	1	1	1	82.4	97.5	75.9	15787	20352	15835
	gap	25	–	–	–	1	1	1	63.6	97.5	84.6	12682	20352	14965
	gap	50	–	–	–	1	1	1	82.0	97.1	72.9	16545	20352	15293
	leaf freq	10	–	–	–	–	–	–	53.6	53.6	52.4	13437	13378	13128
	leaf freq	25	–	–	–	–	–	–	53.5	53.8	52.8	13437	13378	13128
	leaf freq	50	–	–	–	–	–	–	53.6	53.8	52.5	13437	13378	13128
	ssg	10	–	–	–	1	1	1	78.5	103.3	68.8	15622	17480	12694
	ssg	25	–	–	–	1	1	1	81.6	82.8	72.1	12517	15040	15864
	ssg	50	–	–	–	1	1	1	84.8	80.9	80.3	17482	16543	17790
	tree weight	10	–	–	–	1	1	1	83.0	67.1	81.3	12569	14561	13751
	tree weight	25	–	–	–	1	1	1	77.4	83.4	82.8	14898	16122	13851
	tree weight	50	–	–	–	1	1	1	80.3	72.1	79.4	17027	13521	14534
	no clairvoyant	–	–	–	–	–	–	–	54.4	53.8	52.5	13437	13378	13128
	0-restart	–	–	–	–	–	–	–	53.6	53.8	52.1	13437	13378	13128
	monotone	10	–	–	–	–	–	–	t	t	t	200835	139861	245574
	monotone	25	–	–	–	–	–	–	t	t	t	199838	140690	246301
	monotone	50	–	–	–	–	–	–	t	t	t	198405	140916	246106
	reg forest	10	–	–	–	1	1	1	t	t	t	127228	168342	219781
	reg forest	25	–	–	–	–	–	–	t	t	t	199995	140017	244612
	reg forest	50	–	–	–	–	–	–	t	t	t	199286	138531	247654
	gap	10	–	–	–	1	1	1	t	t	t	279559	300508	192786
	gap	25	–	–	–	1	1	1	t	t	t	279522	302165	193523
	gap	50	–	–	–	1	1	1	t	t	t	280857	303407	191393
	leaf freq	10	–	–	–	1	1	1	t	t	t	95006	77995	86419
	leaf freq	25	–	–	–	1	1	1	t	t	t	97224	95076	84996
	leaf freq	50	–	–	–	1	1	1	t	t	t	166806	88851	86739
	ssg	10	–	–	–	1	1	1	t	t	t	167563	303829	160144
	ssg	25	–	–	–	1	1	1	t	t	t	159668	118745	165723
	ssg	50	–	–	–	1	1	1	t	t	t	189378	150034	158820
	tree weight	10	–	–	–	1	1	1	t	t	t	128499	234954	279448
	tree weight	25	–	–	–	1	1	1	t	t	t	97968	168731	93663
	tree weight	50	–	–	–	1	1	1	t	t	t	187399	194612	114907
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	199675	140518	243303
	0-restart	–	–	–	–	–	–	–	t	t	t	199591	141613	251940
rococoB10-011000	monotone	10	–	–	–	1	1	1	t	t	t	75365	70343	61792
	monotone	25	–	–	–	–	1	–	t	t	t	63029	70425	61164
	monotone	50	–	–	–	–	–	–	t	t	t	62875	85967	60912
	reg forest	10	–	–	–	1	1	–	t	t	t	75874	66005	60902
	reg forest	25	–	–	–	–	–	–	t	t	t	62832	85884	60806
	reg forest	50	–	–	–	–	–	–	t	t	t	62804	85795	60862
	gap	10	–	–	–	1	1	1	t	t	t	70533	75537	95833
	gap	25	–	–	–	1	1	1	t	t	t	65282	70225	61999
	gap	50	–	–	–	1	1	1	t	t	t	66018	75381	59909
	leaf freq	10	–	–	–	1	1	1	t	t	t	53739	68983	57629
	leaf freq	25	–	–	–	1	1	1	t	t	t	54668	68365	53403
	leaf freq	50	–	–	–	–	–	–	t	t	t	62616	85888	61374
	ssg	10	–	–	–	1	1	1	t	t	t	66346	70506	78543
	ssg	25	–	–	–	1	1	1	t	t	t	53753	71029	66792
	ssg	50	–	–	–	1	1	1	t	t	t	59318	68121	66037
	tree weight	10	–	–	–	1	1	1	t	t	t	57210	77480	72023
	tree weight	25	–	–	–	1	1	1	t	t	t	57209	77714	71903
	tree weight	50	–	–	–	1	1	1	t	t	t	57172	77401	71487
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	62752	86202	60916
	0-restart	–	–	–	–	–	–	–	t	t	t	62894	86132	61069
rococoC10-001000	monotone	10	–	–	–	1	1	1	1654.2	2228.1	521.1	118327	144244	38823
	monotone	25	–	–	–	–	–	1	597.2	715.4	520.8	32105	47263	38823
	monotone	50	–	–	–	–	–	–	598.2	713.8	661.8	32105	47263	51034
	reg forest	10	–	–	–	1	1	1	768.0	2088.3	523.6	56678	122598	38823
	reg forest	25	–	–	–	–	–	1	601.5	715.8	539.7	32105	47263	38461
	reg forest	50	–	–	–	–	–	–	601.2	716.2	665.9	32105	47263	51034
	gap	10	–	–	–	1	1	1	1202.0	2317.4	520.5	91279	175214	38823
	gap	25	–	–	–	1	1	1	736.5	1539.2	493.4	57321	109343	37453
	gap	50	–	–	–	–	–	1	596.6	716.6	536.3	32105	47263	36576
	leaf freq	10	–	–	–	1	1	1	1176.2	667.2	693.1	73521	43796	48908
	leaf freq	25	–	–	–	–	1	1	596.2	1192.0	694.8	32105	82409	48908
	leaf freq	50	–	–	–	–	–	–	599.1	714.9	661.5	32105	47263	51034
	ssg	10	–	–	–	1	1	1	452.3	2208.2	520.2	26307	135021	38823
	ssg	25	–	–	–	1	1	1	492.0	t	519.9	25082	225966	38823
	ssg	50	–	–	–	1	1	1	564.0	699.0	493.9	32226	41554	37453
	tree weight	10	–	–	–	1	1	1	1207.8	2230.7	1329.0	91279	144244	107746
	tree weight	25	–	–	–	1	1	1	737.0	2229.6	1001.8	57321	144244	71939
	tree weight	50	–	–	–	1	1	1	1076.3	643.2	835.9	55950	39934	53039
	no clairvoyant	–	–	–	–	–	–	–	600.5	715.7	659.4	32105	47263	51034
	0-restart	–	–	–	–	–	–	–	596.9	713.6	661.7	32105	47263	51034

cont. on next page ...

E. Appendix E

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
roi2alpha3n4	monotone	10	–	–	–	–	–	–	1290.1	1394.2	1081.7	13728	14224	9771
	monotone	25	–	–	–	–	–	–	1312.0	1410.8	1082.1	13728	14224	9771
	monotone	50	–	–	–	–	–	–	1312.4	1406.4	1078.8	13728	14224	9771
	reg forest	10	–	–	–	–	–	–	1313.0	1390.7	1092.3	13728	14224	9771
	reg forest	25	–	–	–	–	–	–	1315.5	1396.5	1076.8	13728	14224	9771
	reg forest	50	–	–	–	–	–	–	1304.2	1393.6	1088.4	13728	14224	9771
	gap	10	–	–	–	1	1	1	1495.2	1526.8	1624.0	14234	16055	16250
	gap	25	–	–	–	1	1	1	1648.9	1479.2	1597.7	16791	14209	13953
	gap	50	–	–	–	1	1	–	1783.9	1905.5	1073.8	17031	19396	9771
	leaf freq	10	–	–	–	–	–	–	1318.5	1380.6	1088.5	13728	14224	9771
	leaf freq	25	–	–	–	–	–	–	1307.8	1394.3	1072.5	13728	14224	9771
	leaf freq	50	–	–	–	–	–	–	1299.5	1401.8	1083.9	13728	14224	9771
	ssg	10	–	–	–	1	1	1	1623.2	1738.8	1342.6	17508	19907	13454
	ssg	25	–	–	–	1	1	1	1421.0	1709.6	1324.6	13070	19907	12518
	ssg	50	–	–	–	1	1	1	1547.6	1596.2	1332.1	14235	14747	12899
	tree weight	10	–	–	–	1	1	1	1532.9	1709.9	1393.5	15589	19565	12697
	tree weight	25	–	–	–	1	1	1	1494.9	1702.5	1582.8	14992	19932	13669
	tree weight	50	–	–	–	1	1	1	1513.3	1878.5	1536.4	14531	19739	11910
	no clairvoyant	–	–	–	–	–	–	–	1316.2	1391.2	1084.2	13728	14224	9771
	0-restart	–	–	–	–	–	–	–	1307.5	1391.8	1089.4	13728	14224	9771
roi5alpha10n8	monotone	10	–	–	–	1	1	1	t	t	t	2255	3345	1324
	monotone	25	–	–	–	1	–	–	t	t	t	2220	4757	2951
	monotone	50	–	–	–	–	–	–	t	t	t	2254	4757	2883
	reg forest	10	–	–	–	1	1	1	t	t	t	2237	3303	1326
	reg forest	25	–	–	–	1	1	1	t	t	t	1916	4046	1325
	reg forest	50	–	–	–	–	–	–	t	t	t	2273	4757	2839
	gap	10	–	–	–	1	1	1	t	t	t	2174	3337	1916
	gap	25	–	–	–	1	1	–	t	t	t	2074	2177	2902
	gap	50	–	–	–	1	1	–	t	t	t	2006	1548	2894
	leaf freq	10	–	–	–	–	–	–	t	t	t	2254	4757	2894
	leaf freq	25	–	–	–	–	–	–	t	t	t	2254	4757	2940
	leaf freq	50	–	–	–	–	–	–	t	t	t	2242	4757	2936
	ssg	10	–	–	–	1	1	1	t	t	t	1710	2733	2120
	ssg	25	–	–	–	1	1	1	t	t	t	1620	2701	1575
	ssg	50	–	–	–	1	1	1	t	t	t	1687	3737	1623
	tree weight	10	–	–	–	1	1	1	t	t	t	1939	3359	1323
	tree weight	25	–	–	–	1	1	1	t	t	t	1935	3440	1334
	tree weight	50	–	–	–	1	1	1	t	t	t	1915	3229	2307
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	2258	4757	2953
	0-restart	–	–	–	–	–	–	–	t	t	t	2254	4757	2916
roll3000	monotone	10	–	–	–	–	–	–	24.1	37.4	29.3	789	1221	1391
	monotone	25	–	–	–	–	–	–	24.1	37.4	29.2	789	1221	1391
	monotone	50	–	–	–	–	–	–	24.1	37.3	29.2	789	1221	1391
	reg forest	10	–	–	–	–	–	–	24.4	37.7	29.7	789	1221	1391
	reg forest	25	–	–	–	–	–	–	24.4	37.9	29.7	789	1221	1391
	reg forest	50	–	–	–	–	–	–	24.4	37.6	29.9	789	1221	1391
	gap	10	–	–	–	–	–	–	24.0	37.3	29.3	789	1221	1391
	gap	25	–	–	–	–	–	–	24.1	37.2	29.2	789	1221	1391
	gap	50	–	–	–	–	–	–	24.0	37.4	29.3	789	1221	1391
	leaf freq	10	–	–	–	–	–	–	24.0	37.2	29.2	789	1221	1391
	leaf freq	25	–	–	–	–	–	–	23.9	37.4	29.3	789	1221	1391
	leaf freq	50	–	–	–	–	–	–	23.9	37.3	29.3	789	1221	1391
	ssg	10	–	–	–	–	1	–	24.0	44.6	29.3	789	1324	1391
	ssg	25	–	–	–	–	1	–	24.1	44.6	29.4	789	1324	1391
	ssg	50	–	–	–	–	1	–	24.1	44.7	29.3	789	1324	1391
	tree weight	10	–	–	–	–	–	–	24.0	37.3	29.3	789	1221	1391
	tree weight	25	–	–	–	–	–	–	23.9	37.3	29.3	789	1221	1391
	tree weight	50	–	–	–	–	–	–	24.0	37.3	30.7	789	1221	1391
	no clairvoyant	–	–	–	–	–	–	–	24.4	37.2	29.1	789	1221	1391
	0-restart	–	–	–	–	–	–	–	24.0	37.4	29.3	789	1221	1391
s100	monotone	10	–	–	–	–	–	–	t	t	t	1	1	1
	monotone	25	–	–	–	–	–	–	t	t	t	1	1	1
	monotone	50	–	–	–	–	–	–	t	t	t	1	1	1
	reg forest	10	–	–	–	–	–	–	t	t	t	1	1	1
	reg forest	25	–	–	–	–	–	–	t	t	t	1	1	1
	reg forest	50	–	–	–	–	–	–	t	t	t	1	1	1
	gap	10	–	–	–	–	–	–	t	t	t	1	1	1
	gap	25	–	–	–	–	–	–	t	t	t	1	1	1
	gap	50	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	10	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	25	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	50	–	–	–	–	–	–	t	t	t	1	1	1
	ssg	10	–	–	–	–	–	–	t	t	t	1	1	1
	ssg	25	–	–	–	–	–	–	t	t	t	1	1	1
	ssg	50	–	–	–	–	–	–	t	t	t	1	1	1
	tree weight	10	–	–	–	–	–	–	t	t	t	1	1	1

cont. on next page ...

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
s250r10	tree weight	25	–	–	–	–	–	–	t	t	t	1	1	1
	tree weight	50	–	–	–	–	–	–	t	t	t	1	1	1
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	1	1	1
	0-restart	–	–	–	–	–	–	–	t	t	t	1	1	1
	monotone	10	6	5	5	–	–	–	2805.2	t	t	21678	36775	43256
	monotone	25	6	5	5	–	–	–	2825.9	t	t	21678	37041	43225
	monotone	50	6	5	5	–	–	–	2823.5	t	t	21678	37377	42834
	reg forest	10	6	5	5	–	–	–	2843.0	t	t	21678	36660	43887
	reg forest	25	6	5	5	–	–	–	2852.9	t	t	21678	36420	43406
	reg forest	50	6	5	5	–	–	–	2845.0	t	t	21678	37089	43443
	gap	10	6	5	5	–	–	–	2837.2	t	t	21678	36735	43463
	gap	25	6	5	5	–	–	–	2829.7	t	t	21678	37959	43390
	gap	50	6	5	5	–	–	–	2822.8	t	t	21678	36153	43178
	leaf freq	10	6	5	5	–	–	–	2824.2	t	t	21678	37310	43231
	leaf freq	25	6	5	5	–	–	–	2829.9	t	t	21678	36844	43296
	leaf freq	50	6	5	5	–	–	–	2825.1	t	t	21678	36234	42974
	ssg	10	6	5	5	–	–	–	2819.6	t	t	21678	37031	43031
	ssg	25	6	5	5	–	–	–	2823.4	t	t	21678	37081	43275
	ssg	50	6	5	5	–	–	–	2820.8	t	t	21678	37446	43467
	tree weight	10	6	5	5	–	–	–	2829.6	t	t	21678	37337	42877
	tree weight	25	6	5	5	–	–	–	2826.8	t	t	21678	36556	43247
	tree weight	50	6	5	5	–	–	–	2826.7	t	t	21678	37922	43366
	no clairvoyant	–	6	5	5	–	–	–	2824.9	t	t	21678	36556	43312
	0-restart	–	–	–	–	–	–	–	t	t	t	3347	2474	2212
satellites2-40	monotone	10	1	1	–	–	–	–	t	t	t	2	2	1
	monotone	25	1	1	–	–	–	–	t	t	t	2	2	1
	monotone	50	1	1	–	–	–	–	t	t	t	2	2	1
	reg forest	10	1	1	–	–	–	–	t	t	t	2	2	1
	reg forest	25	1	1	–	–	–	–	t	t	t	2	2	1
	reg forest	50	1	1	–	–	–	–	t	t	t	2	2	1
	gap	10	1	1	–	–	–	–	t	t	t	2	2	1
	gap	25	1	1	–	–	–	–	t	t	t	2	2	1
	gap	50	1	1	–	–	–	–	t	t	t	2	2	1
	leaf freq	10	1	1	–	–	–	–	t	t	t	2	2	1
	leaf freq	25	1	1	–	–	–	–	t	t	t	2	2	1
	leaf freq	50	1	1	–	–	–	–	t	t	t	2	2	1
	ssg	10	1	1	–	–	–	–	t	t	t	2	2	1
	ssg	25	1	1	–	–	–	–	t	t	t	2	2	1
	ssg	50	1	1	–	–	–	–	t	t	t	2	2	1
	tree weight	10	1	1	–	–	–	–	t	t	t	2	2	1
	tree weight	25	1	1	–	–	–	–	t	t	t	2	2	1
	tree weight	50	1	1	–	–	–	–	t	t	t	2	2	1
	no clairvoyant	–	1	1	–	–	–	–	t	t	t	2	2	1
	0-restart	–	–	–	–	–	–	–	t	t	t	2	8	1
sat.-60-fs	monotone	10	–	–	–	–	–	–	t	t	t	20	1	29
	monotone	25	–	–	–	–	–	–	t	t	t	20	1	29
	monotone	50	–	–	–	–	–	–	t	t	t	20	1	29
	reg forest	10	–	–	–	–	–	–	t	t	t	20	1	29
	reg forest	25	–	–	–	–	–	–	t	t	t	20	1	29
	reg forest	50	–	–	–	–	–	–	t	t	t	20	1	29
	gap	10	–	–	–	–	–	–	t	t	t	20	1	29
	gap	25	–	–	–	–	–	–	t	t	t	20	1	29
	gap	50	–	–	–	–	–	–	t	t	t	20	1	29
	leaf freq	10	–	–	–	–	–	–	t	t	t	20	1	29
	leaf freq	25	–	–	–	–	–	–	t	t	t	20	1	29
	leaf freq	50	–	–	–	–	–	–	t	t	t	20	1	29
	ssg	10	–	–	–	–	–	–	t	t	t	20	1	29
	ssg	25	–	–	–	–	–	–	t	t	t	20	1	29
	ssg	50	–	–	–	–	–	–	t	t	t	20	1	29
	tree weight	10	–	–	–	–	–	–	t	t	t	20	1	29
	tree weight	25	–	–	–	–	–	–	t	t	t	20	1	29
	tree weight	50	–	–	–	–	–	–	t	t	t	20	1	29
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	20	1	29
	0-restart	–	–	–	–	–	–	–	t	t	t	20	1	29
savsched1	monotone	10	–	–	–	–	–	–	t	t	t	1	1	1
	monotone	25	–	–	–	–	–	–	t	t	t	1	1	1
	monotone	50	–	–	–	–	–	–	t	t	t	1	1	1
	reg forest	10	–	–	–	–	–	–	t	t	t	1	1	1
	reg forest	25	–	–	–	–	–	–	t	t	t	1	1	1
	reg forest	50	–	–	–	–	–	–	t	t	t	1	1	1
	gap	10	–	–	–	–	–	–	t	t	t	1	1	1
	gap	25	–	–	–	–	–	–	t	t	t	1	1	1
	gap	50	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	10	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	25	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	50	–	–	–	–	–	–	t	t	t	1	1	1

cont. on next page ...

E. Appendix E

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
sct2	ssg	10	–	–	–	–	–	–	t	t	t	1	1	1
	ssg	25	–	–	–	–	–	–	t	t	t	1	1	1
	ssg	50	–	–	–	–	–	–	t	t	t	1	1	1
	tree weight	10	–	–	–	–	–	–	t	t	t	1	1	1
	tree weight	25	–	–	–	–	–	–	t	t	t	1	1	1
	tree weight	50	–	–	–	–	–	–	t	t	t	1	1	1
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	1	1	1
	0-restart	–	–	–	–	–	–	–	t	t	t	1	1	1
	monotone	10	–	–	–	–	–	–	t	t	t	409908	353185	385078
	monotone	25	–	–	–	–	–	–	t	t	t	410378	352697	384186
	monotone	50	–	–	–	–	–	–	t	t	t	409442	353181	381795
	reg forest	10	–	–	–	–	–	–	t	t	t	409594	352217	383684
	reg forest	25	–	–	–	–	–	–	t	t	t	408918	353015	382637
	reg forest	50	–	–	–	–	–	–	t	t	t	410135	352070	382596
	gap	10	–	–	–	–	–	–	t	t	t	409939	352725	383585
	gap	25	–	–	–	–	–	–	t	t	t	409567	353113	382907
	gap	50	–	–	–	–	–	–	t	t	t	408915	353335	383712
	leaf freq	10	–	–	–	–	–	–	t	t	t	409815	353250	383403
	leaf freq	25	–	–	–	–	–	–	t	t	t	410358	353335	383596
	leaf freq	50	–	–	–	–	–	–	t	t	t	409848	353217	383864
seymour	ssg	10	–	–	–	–	–	–	t	t	t	409687	353191	383318
	ssg	25	–	–	–	–	–	–	t	t	t	409267	352929	383239
	ssg	50	–	–	–	–	–	–	t	t	t	409913	353048	385555
	tree weight	10	–	–	–	–	–	–	t	t	t	409865	353217	383766
	tree weight	25	–	–	–	–	–	–	t	t	t	411078	352997	383973
	tree weight	50	–	–	–	–	–	–	t	t	t	410161	353258	384234
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	409391	353347	382551
	0-restart	–	–	–	–	–	–	–	t	t	t	410273	353191	381850
	monotone	10	–	–	–	1	1	1	t	t	t	107832	107616	108113
	monotone	25	–	–	–	1	–	1	t	t	t	108431	117022	108641
	monotone	50	–	–	–	1	–	–	t	t	t	108573	116911	102026
	reg forest	10	–	–	–	–	–	–	t	t	t	109680	117299	101793
	reg forest	25	–	–	–	–	–	–	t	t	t	110112	116768	101436
	reg forest	50	–	–	–	–	–	–	t	t	t	109838	116433	101483
	gap	10	–	–	–	1	1	1	t	t	t	119018	111531	114418
	gap	25	–	–	–	–	–	–	t	t	t	110481	117461	102132
	gap	50	–	–	–	–	–	–	t	t	t	110112	116592	101705
	leaf freq	10	–	–	–	1	1	1	t	t	t	113561	108779	100870
	leaf freq	25	–	–	–	1	1	1	t	t	t	97420	108083	95779
	leaf freq	50	–	–	–	1	1	1	t	t	t	108498	111138	98698
seymour1	ssg	10	–	–	–	1	1	1	t	t	t	108248	95691	110216
	ssg	25	–	–	–	1	1	1	t	t	t	108333	95893	110095
	ssg	50	–	–	–	1	1	1	t	t	t	104085	92856	108458
	tree weight	10	–	–	–	1	1	1	t	t	t	90496	94086	108457
	tree weight	25	–	–	–	1	1	1	t	t	t	93765	93211	116384
	tree weight	50	–	–	–	1	1	1	t	t	t	104281	116620	113607
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	110315	117038	102025
	0-restart	–	–	–	–	–	–	–	t	t	t	110103	116816	101808
	monotone	10	–	–	–	–	–	–	71.0	44.0	55.9	1564	912	1192
	monotone	25	–	–	–	–	–	–	70.6	44.0	55.8	1564	912	1192
	monotone	50	–	–	–	–	–	–	70.8	43.9	55.9	1564	912	1192
	reg forest	10	–	–	–	–	–	–	71.2	44.5	56.2	1564	912	1192
	reg forest	25	–	–	–	–	–	–	71.2	44.4	56.1	1564	912	1192
	reg forest	50	–	–	–	–	–	–	71.2	44.4	56.3	1564	912	1192
	gap	10	–	–	–	–	–	–	70.9	44.1	55.8	1564	912	1192
	gap	25	–	–	–	–	–	–	70.7	44.1	55.8	1564	912	1192
	gap	50	–	–	–	–	–	–	70.6	44.0	56.1	1564	912	1192
	leaf freq	10	–	–	–	–	–	–	70.8	44.1	55.9	1564	912	1192
	leaf freq	25	–	–	–	–	–	–	70.7	44.1	55.8	1564	912	1192
	leaf freq	50	–	–	–	–	–	–	70.8	44.0	55.9	1564	912	1192
sing326	ssg	10	–	–	–	–	–	–	70.8	44.3	55.9	1564	912	1192
	ssg	25	–	–	–	–	–	–	70.7	44.0	55.9	1564	912	1192
	ssg	50	–	–	–	–	–	–	70.4	44.2	55.9	1564	912	1192
	tree weight	10	–	–	–	–	–	–	70.8	44.0	55.9	1564	912	1192
	tree weight	25	–	–	–	–	–	–	70.8	44.1	56.0	1564	912	1192
	tree weight	50	–	–	–	–	–	–	70.7	44.0	55.8	1564	912	1192
	no clairvoyant	–	–	–	–	–	–	–	71.3	44.0	55.7	1564	912	1192
	0-restart	–	–	–	–	–	–	–	70.7	44.0	55.9	1564	912	1192
	monotone	10	–	–	–	1	1	1	t	t	t	1145	918	1507
	monotone	25	–	–	–	1	1	1	t	t	t	1183	918	1531
	monotone	50	–	–	–	1	1	–	t	t	t	1159	918	1659
	reg forest	10	–	–	–	1	–	1	t	t	t	1494	1243	1470
	reg forest	25	–	–	–	–	–	–	t	t	t	2216	1243	1659
	reg forest	50	–	–	–	–	–	–	t	t	t	2217	1242	1659
	gap	10	–	–	–	–	–	–	t	t	t	2212	1224	1659
	gap	25	–	–	–	–	–	–	t	t	t	2216	1207	1659

cont. on next page ...

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
sing44	gap	50	–	–	–	–	–	–	t	t	t	2216	1242	1656
	leaf freq	10	–	–	–	–	–	–	t	t	t	2206	1242	1670
	leaf freq	25	–	–	–	–	–	–	t	t	t	2216	1204	1665
	leaf freq	50	–	–	–	–	–	–	t	t	t	2215	1204	1654
	ssg	10	–	–	–	1	1	1	t	t	t	1637	1101	1470
	ssg	25	–	–	–	1	1	1	t	t	t	1498	1131	1272
	ssg	50	–	–	–	1	–	1	t	t	t	1486	1249	1420
	tree weight	10	–	–	–	1	–	–	t	t	t	1502	1242	1659
	tree weight	25	–	–	–	1	–	–	t	t	t	1894	1242	1680
	tree weight	50	–	–	–	1	–	–	t	t	t	1904	1242	1659
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	2231	1241	1699
	0-restart	–	–	–	–	–	–	–	t	t	t	2216	1172	1659
	monotone	10	–	1	–	–	–	1	t	t	t	615	1369	1103
	monotone	25	–	1	–	–	–	–	t	t	t	606	1369	1401
	monotone	50	–	1	–	–	–	–	t	t	t	626	1369	1423
	reg forest	10	–	1	–	–	–	–	t	t	t	626	1369	1465
	reg forest	25	–	1	–	–	–	–	t	t	t	633	1365	1405
	reg forest	50	–	1	–	–	–	–	t	t	t	626	1369	1424
	gap	10	–	1	–	–	–	–	t	t	t	606	1369	1421
	gap	25	–	1	–	–	–	–	t	t	t	614	1369	1424
	gap	50	–	1	–	–	–	–	t	t	t	615	1365	1405
	leaf freq	10	–	1	–	–	–	–	t	t	t	615	1367	1424
	leaf freq	25	–	1	–	–	–	–	t	t	t	617	1369	1424
	leaf freq	50	–	1	–	–	–	–	t	t	t	626	1369	1402
	ssg	10	–	1	–	–	1	1	t	t	t	634	1156	1078
	ssg	25	–	1	–	–	1	1	t	t	t	607	1307	1139
	ssg	50	–	1	–	–	–	–	t	t	t	630	1367	1421
	tree weight	10	–	1	–	–	1	1	t	t	t	610	1156	1228
	tree weight	25	–	1	–	–	–	1	t	t	t	642	1369	1070
	tree weight	50	–	1	–	–	–	1	t	t	t	630	1369	1100
	no clairvoyant	–	–	1	–	–	–	–	t	t	t	615	1369	1424
	0-restart	–	–	–	–	–	–	–	t	t	t	606	1498	1463
snp-02-004-104	monotone	10	–	–	–	–	–	–	t	t	t	8070	6964	6031
	monotone	25	–	–	–	–	–	–	t	t	t	8070	7122	6031
	monotone	50	–	–	–	–	–	–	t	t	t	8070	6964	6031
	reg forest	10	–	–	–	–	–	–	t	t	t	8070	6964	6031
	reg forest	25	–	–	–	–	–	–	t	t	t	8070	6964	6031
	reg forest	50	–	–	–	–	–	–	t	t	t	8070	6964	6031
	gap	10	–	–	–	–	–	–	t	t	t	8070	6964	6031
	gap	25	–	–	–	–	–	–	t	t	t	8070	6964	6031
	gap	50	–	–	–	–	–	–	t	t	t	8070	6964	6031
	leaf freq	10	–	–	–	–	1	1	t	t	t	8070	8041	5391
	leaf freq	25	–	–	–	–	–	–	t	t	t	8070	6964	6031
	leaf freq	50	–	–	–	–	–	–	t	t	t	8070	6964	6031
	ssg	10	–	–	–	–	1	1	t	t	t	8070	8146	7649
	ssg	25	–	–	–	–	1	–	t	t	t	8070	7455	6031
	ssg	50	–	–	–	–	–	–	t	t	t	8070	6964	6031
	tree weight	10	–	–	–	1	1	1	t	t	t	10438	8537	7317
	tree weight	25	–	–	–	1	1	1	t	t	t	10558	8537	7317
	tree weight	50	–	–	–	1	1	1	t	t	t	10508	8537	7317
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	8070	6964	6031
	0-restart	–	–	–	–	–	–	–	t	t	t	8070	6964	6031
sorrell13	monotone	10	–	–	–	–	–	–	t	t	t	4219	3572	3626
	monotone	25	–	–	–	–	–	–	t	t	t	4200	3573	3541
	monotone	50	–	–	–	–	–	–	t	t	t	4171	3585	3561
	reg forest	10	–	–	–	–	–	–	t	t	t	4236	3585	3561
	reg forest	25	–	–	–	–	–	–	t	t	t	4149	3612	3508
	reg forest	50	–	–	–	–	–	–	t	t	t	4059	3646	3514
	gap	10	–	–	–	–	–	–	t	t	t	4142	3525	3514
	gap	25	–	–	–	–	–	–	t	t	t	4172	3557	3528
	gap	50	–	–	–	–	–	–	t	t	t	4202	3529	3634
	leaf freq	10	–	–	–	–	–	–	t	t	t	4145	3572	3538
	leaf freq	25	–	–	–	–	–	–	t	t	t	4145	3585	3540
	leaf freq	50	–	–	–	–	–	–	t	t	t	4151	3506	3558
	ssg	10	–	–	–	–	–	–	t	t	t	4171	3500	3514
	ssg	25	–	–	–	–	–	–	t	t	t	4232	3586	3518
	ssg	50	–	–	–	–	–	–	t	t	t	4266	3586	3597
	tree weight	10	–	–	–	–	–	–	t	t	t	4239	3552	3528
	tree weight	25	–	–	–	–	–	–	t	t	t	4271	3624	3572
	tree weight	50	–	–	–	–	–	–	t	t	t	4241	3633	3528
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	4200	3506	3540
	0-restart	–	–	–	–	–	–	–	t	t	t	4173	3585	3616
sp150x300d	monotone	10	–	–	–	–	–	–	0.6	1.2	0.7	244	714	258
	monotone	25	–	–	–	–	–	–	0.6	1.2	0.7	244	714	258
	monotone	50	–	–	–	–	–	–	0.6	1.2	0.7	244	714	258
	reg forest	10	–	–	–	–	–	–	0.8	1.4	0.8	244	714	258

cont. on next page ...

E. Appendix E

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
sp97ar	reg forest	25	–	–	–	–	–	–	0.8	1.4	0.9	244	714	258
	reg forest	50	–	–	–	–	–	–	0.8	1.4	0.8	244	714	258
	gap	10	–	–	–	–	–	–	0.6	1.2	0.7	244	714	258
	gap	25	–	–	–	–	–	–	0.5	1.2	0.7	244	714	258
	gap	50	–	–	–	–	–	–	0.6	1.2	0.7	244	714	258
	leaf freq	10	–	–	–	–	–	–	0.6	1.2	0.7	244	714	258
	leaf freq	25	–	–	–	–	–	–	0.6	1.2	0.7	244	714	258
	leaf freq	50	–	–	–	–	–	–	0.6	1.1	0.7	244	714	258
	ssg	10	–	–	–	–	–	–	0.6	1.2	0.7	244	714	258
	ssg	25	–	–	–	–	–	–	0.6	1.2	0.7	244	714	258
	ssg	50	–	–	–	–	–	–	0.6	1.2	0.7	244	714	258
	tree weight	10	–	–	–	–	–	–	0.6	1.2	0.6	244	714	258
	tree weight	25	–	–	–	–	–	–	0.6	1.2	0.7	244	714	258
	tree weight	50	–	–	–	–	–	–	0.6	1.2	0.7	244	714	258
	no clairvoyant	–	–	–	–	–	–	–	0.7	1.2	0.6	244	714	258
	0-restart	–	–	–	–	–	–	–	0.6	1.2	0.7	244	714	258
	monotone	10	–	–	–	1	1	1	t	t	t	47257	50127	46675
	monotone	25	–	–	–	1	1	1	t	t	t	47548	50110	46529
	monotone	50	–	–	–	1	1	1	t	t	t	47261	50133	46586
	reg forest	10	–	–	–	1	1	1	t	t	t	47244	50011	46561
	reg forest	25	–	–	–	–	–	–	t	t	t	37493	38777	53849
	reg forest	50	–	–	–	–	–	–	t	t	t	37549	38767	53378
	gap	10	1	–	–	1	1	1	t	t	t	36736	39384	46723
	gap	25	–	–	–	–	–	1	t	t	t	37682	38842	46537
	gap	50	–	–	–	–	–	1	t	t	t	37549	38438	46586
	leaf freq	10	–	–	–	1	1	1	t	t	t	38701	34124	46909
	leaf freq	25	–	1	–	–	1	1	t	t	t	37304	37236	46809
	leaf freq	50	–	–	–	–	1	1	t	t	t	37571	37080	46536
	ssg	10	–	–	–	1	1	1	t	t	t	32387	43440	46748
	ssg	25	–	–	–	1	1	1	t	t	t	44656	37147	46723
	ssg	50	–	–	–	1	1	1	t	t	t	41828	37547	46737
	tree weight	10	–	–	–	1	1	1	t	t	t	47312	50436	46784
	tree weight	25	–	–	–	1	1	1	t	t	t	47013	37724	46957
	tree weight	50	–	–	–	1	1	1	t	t	t	46982	37865	46793
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	37822	38754	53884
	0-restart	–	–	–	–	–	–	–	t	t	t	37669	38721	54080
sp98ar	monotone	10	–	–	–	1	1	1	t	t	t	59727	48748	72315
	monotone	25	–	–	–	1	1	1	t	t	t	60059	48802	72267
	monotone	50	–	–	–	1	1	1	t	t	t	59441	48871	72272
	reg forest	10	–	–	–	1	1	1	t	t	t	59946	58284	62736
	reg forest	25	–	–	–	–	–	–	t	t	t	55256	63311	60557
	reg forest	50	–	–	–	–	–	–	t	t	t	55051	63117	60344
	gap	10	–	–	–	–	–	1	t	t	t	55597	62845	70411
	gap	25	–	–	–	–	–	–	t	t	t	55625	63321	60363
	gap	50	–	–	–	–	–	–	t	t	t	55609	63481	60069
	leaf freq	10	1	1	–	1	1	1	t	t	t	75218	72895	62778
	leaf freq	25	1	1	1	1	1	1	t	t	t	58359	68077	55524
	leaf freq	50	–	1	–	–	1	–	t	t	t	55669	67355	60428
	ssg	10	–	–	–	1	1	1	t	t	t	59882	59095	63637
	ssg	25	–	–	–	1	1	1	t	t	t	59841	70422	67348
	ssg	50	–	–	–	1	1	1	t	t	t	59990	65561	67164
	tree weight	10	–	–	–	1	1	1	t	t	t	59995	61312	72250
	tree weight	25	–	–	–	1	1	1	t	t	t	59972	63016	74152
	tree weight	50	–	–	–	1	1	1	t	t	t	59978	64745	67649
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	55117	63020	60292
	0-restart	–	–	–	–	–	–	–	t	t	t	55508	63070	60451
	monotone	10	–	–	–	–	–	–	t	t	t	126	129	134
	monotone	25	–	–	–	–	–	–	t	t	t	126	129	134
	monotone	50	–	–	–	–	–	–	t	t	t	126	129	134
	reg forest	10	–	–	–	–	–	–	t	t	t	126	129	134
	reg forest	25	–	–	–	–	–	–	t	t	t	126	129	134
	reg forest	50	–	–	–	–	–	–	t	t	t	126	129	134
	gap	10	–	–	–	–	–	–	t	t	t	126	129	134
	gap	25	–	–	–	–	–	–	t	t	t	126	129	134
	gap	50	–	–	–	–	–	–	t	t	t	126	129	134
	leaf freq	10	–	–	–	–	–	–	t	t	t	126	123	134
	leaf freq	25	–	–	–	–	–	–	t	t	t	126	129	135
	leaf freq	50	–	–	–	–	–	–	t	t	t	126	129	134
	ssg	10	–	–	–	–	–	–	t	t	t	126	129	134
	ssg	25	–	–	–	–	–	–	t	t	t	126	129	134
	ssg	50	–	–	–	–	–	–	t	t	t	126	129	134
	tree weight	10	–	–	–	–	–	–	t	t	t	126	129	134
	tree weight	25	–	–	–	–	–	–	t	t	t	126	129	134
	tree weight	50	–	–	–	–	–	–	t	t	t	126	129	134
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	126	129	134
	0-restart	–	–	–	–	–	–	–	t	t	t	126	128	134
splice1k1	monotone	10	–	–	–	–	–	–	t	t	t	126	129	134
	monotone	25	–	–	–	–	–	–	t	t	t	126	129	134
	monotone	50	–	–	–	–	–	–	t	t	t	126	129	134
	reg forest	10	–	–	–	–	–	–	t	t	t	126	129	134
	reg forest	25	–	–	–	–	–	–	t	t	t	126	129	134
	reg forest	50	–	–	–	–	–	–	t	t	t	126	129	134
	gap	10	–	–	–	–	–	–	t	t	t	126	129	134
	gap	25	–	–	–	–	–	–	t	t	t	126	129	134
	gap	50	–	–	–	–	–	–	t	t	t	126	129	134
	leaf freq	10	–	–	–	–	–	–	t	t	t	126	123	134
	leaf freq	25	–	–	–	–	–	–	t	t	t	126	129	135
	leaf freq	50	–	–	–	–	–	–	t	t	t	126	129	134
	ssg	10	–	–	–	–	–	–	t	t	t	126	129	134
	ssg	25	–	–	–	–	–	–	t	t	t	126	129	134
	ssg	50	–	–	–	–	–	–	t	t	t	126	129	134
	tree weight	10	–	–	–	–	–	–	t	t	t	126	129	134
	tree weight	25	–	–	–	–	–	–	t	t	t	126	129	134
	tree weight	50	–	–	–	–	–	–	t	t	t	126	129	134
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	126	129	134
	0-restart	–	–	–	–	–	–	–	t	t	t	126	128	134

cont. on next page ...

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
square41	monotone	10	–	–	–	–	–	–	t	t	t	14	15	16
	monotone	25	–	–	–	–	–	–	t	t	t	14	13	16
	monotone	50	–	–	–	–	–	–	t	t	t	14	15	16
	reg forest	10	–	–	–	–	–	–	t	t	t	14	13	16
	reg forest	25	–	–	–	–	–	–	t	t	t	14	15	16
	reg forest	50	–	–	–	–	–	–	t	t	t	15	13	16
	gap	10	–	–	–	–	–	–	t	t	t	15	15	16
	gap	25	–	–	–	–	–	–	t	t	t	14	15	16
	gap	50	–	–	–	–	–	–	t	t	t	15	15	16
	leaf freq	10	–	–	–	–	–	–	t	t	t	15	13	16
	leaf freq	25	–	–	–	–	–	–	t	t	t	15	15	16
	leaf freq	50	–	–	–	–	–	–	t	t	t	14	15	16
	ssg	10	–	–	–	–	–	–	t	t	t	15	15	16
	ssg	25	–	–	–	–	–	–	t	t	t	14	13	16
	ssg	50	–	–	–	–	–	–	t	t	t	15	15	16
	tree weight	10	–	–	–	–	–	–	t	t	t	15	13	16
	tree weight	25	–	–	–	–	–	–	t	t	t	14	17	16
	tree weight	50	–	–	–	–	–	–	t	t	t	14	15	16
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	14	15	16
	0-restart	–	–	–	–	–	–	–	t	t	t	14	15	16
square47	monotone	10	–	–	–	–	–	–	t	t	t	5	5	4
	monotone	25	–	–	–	–	–	–	t	t	t	5	5	4
	monotone	50	–	–	–	–	–	–	t	t	t	5	5	4
	reg forest	10	–	–	–	–	–	–	t	t	t	5	5	4
	reg forest	25	–	–	–	–	–	–	t	t	t	5	5	4
	reg forest	50	–	–	–	–	–	–	t	t	t	5	4	4
	gap	10	–	–	–	–	–	–	t	t	t	5	5	4
	gap	25	–	–	–	–	–	–	t	t	t	5	4	4
	gap	50	–	–	–	–	–	–	t	t	t	5	5	4
	leaf freq	10	–	–	–	–	–	–	t	t	t	5	5	4
	leaf freq	25	–	–	–	–	–	–	t	t	t	5	5	4
	leaf freq	50	–	–	–	–	–	–	t	t	t	5	5	4
	ssg	10	–	–	–	–	–	–	t	t	t	5	5	4
	ssg	25	–	–	–	–	–	–	t	t	t	5	5	4
	ssg	50	–	–	–	–	–	–	t	t	t	5	5	4
	tree weight	10	–	–	–	–	–	–	t	t	t	5	4	4
	tree weight	25	–	–	–	–	–	–	t	t	t	5	5	4
	tree weight	50	–	–	–	–	–	–	t	t	t	5	4	4
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	5	5	4
	0-restart	–	–	–	–	–	–	–	t	t	t	5	5	4
supportcase10	monotone	10	–	–	–	–	–	–	t	t	t	1	1	1
	monotone	25	–	–	–	–	–	–	t	t	t	1	1	1
	monotone	50	–	–	–	–	–	–	t	t	t	1	1	1
	reg forest	10	–	–	–	–	–	–	t	t	t	1	1	1
	reg forest	25	–	–	–	–	–	–	t	t	t	1	1	1
	reg forest	50	–	–	–	–	–	–	t	t	t	1	1	1
	gap	10	–	–	–	–	–	–	t	t	t	1	1	1
	gap	25	–	–	–	–	–	–	t	t	t	1	1	1
	gap	50	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	10	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	25	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	50	–	–	–	–	–	–	t	t	t	1	1	1
	ssg	10	–	–	–	–	–	–	t	t	t	1	1	1
	ssg	25	–	–	–	–	–	–	t	t	t	1	1	1
	ssg	50	–	–	–	–	–	–	t	t	t	1	1	1
	tree weight	10	–	–	–	–	–	–	t	t	t	1	1	1
	tree weight	25	–	–	–	–	–	–	t	t	t	1	1	1
	tree weight	50	–	–	–	–	–	–	t	t	t	1	1	1
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	1	1	1
	0-restart	–	–	–	–	–	–	–	t	t	t	1	1	1
supportcase12	monotone	10	–	–	–	–	–	1	t	t	t	910	881	1064
	monotone	25	–	–	–	–	–	1	t	t	t	911	879	1064
	monotone	50	–	–	–	–	–	1	t	t	t	906	879	1064
	reg forest	10	–	–	–	–	–	1	t	t	t	911	882	1065
	reg forest	25	–	–	–	–	–	–	t	t	t	911	877	1364
	reg forest	50	–	–	–	–	–	–	t	t	t	911	881	1367
	gap	10	–	–	–	–	–	–	t	t	t	913	879	1370
	gap	25	–	–	–	–	–	–	t	t	t	911	876	1370
	gap	50	–	–	–	–	–	–	t	t	t	910	879	1369
	leaf freq	10	–	–	–	–	–	–	t	t	t	912	877	1368
	leaf freq	25	–	–	–	–	–	–	t	t	t	914	881	1369
	leaf freq	50	–	–	–	–	–	–	t	t	t	909	881	1369
	ssg	10	–	–	–	–	–	1	t	t	t	907	881	1104
	ssg	25	–	–	–	–	–	1	t	t	t	911	886	1101
	ssg	50	–	–	–	–	–	1	t	t	t	911	882	1101
	tree weight	10	–	–	–	–	–	–	t	t	t	911	883	1369

cont. on next page ...

E. Appendix E

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
supportcase18	tree weight	25	–	–	–	–	–	–	t	t	t	913	874	1369
	tree weight	50	–	–	–	–	–	–	t	t	t	920	878	1368
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	911	879	1369
	0-restart	–	–	–	–	–	–	–	t	t	t	913	883	1368
	monotone	10	–	–	–	1	–	1	t	t	t	42861	202406	85804
	monotone	25	–	–	–	1	–	1	t	t	t	42861	204878	85787
	monotone	50	–	–	–	–	–	1	t	t	t	147821	202827	85805
	reg forest	10	–	–	–	1	–	1	t	t	t	42875	202076	85527
	reg forest	25	–	–	–	–	–	1	t	t	t	147577	202843	57435
	reg forest	50	–	–	–	–	–	–	t	t	t	147179	202057	99976
	gap	10	–	–	–	1	1	1	t	t	t	71093	137774	115174
	gap	25	–	–	–	–	1	–	t	t	t	147770	138103	100207
	gap	50	–	–	–	–	1	–	t	t	t	147910	137822	100016
	leaf freq	10	–	–	–	1	1	1	t	t	t	147863	133534	125193
	leaf freq	25	–	–	–	1	1	1	t	t	t	67014	133123	125358
	leaf freq	50	–	–	–	–	1	1	t	t	t	147580	133443	55560
	ssg	10	–	–	–	1	–	1	t	t	t	42857	203842	114260
	ssg	25	–	–	–	1	–	1	t	t	t	42861	204313	150738
	ssg	50	–	–	–	1	–	1	t	t	t	42857	202539	96439
supportcase19	tree weight	10	–	–	–	1	1	1	t	t	t	133340	195285	142392
	tree weight	25	–	–	–	1	1	1	t	t	t	62963	121191	83416
	tree weight	50	–	–	–	1	1	1	t	t	t	126458	121298	55687
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	147414	203165	100204
	0-restart	–	–	–	–	–	–	–	t	t	t	147823	204083	100255
	monotone	10	–	–	–	–	–	–	t	t	t	1	1	1
	monotone	25	–	–	–	–	–	–	t	t	t	1	1	1
	monotone	50	–	–	–	–	–	–	t	t	t	1	1	1
	reg forest	10	–	–	–	–	–	–	t	t	t	1	1	1
	reg forest	25	–	–	–	–	–	–	t	t	t	1	1	1
	reg forest	50	–	–	–	–	–	–	t	t	t	1	1	1
	gap	10	–	–	–	–	–	–	t	t	t	1	1	1
	gap	25	–	–	–	–	–	–	t	t	t	1	1	1
	gap	50	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	10	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	25	–	–	–	–	–	–	t	t	t	1	1	1
	leaf freq	50	–	–	–	–	–	–	t	t	t	1	1	1
	ssg	10	–	–	–	–	–	–	t	t	t	1	1	1
	ssg	25	–	–	–	–	–	–	t	t	t	1	1	1
	ssg	50	–	–	–	–	–	–	t	t	t	1	1	1
supportcase22	tree weight	10	–	–	–	–	–	–	t	t	t	1	1	1
	tree weight	25	–	–	–	–	–	–	t	t	t	1	1	1
	tree weight	50	–	–	–	–	–	–	t	t	t	1	1	1
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	1	1	1
	0-restart	–	–	–	–	–	–	–	t	t	t	1	1	1
	monotone	10	–	–	–	–	–	–	t	t	t	20	3	2
	monotone	25	–	–	–	–	–	–	t	t	t	20	3	2
	monotone	50	–	–	–	–	–	–	t	t	t	20	3	2
	reg forest	10	–	–	–	–	–	–	t	t	t	20	3	2
	reg forest	25	–	–	–	–	–	–	t	t	t	20	3	2
	reg forest	50	–	–	–	–	–	–	t	t	t	20	3	2
	gap	10	–	–	–	–	–	–	t	t	t	20	3	2
	gap	25	–	–	–	–	–	–	t	t	t	20	3	2
	gap	50	–	–	–	–	–	–	t	t	t	20	3	2
	leaf freq	10	–	–	–	–	–	–	t	t	t	20	3	2
	leaf freq	25	–	–	–	–	–	–	t	t	t	20	3	2
	leaf freq	50	–	–	–	–	–	–	t	t	t	20	3	2
	ssg	10	–	–	–	–	–	–	t	t	t	20	3	2
	ssg	25	–	–	–	–	–	–	t	t	t	20	3	2
	ssg	50	–	–	–	–	–	–	t	t	t	20	3	2
supportcase26	tree weight	10	–	–	–	–	–	–	t	t	t	20	3	2
	tree weight	25	–	–	–	–	–	–	t	t	t	20	3	2
	tree weight	50	–	–	–	–	–	–	t	t	t	20	3	2
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	20	3	2
	0-restart	–	–	–	–	–	–	–	t	t	t	20	3	2
	monotone	10	1	1	1	1	1	1	t	t	t	3517290	3411645	3771093
	monotone	25	1	1	1	–	1	–	t	t	t	3910297	3394819	3752831
	monotone	50	1	1	1	–	–	–	t	t	t	3915560	3142271	3757370
	reg forest	10	1	1	1	1	1	1	t	t	t	3507598	3404709	3789070
	reg forest	25	1	1	1	1	1	1	t	t	t	3530427	3405908	3780989
	reg forest	50	1	1	1	–	–	–	t	t	t	3764566	3050424	3619929
	gap	10	1	1	1	1	1	1	t	t	t	3762922	3618087	3741318
	gap	25	1	1	1	1	1	1	t	t	t	3783434	3631301	3723588
	gap	50	1	1	1	1	1	1	t	t	t	3282791	3659905	3713179
	leaf freq	10	1	1	1	1	1	1	t	t	t	3208046	3672842	3512339
	leaf freq	25	1	1	1	1	1	1	t	t	t	3565060	2805478	3213717
	leaf freq	50	1	1	1	1	1	1	t	t	t	3338547	3132113	3212803

cont. on next page ...

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
supportcase33	ssg	10	1	1	1	1	1	1	t	t	t	3797975	3410005	3810031
	ssg	25	1	1	1	1	1	1	t	t	t	3613185	3405182	3822511
	ssg	50	1	1	1	1	1	1	t	t	t	3788192	3571718	3710767
	tree weight	10	1	1	1	1	1	1	t	t	t	4030249	3416329	3787981
	tree weight	25	1	1	1	1	1	1	t	t	t	3474124	3375227	3880693
	tree weight	50	1	1	1	1	1	1	t	t	t	3486353	3469622	3903871
	no clairvoyant	–	1	1	1	–	–	–	t	t	t	3911501	3132386	3745811
	0-restart	–	–	–	–	–	–	–	t	t	t	4103253	4590357	4458099
	monotone	10	–	1	1	–	–	–	3167.7	225.0	195.7	55354	8625	5203
	monotone	25	–	1	1	–	–	–	3203.2	225.2	196.8	55354	8625	5203
	monotone	50	–	1	1	–	–	–	3208.6	225.5	196.7	55354	8625	5203
	reg forest	10	–	1	1	–	–	–	2923.0	227.4	198.0	43560	8625	5203
	reg forest	25	–	1	1	–	–	–	3223.1	226.5	197.9	55354	8625	5203
	reg forest	50	–	1	1	–	–	–	3194.0	227.3	197.8	55354	8625	5203
	gap	10	–	1	1	1	1	1	1767.4	317.0	174.8	43191	11671	3715
	gap	25	–	1	1	1	–	–	1277.5	224.3	196.8	30544	8625	5203
	gap	50	–	1	1	1	–	–	1941.9	224.8	196.6	30816	8625	5203
	leaf freq	10	–	1	1	1	–	–	2049.0	224.4	197.0	33531	8625	5203
	leaf freq	25	–	1	1	1	–	–	2843.8	225.4	195.5	51897	8625	5203
	leaf freq	50	–	1	1	–	–	–	3198.9	225.3	196.2	55354	8625	5203
	ssg	10	–	1	1	1	1	1	1176.0	252.3	192.8	22535	8447	3944
	ssg	25	–	1	1	1	1	1	1072.3	317.8	185.8	30211	10737	4584
	ssg	50	–	1	1	1	1	1	1018.0	311.2	211.8	28924	11117	5454
	tree weight	10	–	1	1	1	1	1	945.4	227.9	183.7	34850	5832	3647
	tree weight	25	–	1	1	1	1	1	1147.8	234.6	183.9	17999	6596	4236
	tree weight	50	–	1	1	1	1	1	769.9	294.8	163.6	17991	10315	3980
supportcase40	no clairvoyant	–	–	1	1	–	–	–	3203.7	224.1	196.0	55354	8625	5203
	0-restart	–	–	–	–	–	–	–	3183.7	345.3	259.6	55354	10792	5670
	monotone	10	–	–	–	–	–	–	1175.2	1300.0	1060.2	12438	14026	9931
	monotone	25	–	–	–	–	–	–	1179.2	1308.0	1058.2	12438	14026	9931
	monotone	50	–	–	–	–	–	–	1179.8	1294.9	1057.3	12438	14026	9931
	reg forest	10	–	–	–	–	–	–	1183.0	1289.8	1062.4	12438	14026	9931
	reg forest	25	–	–	–	–	–	–	1182.1	1298.8	1059.4	12438	14026	9931
	reg forest	50	–	–	–	–	–	–	1180.1	1301.1	1059.1	12438	14026	9931
	gap	10	–	–	–	1	1	1	1607.5	1710.1	1262.4	17103	19951	10414
	gap	25	–	–	–	–	–	1	1179.0	1299.2	1266.1	12438	14026	10414
	gap	50	–	–	–	–	–	–	1173.3	1297.0	1059.0	12438	14026	9931
	leaf freq	10	–	–	–	–	–	–	1178.5	1301.3	1061.5	12438	14026	9931
	leaf freq	25	–	–	–	–	–	–	1180.7	1294.0	1060.3	12438	14026	9931
	leaf freq	50	–	–	–	–	–	–	1179.2	1304.5	1062.2	12438	14026	9931
	ssg	10	–	–	–	1	1	1	1287.2	1393.6	1351.6	13582	13573	12725
	ssg	25	–	–	–	1	1	1	1013.1	1419.2	1354.6	8265	13045	12725
	ssg	50	–	–	–	1	1	1	1324.3	1428.2	1350.7	14271	13017	12725
	tree weight	10	–	–	–	1	1	1	1461.6	1300.0	1296.7	15442	11414	12037
	tree weight	25	–	–	–	1	1	1	1408.4	1495.1	1362.5	14813	14441	12429
	tree weight	50	–	–	–	1	1	1	1400.8	1578.2	1367.4	15742	16158	12009
	no clairvoyant	–	–	–	–	–	–	–	1181.2	1298.9	1060.6	12438	14026	9931
	0-restart	–	–	–	–	–	–	–	1177.7	1294.8	1061.3	12438	14026	9931
supportcase42	monotone	10	1	1	1	1	1	1	t	t	t	146221	226492	168494
	monotone	25	1	1	1	1	1	1	t	t	t	147056	226489	168012
	monotone	50	1	1	1	1	1	1	t	t	t	151499	224007	168063
	reg forest	10	1	1	1	1	1	1	t	t	t	146759	225373	168357
	reg forest	25	1	1	1	–	–	–	t	t	t	169891	144936	122122
	reg forest	50	1	1	1	–	–	–	t	t	t	170552	146162	121807
	gap	10	1	1	1	–	1	1	t	t	t	172404	225931	166923
	gap	25	1	1	1	–	1	1	t	t	t	172284	226168	168761
	gap	50	1	1	1	–	1	1	t	t	t	171170	225477	167982
	leaf freq	10	1	1	1	1	1	1	t	t	t	143238	226157	168386
	leaf freq	25	1	1	1	1	1	1	t	t	t	135843	225467	167816
	leaf freq	50	1	1	1	1	1	1	t	t	t	127460	226482	168302
	ssg	10	1	1	1	1	1	1	t	t	t	169212	224876	168516
	ssg	25	1	1	1	1	1	1	t	t	t	155426	226111	168004
	ssg	50	1	1	1	1	1	1	t	t	t	150829	226083	169227
	tree weight	10	1	1	1	1	1	1	t	t	t	140652	224830	167766
	tree weight	25	1	1	1	1	1	1	t	t	t	141726	224855	168527
	tree weight	50	1	1	1	1	1	1	t	t	t	141659	225508	168473
	no clairvoyant	–	1	1	1	–	–	–	t	t	t	172409	147001	122233
	0-restart	–	–	–	–	–	–	–	t	t	t	79714	89795	81680
supportcase6	monotone	10	–	1	1	–	–	–	t	t	t	3760	7313	5509
	monotone	25	–	1	1	–	–	–	t	t	t	3842	7295	5489
	monotone	50	–	1	1	–	–	–	t	t	t	4039	7212	5512
	reg forest	10	–	1	1	–	–	–	t	t	t	4082	7222	5530
	reg forest	25	–	1	1	–	–	–	t	t	t	4017	7388	5558
	reg forest	50	–	1	1	–	–	–	t	t	t	3825	7409	5529
	gap	10	–	1	1	–	1	–	t	t	t	3813	6807	5531
	gap	25	–	1	1	–	–	–	t	t	t	3850	7410	5460

cont. on next page ...

E. Appendix E

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
supportcase7	gap	50	–	1	1	–	–	–	t	t	t	3872	7249	5469
	leaf freq	10	–	1	1	–	–	–	t	t	t	3957	7358	5475
	leaf freq	25	–	1	1	–	–	–	t	t	t	3852	7345	5466
	leaf freq	50	–	1	1	–	–	–	t	t	t	3913	7345	5509
	ssg	10	1	1	1	1	1	1	t	t	t	4914	6598	6906
	ssg	25	–	1	1	–	1	1	t	t	t	3787	5919	7449
	ssg	50	–	1	1	–	1	1	t	t	t	3914	6222	5033
	tree weight	10	–	1	1	–	1	1	t	t	t	3914	6832	5340
	tree weight	25	–	1	1	–	1	1	t	t	t	4088	6366	5293
	tree weight	50	–	1	1	–	1	1	t	t	t	4023	6499	5352
	no clairvoyant	–	–	1	1	–	–	–	t	t	t	3634	7288	5531
	0-restart	–	–	–	–	–	–	–	t	t	t	3723	4534	4124
	monotone	10	–	–	–	–	–	–	146.7	134.7	217.4	33	9	58
	monotone	25	–	–	–	–	–	–	146.9	135.1	217.1	33	9	58
	monotone	50	–	–	–	–	–	–	146.9	134.4	216.8	33	9	58
	reg forest	10	–	–	–	–	–	–	147.2	134.8	217.0	33	9	58
	reg forest	25	–	–	–	–	–	–	147.2	134.9	217.9	33	9	58
	reg forest	50	–	–	–	–	–	–	147.3	135.2	217.2	33	9	58
	gap	10	–	–	–	–	–	–	146.9	134.6	215.8	33	9	58
	gap	25	–	–	–	–	–	–	146.9	134.6	214.8	33	9	58
	gap	50	–	–	–	–	–	–	146.9	134.5	216.9	33	9	58
	leaf freq	10	–	–	–	–	–	–	146.7	135.3	216.7	33	9	58
	leaf freq	25	–	–	–	–	–	–	147.2	134.9	216.7	33	9	58
	leaf freq	50	–	–	–	–	–	–	147.9	135.4	216.7	33	9	58
	ssg	10	–	–	–	–	–	–	147.6	135.1	217.8	33	9	58
	ssg	25	–	–	–	–	–	–	146.8	133.8	216.8	33	9	58
	ssg	50	–	–	–	–	–	–	146.6	135.8	216.8	33	9	58
	tree weight	10	–	–	–	–	–	–	147.8	135.4	216.5	33	9	58
	tree weight	25	–	–	–	–	–	–	146.3	134.7	217.1	33	9	58
	tree weight	50	–	–	–	–	–	–	146.5	135.1	217.1	33	9	58
	no clairvoyant	–	–	–	–	–	–	–	147.0	134.8	216.7	33	9	58
	0-restart	–	–	–	–	–	–	–	145.3	135.4	217.3	33	9	58
swath1	monotone	10	1	1	1	–	–	–	11.4	12.7	13.1	256	330	368
	monotone	25	1	1	1	–	–	–	11.5	12.6	13.1	256	330	368
	monotone	50	1	1	1	–	–	–	11.4	12.5	13.1	256	330	368
	reg forest	10	1	1	1	–	–	–	11.6	12.8	13.3	256	330	368
	reg forest	25	1	1	1	–	–	–	11.6	12.9	13.5	256	330	368
	reg forest	50	1	1	1	–	–	–	11.6	12.9	13.3	256	330	368
	gap	10	1	1	1	–	–	–	11.5	12.7	13.0	256	330	368
	gap	25	1	1	1	–	–	–	11.5	12.6	12.9	256	330	368
	gap	50	1	1	1	–	–	–	11.3	12.6	12.9	256	330	368
	leaf freq	10	1	1	1	–	–	–	11.4	12.7	13.1	256	330	368
	leaf freq	25	1	1	1	–	–	–	11.5	12.6	13.2	256	330	368
	leaf freq	50	1	1	1	–	–	–	11.4	12.7	13.1	256	330	368
	ssg	10	1	1	1	–	–	–	11.4	12.7	13.2	256	330	368
	ssg	25	1	1	1	–	–	–	11.4	12.6	13.1	256	330	368
	ssg	50	1	1	1	–	–	–	11.4	12.6	13.1	256	330	368
	tree weight	10	1	1	1	–	–	–	11.4	12.7	13.0	256	330	368
	tree weight	25	1	1	1	–	–	–	11.4	12.7	13.1	256	330	368
	tree weight	50	1	1	1	–	–	–	11.4	12.6	13.1	256	330	368
	no clairvoyant	–	1	1	1	–	–	–	11.5	12.7	13.4	256	330	368
	0-restart	–	–	–	–	–	–	–	13.0	14.6	13.0	271	279	241
swath3	monotone	10	1	1	–	–	–	–	510.8	285.3	426.9	77119	54933	63346
	monotone	25	1	1	–	–	–	–	512.7	285.4	426.1	77119	54933	63346
	monotone	50	1	1	–	–	–	–	512.9	285.3	429.4	77119	54933	63346
	reg forest	10	1	1	–	–	1	–	520.1	243.7	429.5	77119	45580	63346
	reg forest	25	1	1	–	–	–	–	515.9	290.7	431.4	77119	54933	63346
	reg forest	50	1	1	–	–	–	–	516.8	289.5	429.4	77119	54933	63346
	gap	10	2	1	–	1	1	1	299.9	185.8	175.8	50731	31645	24739
	gap	25	2	1	–	1	1	1	218.6	274.0	257.7	28247	51467	45290
	gap	50	2	1	–	1	1	1	179.8	231.2	186.1	18302	42955	29578
	leaf freq	10	2	1	–	1	1	1	277.8	249.9	288.0	41251	40939	40108
	leaf freq	25	2	1	1	1	1	1	318.7	212.0	302.7	39435	31681	42244
	leaf freq	50	1	1	–	–	–	–	513.1	285.7	425.4	77119	54933	63346
	ssg	10	2	1	1	1	1	1	187.9	252.6	256.8	27250	48785	38622
	ssg	25	2	1	–	1	1	1	92.9	504.6	218.1	8194	105107	36711
	ssg	50	2	1	1	1	1	1	207.7	303.7	312.1	31503	55829	44942
	tree weight	10	2	1	1	1	1	1	193.4	500.0	321.9	26415	101203	54189
	tree weight	25	2	1	1	1	1	1	281.4	184.1	160.2	50772	30348	18472
	tree weight	50	2	1	1	1	1	1	348.1	242.4	201.8	56187	45036	27106
	no clairvoyant	–	1	1	–	–	–	–	512.1	285.1	426.3	77119	54933	63346
	0-restart	–	–	–	–	–	–	–	387.6	192.3	426.0	58894	28724	63346
tbfp-network	monotone	10	–	–	–	–	–	–	799.2	946.9	3391.2	65	213	1983
	monotone	25	–	–	–	–	–	–	800.0	951.1	3381.0	65	213	1983
	monotone	50	–	–	–	–	–	–	798.5	950.8	3376.6	65	213	1983
	reg forest	10	–	–	–	–	–	–	799.9	955.0	3390.8	65	213	1983

cont. on next page ...

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
thor50dday	reg forest	25	–	–	–	–	–	–	799.2	952.5	3393.1	65	213	1983
	reg forest	50	–	–	–	–	–	–	801.2	951.6	3360.9	65	213	1983
	gap	10	–	–	–	–	–	–	798.9	951.5	3378.1	65	213	1983
	gap	25	–	–	–	–	–	–	800.3	946.7	3385.2	65	213	1983
	gap	50	–	–	–	–	–	–	795.3	948.9	3384.5	65	213	1983
	leaf freq	10	–	–	–	–	–	–	796.8	950.9	3382.5	65	213	1983
	leaf freq	25	–	–	–	–	–	–	800.2	946.9	3380.1	65	213	1983
	leaf freq	50	–	–	–	–	–	–	804.4	955.5	3377.6	65	213	1983
	ssg	10	–	–	2	–	–	1	801.8	949.4	t	65	213	1979
	ssg	25	–	–	2	–	–	1	800.0	950.2	t	65	213	1976
	ssg	50	–	–	2	–	–	1	797.4	953.5	t	65	213	1968
	tree weight	10	–	–	–	–	–	–	799.7	951.7	3374.7	65	213	1983
	tree weight	25	–	–	–	–	–	–	797.1	949.9	3376.2	65	213	1983
	tree weight	50	–	–	–	–	–	–	801.9	947.4	3376.0	65	213	1983
	no clairvoyant	–	–	–	–	–	–	–	798.0	948.9	3471.2	65	213	1983
	0-restart	–	–	–	–	–	–	–	803.0	949.4	3384.0	65	213	1983
	monotone	10	1	–	–	–	–	–	t	t	t	2	1	1
	monotone	25	1	–	–	–	–	–	t	t	t	2	1	1
	monotone	50	1	–	–	–	–	–	t	t	t	2	1	1
	reg forest	10	1	–	–	–	–	–	t	t	t	2	1	1
	reg forest	25	1	–	–	–	–	–	t	t	t	2	1	1
	reg forest	50	1	–	–	–	–	–	t	t	t	2	1	1
	gap	10	1	–	–	–	–	–	t	t	t	2	1	1
	gap	25	1	–	–	–	–	–	t	t	t	2	1	1
	gap	50	1	–	–	–	–	–	t	t	t	2	1	1
	leaf freq	10	1	–	–	–	–	–	t	t	t	2	1	1
	leaf freq	25	1	–	–	–	–	–	t	t	t	2	1	1
	leaf freq	50	1	–	–	–	–	–	t	t	t	2	1	1
	ssg	10	1	–	–	–	–	–	t	t	t	2	1	1
	ssg	25	1	–	–	–	–	–	t	t	t	2	1	1
	ssg	50	1	–	–	–	–	–	t	t	t	2	1	1
	tree weight	10	1	–	–	–	–	–	t	t	t	2	1	1
	tree weight	25	1	–	–	–	–	–	t	t	t	2	1	1
	tree weight	50	1	–	–	–	–	–	t	t	t	2	1	1
	no clairvoyant	–	1	–	–	–	–	–	t	t	t	2	1	1
	0-restart	–	–	–	–	–	–	–	t	t	t	3	1	1
timtab1	monotone	10	–	–	–	–	1	–	54.9	59.9	63.6	39949	39680	57229
	monotone	25	–	–	–	–	–	–	55.1	41.5	63.7	39949	31062	57229
	monotone	50	–	–	–	–	–	–	54.9	41.3	63.8	39949	31062	57229
	reg forest	10	–	–	–	1	1	1	60.8	71.4	62.6	42269	48832	51707
	reg forest	25	–	–	–	–	1	1	57.8	55.3	62.3	39949	37753	51707
	reg forest	50	–	–	–	–	–	–	57.4	44.0	67.5	39949	31062	57229
	gap	10	–	–	–	1	1	1	61.5	70.3	76.3	40509	48832	47180
	gap	25	–	–	–	–	1	1	62.6	70.2	74.6	39751	40257	45932
	gap	50	–	–	–	1	1	–	58.1	54.7	63.6	39327	40642	57229
	leaf freq	10	–	–	–	1	–	1	81.4	41.5	81.5	46010	31062	56692
	leaf freq	25	–	–	–	–	–	1	54.9	41.4	77.1	39949	31062	54878
	leaf freq	50	–	–	–	–	–	1	55.4	41.5	69.1	39949	31062	46526
	ssg	10	–	–	–	1	1	1	53.8	47.1	74.7	33154	32811	45882
	ssg	25	–	–	–	1	1	1	67.6	46.8	66.7	46482	29193	41136
	ssg	50	–	–	–	1	1	1	52.0	68.6	85.5	33508	50541	53398
	tree weight	10	–	–	–	1	1	1	68.9	68.7	89.2	36280	42237	56083
	tree weight	25	–	–	–	1	1	1	51.8	80.7	69.7	38183	42022	44167
	tree weight	50	–	–	–	1	1	1	68.8	73.1	81.4	54697	33874	57307
	no clairvoyant	–	–	–	–	–	–	–	56.0	41.3	63.8	39949	31062	57229
	0-restart	–	–	–	–	–	–	–	54.8	41.5	63.8	39949	31062	57229
tri2-30	monotone	10	–	–	–	1	1	1	738.7	926.6	656.1	404011	533994	343212
	monotone	25	–	–	–	–	1	1	739.0	924.6	656.4	404011	533994	343212
	monotone	50	–	–	–	–	–	1	631.9	927.5	657.5	301862	533994	343212
	reg forest	10	–	–	–	1	1	1	743.2	932.1	659.2	404011	533994	343212
	reg forest	25	–	–	–	–	–	–	645.0	1067.6	718.1	301862	629526	361096
	reg forest	50	–	–	–	–	–	–	645.4	1075.2	719.5	301862	629526	361096
	gap	10	–	–	–	–	–	–	633.7	1048.9	699.9	301862	629526	361096
	gap	25	–	–	–	–	–	–	634.0	1046.7	703.9	301862	629526	361096
	gap	50	–	–	–	–	–	–	632.1	1052.7	703.8	301862	629526	361096
	leaf freq	10	–	–	–	1	1	1	694.9	963.2	652.3	367376	493795	323371
	leaf freq	25	–	–	–	1	1	1	729.4	799.7	738.9	354449	405802	354051
	leaf freq	50	–	–	–	1	1	1	766.9	800.0	666.2	383229	405802	314678
	ssg	10	–	–	–	1	1	1	740.1	928.3	655.5	404011	533994	343212
	ssg	25	–	–	–	1	1	1	739.4	990.9	659.4	404011	562046	343212
	ssg	50	–	–	–	1	1	1	740.9	1022.9	655.4	404011	531029	343212
	tree weight	10	–	–	–	1	1	1	853.2	1025.9	769.4	470435	571017	375522
	tree weight	25	–	–	–	1	1	1	684.5	959.2	685.7	346737	509057	345059
	tree weight	50	–	–	–	1	1	1	721.3	958.1	633.1	372347	519611	298443
	no clairvoyant	–	–	–	–	–	–	–	636.1	1048.7	704.6	301862	629526	361096
	0-restart	–	–	–	–	–	–	–	632.1	1050.4	705.8	301862	629526	361096

cont. on next page ...

E. Appendix E

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
traininstance2	monotone	10	–	–	–	–	–	1	t	t	t	209213	303924	319546
	monotone	25	–	–	–	–	–	1	t	t	t	209865	303191	322730
	monotone	50	–	–	–	–	–	–	t	t	t	210035	301778	192908
	reg forest	10	–	–	–	1	1	1	t	t	t	315568	242763	316176
	reg forest	25	–	–	–	–	–	–	t	t	t	208437	301260	188821
	reg forest	50	–	–	–	–	–	–	t	t	t	208925	302999	188746
	gap	10	–	–	–	–	–	1	t	t	t	209038	301261	236756
	gap	25	–	–	–	–	–	1	t	t	t	208851	302938	236744
	gap	50	–	–	–	–	–	1	t	t	t	209909	303333	233443
	leaf freq	10	–	–	–	1	1	1	t	t	t	270731	248179	240512
	leaf freq	25	–	–	–	1	1	1	t	t	t	337513	211034	266796
	leaf freq	50	–	–	–	1	1	1	t	t	t	261120	186195	181039
	ssg	10	–	–	–	1	1	1	t	t	t	272045	181456	376391
	ssg	25	–	–	–	1	1	1	t	t	t	272478	215210	482595
	ssg	50	–	–	–	1	1	1	t	t	t	225847	304707	369221
	tree weight	10	–	–	–	1	1	1	t	t	t	308137	242899	313082
	tree weight	25	–	–	–	1	1	1	t	t	t	308098	227499	195711
	tree weight	50	–	–	–	1	1	1	t	t	t	297711	173748	332634
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	209110	303241	192114
	0-restart	–	–	–	–	–	–	–	t	t	t	210600	303199	192659
traininstance6	monotone	10	–	–	–	–	–	–	t	t	t	612082	519206	818057
	monotone	25	–	–	–	–	–	–	t	t	t	612628	520766	819641
	monotone	50	–	–	–	–	–	–	t	t	t	612908	527201	815943
	reg forest	10	–	–	–	–	–	–	t	t	t	615572	515710	809645
	reg forest	25	–	–	–	–	–	–	t	t	t	611436	515251	811275
	reg forest	50	–	–	–	–	–	–	t	t	t	607078	517118	812841
	gap	10	–	–	–	1	1	1	t	t	t	526787	506207	810239
	gap	25	–	–	–	1	1	1	t	t	t	528270	506646	838584
	gap	50	–	–	–	1	1	1	t	t	t	457119	504932	751127
	leaf freq	10	–	–	–	1	1	1	t	t	t	581547	538376	499344
	leaf freq	25	–	–	–	1	1	1	t	t	t	514150	515962	763128
	leaf freq	50	–	–	–	1	–	–	t	t	t	507204	522666	855500
	ssg	10	–	–	–	1	1	1	t	t	t	729706	760112	749456
	ssg	25	–	–	–	1	1	1	t	t	t	572545	706194	1000778
	ssg	50	–	–	–	1	1	1	t	t	t	507549	543561	690190
	tree weight	10	–	–	–	1	1	1	t	t	t	597493	1101522	659611
	tree weight	25	–	–	–	1	1	1	t	t	t	599705	1103126	712760
	tree weight	50	–	–	–	1	1	1	t	t	t	597174	1100712	519543
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	612678	519020	816841
	0-restart	–	–	–	–	–	–	–	t	t	t	613451	515006	815690
trento1	monotone	10	–	–	–	1	1	1	t	t	t	15857	17055	14789
	monotone	25	–	–	–	1	1	1	t	t	t	15755	17417	14788
	monotone	50	–	–	–	1	1	1	t	t	t	16207	17416	14925
	reg forest	10	–	–	–	1	1	1	t	t	t	16044	17393	17259
	reg forest	25	–	–	–	1	–	–	t	t	t	16406	16809	19704
	reg forest	50	–	–	–	–	–	–	t	t	t	27935	16825	19651
	gap	10	–	–	–	1	1	1	t	t	t	18553	14339	17506
	gap	25	–	–	–	–	1	–	t	t	t	27931	15015	19935
	gap	50	–	–	–	–	1	–	t	t	t	27930	17714	19741
	leaf freq	10	1	–	–	1	1	–	t	t	t	30388	13544	19659
	leaf freq	25	–	–	–	–	–	–	t	t	t	28224	16831	19744
	leaf freq	50	–	–	–	–	–	–	t	t	t	27995	16626	19659
	ssg	10	–	–	–	1	1	1	t	t	t	15960	9579	17311
	ssg	25	–	–	–	1	1	1	t	t	t	15948	9606	17339
	ssg	50	–	–	–	1	1	1	t	t	t	15425	13109	17314
	tree weight	10	–	–	–	1	1	1	t	t	t	20382	16695	16574
	tree weight	25	–	–	–	1	1	1	t	t	t	13245	16926	16397
	tree weight	50	–	–	–	1	1	1	t	t	t	19707	23786	11386
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	27845	16808	19741
	0-restart	–	–	–	–	–	–	–	t	t	t	27927	16826	19882
triptim1	monotone	10	2	1	1	–	–	–	531.3	1866.6	335.2	3	22	2
	monotone	25	2	1	1	–	–	–	535.8	1864.2	334.9	3	22	2
	monotone	50	2	1	1	–	–	–	536.0	1866.9	334.6	3	22	2
	reg forest	10	2	1	1	–	–	–	533.3	1868.3	334.6	3	22	2
	reg forest	25	2	1	1	–	–	–	535.9	1868.4	335.1	3	22	2
	reg forest	50	2	1	1	–	–	–	532.6	1875.1	336.4	3	22	2
	gap	10	2	1	1	–	–	–	533.7	1852.3	336.5	3	22	2
	gap	25	2	1	1	–	–	–	537.1	1869.5	330.8	3	22	2
	gap	50	2	1	1	–	–	–	533.5	1874.5	334.6	3	22	2
	leaf freq	10	2	1	1	–	–	–	536.4	1868.9	335.9	3	22	2
	leaf freq	25	2	1	1	–	–	–	537.7	1872.8	334.8	3	22	2
	leaf freq	50	2	1	1	–	–	–	538.8	1860.2	336.9	3	22	2
	ssg	10	2	1	1	–	–	–	537.1	1867.2	335.1	3	22	2
	ssg	25	2	1	1	–	–	–	536.2	1869.2	335.4	3	22	2
	ssg	50	2	1	1	–	–	–	536.2	1882.8	337.9	3	22	2
	tree weight	10	2	1	1	–	–	–	533.6	1862.6	336.1	3	22	2

cont. on next page ...

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
uccase12	tree weight	25	2	1	1	–	–	–	534.4	1876.4	334.8	3	22	2
	tree weight	50	2	1	1	–	–	–	533.8	1868.6	336.5	3	22	2
	no clairvoyant	–	2	1	1	–	–	–	537.1	1870.9	331.4	3	22	2
	0-restart	–	–	–	–	–	–	–	402.7	849.4	1770.8	1	9	85
	monotone	10	1	1	1	–	–	–	t	t	t	33487	37633	37084
	monotone	25	1	1	1	–	–	–	t	t	t	34219	37527	37098
	monotone	50	1	1	1	–	–	–	t	t	t	33593	37786	37422
	reg forest	10	1	1	1	–	–	–	t	t	t	33336	37883	37019
	reg forest	25	1	1	1	–	–	–	t	t	t	33661	37555	37203
	reg forest	50	1	1	1	–	–	–	t	t	t	33593	38175	37042
	gap	10	1	1	1	–	–	–	t	t	t	33673	37839	36905
	gap	25	1	1	1	–	–	–	t	t	t	33541	37743	36770
	gap	50	1	1	1	–	–	–	t	t	t	33716	38178	37084
	leaf freq	10	1	1	1	–	–	–	t	t	t	33317	37853	37084
	leaf freq	25	1	1	1	–	–	–	t	t	t	33863	37791	36993
	leaf freq	50	1	1	1	–	–	–	t	t	t	33595	37669	36965
	ssg	10	1	1	1	–	–	–	t	t	t	33519	37636	37084
	ssg	25	1	1	1	–	–	–	t	t	t	33516	37909	36729
	ssg	50	1	1	1	–	–	–	t	t	t	33593	38084	37150
	tree weight	10	1	1	1	–	–	–	t	t	t	33506	38132	37203
uccase9	tree weight	25	1	1	1	–	–	–	t	t	t	33826	37826	37148
	tree weight	50	1	1	1	–	–	–	t	t	t	33489	37899	37203
	no clairvoyant	–	1	1	1	–	–	–	t	t	t	33355	37978	36852
	0-restart	–	–	–	–	–	–	–	t	t	t	20281	17261	18411
	monotone	10	–	–	–	–	–	–	t	t	t	485	1667	1087
	monotone	25	–	–	–	–	–	–	t	t	t	470	1655	1087
	monotone	50	–	–	–	–	–	–	t	t	t	470	1663	1087
	reg forest	10	–	–	–	–	–	–	t	t	t	482	1654	1120
	reg forest	25	–	–	–	–	–	–	t	t	t	482	1660	1087
	reg forest	50	–	–	–	–	–	–	t	t	t	485	1660	1087
	gap	10	–	–	–	–	–	–	t	t	t	480	1658	1090
	gap	25	–	–	–	–	–	–	t	t	t	466	1652	1087
	gap	50	–	–	–	–	–	–	t	t	t	485	1660	1087
	leaf freq	10	–	–	–	–	–	–	t	t	t	485	1663	1077
	leaf freq	25	–	–	–	–	–	–	t	t	t	474	1652	1087
	leaf freq	50	–	–	–	–	–	–	t	t	t	466	1643	1087
	ssg	10	–	–	–	–	–	–	t	t	t	470	1655	1087
	ssg	25	–	–	–	–	–	–	t	t	t	485	1650	1090
	ssg	50	–	–	–	–	–	–	t	t	t	485	1665	1087
	tree weight	10	–	–	–	–	–	–	t	t	t	475	1641	1087
uct-subprob	tree weight	25	–	–	–	–	–	–	t	t	t	485	1665	1087
	tree weight	50	–	–	–	–	–	–	t	t	t	485	1665	1123
	no clairvoyant	–	–	–	–	–	–	–	t	t	t	485	1665	1087
	0-restart	–	–	–	–	–	–	–	t	t	t	470	1663	1120
	monotone	10	–	–	–	–	–	–	1650.5	1574.7	1153.2	59352	49996	34274
	monotone	25	–	–	–	–	–	–	1658.0	1578.4	1144.9	59352	49996	34274
	monotone	50	–	–	–	–	–	–	1656.0	1576.8	1149.0	59352	49996	34274
	reg forest	10	–	–	–	1	1	1	1730.0	1740.7	1120.3	61018	62470	35175
	reg forest	25	–	–	–	–	–	–	1655.1	1588.3	1151.0	59352	49996	34274
	reg forest	50	–	–	–	–	–	–	1658.7	1581.2	1154.5	59352	49996	34274
	gap	10	–	–	–	1	1	1	1766.4	2268.8	1544.6	61011	76763	41131
	gap	25	–	–	–	1	1	1	2249.0	1884.6	1459.9	70785	61147	39398
	gap	50	–	–	–	–	–	–	1661.1	1574.1	1147.0	59352	49996	34274
	leaf freq	10	–	–	–	1	1	1	2095.0	1974.0	1652.1	67670	61267	42791
	leaf freq	25	–	–	–	1	1	–	2109.2	2384.0	1147.7	67670	79957	34274
	leaf freq	50	–	–	–	–	–	–	1648.8	1574.9	1143.9	59352	49996	34274
	ssg	10	–	–	–	1	1	1	2618.2	1757.9	1131.9	97700	63432	34056
	ssg	25	–	–	–	1	1	1	2634.5	1638.7	1713.3	97700	51106	59847
	ssg	50	–	–	–	1	1	1	1834.1	2310.1	1300.8	58248	77734	35952
	tree weight	10	–	–	–	1	1	1	2198.3	2209.4	1089.5	74201	85874	32083
unitcal_7	tree weight	25	–	–	–	1	1	1	1668.3	1274.7	1072.6	59439	41621	28377
	tree weight	50	–	–	–	1	1	1	1891.9	1822.1	1117.3	66627	59593	32591
	no clairvoyant	–	–	–	–	–	–	–	1648.7	1580.1	1155.6	59352	49996	34274
	0-restart	–	–	–	–	–	–	–	1655.1	1567.4	1153.6	59352	49996	34274
	monotone	10	5	6	5	–	–	–	224.8	249.2	199.9	16	9	6
	monotone	25	5	6	5	–	–	–	225.2	249.1	200.7	16	9	6
	monotone	50	5	6	5	–	–	–	224.5	248.3	200.8	16	9	6
	reg forest	10	5	6	5	–	–	–	226.2	249.9	200.6	16	9	6
	reg forest	25	5	6	5	–	–	–	225.8	250.1	200.7	16	9	6
	reg forest	50	5	6	5	–	–	–	225.4	248.6	200.7	16	9	6
	gap	10	5	6	5	–	–	–	225.7	248.6	200.0	16	9	6
	gap	25	5	6	5	–	–	–	225.1	248.3	200.1	16	9	6
	gap	50	5	6	5	–	–	–	225.7	248.7	199.0	16	9	6
	leaf freq	10	5	6	5	–	–	–	225.0	249.2	199.5	16	9	6
	leaf freq	25	5	6	5	–	–	–	225.7	249.4	200.5	16	9	6
	leaf freq	50	5	6	5	–	–	–	225.7	249.1	199.7	16	9	6

cont. on next page ...

E. Appendix E

Table E.1 cont.

mip	settings	ϕ^{clair}	root			tree			time			nodes		
			0	1	2	0	1	2	0	1	2	0	1	2
var--m6j6	ssg	10	5	6	5	–	–	–	225.6	248.9	201.8	16	9	6
	ssg	25	5	6	5	–	–	–	225.6	249.0	201.1	16	9	6
	ssg	50	5	6	5	–	–	–	224.2	248.5	201.0	16	9	6
	tree weight	10	5	6	5	–	–	–	225.7	248.8	200.2	16	9	6
	tree weight	25	5	6	5	–	–	–	225.2	248.5	200.4	16	9	6
	tree weight	50	5	6	5	–	–	–	225.2	249.8	200.5	16	9	6
	no clairvoyant	–	5	6	5	–	–	–	226.1	248.2	199.8	16	9	6
	0-restart	–	–	–	–	–	–	–	267.9	239.8	198.2	143	145	172
	monotone	10	1	1	1	1	–	–	t	t	t	260853	241264	245898
	monotone	25	1	1	1	–	–	–	t	t	t	262641	240492	245522
	monotone	50	1	1	1	–	–	–	t	t	t	261522	240584	244762
	reg forest	10	1	1	1	1	1	1	t	t	t	259877	245806	235551
	reg forest	25	1	1	1	–	–	–	t	t	t	259926	239588	243915
	reg forest	50	1	1	1	–	–	–	t	t	t	260449	238037	243312
	gap	10	1	1	1	1	–	1	t	t	t	264754	241363	229632
	gap	25	1	1	1	–	–	–	t	t	t	261195	240896	245927
	gap	50	1	1	1	–	–	–	t	t	t	262217	240712	245423
	leaf freq	10	1	1	1	1	1	1	t	t	t	198189	231823	225644
	leaf freq	25	1	1	1	1	1	1	t	t	t	35412	220766	219677
	leaf freq	50	1	1	1	1	1	1	t	t	t	94932	83471	221107
	ssg	10	1	1	1	1	1	1	t	t	t	151946	237598	241972
	ssg	25	1	1	1	1	1	1	t	t	t	174878	251171	231325
	ssg	50	1	1	1	1	1	1	t	t	t	152019	220259	226251
	tree weight	10	1	1	1	1	1	1	t	t	t	243046	232755	244314
	tree weight	25	1	1	1	1	1	1	t	t	t	227102	231695	220068
	tree weight	50	1	1	1	1	1	1	t	t	t	214684	232434	256259
wachplan	no clairvoyant	–	1	1	1	–	–	–	t	t	t	261975	241971	245564
	0-restart	–	–	–	–	–	–	–	t	t	t	130787	141630	148556
	monotone	10	–	–	–	–	–	–	2793.2	2338.4	2218.2	208422	162732	165428
	monotone	25	–	–	–	–	–	–	2802.9	2332.1	2216.9	208422	162732	165428
	monotone	50	–	–	–	–	–	–	2791.5	2340.3	2217.0	208422	162732	165428
	reg forest	10	–	–	–	–	–	–	2806.9	2360.7	2237.7	208422	162732	165428
	reg forest	25	–	–	–	–	–	–	2806.8	2351.0	2244.0	208422	162732	165428
	reg forest	50	–	–	–	–	–	–	2809.6	2356.1	2240.8	208422	162732	165428
	gap	10	–	–	–	–	–	–	2786.1	2330.6	2222.4	208422	162732	165428
	gap	25	–	–	–	–	–	–	2791.6	2335.0	2209.7	208422	162732	165428
	gap	50	–	–	–	–	–	–	2794.9	2335.0	2218.0	208422	162732	165428
	leaf freq	10	–	–	–	–	–	–	2792.2	2334.7	2213.4	208422	162732	165428
	leaf freq	25	–	–	–	–	–	–	2785.7	2341.4	2226.6	208422	162732	165428
	leaf freq	50	–	–	–	–	–	–	2785.3	2334.1	2214.2	208422	162732	165428
	ssg	10	–	–	–	–	–	–	2790.0	2333.9	2213.0	208422	162732	165428
	ssg	25	–	–	–	–	–	–	2788.0	2346.1	2215.2	208422	162732	165428
	ssg	50	–	–	–	–	–	–	2792.8	2336.1	2210.9	208422	162732	165428
	tree weight	10	–	–	–	–	–	–	2788.5	2336.6	2217.1	208422	162732	165428
	tree weight	25	–	–	–	–	–	–	2794.9	2331.8	2221.2	208422	162732	165428
	tree weight	50	–	–	–	–	–	–	2797.5	2330.5	2214.5	208422	162732	165428
	no clairvoyant	–	–	–	–	–	–	–	2812.9	2331.7	2217.6	208422	162732	165428
	0-restart	–	–	–	–	–	–	–	2783.1	2343.1	2223.4	208422	162732	165428