Foundations of computing Volume 12

Vincent Froese

Fine-Grained Complexity Analysis of Some Combinatorial Data Science Problems

Universitätsverlag der TU Berlin

Technische Universität Berlin

Vincent Froese

Fine-Grained Complexity Analysis of Some Combinatorial Data Science Problems

The scientific series *Foundations of computing* of the Technische Universität Berlin is edited by:
Prof. Dr. Rolf Niedermeier,
Prof. Dr. Uwe Nestmann,
Prof. Dr. Stephan Kreutzer.

Foundations of computing | 12

Vincent Froese

Fine-Grained Complexity Analysis of Some Combinatorial Data Science Problems

Universitätsverlag der TU Berlin

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available on the internet at http://dnb.dnb.de.

Universitätsverlag der TU Berlin, 2018
http://www.verlag.tu-berlin.de

Fasanenstr. 88, 10623 Berlin
Tel.: +49 (0)30 314 76131 / Fax: -76133
E-Mail: publikationen@ub.tu-berlin.de

Zugl.: Berlin, Techn. Univ., Diss., 2018
Gutachter: Prof. Dr. Rolf Niedermeier
Gutachter: Prof. Dr. Tobias Friedrich
Gutachter: Prof. Dr. Marek Cygan
Die Arbeit wurde am 28. Mai 2018 an der Fakultät IV unter Vorsitz von Prof. Dr. Uwe Nestmann erfolgreich verteidigt.

Print: docupoint GmbH
Layout/Typesetting: Vincent Froese

ISBN 978-3-7983-3003-0 (print)
ISBN 978-3-7983-3004-7 (online)

ISSN 2199-5249 (print)
ISSN 2199-5257 (online)

Published online on the institutional repository of the Technische Universität Berlin:
DOI 10.14279/depositonce-7123
http://dx.doi.org/10.14279/depositonce-7123

Zusammenfassung

Die vorliegende Dissertation befasst sich mit der Analyse der Berechnungskomplexität von NP-schweren Problemen aus dem Bereich Data Science. Für die meisten der hier betrachteten Probleme wurde die Berechnungskomplexität bisher nicht sehr detailliert untersucht. Wir führen daher eine genaue Komplexitätsanalyse dieser Probleme durch, mit dem Ziel, effizient lösbare Spezialfälle zu identifizieren. Zu diesem Zweck nehmen wir eine parametrisierte Perspektive ein, bei der wir bestimmte Parameter definieren, welche Eigenschaften einer konkreten Probleminstanz beschreiben, die es ermöglichen, diese Instanz effizient zu lösen. Wir entwickeln dabei spezielle Algorithmen, deren Laufzeit für konstante Parameterwerte polynomiell ist. Darüber hinaus untersuchen wir, in welchen Fällen die Probleme selbst bei kleinen Parameterwerten berechnungsschwer bleiben. Somit skizzieren wir die Grenze zwischen schweren und handhabbaren Probleminstanzen, um ein besseres Verständnis der Berechnungskomplexität für die folgenden praktisch motivierten Probleme zu erlangen.

Beim General Position Subset Selection Problem ist eine Menge von Punkten in der Ebene gegeben und das Ziel ist es, möglichst viele Punkte in allgemeiner Lage davon auszuwählen. Punktmengen in allgemeiner Lage sind in der Geometrie gut untersucht und spielen unter anderem im Bereich der Datenvisualisierung eine Rolle. Wir beweisen etliche Härteergebnisse und zeigen, wie das Problem mittels Polynomzeitdatenreduktion gelöst werden kann, falls die Anzahl gesuchter Punkte in allgemeiner Lage sehr klein oder sehr groß ist.

Distinct Vectors ist das Problem, möglichst wenige Spalten einer gegebenen Matrix so auszuwählen, dass in der verbleibenden Submatrix alle Zeilen paarweise verschieden sind. Dieses Problem hat Anwendungen im Bereich der kombinatorischen Merkmalsselektion. Wir betrachten Kombinationen aus maximalem und minimalem paarweisen Hamming-Abstand der Zeilenvektoren und beweisen eine Komplexitätsdichotomie für Binärmatrizen, welche die NP-schweren von den polynomzeitlösbaren Kombinationen unterscheidet.

Co-Clustering ist ein bekanntes Matrix-Clustering-Problem aus dem Gebiet Data-Mining. Ziel ist es, eine Matrix in möglichst homogene Submatrizen zu partitionieren. Wir führen eine umfangreiche multivariate Komplexitätsana-

lyse durch, in der wir zahlreiche NP-schwere, sowie polynomzeitlösbare und festparameterhandhabbare Spezialfälle identifizieren.

Bei F-free Editing handelt es sich um ein generisches Graphmodifikationsproblem, bei dem ein Graph durch möglichst wenige Kantenmodifikationen so abgeändert werden soll, dass er keinen induzierten Teilgraphen mehr enthält, der isomorph zum Graphen F ist. Wir betrachten die drei folgenden Spezialfälle dieses Problems: Das Graph-Clustering-Problem Cluster Editing aus dem Bereich des Maschinellen Lernens, das Triangle Deletion Problem aus der Netzwerk-Cluster-Analyse und das Problem Feedback Arc Set in Tournaments mit Anwendungen bei der Aggregation von Rankings. Wir betrachten eine neue Parametrisierung mittels der Differenz zwischen der maximalen Anzahl Kantenmodifikationen und einer unteren Schranke, welche durch eine Menge von induzierten Teilgraphen bestimmt ist. Wir zeigen Festparameterhandhabbarkeit der drei obigen Probleme bezüglich dieses Parameters. Darüber hinaus beweisen wir etliche NP-Schwereergebnisse für andere Problemvarianten von F-free Editing bei konstantem Parameterwert.

DTW-Mean ist das Problem, eine Durchschnittszeitreihe bezüglich der Dynamic-Time-Warping-Distanz für eine Menge gegebener Zeitreihen zu berechnen. Hierbei handelt es sich um ein grundlegendes Problem der Zeitreihenanalyse, dessen Komplexität bisher unbekannt ist. Wir entwickeln einen exakten Exponentialzeitalgorithmus für DTW-Mean und zeigen, dass der Spezialfall binärer Zeitreihen in polynomieller Zeit lösbar ist.

Abstract

This thesis is concerned with analyzing the computational complexity of NP-hard problems related to data science. For most of the problems considered in this thesis, the computational complexity has not been intensively studied before. We focus on the complexity of computing exact problem solutions and conduct a detailed analysis identifying tractable special cases. To this end, we adopt a parameterized viewpoint in which we spot several parameters which describe properties of a specific problem instance that allow to solve the instance efficiently. We develop specialized algorithms whose running times are polynomial if the corresponding parameter value is constant. We also investigate in which cases the problems remain intractable even for small parameter values. We thereby chart the border between tractability and intractability for some practically motivated problems which yields a better understanding of their computational complexity. In particular, we consider the following problems.

General Position Subset Selection is the problem to select a maximum number of points in general position from a given set of points in the plane. Point sets in general position are well-studied in geometry and play a role in data visualization. We prove several computational hardness results and show how polynomial-time data reduction can be applied to solve the problem if the sought number of points in general position is very small or very large.

The Distinct Vectors problem asks to select a minimum number of columns in a given matrix such that all rows in the selected submatrix are pairwise distinct. This problem is motivated by combinatorial feature selection. We prove a complexity dichotomy with respect to combinations of the minimum and the maximum pairwise Hamming distance of the rows for binary input matrices, thus separating polynomial-time solvable from NP-hard cases.

Co-Clustering is a well-known matrix clustering problem in data mining where the goal is to partition a matrix into homogenous submatrices. We conduct an extensive multivariate complexity analysis revealing several NP-hard and some polynomial-time solvable and fixed-parameter tractable cases.

The generic F-free Editing problem is a graph modification problem in which a given graph has to be modified by a minimum number of edge

modifications such that it does not contain any induced subgraph isomorphic to the graph F. We consider three special cases of this problem: The graph clustering problem Cluster Editing with applications in machine learning, the Triangle Deletion problem which is motivated by network cluster analysis, and Feedback Arc Set in Tournaments with applications in rank aggregation. We introduce a new parameterization by the number of edge modifications above a lower bound derived from a packing of induced forbidden subgraphs and show fixed-parameter tractability for all of the three above problems with respect to this parameter. Moreover, we prove several NP-hardness results for other variants of F-free Editing for a constant parameter value.

The problem DTW-Mean is to compute a mean time series of a given sample of time series with respect to the dynamic time warping distance. This is a fundamental problem in time series analysis the complexity of which is unknown. We give an exact exponential-time algorithm for DTW-Mean and prove polynomial-time solvability for the special case of binary time series.

Preface

This thesis contains some outcomes of my research activity at TU Berlin in the Algorithmics and Computational Complexity group of Prof. Rolf Niedermeier from December 2012 until December 2017 (including a parental leave from August 2016 to March 2017). From December 2012 to February 2015, I gratefully received financial support by Deutsche Forschungsgemeinschaft in the project DAMM (NI 369/13).

The presented results are partly based on journal and conference publications which were prepared in close collaboration with several coauthors, which are, in alphabetical order, René van Bevern, Markus Brill, Laurent Bulteau, Till Fluschnik, Sepp Hartung, Brijnesh Johannes Jain, Iyad Kanj, Christian Komusiewicz, André Nichterlein, Rolf Niedermeier, David Schultz, and Manuel Sorge.

In the following, I will briefly elaborate on my contributions to the respective publication corresponding to each chapter.

Chapter 3. Iyad Kanj (visiting TU Berlin from October 2014 to March 2015) proposed to study the General Position Subset Selection problem which I found appealing due to its simple and natural definition. Basically all results were jointly developed by all authors. I was especially involved in proving the kernelization results and wrote down major parts of our results for a conference version. Iyad Kanj presented the results at the *25th Fall Workshop on Computational Geometry (FWCG '15)* and at the *28th Canadian Conference on Computational Geometry (CCCG '16)* [Fro+16b]. I was also responsible for preparing a journal version for the *International Journal of Computational Geometry & Applications* [Fro+17].

Chapter 4. In my master thesis [Fro12], I studied several combinatorial feature selection problems, among which was Distinct Vectors. The results included the NP-hardness for binary matrices with maximum row Hamming distance four and minimum row Hamming distance two and polynomial-time

solvability for maximum Hamming distance three. I presented the results at the *38th International Symposium on Mathematical Foundations of Computer Science (MFCS '13)* [Fro+13]. Afterwards, I realized that the polynomial-time solvable cases can actually be generalized to arbitrary maximum Hamming distance depending on the minimum Hamming distance. I wrote down the proof for the full dichotomy which is published in the *Journal of Computer and System Sciences* [Fro+16a]. The chapter of this thesis consists of this dichotomy result.

Chapter 5. Rolf Niedermeier suggested to study the complexity of co-clustering problems. I contributed the hardness results. The idea for the reduction to SAT was by Laurent Bulteau. I was involved in developing the algorithmic results based on the SAT reduction and also in writing major parts for a conference version which I presented at the *25th International Symposium on Algorithms and Computation (ISAAC '14)* [Bul+14]. I was also responsible for preparing the journal version for which I implemented the SAT solving approach and conducted some preliminary experiments. The article appeared in *Algorithms* [Bul+16a].

Chapter 6. René van Bevern and Christian Komusiewicz had the idea to parameterize edge modification problems above lower bounds obtained by packings of induced forbidden subgraphs. They already had some preliminary results for Triangle Deletion with triangle packings when I joined the project. Together, we developed the generic framework for vertex-disjoint packings of arbitrary cost-t subgraphs. Additionally, I contributed the results for Feedback Arc Set in Tournaments. We prepared a conference version which was presented by René van Bevern at the *11th International Computer Science Symposium in Russia (CSR '16)* [BFK16]. A journal version was invited to a special issue of *Theory of Computing Systems* [BFK18]. For this thesis, I also revised some of the proofs.

Chapter 7. In January 2017, Brijnesh Johannes Jain contacted me with a question regarding the complexity of the DTW-Mean problem. On the annual research group retreat in April 2017 near Boiensdorf, we started to investigate the problem together with Markus Brill, Till Fluschnik and Rolf Niedermeier. We developed some first ideas for a polynomial-time algorithm for the binary case. Back in Berlin, I figured out the details and wrote down the proof.

Together, we discovered some flaws in some claims from the literature regarding the computation of exact solutions. I developed the dynamic program solving DTW-Mean exactly and implemented it for some benchmark experiments. A conference version was presented by Till Fluschnik at the *SIAM International Conference on Data Mining (SDM '18)* [Bri+18]. I presented the results at the *Workshop on Optimization, Machine Learning, and Data Science* and also at the *3rd Highlights of Algorithms Conference*.

Besides the abovementioned research, I also contributed to projects, which are not covered by my thesis. I was involved in studying vertex dissolution problems in networks [Bev+15], degree-based graph anonymization [Bre+15b], triangle counting in graph streams [Bul+16b], diffusion games on graphs [BFT16], degree-constrained editing of undirected graphs [FNN16] and directed graphs [Bre+18], influence propagation in social networks [Bul+17], graph partitioning problems [Bev+17], and complexity aspects of point visibility graphs [Him+18].

Acknowledgements. I am thankful to Rolf Niedermeier for teaching me as a master's student and giving me the opportunity to work in his group as a PhD student. Starting in 2010, I attended his lectures on algorithmics and complexity at TU Berlin. In 2012, he supervised my master thesis. Subsequently, he supervised my work on this PhD thesis until 2018.

Furthermore, I would like to thank all my former and current colleagues Matthias Bentert, René van Bevern, Robert Bredereck, Markus Brill, Laurent Bulteau, Jiehua Chen, Stefan Fafianie, Till Fluschnik, Marcelo Garlet Millani, Sepp Hartung, Falk Hüffner, Andrzej Kaczmarczyk, Christian Komusiewicz, Stefan Kratsch, Junjie Luo, Hendrik Molter, André Nichterlein, Piotr Skowron, Manuel Sorge, Nimrod Talmon, Christlinde Thielcke, Anh Quyen Vuong, Mathias Weller, Gerhard J. Woeginger, and Philipp Zschoche.

Moreover, I enjoyed working with my groupexternal coauthors Anne-Sophie Himmel, Clemens Hoffmann, Brijnesh Johannes Jain, Iyad Kanj, Marcel Koseler, Pascal Kunz, Konstantin Kutzkov, Rasmus Pagh, and David Schultz.

Finally, I am thankful to Deutsche Forschungsgemeinschaft for financial support.

Contents

Chapter 1

Introduction

In this thesis, we perform an in-depth computational complexity analysis of various computationally hard problems related to data science questions. The goal is to obtain a fine-grained picture of their complexity landscapes. Such a detailed analysis provides insights into which properties make a problem hard to solve, thereby allowing one to devise specialized algorithms to efficiently compute a solution for certain special cases. Thus, we chart the border between intractable and tractable cases (always considering exact solutions).

To this end, we adopt a *parameterized* viewpoint, where a specific property of a problem instance is measured in terms of a *parameter* which usually comprises a single integer value (for example, the number of rows of an input matrix). This approach is motivated by the idea that, in practice, real-world instances often have an underlying structure with certain properties (that is, small parameter values) that may render these instances tractable. As a tool for designing efficient algorithms, we adopt the concept of *fixed-parameter tractability*, which has successfully been applied over the last years to overcome the intractability of NP-hard problems. The notion of fixed-parameter tractability describes the fact that an instance with a small parameter value can be solved efficiently (a constant parameter value implying a polynomial running time where the degree of the polynomial does not depend on the parameter). In some cases, where fixed-parameter tractability is unknown or unlikely, we show that it is still possible to spot polynomial-time solvability for specific constant parameter values.

While a majority of parameterized complexity research so far mainly focused on (classic) graph-theoretic problems, we study a diverse selection of combinatorial problems originating from data science-related domains which have not been intensively studied from a parameterized viewpoint until now, such as machine learning, data mining, time series analysis, and also computational

geometry (geometric questions play a role in data visualization for example). The contribution of this research is twofold:

First, we broaden the field of application of parameterized algorithmics to some "non-standard" problems such as finding points in general position, feature selection, co-clustering, or time series averaging. Problems like these have so far rarely been investigated from a parameterized viewpoint. In fact, some of the problems we study have drawn no or only little attention concerning their (classical) computational complexity at all. Hence, in parts, our results initialize complexity-theoretic studies of some practically motivated problems (which are often solved heuristically) and encourage further research in this direction. Moreover, by studying problems on a variety of discrete structures such as point sets, matrices, (di-) graphs, and time series, we showcase the versatility and the merit of a parameterized view for a fine-grained complexity analysis[1].

Second, in conducting our complexity analysis we en passant enrich the toolbox of parameterized and exact algorithmics by some further algorithmic ideas and facets. For example, this includes the following aspects:

- A crucial part of deriving fixed-parameter tractability is the choice of parameter. While often the so-called "standard parameter" (usually the size or the quality of a sought solution) is considered for many problems, we aim at identifying "non-standard" parameters for our problems. For example, we consider *multivariate* parameters which are combinations of multiple single parameters. This provides a systematic understanding of the inherent computational hardness of a problem. In addition, we consider *above-guarantee* parameterization, where the parameter is the difference between the size of a sought solution and a lower bound for the size of a solution for the given input instance. This "data-driven" parameter can be computed in advance for a given input instance and can be much smaller than the standard parameter solution size. Hence, this might lead to faster algorithms in practice.

- Fixed-parameter tractability results often involve a detailed analysis of the combinatorial structure of a problem instance (it has even been described as "polynomial-time extremal structure theory" [Est+05]). Sometimes it is possible to make use of known results from pure combinatorics in

[1] This is in the spirit of the *Workshop on Parameterized Complexity - "Not About Graphs"* (organized by Michael Fellows and Frances Rosamond) which first took place in 2011 at Charles Darwin University.

order to achieve an algorithmic result. A famous example is the use of the *sunflower lemma* for kernelizations of Hitting Set [FG06, Chapter 9]. This thesis contains two further examples of how to make algorithmic use of extremal combinatorics for a geometry problem and a matrix problem.

- A prominent technique to derive fixed-parameter tractability is to apply polynomial-time data reduction to obtain problem kernels. We provide some generic data reduction rules which are equipped with an additional parameter that allows for a tradeoff between running time and effectiveness. The possibility of such a fine-tuned data reduction might be valuable in practice.

In the following, we give a brief overview of this work.

To start with, we provide preliminaries such as the theoretical background of (parameterized) computational complexity theory in Chapter 2.

Chapter 3 studies a point set problem from computational geometry, where the goal is to select a maximum number of points in general position from a given set of points in the plane. The computational complexity of this problem has not been studied before. We prove several intractability results and also show how polynomial-time data reduction can be applied in order to obtain (tight) problem kernels based on results from combinatorial geometry. We also note that existing ideas in the literature generalize to a framework for showing kernel lower bounds for other point set problems.

In Chapter 4, we again make use of a combinatorial result (this time from extremal set theory). Notably, in doing so, we even achieve polynomial-time solvability (in contrast to fixed-parameter tractability in Chapter 3). The problem we study is motivated by combinatorial feature selection tasks in data mining and machine learning and asks to select a minimum number of columns of a matrix that are sufficient to distinguish all rows. We consider the combination of the maximum and the minimum Hamming distance between any pair of rows as a parameter. This allows us to define a clear complexity border, that is, we prove a dichotomy fully separating NP-hard cases from polynomial-time solvable cases (for binary matrices).

Chapter 5 is concerned with a data mining problem which is to cluster a matrix into homogenous submatrices by partitioning both the rows and the columns of the matrix. While in Chapter 4 we considered a combination of two parameters, we now conduct an extensive multivariate complexity analysis (involving five parameters) revealing NP-hardness for several parameter combinations and some polynomial-time solvable and also fixed-parameter tractable cases (yet, we do

not reach a complete dichotomy as in Chapter 4, since some cases remain open). Interestingly, the polynomial-time and fixed-parameter tractability results are achieved via a reduction to SAT solving, which draws a connection between two seemingly remote domains.

In Chapter 6, we continue with graph-based clustering. We develop a general framework for deriving fixed-parameter tractability (problem kernels and algorithms) of graph (edge) modification problems with applications in machine learning and bioinformatics such as correlation clustering (or cluster editing), that is, to find a clustering that maximizes similarity (or agreement) between given data points. We consider a parameterization above a given lower bound of required edge modifications computed from a packing of induced subgraphs. Here, the packing incorporates structural information about the input graph which can be used to apply data reduction. We apply the framework to three example problems including cluster editing and two other problems motivated by network cluster analysis and rank aggregation. Moreover, we prove NP-hardness for a constant parameter value for several other problems, thus outlining the border of what is achievable with our parameterization.

Finally, Chapter 7 discusses a problem related to clustering time series, namely, to compute a mean of a sample of time series in dynamic time warping spaces. This is a fundamental problem in time series analysis whose computational complexity is unknown. In practice, the task is solved only by heuristics without provable performance guarantees. We give a first exact algorithm for this problem running in exponential time (the running time is polynomial for a constant number of input time series). As in Chapters 4 and 5, we also spot a polynomial-time solvable special case, namely for binary time series. We recently proved several hardness results and a running time lower bound [BFN18] (not included in this thesis).

Chapter 8 concludes the thesis and outlines possible directions for future research.

Chapter 2

Preliminaries and Notation

In this section, we define notation and introduce basic concepts used in this thesis.

2.1 Numbers, Sets, and Matrices

We use the following notation for sets of numbers

- $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$ the set of natural numbers,
- $\mathbb{Z}$ the set of integers,
- $\mathbb{Q}$ the set of rational numbers,
- $\mathbb{R}$ the set of real numbers.

A superscript $+$ (or $-$) indicates the subset of positive (or negative) numbers and a subscript 0 indicates that 0 is also contained in the set. For $n, i, j \in \mathbb{N}$, we define $[n] := \{1, 2, \ldots, n\}$ and $[i, j] := \{i, i+1, \ldots, j\}$. The set of all size-$k$ subsets of a set X is denoted by $\binom{X}{k}$.

We use standard terminology for matrices. A matrix $A = (a_{ij}) \in \mathbb{Q}^{m \times n}$ consists of m rows and n columns where a_{ij} denotes the entry in row i and column j. We denote the i-th row vector of A by a_i and the j-th column vector by a_{*j}. For subsets $I \subseteq [m]$ and $J \subseteq [n]$ of row and column indices, we write $A[I, J] := (a_{ij})_{(i,j) \in I \times J}$ for the $(|I| \times |J|)$-submatrix of A containing only the rows with indices in I and the columns with indices in J. For a vector x, we denote by $(x)_j$ the j-th entry of x. The null vector is denoted by $\mathbf{0} := (0, \ldots, 0)$.

2.2 Graph Theory

This section provides some basic definitions from graph theory.

Undirected Graphs. An undirected graph (or graph) is a pair $G = (V, E)$ containing a *vertex set* $V(G) := V$ and an *edge set* $E(G) := E \subseteq \binom{V}{2}$. We set $n := |V|$, $m := |E|$ and $|G| := n + m$.

A vertex $v \in V$ is a *neighbor* of (or is *adjacent* to) a vertex $u \in V$ if $\{u, v\} \in E$ and a vertex $v \in V$ is *incident* with an edge $e \in E$ if $v \in e$. The *open neighborhood* of vertex v is the set $N_G(v) := \{u \in V \mid \{u, v\} \in E\}$ of its neighbors. The *closed neighborhood* of v is $N_G[v] := N_G(v) \cup \{v\}$. For a vertex set $V' \subseteq V$, we define $N_G(V') := \bigcup_{v \in V'} N_G(v)$. The *degree* of a vertex v is the number $\deg_G(v) := |N_G(v)|$ of its neighbors. We omit subscripts if the corresponding graph is clear from the context.

We denote by $\overline{G} := (V, \binom{V}{2} \setminus E)$ the *complement graph* of G. For a vertex subset $V' \subseteq V$, the *subgraph induced by* V' is $G[V'] := (V', E \cap \binom{V'}{2})$. For an edge subset $E' \subseteq \binom{V}{2}$, $V(E')$ denotes the set of all *endpoints* of edges in E' and we define the induced subgraph $G[E'] := (V(E'), E')$. For a set $E' \subseteq \binom{V}{2}$, we denote by $G + E' := (V, E \cup E')$ the graph that results from inserting all edges in E' into G and by $G - E' := (V, E \setminus E')$ we denote the graph that results from deleting the edges in E'. For a vertex set $V' \subseteq V$, we define the graph $G - V' := G[V \setminus V']$.

Directed Graphs. A directed graph (or digraph) $G = (V, A)$ consists of a vertex set $V(G)$ and an *arc set* $A(G) := A \subseteq \{(u, v) \in V^2 \mid u \neq v\}$.

A *tournament* is a directed graph (V, A) such that, for each pair $\{u, v\} \in \binom{V}{2}$, either $(u, v) \in A$ or $(v, u) \in A$.

Hypergraphs. A hypergraph is a pair $H = (V, E)$ containing a vertex set V and a set E of *hyperedges*. Each hyperedge in E is a subset $F \subseteq V$ of vertices. For $d \in \mathbb{N}$, a hypergraph H is called *d-uniform* if all hyperedges in H contain d elements. Thus, an undirected graph is a 2-uniform hypergraph.

2.3 Computational Complexity

In this section, we briefly recall the basic concepts of classical complexity theory (we refer to the books by Garey and Johnson [GJ79], Papadimitriou [Pap94], and Arora and Barak [AB09] for a detailed introduction). Complexity theory investigates the computational complexity of *decision problems* usually in terms of the *worst-case* running time required to solve a given problem algorithmically (often formalized by Turing machines).

Formally, given a finite alphabet Σ (usually the binary alphabet $\Sigma = \{0, 1\}$), a decision problem is to decide whether a given word (or *instance*) $x \in \Sigma^*$ is contained in a given language $L \subseteq \Sigma^*$. For example, an instance x might be an appropriate encoding of numbers, a matrix or a graph. If $x \in L$, then x is called a *yes-instance*, otherwise, x is a *no-instance*.

The two most important complexity classes are called P and NP. The class P contains all languages which can be decided in *polynomial time*, that is, in $O(|x|^c)$ time for some constant c, where $|x|$ denotes the length of the input instance, by a *deterministic* Turing machine. Problems contained in P are considered to be computationally tractable. The class NP contains all problems that can be solved in polynomial time by a *nondeterministic* Turing machine. It is clear that $\mathrm{P} \subseteq \mathrm{NP}$ but it is a famous open problem whether $\mathrm{P} = \mathrm{NP}$ holds. In fact, it is widely conjectured that $\mathrm{P} \neq \mathrm{NP}$. Thus, the hardest problems in NP are believed to be intractable.

NP-hardness is defined in terms of reductions.

Definition 2.1. Let $A, B \subseteq \Sigma^*$. A *polynomial-time many-one reduction* from A to B is a polynomial-time computable function $f : \Sigma^* \to \Sigma^*$ such that for all $x \in \Sigma^*$ it holds $x \in A \Leftrightarrow f(x) \in B$.

A problem A is *NP-hard* if for every problem B in NP there exists a polynomial-time many-one reduction from B to A. If A is also contained in NP, then A is called *NP-complete*. A polynomial-time algorithm solving an NP-complete problem implies $\mathrm{P} = \mathrm{NP}$. Hence, all NP-complete problems are presumably intractable.

Exponential Time Hypothesis. The assumption $\mathrm{P} \neq \mathrm{NP}$ asserts that no NP-complete problem can be solved in polynomial time. There exist even stronger conjectures arising from the experienced difficulty to find *sub-exponential-time* algorithms (that is, running in $2^{o(|x|)}$ time) for certain NP-complete problems. One of those problems is the following:

k-CNF-Satisfiability (k-SAT)

Input: Boolean formula ϕ in conjunctive normal form with at most k literals per clause.

Question: Is there a truth assignment of the variables that satisfies ϕ?

Impagliazzo and Paturi [IP01] formulated the following conjecture that 3-SAT cannot be solved in time sub-exponential in the number of variables.

Conjecture 2.1 (Exponential Time Hypothesis (ETH))**.** *There exists a constant $c > 0$ such that* 3-SAT *cannot be solved in $O(2^{cn})$ time, where n is the number of variables in the input formula.*

Note that the Exponential Time Hypothesis implies P $\neq$ NP. Based on the Exponential Time Hypothesis, it is also possible to derive exponential running time lower bounds for other problems. Assuming the Exponential Time Hypothesis, it is however still possible that k-SAT with n-variable formulas can be solved in 2^{cn} time for some constant c with $0 < c < 1$ for every k. An even stronger conjecture (implying ETH) asserts that this is not possible [IP01].

Conjecture 2.2 (Strong Exponential Time Hypothesis (SETH))**.** *There exists no constant $c < 1$ such that k-*SAT *can be solved in $O(2^{cn})$ time for every $k \geq 3$, where n is the number of variables in the input formula.*

2.4 Parameterized Complexity

Parameterized complexity theory was developed by Downey and Fellows [DF13, DF99] in order to cope with the inctractability of NP-hard problems. The idea is to measure the time complexity of a problem not only with respect to the input size but also with respect to an additional *parameter* value (usually a natural number). Intuitively, the parameter holds some information about the input instance and its structure which might be used algorithmically.

We briefly introduce the formal definitions of relevant notions from parameterized algorithmics (refer to the monographs [Cyg+15, DF13, FG06, Nie06] for a detailed introduction).

Definition 2.2. A *parameterized problem* is a language $L \subseteq \Sigma^* \times \mathbb{N}$. A parameterized instance (x, k) consists of the *input* $x \in \Sigma^*$ and the *parameter* $k \in \mathbb{N}$.

A parameterized problem is classified as tractable if it can be solved by an algorithm for which the superpolynomial part of the running time solely depends on the parameter. Hence, if the parameter of an instance is small, then this algorithm solves the instance efficiently. This motivates the following definition.

Definition 2.3. A parameterized problem is *fixed-parameter tractable* if there exists an algorithm that decides any instance (x, k) in $f(k) \cdot |x|^{O(1)}$ time, where f is a computable function only depending on k.

The class FPT contains all fixed-parameter tractable parameterized problems. If the parameter of a fixed-parameter tractable problem is a constant, then the problem is polynomial-time solvable (that is, the corresponding classical decision problem is contained in P). Moreover, note that the degree of the polynomial $|x|^{O(1)}$ is a constant not depending on the parameter k. If the degree of the polynomial depends on k (for example $|x|^{O(k^2)}$) then the problem is contained in the class XP. Clearly, FPT $\subseteq$ XP. A *multivariate* parameter is a tuple $(k_1, \ldots, k_d)$ and the parameter value is defined as the sum $k_1 + \ldots + k_d$.

Kernelization. A common approach to show that a problem is fixed-parameter tractable is to apply polynomial-time data reduction as a preprocessing of a given instance. The goal is to obtain an equivalent instance whose size is upper-bounded by a function of the parameter. This is known as *kernelization*.

Definition 2.4. A *problem kernelization* for a parameterized problem A is an algorithm that given an instance (x, k) outputs an instance (x', k') in $(|x|+k)^{O(1)}$ time such that

(i) $|x'| \leq \lambda(k)$ for some computable function λ,

(ii) $k' \leq \lambda(k)$, and

(iii) $(x, k) \in A \Leftrightarrow (x', k') \in A$.

The instance (x', k') is called a *problem kernel* and λ is called its *size*. If λ is a polynomial, then (x', k') is called a *polynomial problem kernel*. We say that (x', k') and (x, k) are *equivalent* instances. A decidable parameterized problem is in FPT if and only if it has a problem kernelization [Cai+97].

A problem kernelization is often achieved by describing a set of polynomial-time *data reduction rules* which are applied to an instance (x, k) of a problem and yield an instance (x', k'). We say that a data reduction rule is *correct* if (x, k) and (x', k') are equivalent instances.

Kernelization Lower Bounds. It is known that every parameterized problem in FPT has a problem kernel, but it is not clear whether every fixed-parameter tractable probem has a "small", that is, a polynomial kernel. Several techniques have been developed to exclude the existence of polynomial problem kernels for some problems based on the complexity-theoretic assumptions that NP $\not\subseteq$ coNP/poly (or coNP $\not\subseteq$ NP/poly) [BJK14, Bod+09, DM14, Dru15]. Here, NP/poly is the (nonuniform) class of problems that can be solved by a polynomial-time nondeterministic Turing machine with polynomial advice (and coNP/poly contains all complements of languages in NP/poly). Yap [Yap83] showed that NP $\subseteq$ coNP/poly (and also coNP $\subseteq$ NP/poly) implies that the polynomial hierarchy collapses to its third level which is considered unlikely in complexity theory.

One way to show that a problem presumably has no polynomial problem kernel is to use a *polynomial parameter transformation.* A polynomial parameter transformation from a parameterized problem L to another parameterized problem Q is a polynomial-time algorithm mapping an instance (x, k) of L to an equivalent instance $(x', p(k))$ of Q, where p is a polynomial [BTY11]. If L does not have a polynomial problem kernel and the classical (non-parameterized) decision problem of L is NP-hard and the decision problem of Q is in P, then Q also has no polynomial problem kernel [BJK14].

Parameterized Intractability. Some parameterized problems turn out to be presumably not fixed-parameter tractable. In parameterized complexity theory, these problems are captured in the classes of the W-hierarchy:

$$\text{FPT} \subseteq \text{W[1]} \subseteq \text{W[2]} \subseteq \ldots \subseteq \text{XP}.$$

It is unknown whether FPT = W[1] and it is believed that in fact all above inclusions are strict.

A parameterized problem A is W[t]-hard for $t \geq 1$ if for every problem B in W[t] there exists a *parameterized reduction* from B to A.

Definition 2.5. Let $A, B \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems. A *parameterized reduction* from A to B is a function $f : (\Sigma^* \times \mathbb{N}) \to (\Sigma^* \times \mathbb{N})$ such that for all $(x, k) \in (\Sigma^* \times \mathbb{N})$ the following holds:

(i) $(x', k') := f((x, k))$ can be computed in $g(k) \cdot |x|^{O(1)}$ time for some computable function g,

(ii) $k' \leq h(k)$ for some computable function h, and

(iii) $(x, k) \in A \Leftrightarrow (x', k') \in B$.

2.5 Approximation

Another way to deal with NP-hard problems is to relax the optimality condition of a solution. That is, instead of solving a decision problem, we aim at finding an approximate solution for the corresponding optimization problem in polynomial time such that this solution is close to optimal. We briefly introduce the basic concepts. Refer to the books by Vazirani [Vaz01] and Williamson and Shmoys [WS11] for a more comprehensive discussion of approximation algorithms.

Definition 2.6. An optimization problem Q is a quadruple (I, S, f, g) where

1. $I \subseteq \Sigma^*$ is the set of input instances;
2. for each instance $x \in I$, $S(x)$ is the set of *feasible solutions* for x. The length of each solution $y \in S(x)$ is upper-bounded by a polynomial in $|x|$ and it is polynomial-time decidable whether $y \in S(x)$ holds for given y and x;
3. f is the *objective* function mapping an instance $x \in I$ and a solution $y \in S(x)$ to a natural number $f(x, y) \in \mathbb{N}$. The function f is computable in polynomial time;
4. $g \in \{\max, \min\}$ is the *goal* function.

We speak of a *maximization problem* if $g = \max$ and of a *minimization problem* if $g = \min$.

For an instance $x \in I$, we denote by $\operatorname{opt}(x) := g\{f(x, y) \mid y \in S(x)\}$ the optimal objective value of a solution for x. The task is to find a solution $y \in S(x)$ that achieves the optimal objective value, that is, $f(x, y) = \operatorname{opt}(x)$ holds. A feasible solution $y \in S(x)$ has an *approximation factor* of

$$r(x, y) := \begin{cases} \operatorname{opt}(x)/f(x, y), & \text{if } g = \max \\ f(x, y)/\operatorname{opt}(x), & \text{if } g = \min \end{cases} \geq 1.$$

Definition 2.7. Let $\rho : \mathbb{N} \to \mathbb{Q}$ be a function. A *ρ-approximation algorithm* for an optimization problem Q is a polynomial-time algorithm that for every instance $x \in I$ returns a feasible solution $y(x) \in S(x)$ such that $r(x, y(x)) \leq \rho(|x|)$.

An optimization problem Q has a *constant-factor approximation algorithm* if it has a ρ-approximation algorithm for a constant function ρ. The class of all optimization problems that admit constant-factor approximation algorithms is denoted by APX. The class APX contains problems which can even be solved arbitrarily close to optimal in polynomial time. These problems are said to admit a polynomial-time approximation scheme.

Definition 2.8. A *polynomial-time approximation scheme* (PTAS) for an optimization problem Q is an algorithm that takes an instance $x \in I$ and a constant $\epsilon > 0$ as input and returns in $O(n^{\lambda(\epsilon)})$ time (for some computable function λ) a feasible solution $y \in S(x)$ such that $r(x, y) \leq 1 + \epsilon$.

There also exists an inctractability theory concerning polynomial-time approximation. An optimization problem that is APX-hard does not admit a PTAS unless P = NP. APX-hardness is defined in terms of so-called PTAS-reductions.

Definition 2.9. Given two optimization problems $Q = (I, S, f, g)$ and $Q' = (I', S', f', g')$, a *PTAS-reduction* from Q to Q' consists of three polynomial-time computable functions h, h' and $\alpha\colon \mathbb{Q} \to \mathbb{Q}^+$ such that:

(i) $h : I \to I'$ maps an instance of Q to an instance of Q', and

(ii) for any instance $x \in I$ of Q and any solution $y' \in S'(h(x))$ of $h(x) \in I'$ and for any $\epsilon \in \mathbb{Q}^+$, $h'(x, y', \epsilon) \in S(x)$ is a solution of x such that

$$r(h(x), y') \leq 1 + \alpha(\epsilon) \;\Rightarrow\; r(x, h'(x, y', \epsilon)) \leq 1 + \epsilon.$$

Chapter 3

General Position Subset Selection

In this chapter, we study the problem of selecting a maximum number of points in general position from a given point set in the plane. Even though the concept of general position is fundamental in geometry, this problem has not been studied with respect to its computational complexity. We prove several intractability results for this problem as well as fixed-parameter tractability results. In particular, we use a result from combinatorial geometry to obtain a cubic-point problem kernel with respect to the number of selected points based on a simple reduction rule (improving on the kernel size known from the literature). We also prove a quadratic-point problem kernel for the dual parameter (that is, the number of points to delete in order to obtain a point set in general position) while excluding strongly subquadratic kernels, in the course of which we observe a framework for proving kernel lower bounds for point set problems as a side result.

3.1 Introduction

For a set $P = \{p_1, \ldots, p_n\}$ of n points in the plane, a subset $S \subseteq P$ is in *general position* if no three points in S are *collinear* (that is, lie on the same straight line). A frequent assumption for point set problems in computational geometry is that the given point set is in general position. In fact, testing whether a given point set is in general position is a key problem in computational geometry [Bar+17]. We consider the generalization of this problem which is to select a maximum-cardinality subset of points in general position from a given set of points. Point sets in general position occur in many different fields including graph drawing [FKV14] and pattern recognition [Cov65, Joh14].

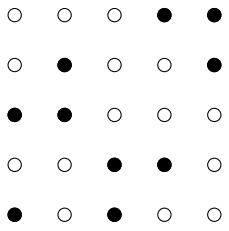


Figure 3.1: Example of a point set consisting of 25 points on a (5×5)-grid. The ten black points are in general position. Since all the points can be covered by five straight lines, it follows that there can be at most ten points in general position (two from each covering line).

Formally, the decision version of the problem is as follows:

General Position Subset Selection
Input: A set P of points in the plane and $k \in \mathbb{N}$.
Question: Is there a subset $S \subseteq P$ in general position of cardinality at least k?

See Figure 3.1 for an example. This problem has received quite some attention from the combinatorial geometry perspective, but it was hardly considered from the computational complexity perspective. In particular, the classical complexity of General Position Subset Selection was unknown.

A well-known special case of General Position Subset Selection, called the No-Three-In-Line problem, asks to place a maximum number of points in general position on an $(n \times n)$-grid. Since at most two points can be placed on any grid-line, the maximum number of points in general position that can be placed on an $(n \times n)$-grid is at most $2n$. Indeed, only for small n it is known that $2n$ points can always be placed on the $(n \times n)$-grid (Figure 3.1 shows a solution for $n = 5$). Erdős [Rot51] observed that, for sufficiently large n, one can place at least $(1 - \epsilon)n$ points in general position on the $(n \times n)$-grid, for any $\epsilon > 0$. This lower bound was improved by Hall et al. [Hal+75] to $(\frac{3}{2} - \epsilon)n$. It was conjectured by Guy and Kelly [GK68] that, for sufficiently large n, one can place no more than $\frac{\pi}{\sqrt{3}}n$ points in general position on an $(n \times n)$-grid. This conjecture remains unresolved, hinting at the challenging combinatorial nature of No-Three-In-Line, and hence of General Position Subset Selection as well.

A problem closely related to General Position Subset Selection is Point Line Cover: Given a point set in the plane, find a minimum-cardinality set of straight lines, the size of which is called the *line cover number*, that cover all points. Interestingly, the size of a maximum subset in general position is related to the line cover number in the following way (see also Observation 3.11). If there are k points in general position, then at least $k/2$ lines are required to cover all points since each line can cover at most two of the k points in general position. Also, if k is the maximum number of points in general position, then at most $\binom{k}{2}$ lines are required to cover all points since every point lies on some line defined by two of the k points in general position. Thus, knowing the maximum number of points in general position allows to derive lower and upper bound on the line cover number.

While Point Line Cover has been intensively studied with respect to its computational complexity, we start filling the gap for General Position Subset Selection by providing both computational hardness and fixed-parameter tractability results for the problem. In doing so, we particularly consider the parameters solution size k (size of the sought subset in general position) and its dual $h := n - k$ (number of points to delete to obtain a subset in general position), and investigate their impact on the computational complexity of General Position Subset Selection.

Related Work. Payne and Wood [PW13] proved lower bounds on the size of a point set in general position, a question originally studied by Erdős [Erd86]. In his Master's thesis, Cao [Cao12] showed a problem kernel with $O(k^4)$ points for General Position Subset Selection (there called Non-Collinear Packing problem) and a simple greedy factor-$O(\sqrt{\text{opt}})$ approximation algorithm for the maximization version. An improved approximation algorithm for General Position Subset Selection was recently published by Rudi [Rud17] and finds a subset in general position of size at least $\max\{2n^2/(\text{coll}(P)+2n), \sqrt{\text{opt}}\}$, where $\text{coll}(P)$ denotes the total number of collinear pairs in lines with at least three points in P. Cao [Cao12] also presented an Integer Linear Program formulation and showed that it is in fact the dual of an Integer Linear Program formulation for Point Line Cover.

The General Position Testing problem is to decide whether any three points of a given point set are collinear (this corresponds to the special case of General Position Subset Selection where $k = |P|$). It is a major open question in computational geometry whether this problem can be solved

in subquadratic time. Recently, Barba et al. [Bar+17] showed how to solve General Position Testing in subquadratic time for special point sets.

As to results for Point Line Cover (refer to Kratsch et al. [KPR16] and the work cited therein): It is known to be NP- and APX-hard (a factor-$O(\log(\text{opt}))$ approximation algorithm is known). It is fixed-parameter tractable with respect to the number k of lines and solvable in $(k/1.35)^k \cdot n^{O(1)}$ time. Also, a problem kernel containing k^2 points is known and there is no problem kernel with $O(k^{2-\epsilon})$ points for any $\epsilon > 0$ unless coNP $\subseteq$ NP/poly.

Our Contributions. In Section 3.2, we show that General Position Subset Selection is NP- and APX-hard (as is Point Line Cover) and we prove an exponential running time lower bound based on the Exponential Time Hypothesis. Our main algorithmic results concern the power of polynomial-time data reduction for General Position Subset Selection. In Section 3.3, we give an $O(k^3)$-point problem kernel and an $O(h^2)$-point problem kernel for the dual parameter $h := n - k$, and show that the latter kernel is asymptotically optimal under a reasonable complexity-theoretic assumption. Table 3.1 summarizes our main results. Note that our kernelization results for the dual parameter h match the kernelization results for Point Line Cover with respect to the number of lines (in fact, our lower bound result is based on the ideas of Kratsch et al. [KPR16]).

Preliminaries. We consider points in the plane encoded as pairs of coordinates and assume all coordinates to be rational numbers. The *collinearity* of a set of points $P \subseteq \mathbb{Q}^2$ is the maximum number of points in P that lie on the same straight line. A *blocker* for two points p, q is a point on the open line segment pq.

3.2 Hardness Results

In this section, we prove that General Position Subset Selection is NP-hard, APX-hard, and presumably not solvable in sub-exponential time. Our hardness results follow from a polynomial-time transformation mapping arbitrary graphs to point sets. The construction is due to Ghosh and Roy [GR15, Section 5], which they used to prove NP-hardness of the Independent Set problem on so-called *point visibility graphs*. This transformation, henceforth called Φ, allows us to obtain the above-mentioned hardness results (using polynomial-time many-one reductions from NP-hard restrictions of Independent Set to General

Table 3.1: Overview of our results for General Position Subset Selection, where n is the number of input points, k is the parameter size of the sought subset in general position, $h = n - k$ is the dual parameter, and ℓ is the line cover number.

	Result	Reference
Hardness	NP-hard	Th. 3.7
	APX-hard	Th. 3.7
	no $2^{o(n)} \cdot n^{O(1)}$-time algorithm (unless the ETH fails)	Th. 3.7
	no $O(h^{2-\epsilon})$-point kernel (unless coNP $\subseteq$ NP/poly)	Th. 3.20
Tractability	$O(k^3)$-point kernel (computable in $O(n^2 \log n)$ time)	Th. 3.9
	$O(n^2 \log n + \alpha^k k^{2k})$-time solvable (for some constant α)	Cor. 3.10
	$O(h^2)$-point kernel (computable in $O(n^2)$ time)	Th. 3.16
	$O(2.08^h + n^3)$-time solvable	Prop. 3.14
	$O(\ell^3)$-point kernel (computable in $O(n^2 \log n)$ time)	Cor. 3.12
	$O(n^2 \log n + \alpha^{2\ell} \ell^{4\ell})$-time solvable (for some constant α)	Cor. 3.12

Position Subset Selection). Moreover, in Section 3.3.2, we will use Φ to give a polynomial-time many-one reduction from Vertex Cover to General Position Subset Selection in order to obtain kernel size lower bounds with respect to the dual parameter (see Theorem 3.17 and Theorem 3.20). We start by formally defining some properties that are required for the output point set of the transformation Φ. As a next step, we prove that such a point set can be computed in polynomial time.

Let G be a graph with vertex set $V(G) = \{v_1, \ldots, v_n\}$. Let $\Phi(G) := C \cup B$ be a set of points where $C = \{p_1, \ldots, p_n\}$ are points in strictly convex position, that is, the points in C are vertices of a convex polygon ($p_i \in C$ corresponds to v_i, $i = 1, \ldots, n$). For each edge $e = \{v_i, v_j\} \in E(G)$, we place a *blocker* $b_e \in B$ on the line segment $p_i p_j$ such that the following three conditions are satisfied:

(I) For any edge $e \in E(G)$ and for any two points $p_i, p_j \in C$, if b_e, p_i, p_j are collinear, then p_i, p_j are the points in C corresponding to the endpoints of edge e.

(II) Any two distinct blockers $b_e, b_{e'}$ are not collinear with any point $p_i \in C$.

(III) The set $B := \{b_e \mid e \in E(G)\}$ of blockers is in general position.

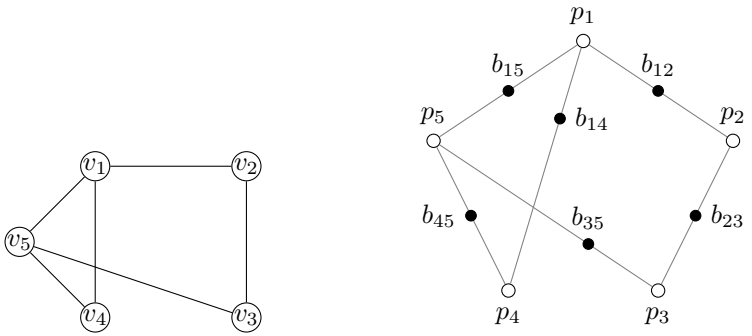


Figure 3.2: Example of a graph (left) and a point set (right) satisfying Conditions (I) to (III). The set contains a point (white) for each vertex and a blocker (black) for each edge in the graph such that the only collinear triples are (p_i, p_j, b_{ij}) for every edge $\{v_i, v_j\}$. Also, no four points in the set are collinear.

Figure 3.2 depicts an example of the transformation described above.

Proposition 3.1. *There is an $O(m \cdot (\max\{n, m\})^2)$-time computable transformation Φ which maps an arbitrary graph G to a point set $\Phi(G) = C \cup B$ that satisfies Conditions (I) to (III). Moreover, no four points in $\Phi(G)$ are collinear.*

Proof. Given a graph G, let $n = |V(G)|$, $m = |E(G)|$ and let $C = \{p_1, \ldots, p_n\}$ be a set of rational points that are in a strictly convex position; for instance, let $p_j := (\frac{2j}{1+j^2}, \frac{1-j^2}{1+j^2})$ for $j \in \{1, \ldots, n\}$ be n rational points (computable in $O(n)$ time) on the unit circle centered at the origin [Tan96]. To choose the set B of blockers, suppose (inductively) that we have chosen a subset B' of blockers such that all blockers in B' are rational points and satisfy Conditions (I) to (III). Let $b_e \notin B'$ be a blocker corresponding to an edge $e = \{v_i, v_j\}$ in G. To determine the coordinates of b_e, we first mark the (rational) intersection points (if any) between the line segment p_ip_j and the lines formed by every pair of distinct blockers in B', every pair of distinct points in $C \setminus \{p_i, p_j\}$, and every pair consisting of a blocker in B' and a point in $C \setminus \{p_i, p_j\}$. We then choose b_e to be an interior point of the segment p_ip_j with rational coordinates that is distinct from all marked intersection points. To this end, let q be the first marked point on the segment p_ip_j starting from p_i (if there is no marked point on p_ip_j, then set $q := p_j$) and let b_e be the midpoint of p_iq. This point is rational since it is the midpoint of rational points. Determining the coordinates

of a blocker can thus be done in $O(m^2 + n^2 + nm)$ time. The overall running time is thus in $O(m \cdot (\max\{n, m\})^2)$ Clearly, all points in $C \cup B$ are rational and satisfy Conditions (I) to (III). Moreover, the construction of $C \cup B$ ensures that no four points in $C \cup B$ are collinear. □

In what follows, we will use transformation Φ as a reduction from Independent Set to General Position Subset Selection in order to prove our hardness results. The following observation will be helpful in proving the correctness of the reduction.

Observation 3.2. *Let G be an arbitrary graph, and let $P := \Phi(G) = C \cup B$. For any point set $S \subseteq P$ that is in general position, there is a general position set of size at least $|S|$ that contains the set of blockers B.*

Proof. Suppose that $S \subseteq P$ is in general position, and suppose that there is a point $b \in B \setminus S$. If b does not lie on a line defined by any two points in S, then $S \cup \{b\}$ is in general position. Otherwise, b lies on a line defined by two points $p, q \in S$. By Conditions (I) and (II), it holds that $p, q \in C$. Moreover, p and q are the only two points in S that are collinear with b. Hence, we exchange one of them with b to obtain a set of points in general position of the same cardinality as S. Since $b \in B$ was arbitrarily chosen, we can repeat the above argument to obtain a subset in general position of cardinality at least $|S|$ that contains B. □

Using Observation 3.2, we give a polynomial-time many-one reduction from Independent Set to General Position Subset Selection based on transformation Φ.

Independent Set
Input: An undirected graph G and $k \in \mathbb{N}$.
Question: Is there a subset $I \subseteq V(G)$ of size at least k such that no two vertices in I are connected by an edge?

Lemma 3.3. *There is a polynomial-time many-one reduction from* Independent Set *to* General Position Subset Selection*. Moreover, each instance of* General Position Subset Selection *produced by this reduction satisfies the property that no four points in the instance are collinear.*

Proof. Let (G, k) be an instance of Independent Set, where $k \in \mathbb{N}$. The General Position Subset Selection instance is defined as $(P := \Phi(G), k +$

$|E(G)|$). Clearly, by Proposition 3.1, the set P can be computed in polynomial time, and no four points in P are collinear. We show that G has an independent set of cardinality k if and only if P has a subset in general position of cardinality $k + |E(G)|$.

Suppose that $I \subseteq V(G)$ is an independent set of cardinality k, and let $S := \{p_i \mid v_i \in I\} \cup B$, where B is the set of blockers in P. Since $|B| = |E(G)|$, we have $|S| = k + |E(G)|$. Suppose towards a contradiction that S is not in general position, and let q, r, s be three distinct collinear points in S. By Conditions (II) and (III), and since the points in C are in a strictly convex position, it follows that exactly two of the points q, r, s must be in C. Suppose, without loss of generality, that $q = p_i, r = p_j \in C$ and $s \in B$. By Condition (I), there is an edge between the vertices v_i and v_j in G that correspond to the points $p_i, p_j \in C$, contradicting that $v_i, v_j \in I$. It follows that S is a subset in general position of cardinality $k + |E(G)|$.

Conversely, assume that $S \subseteq P$ is in general position and that $|S| = k+|E(G)|$. By Observation 3.2, we may assume that $B \subseteq S$. Let I be the set of vertices corresponding to the points in $S \setminus B$, and note that $|I| = k$. Since $B \subseteq S$, no two points v_i, v_j in I can be adjacent; otherwise, their corresponding points p_i, p_j and the blocker of edge $\{v_i, v_j\}$ would be three collinear points in S. It follows that I is an independent set of cardinality k in G. □

Lemma 3.3 implies the NP-hardness of General Position Subset Selection. Furthermore, a careful analysis of the proof of Lemma 3.3 reveals the intractability of an extension variant of General Position Subset Selection, where as an additional input to the problem a subset $S' \subseteq P$ of points in general position is given and the task is to find k *additional* points in general position, that is, one looks for a point subset $S \subseteq P$ in general position such that $S' \subset S$ and $|S| \geq |S'| + k$. By Observation 3.2, we can assume for the instance created by transformation Φ that B (the set of blockers) is contained in a maximum-cardinality point subset in general position. Thus, we can set $S' := B$. The proof of Lemma 3.3 then shows that k points can be added to S' if and only if the graph G contains an independent set of size k. Since Independent Set is W[1]-hard with respect to the solution size [DF13], we can observe the following:

Observation 3.4. *The extension variant of* General Position Subset Selection *described above is W[1]-hard when parameterized by the number k of additional points.*

Hence, this extension is *not* fixed-parameter tractable with respect to k, unless W[1] = FPT. The W[1]-hardness for the extension variant is in contrast to the fixed-parameter tractability results for General Position Subset Selection that will be shown in Section 3.3.1.

Next, we turn our attention to approximation, that is, we consider the following optimization problem.

Maximum General Position Subset Selection
Input: A set P of points in the plane.
Task: Select a subset $S \subseteq P$ in general position of maximum cardinality.

By IS-3 we denote the Maximum Independent Set problem restricted to graphs of maximum degree at most three.

IS-3
Input: An undirected graph G of maximum degree at most three.
Task: Find an independent set $I \subseteq V(G)$ of maximum size.

The next lemma implies APX-hardness of Maximum General Position Subset Selection. Recall the definition of a PTAS-reduction from Section 2.5.

Definition 3.1. Given two maximization problems $Q = (I, S, f, g)$ and $Q' = (I', S', f', g')$, a *PTAS-reduction* from Q to Q' consists of three polynomial-time computable functions h, h' and $\alpha\colon \mathbb{Q} \to \mathbb{Q}^+$ such that:

(i) $h : I \to I'$ maps an instance of Q to an instance of Q', and

(ii) for any instance $x \in I$ of Q and any solution $y' \in S'(h(x))$ of $h(x) \in I'$ and for any $\epsilon \in \mathbb{Q}^+$, $h'(x, y', \epsilon) \in S(x)$ is a solution of x such that

$$\frac{\mathrm{opt}(h(x))}{f'(h(x), y')} \leq 1 + \alpha(\epsilon) \quad \Rightarrow \quad \frac{\mathrm{opt}(x)}{f(x, h'(x, y', \epsilon))} \leq 1 + \epsilon.$$

Lemma 3.5. *There is a PTAS-reduction from* IS-3 *to* Maximum General Position Subset Selection.

Proof. Let G be an instance of IS-3, and note that $|E(G)| \leq 3|V(G)|/2$. It is easy to see that G has an independent set I_0 of cardinality at least $|I_0| > |V(G)|/4$ that can be obtained in polynomial time by repeatedly selecting a vertex in G of minimum degree and discarding all its neighbors until the graph is empty.

We define the computable functions h, h' and α in Definition 3.1 as follows. For a graph G, the function h is defined as $h(G) := \Phi(G)$ and is computable in polynomial time by Proposition 3.1. Let $P := h(G) = C \cup B$ and let $S \subseteq P$ be a subset in general position. For every $\epsilon > 0$, we define an independent set $h'(G, S, \epsilon) \subseteq V(G)$ as follows: If $|S \setminus B| < |V(G)|/4$, then $h'(G, S, \epsilon) := I_0$. Otherwise, we have $|S \setminus B| \geq |V(G)|/4$. If $B \subseteq S$, then $h'(G, S, \epsilon)$ is the set of vertices corresponding to the points in $S \setminus B$ (by the proof of Lemma 3.3, these vertices form an independent set in G). If $B \not\subseteq S$, then we first compute in polynomial time a subset $S' \subseteq P$ in general position such that $B \subseteq S'$ and $|S'| \geq |S|$ (as described in the proof of Observation 3.2). Now, if $|S' \setminus B| < |V(G)|/4$, then we set $h'(G, S, \epsilon) := I_0$ again. Otherwise, we define $h'(G, S, \epsilon)$ to be the independent set corresponding to the points in $S' \setminus B$. This finishes the definition of h'. Note that h' is clearly polynomial-time computable and that $h'(G, S, \epsilon)$ is always of size at least $|V(G)|/4$. Finally, we define $\alpha(\epsilon) := \epsilon/7$.

Let G be a graph, $P := h(G) = C \cup B$ be a point set and let $S \subseteq P$ be a subset in general position. Further, let $\mathrm{opt}(G) \geq |V(G)|/4$ denote the cardinality of a maximum independent set in G, and let $\mathrm{opt}(P)$ be the cardinality of a largest subset in general position in P. From Lemma 3.3, it follows that $\mathrm{opt}(P) = |B| + \mathrm{opt}(G)$. To finish the proof, we need to show that

$$\frac{\mathrm{opt}(P)}{|S|} \leq 1 + \alpha(\epsilon) \quad \Rightarrow \quad \frac{\mathrm{opt}(G)}{|h'(G, S, \epsilon)|} \leq 1 + \epsilon$$

holds for every $\epsilon > 0$. Let $I := h'(G, S, \epsilon)$. Then, we have

$$\begin{aligned}
\frac{\mathrm{opt}(P)}{|S|} = \frac{|B| + \mathrm{opt}(G)}{|B| + |S \setminus B|} &\leq 1 + \alpha(\epsilon) \\
\iff \qquad \mathrm{opt}(G) &\leq (1 + \alpha(\epsilon))(|S \cap B| + |S \setminus B|) - |B| \\
\iff \qquad \frac{\mathrm{opt}(G)}{|I|} &\leq (1 + \alpha(\epsilon)) \cdot \frac{|S \cap B| + |S \setminus B|}{|I|} - \frac{|B|}{|I|}
\end{aligned}$$

Since $|S \cap B| \leq |B|$, we obtain

$$\frac{\mathrm{opt}(G)}{|I|} \leq \alpha(\epsilon)\frac{|B|}{|I|} + (1 + \alpha(\epsilon))\frac{|S \setminus B|}{|I|}$$

Using $|B| = |E(G)| \leq 3|V(G)|/2$ and $|I| \geq |V(G)|/4$, we have

$$\frac{\mathrm{opt}(G)}{|I|} \leq \alpha(\epsilon) \cdot 6 + (1 + \alpha(\epsilon))\frac{|S \setminus B|}{|I|}.$$

Now, if $|S \setminus B| \leq |V(G)|/4$ or $B \subseteq S$, then, by definition of h', it holds $|S \setminus B|/|I| \leq 1$, and thus

$$\frac{\text{opt}(G)}{|I|} \leq \alpha(\epsilon) \cdot 6 + (1 + \alpha(\epsilon)) = 1 + 7\alpha(\epsilon) = 1 + \epsilon.$$

If $B \not\subseteq S$ and $|S \setminus B| > |V(G)|/4$, then consider the set S' from which I is determined by definition of h' and note that since $|S'| \geq |S|$, we also have

$$\frac{\text{opt}(P)}{|S'|} \leq (1 + \alpha(\epsilon)) \iff \frac{\text{opt}(G)}{|I|} \leq \alpha(\epsilon) \cdot 6 + (1 + \alpha(\epsilon))\frac{|S' \setminus B|}{|I|}.$$

By definition of h', it holds $|S' \setminus B| \leq |I|$ and thus

$$\frac{\text{opt}(G)}{|I|} \leq 1 + 7\alpha(\epsilon) = 1 + \epsilon.$$

□

Finally, we prove that transformation Φ yields a polynomial-time many-one reduction from IS-3 to Maximum General Position Subset Selection, where the number of points in the point set depends linearly on the number of vertices in the graph. This implies an exponential-time lower bound based on the Exponential Time Hypothesis (ETH) [IPZ01].

Lemma 3.6. *There is a polynomial-time many-one reduction from* Independent Set *on graphs with maximum degree three to* General Position Subset Selection *mapping a graph G to a point set P of size $O(|V(G)|)$.*

Proof. For an instance (G, k) of Independent Set, where G has maximum degree three, the point set $P := \Phi(G)$ is of cardinality $|P| = |V(G)| + |E(G)| \leq |V(G)| + 3|V(G)|/2 \in O(|V(G)|)$. By the proof of Lemma 3.3, Φ is a polynomial-time many-one reduction. □

We summarize the consequences of Lemmas 3.3, 3.5 and 3.6 in the following theorem:

Theorem 3.7. *The following statements are true:*
(a) General Position Subset Selection *is NP-hard.*
(b) Maximum General Position Subset Selection *is APX-hard.*

(c) Unless ETH fails, General Position Subset Selection *is not solvable in* $2^{o(n)} \cdot n^{O(1)}$ *time, where* n *is the number of points in the input.*

The above hardness results even hold for instances in which no four points are collinear.

Proof. Part (a) follows from the NP-hardness of Independent Set [GJ79], combined with Proposition 3.1 and Lemma 3.3. Part (b) follows from the APX-hardness of IS-3 [AK00], combined with Proposition 3.1 and Lemma 3.5. Concerning Part (c), it is well known that, unless the ETH fails, Independent Set is not solvable in sub-exponential time [IPZ01], and the same is true for the restriction to graphs of maximum degree three by the result of Johnson and Szegedy [JS99]. Hence, by the reduction in Lemma 3.6, General Position Subset Selection cannot be solved in sub-exponential time since this would imply a sub-exponential-time algorithm for Independent Set on graphs of maximum degree three.

Parts (a)–(c) remain true for instances in which no four points are collinear because the point set produced by transformation Φ satisfies this property (see Proposition 3.1). □

3.3 Fixed-Parameter Tractability

In this section, we prove several fixed-parameter tractability results for General Position Subset Selection. In Section 3.3.1 we develop problem kernels with a cubic number of points for the parameter size k of the sought subset in general position, and for the line cover number ℓ. In Section 3.3.2, we show a quadratic-size problem kernel with respect to the *dual parameter* $h := n - k$, that is, the number of points whose deletion leaves a set of points in general position. Moreover, we prove that this problem kernel is essentially optimal, unless coNP $\subseteq$ NP/poly (which implies an unlikely collapse in the polynomial hierarchy).

3.3.1 Fixed-Parameter Tractability Results for the Parameter Solution Size k

Let (P, k) be an instance of General Position Subset Selection, and let $n = |P|$. Cao [Cao12] gave a problem kernel for General Position Subset Selection with $O(k^4)$ points based on the following idea. Suppose that there

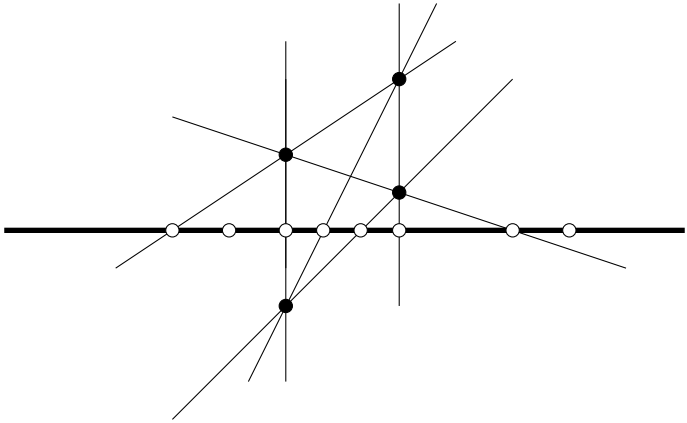


Figure 3.3: Example of Reduction Rule 3.1 for $k = 6$. The thick horizontal line L contains $\binom{6-2}{2} + 2 = 8$ points. Any two of the other four (black) points in general position are collinear with only one point on L. Hence, there are two points on L forming a subset in general position together with the four other points.

is a line L containing at least $\binom{k-2}{2} + 2$ points from P. For any subset $S' \subset P$ in general position with $|S'| = k - 2$, there can be at most $\binom{k-2}{2}$ points on L such that each is collinear with two points in S'. Hence, we can always find at least two points on L that together with the points in S' form a subset S in general position of cardinality k (see Figure 3.3 for an example). Based on this idea, Cao [Cao12] introduced the following data reduction rule:

Reduction Rule 3.1 ([Cao12])**.** *Let (P, k) be an instance of* General Position Subset Selection*. If there is a line L that contains at least $\binom{k-2}{2} + 2$ points from P, then remove all points on L and set $k := k - 2$.*

Cao showed that Reduction Rule 3.1 can be applied in $O(n^3)$ time ([Cao12, Lemma B.1.]), and he showed its correctness, that is, applying Reduction Rule 3.1 to an instance (P, k) yields an instance (P', k') which is a yes-instance if and only if (P, k) is ([Cao12, Theorem B.2.]). Using Reduction Rule 3.1, he gave a problem kernel for General Position Subset Selection containing $O(k^4)$ points that is computable in $O(n^3 k)$ time ([Cao12, Theorem B.3.]). We shall improve on Cao's result, both in terms of the kernel size and the running time of the kernelization algorithm. We start by showing how, using a result

by Guibas et al. [GOR96, Theorem 3.2], Reduction Rule 3.1 can be applied exhaustively in $O(n^2 \log n)$ time. Notably, the idea of reducing lines with many points (based on Guibas et al. [GOR96]) also yields a kernelization result for Point Line Cover [LM05].

Lemma 3.8. *For an instance (P, k) of* General Position Subset Selection *where $|P| = n$, we can compute in $O(n^2 \log n)$ time an equivalent instance (P', k') such that the collinearity of P' is at most $\binom{k-2}{2} + 1$.*

Proof. Let $\lambda = \binom{k-2}{2} + 2$. We start by computing the set $\mathcal{L}$ of all lines that contain at least λ points from P. By a result of Guibas et al. [GOR96, Theorem 3.2], this can be performed in $O(n^2 \log(n/\lambda)/\lambda)$ time (the algorithm also yields for every such line the points of P lying on that line). We then iterate over each line $L \in \mathcal{L}$, checking whether L, at the current iteration, still contains at least λ points; if it does, we remove all points on L from P and decrement k by 2. For each line L, the running time of the preceding step is $O(\lambda)$, which is the time to check whether L contains at least λ points. Additionally, we might need to remove all points on L. If k reaches zero, then we can return a trivial yes-instance (P', k') of General Position Subset Selection in constant time. Otherwise, after iterating over all lines in $\mathcal{L}$, by Reduction Rule 3.1, the resulting instance (P', k') is an equivalent instance to (P, k) satisfying that no line in P' contains λ points, and hence the collinearity of P' is at most $\binom{k-2}{2} + 1$. Overall, the above can be implemented in time $O((n^2 \log(n/\lambda)/\lambda) \cdot \lambda) \subseteq O(n^2 \log n)$. □

We move on to improving the size of the problem kernel. Payne and Wood [PW13, Theorem 2.3] proved a lower bound on the maximum cardinality of a subset in general position when an upper bound on the collinearity of the point set is known. We show next how to obtain a kernel for General Position Subset Selection containing $O(k^3)$ points based on this result.

Theorem 3.9. General Position Subset Selection *admits a problem kernel containing $O(k^3)$ points that is computable in $O(n^2 \log n)$ time.*

Proof. By Lemma 3.8, after $O(n^2 \log n)$ preprocessing time, we can either return an equivalent yes-instance of (P, k) of constant size, or obtain an equivalent instance with a point set of cardinality $O(k^4)$ ([Cao12, Theorem B.3.]) and collinearity at most $\binom{k-2}{2} + 1$. Hence, without loss of generality, we assume in what follows that $|P| \in O(k^4)$ and that the collinearity of P is $\lambda \leq \binom{k-2}{2} + 1$.

Payne and Wood [PW13, Theorem 2.3] showed that any set of n points whose collinearity is at most λ contains a subset of points in general position of size at least

$$\frac{\alpha n}{\sqrt{n \ln \lambda + \lambda^2}}$$

for some constant $\alpha > 0$. Since $\lambda \leq \binom{k-2}{2} + 1 \leq k^2$, we have at least

$$\frac{\alpha n}{\sqrt{2n \ln k + k^4}}$$

points in general position. Note that this number is monotonically increasing in n. Thus, if $n \geq c_n k^3$ for a constant $c_n > 0$, then there are at least

$$\frac{\alpha c_n k^3}{\sqrt{2c_n k^3 \ln k + k^4}}$$

points in general position. Since $2c_n k^3 \ln k \in o(k^4)$, it follows that for large enough $k \geq c_k$ (for some constant $c_k > 0$), the number of points in general position is at least

$$\frac{\alpha c_n k^3}{\sqrt{2k^4}} = \frac{\alpha c_n k^3}{\sqrt{2}k^2} = \frac{\alpha c_n}{\sqrt{2}} \cdot k.$$

Hence, for $c_n \geq \sqrt{2}/\alpha$, the number of points in general position is at least k. Thus, we derived that there exists a constant c_n (depending on α) such that there exists a constant c_k (depending on c_n) such that, for $n \geq c_n k^3$ and $k > c_k$, there exist at least k points in general position.

The kernelization algorithm distinguishes the following three cases: First, if $k \leq c_k$, then the instance is of constant size and the algorithm decides it in $O(1)$ time and returns an equivalent constant-size instance. Second, if $k > c_k$ and $|P| \geq c_n k^3$, then the algorithm returns a trivial yes-instance of constant size. Third, if none of the two above cases applies, then the algorithm returns the (preprocessed) instance (P, k) which satisfies $|P| \in O(k^3)$. □

We can derive the following result by a brute-force algorithm on the above problem kernel:

Corollary 3.10. General Position Subset Selection *can be solved in* $O(n^2 \log n + \alpha^k k^{2k})$ *time for some constant* α.

Proof. Let (P, k) be an instance of General Position Subset Selection. By Theorem 3.9, after $O(n^2 \log n)$ preprocessing time, we can assume that $|P| \leq ck^3$ for some constant $c > 0$. We enumerate every subset of size k in P, and for each such subset, we use the result of Guibas et al. [GOR96, Theorem 3.2] to check in $O(k^2 \log k)$ time whether the subset is in general position. If we find such a subset, then we answer positively; otherwise, we answer negatively. The number of enumerated subsets is

$$\binom{|P|}{k} \leq \binom{ck^3}{k} \leq \frac{(ck^3)^k}{k!} \leq \frac{(ck^3)^k}{(k/e)^k} \leq (ce)^k k^{2k},$$

where e is the base of the natural logarithm and $k! \geq (k/e)^k$ follows from Stirling's formula [Rob55]. Putting everything together, we obtain an algorithm for General Position Subset Selection with a running time in

$$O(n^2 \log n + (ce)^k k^{2k} \cdot k^2 \log k) \subseteq O(n^2 \log n + \alpha^k k^{2k})$$

for some constant $\alpha > ce$. □

We can transfer the above results for the parameter k to the *line cover number* ℓ (that is, the minimum number of lines that cover all points in the point set) using the following observation made by Cao [Cao12].

Observation 3.11 ([Cao12])**.** *For a set P of points let $S \subseteq P$ be a maximum subset in general position and let ℓ be the line cover number of P. Then, $\sqrt{\ell} \leq |S| \leq 2\ell$.*

As a consequence of Observation 3.11, we can assume that $k \leq 2\ell$ and, thus, the following corollary is immediate.

Corollary 3.12. General Position Subset Selection *can be solved in $O(n^2 \log n + \alpha^{2\ell} \cdot (2\ell)^{4\ell})$ time, and there is a kernelization algorithm that, given an instance (P, k) of* General Position Subset Selection*, computes an equivalent instance containing $O(\ell^3)$ points in $O(n^2 \log n)$ time.*

We close this section with an improved problem kernel for a special case of General Position Subset Selection. In practice, it might be unlikely that many points lie on the same line (for example, when the data points are measurements involving some inaccuracy). That is, the collinearity λ of the point set will be small. Note that if λ is bounded by a constant, then

Observation 3.11 implies an $O(k^2)$-point kernel (the line cover number is at least n/λ, thus, there are $\Omega(\sqrt{n})$ points in general position). For the case $\lambda = 3$, we can show an even smaller problem kernel.

Let 3-General Position Subset Selection denote the restriction of General Position Subset Selection to instances in which the point set contains no four collinear points (that is, collinearity $\lambda \leq 3$). By Theorem 3.7, 3-General Position Subset Selection is NP-hard. Füredi [Für91, Theorem 1] showed that every set P of n points in which no four points are collinear contains a subset in general position of size $\Omega(\sqrt{n \log n})$. Based on Füredi's result, we get the following:

Theorem 3.13. *3-General Position Subset Selection admits a problem kernel containing $O(k^2/\log k)$ points that is computable in $O(n)$ time.*

Proof. Let (P, k) be an instance of 3-General Position Subset Selection with $|P| = n$. Note that the line cover number ℓ of P is at least $n/3$ since at most three points lie on the same line. Therefore, if $n \geq 3k^2$, then by Observation 3.11 we know that P contains at least k points in general position. Thus, we can return a trivial constant-size yes-instance.

We henceforth assume that $n < 3k^2$. Then, there exists a constant $\alpha > 0$ such that P contains at least $\alpha\sqrt{n \log n}$ points in general position [Für91, Theorem 1]. Let $n \geq c_n k^2/\log k$ for some constant $c_n > 0$. Then, P contains at least

$$\alpha\sqrt{\frac{c_n k^2}{\log k}\log\left(\frac{c_n k^2}{\log k}\right)} = \alpha\sqrt{c_n k^2\left(2 + \frac{\log c_n - \log\log k}{\log k}\right)}$$

points in general position. Since there exists a constant c_k (depending on c_n) such that

$$\frac{\log c_n - \log\log k}{\log k} \geq -1$$

holds for all $k \geq c_k$, it follows that there are at least $\alpha\sqrt{c_n k^2(2-1)} = \alpha\sqrt{c_n} \cdot k$ points in general position for large enough k. Hence, for $c_n \geq \alpha^{-2}$, we have a yes-instance.

The kernelization algorithm works as follows: If $k < c_k$, then P is of constant size and we decide the instance in constant time returning an equivalent constant-size instance. If $k \geq c_k$ and $n \geq c_n k^2/\log k$, then we return a constant-size yes-instance. Otherwise, we simply return the instance (P, k) with $|P| \in O(k^2/\log k)$. The running time is in $O(n)$ for counting the number of points in the input. □

We remark that the presented problem kernels (relying on non-constructive lower bounds) only can be used to solve the decision problem, that is, they do not allow to actually find a point set in general position. It is not clear how to constructively find the points in general position for large yes-instances.

3.3.2 Fixed-Parameter Tractability Results for the Dual Parameter h

In this section we consider the dual parameter number $h := n - k$ of points that have to be *deleted* (that is, excluded from the sought point set in general position) so that the remaining points are in general position. We show a problem kernel containing $O(h^2)$ points for General Position Subset Selection. Moreover, we show that most likely this problem kernel is essentially *tight*, that is, there is presumably no problem kernel with $O(h^{2-\epsilon})$ points for any $\epsilon > 0$.

We start with the problem kernel that relies essentially on a problem kernel for the 3-Hitting Set problem:

3-Hitting Set
Input: A universe U, a collection $\mathcal{C}$ of size-3 subsets of U, and $h \in \mathbb{N}$.
Question: Is there a subset $H \subseteq U$ of size at most h containing at least one element from each subset $S \in \mathcal{C}$?

There is a close connection between General Position Subset Selection and 3-Hitting Set: For any collinear triple $p, q, r \in P$ of distinct points, one of the three points has to be deleted in order to obtain a subset in general position. Hence, the set of deleted points has to be a hitting set for the family of all collinear triples in P. Since 3-Hitting Set can be solved in $O(2.08^h + |\mathcal{C}| + |U|)$ time [Wah07], we get:

Proposition 3.14. General Position Subset Selection *can be solved in* $O(2.08^h + n^3)$ *time.*

3-Hitting Set is known to admit a problem kernel with a universe of size $O(h^2)$ computable in $O(|U| + |\mathcal{C}| + h^{1.5})$ time [Bev14]. Based on this, one can obtain a problem kernel of size $O(h^2)$ computable in $O(n^3)$ time. The bottleneck in this running time is listing all collinear triples. We can improve the running time of this kernelization algorithm by giving a direct kernel exploiting the simple geometric fact that two non-parallel lines intersect in one point. We first need two data reduction rules for which we introduce the following definition:

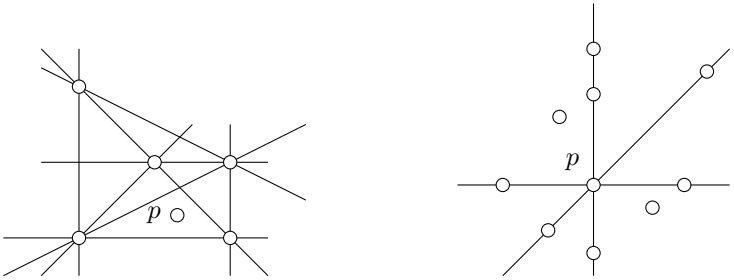


Figure 3.4: Examples illustrating Reduction Rules 3.2 and 3.3. Left: A point p which is not in conflict with any of the other points is always contained in a maximum subset in general position (Reduction Rule 3.2). Right: The point p is in conflict with seven other points on three lines. Either p is deleted or, from each of the three lines, at least all but one of the other points are deleted. Hence, if $h < 4$, then p has to be removed (Reduction Rule 3.3).

Definition 3.2. For a set P of points in the plane, we say that a point $p \in P$ is in *conflict* with a point $q \in P$ if there is a third point $z \in P$ such that p, q, and z are collinear.

Reduction Rule 3.2. *Let (P, k) be an instance of* General Position Subset Selection. *If there is a point $p \in P$ that is not in conflict with any other points in P, then delete p and decrease k by one.*

Clearly, Reduction Rule 3.2 is correct since we can always add a point which is not lying on any line defined by two other points to a general position subset. The next rule deals with points that are in conflict with too many other points. The basic idea here is that if a point lies on more than h distinct lines defined by two other points of P, then it has to be deleted (see Figure 3.4). This is formalized in the next rule.

Reduction Rule 3.3. *Let (P, k) be an instance of* General Position Subset Selection. *For a point $p \in P$, let $\mathcal{L}(p)$ be the set of lines containing p and at least two points of $P \setminus \{p\}$, and for $L \in \mathcal{L}(p)$ let $|L|$ denote the number of points of P on L. Then, delete each point $p \in P$ satisfying $\sum_{L \in \mathcal{L}(p)} (|L| - 2) > h$.*

Lemma 3.15. *Let (P, k) be a* General Position Subset Selection *instance and let (P', k) be the resulting instance after applying Reduction Rule 3.3 to (P, k). Then, (P, k) is a yes-instance if and only if (P', k) is a yes-instance.*

Proof. Let (P, k) be an instance of General Position Subset Selection and let $(P' := P \setminus D, k)$ be the reduced instance, where $D \subseteq P$ denotes the set of removed points.

Clearly, if (P', k) is a yes-instance, then so is (P, k). For the converse, we show that no point $p \in D$ can be contained in a size-k subset of P in general position: For each line $L \in \mathcal{L}(p)$, all but two points need to be deleted. If a subset $S \subseteq P$ in general position contains p, then the points that have to be deleted on the lines in $\mathcal{L}(p)$ are all distinct since any two of these lines only intersect in p. This means that $\sum_{L \in \mathcal{L}(p)}(|L| - 2)$ points need to be deleted. However, since this value is by assumption larger than h, the solution S is of size less than $k = |P| - h$. □

Theorem 3.16. General Position Subset Selection *admits a problem kernel containing at most* $2h^2 + h$ *points that is computable in* $O(n^2)$ *time, where* $h = n - k$.

Proof. Let (P, k) be a General Position Subset Selection instance. We first show that applying Reduction Rule 3.2 exhaustively and then applying Reduction Rule 3.3 once indeed gives a small instance (P', k'). Note that each point $p \in P'$ is in conflict with at least two other points, that is, p is on at least one line containing two other points in P', since the instance is reduced with respect to Reduction Rule 3.2. Moreover, since the instance is reduced with respect to Reduction Rule 3.3, it follows that each point is in conflict with at most $2h$ other points. Thus, deleting h points can give at most $h \cdot 2h$ points in general position. Hence, if P' contains more than $2h^2 + h$ points, then the input instance is a no-instance.

We next show how to exhaustively apply Reduction Rules 3.2 and 3.3 in $O(n^2)$ time. To this end, we follow an approach described by Edelsbrunner et al. [EOS86] and Gómez et al. [GRT97] which uses the dual representation and line arrangements. The dual representation maps points to lines as follows: $(a, b) \mapsto y = ax + b$. A line in the primal representation containing some points of P corresponds in the dual representation to the intersection point of the lines corresponding to these points. Thus, a set of at least three collinear points in the primal corresponds to the intersection of the corresponding lines in the dual. An *arrangement* of lines in the plane is, roughly speaking, the partition of the plane formed by these lines. A representation of an arrangement of n lines can be computed in $O(n^2)$ time [EOS86]. Using an algorithm of Edelsbrunner et al. [EOS86], we compute in $O(n^2)$ time the arrangement $A(P^*)$ of the lines P^* in the dual representation of P.

Reduction Rule 3.2 is now easily executable in $O(n^2)$ time: Initially, mark all points in P as "not in conflict". Then, iterate over the vertices of $A(P^*)$ and whenever the vertex has degree six or more (each line on an intersection contributes two to the degree of the corresponding vertex) mark the points corresponding to the intersecting lines as "in conflict". In a last step, remove all points that are marked as "not in conflict".

Reduction Rule 3.3 can be applied in a similar fashion in $O(n^2)$ time: Assign a counter to each point $p \in P$ and initialize it with zero. We want this counter to store the number $\sum_{L \in \mathcal{L}(p)}(|L| - 2)$ on which Reduction Rule 3.3 is conditioned. To this end, we iterate over the vertices in $A(P^*)$ and for each vertex of degree six or more (each line contributes two to the degree of the intersection vertex) we increase the counter of each point corresponding to a line in the intersection by $d/2 - 2$ where d is the degree of the vertex. After one pass over all vertices in $A(P^*)$ in $O(n^2)$ time, the counters of the points store the correct values and we can delete all points whose counter is more than h. □

Another interesting parameter for geometry problems is the number γ of *inner* points (that is, all points which are not a vertex of the convex hull of P), which successfully has been deployed algorithmically [Deĭ+06, HO06]. Note that Proposition 3.14 and Theorem 3.16 also hold if we replace the parameter h by γ. The reason is that in all non-trivial instances we have $h \leq \gamma$ since removing all inner points yields a set of points in general position.

3.3.3 Excluding $O(h^{2-\epsilon})$-point Kernels

In this section we show that the $O(h^2)$-point problem kernel of Theorem 3.16 is essentially optimal. We start by proving a (conditional) lower bound on the problem kernel *size* (the number of encoding bits) for General Position Subset Selection via a reduction from Vertex Cover.

Vertex Cover
Input: An undirected graph G and $k \in \mathbb{N}$.
Question: Is there a subset $V' \subseteq V(G)$ of size at most k such that every edge is incident to at least one vertex in V' ?

Using a lower bound result by Dell and Melkebeek [DM14] for Vertex Cover, we obtain the following:

Theorem 3.17. *Unless coNP $\subseteq$ NP/poly, for any $\epsilon > 0$,* General Position Subset Selection *admits no problem kernel of size $O(h^{2-\epsilon})$.*

Proof. We give a polynomial-time many-one reduction from Vertex Cover, where the resulting dual parameter h equals the size of the sought vertex cover. The claimed lower bound then follows because, unless coNP $\subseteq$ NP/poly, for any $\epsilon > 0$, Vertex Cover admits no problem kernel of size $O(k^{2-\epsilon})$, where k is the size of the vertex cover [DM14].

Given a Vertex Cover instance (G, k), we first reduce it to the equivalent Independent Set instance $(G, |V(G)| - k)$. We then apply transformation Φ (see Section 3.2) to G to obtain a set of points P, where $|P| = |V(G)| + |E(G)|$; we set $k' := |V(G)| + |E(G)| - k$, and consider the instance (P, k') of General Position Subset Selection. Clearly, G has a vertex cover of cardinality k if and only if G has an independent set of cardinality $|V(G)| - k$, which, by Lemma 3.3, is true if and only if P has a subset in general position of cardinality $|E(G)| + |V(G)| - k$. Hence, the dual parameter $h = |P| - k'$ equals the sought vertex cover size. □

Note that Theorem 3.17 gives a lower bound only on the total size (that is, the number of bits required to encode the instance) of a problem kernel for General Position Subset Selection. The number of points in the instance, however, is of course smaller than the total size since each point requires more than one bit of encoding. Hence, proving a lower bound on the number of points in a problem kernel is an even stronger result. We show a lower bound on the number of points contained in any problem kernel using ideas due to Kratsch et al. [KPR16], which are based on a lower bound framework by Dell and Melkebeek [DM14]. Kratsch et al. [KPR16] showed that there is no polynomial-time algorithm that reduces a Point Line Cover instance (P, k) to an equivalent instance with $O(k^{2-\epsilon})$ points for any $\epsilon > 0$ unless coNP $\subseteq$ NP/poly. The proof is based on a result by Dell and Melkebeek [DM14] who showed that Vertex Cover does not admit a so-called *oracle communication protocol* of cost $O(k^{2-\epsilon})$ for $\epsilon > 0$ unless coNP $\subseteq$ NP/poly. An oracle communication protocol is a two-player protocol, in which the first player is holding the input and is allowed polynomial (computational) time in the length of the input, and the second player is computationally unbounded. The *cost* of the communication protocol is the number of bits communicated from the first player to the second player in order to solve the input instance.

Kratsch et al. [KPR16] devise an oracle communication protocol with cost $O(n \log n)$ for deciding instances of Point Line Cover with n points. Thus, a problem kernel for Point Line Cover with $O(k^{2-\epsilon})$ points implies an oracle communication protocol of cost $O(k^{2-\epsilon'})$ for some $\epsilon' > 0$ since the first

player could simply compute the kernelized instance in polynomial time and subsequently follow the protocol yielding a cost of $O(k^{2-\epsilon} \cdot \log(k^{2-\epsilon}))$, which is in $O(k^{2-\epsilon'})$ for some $\epsilon' > 0$. This again would imply an $O(k^{2-\epsilon''})$-cost oracle communication protocol for Vertex Cover for some $\epsilon'' > 0$ (via a polynomial-time many-one reduction with a linear parameter increase [KPR16, Lemma 6]). We show that there exists a similar oracle communication protocol of cost $O(n \log n)$ for General Position Subset Selection.

The protocol is based on *order types* of point sets. Let $P = \langle p_1, \ldots, p_n \rangle$ be an ordered set of points and denote by $\binom{[n]}{3}$ the set of ordered triples $\langle i, j, k \rangle$ where $i < j < k$, $i, j, k \in [n] := \{1, \ldots, n\}$. The *order type* of P is a function $\sigma : \binom{[n]}{3} \to \{-1, 0, 1\}$, where $\sigma(\langle i, j, k \rangle)$ equals 1 if p_i, p_j, p_k are in counter-clockwise order, equals -1 if they are in clockwise order, and equals 0 if they are collinear. Two point sets P and Q of the same cardinality are *combinatorially equivalent* if there exist orderings P' and Q' of P and Q such that the order types of P' and Q' are identical.

A key step in the development of an oracle communication protocol is to show that two instances of Point Line Cover with combinatorially equivalent point sets are actually equivalent [KPR16, Lemma 2]. We can prove an analogous result for General Position Subset Selection:

Observation 3.18. *Let (P, k) and (Q, k) be two instances of* General Position Subset Selection. *If the point sets P and Q are combinatorially equivalent, then (P, k) and (Q, k) are equivalent instances of* General Position Subset Selection.

Proof. Let P and Q be combinatorially equivalent point sets with $|P| = |Q| = n$ and let $P' = \langle p_1, \ldots, p_n \rangle$ and $Q' = \langle q_1, \ldots, q_n \rangle$ be orderings of P and Q, respectively, having the same order type σ.

Now, a subset $S \subseteq P'$ is in general position if and only if no three points in S are collinear, that is, $\sigma(\langle p_i, p_j, p_k \rangle) \neq 0$ holds for all $p_i, p_j, p_k \in S$. Consequently, it holds that $\sigma(\langle q_i, q_j, q_k \rangle) \neq 0$, and thus the subset $\{q_i \mid p_i \in S\} \subseteq Q'$ is in general position. Hence, (P, k) is a yes-instance if and only if (Q, k) is a yes-instance. □

Based on Observation 3.18, we obtain an oracle communication protocol for General Position Subset Selection. The proof of the following lemma is completely analogous to the proof of Lemma 4.1 in [KPR16]:

Lemma 3.19. *There is an oracle communication protocol of cost $O(n \log n)$ for deciding instances of* General Position Subset Selection *with n points.*

The basic idea of the proof for Lemma 3.19 is that the first player only sends the order type of the input point set so that the computationally unbounded second player can solve the instance (according to Observation 3.18 the order type contains enough information to solve a General Position Subset Selection instance). We conclude with the following lower bound result based on Lemma 3.19.

Theorem 3.20. *Let $\epsilon > 0$. Unless coNP $\subseteq$ NP/poly, there is no polynomial-time algorithm that reduces an instance (P, k) of* General Position Subset Selection *to an equivalent instance with $O(h^{2-\epsilon})$ points.*

Proof. Assuming that such an algorithm exists, the oracle communication protocol of Lemma 3.19 has cost $O(h^{2-\epsilon'})$ for some $\epsilon' > 0$. Since the reduction from Vertex Cover in Theorem 3.17 outputs a General Position Subset Selection instance where the dual parameter h equals the size k of the vertex cover sought, we obtain a communication protocol for Vertex Cover of cost $O(k^{2-\epsilon'})$, which implies that coNP $\subseteq$ NP/poly [DM14, Theorem 2]. □

A Kernel Lower Bound Framework Based on Kratsch et al. [KPR16]. As a final observation, we mention that the results of Kratsch et al. [KPR16] are indeed more generally applicable than stated there since their arguments only rely on the equivalence of instances with respect to order types of point sets.

Observation 3.21. *A parameterized problem on a point set for which*

(i) two instances with combinatorially equivalent point sets are equivalent (see Observation 3.18), and

(ii) there is no oracle communication protocol of cost $O(k^{2-\epsilon})$ for some parameter k and any $\epsilon > 0$ unless coNP $\subseteq$ NP/poly,

has no problem kernel containing $O(k^{2-\epsilon'})$ points for any $\epsilon' > 0$ unless coNP $\subseteq$ NP/poly.

This framework has recently been applied to prove lower bounds for problem kernels for the extension of General Position Subset Selection to d-dimensional point sets (called Hyperplane Subset General Position) [Boi+17].

3.4 Conclusion

We studied the computational complexity of General Position Subset Selection and proved its (approximation) intractability. Then, we started a parameterized complexity analysis obtaining several fixed-parameter tractability results mainly based on problem kernelizations. The considered parameters are the solution size k, its dual h, the line cover number ℓ and the number γ of inner points. The kernelization results mostly rely on *combinatorial* arguments (similar ideas were recently used to develop kernelizations for generalizing General Position Subset Selection to d dimensions [Boi+17]). Thus, a natural question to ask is whether there are any *geometric* properties that can be exploited in order to obtain further improved algorithmic results for General Position Subset Selection. For practical purposes, it would also be interesting to find out which data reduction rules can be applied in linear time.

We conclude with some further questions:

- Can the $O(k^3)$-point kernel (Theorem 3.9) for General Position Subset Selection be asymptotically improved (for example by proving stronger combinatorial bounds or by finding further data reduction rules)? Or can we derive a quadratic (or even a cubic) lower bound on the kernel size (or even on the number of points)?
- For which other problems on point sets can we derive kernel lower bounds using the framework described in Observation 3.21?
- What are further interesting parameterizations for General Position Subset Selection? Maybe there exist other (geometrically motivated) parameters which allow for (stronger) fixed-parameter tractability results. Regarding parameterization above a lower bound (as studied in Chapter 6), however, note that the W[1]-hardness of the extension variant (Observation 3.4) already indicates intractability.

More generally, studying the complexity of other computational geometry problems from a parameterized view (as done by Knauer et al. [KKW15]) could be a rewarding direction for future research. For example, a special case of the Co-Clustering$_\infty$ problem studied in Chapter 5 boils down to the geometric problem of separating points in the plane with few axis-parallel lines. Maybe it is also possible to show fixed-parameter tractability with respect to the number of lines for this problem based on combinatorial results.

Chapter 4

Distinct Vectors

In this chapter, we study the Distinct Vectors problem which asks to delete as many columns as possible from a given matrix such that all rows in the resulting matrix are still pairwise distinct. This NP-complete problem arises in the context of feature selection in data mining and machine learning. Its (classic) computational complexity has previously been studied. We conduct a fine-grained complexity analysis with respect to the maximum (H) and the minimum (h) pairwise Hamming distance between matrix rows. Based on this combined parameterization, we prove a dichotomy result for binary matrices: Distinct Vectors can be solved in polynomial time if $H \leq 2\lceil h/2 \rceil + 1$, and is NP-complete otherwise. Interestingly, as in Chapter 3, we again make use of a combinatorial result: The polynomial-time solvable cases are based on a result from extremal set theory concerning certain set systems called *sunflowers*[1]. Moreover, we reveal a connection to matchings in bipartite graphs.

4.1 Introduction

Feature selection in a high-dimensional feature space means to choose a subset of features (that is, dimensions) such that some desirable data properties are preserved or achieved in the induced subspace. *Combinatorial* feature selection [Cha+00, KS96] is a well-motivated alternative to the more frequently studied affine feature selection. While *affine* feature selection combines features to reduce dimensionality, combinatorial feature selection simply discards some features. The advantage of the latter is that the resulting reduced feature space is easier to interpret. See Charikar et al. [Cha+00] for a more extensive discussion in favor of combinatorial feature selection. Unfortunately, combinatorial feature

[1] Note that the concept of sunflowers has been used before in parameterized complexity (for example to show problem kernels [FG06, Chapter 9]).

1.0	88.1	+	2.0	A	-1.0	x
0.0	88.1	+	2.0	A	-1.0	x
1.0	97.3	−	1.7	C	5.0	y
1.0	51.7	−	1.7	B	5.0	y
1.0	97.3	+	2.7	B	-3.0	y

1.0	+	A
0.0	+	A
1.0	−	C
1.0	−	B
1.0	+	B

Figure 4.1: An illustration of the DISTINCT VECTORS problem: For the given 5×7 input matrix S (left), it is possible to select the three gray columns in order to distinguish all rows (that is, $K = \{1, 3, 5\}$ is a solution). All rows in the matrix $S_{|K}$ (right) are pairwise distinct. Note that there is no solution containing only two column indices.

selection problems are typically computationally very hard to solve (NP-hard and also hard to approximate [Cha+00, CS14, DK08, GE03]), resulting in the use of heuristic approaches in practice [BL97, Das+07, For03, MR15, Put+15].

We refine the analysis of the computational complexity landscape of an easy-to-define combinatorial feature selection problem called DISTINCT VECTORS [Cha+00].

DISTINCT VECTORS
Input: A matrix $S \in \Sigma^{n \times d}$ over a finite alphabet Σ with n distinct rows and $k \in \mathbb{N}$ with $1 \leq k \leq d$.
Question: Is there a subset $K \subseteq [d]$ of column indices with $|K| \leq k$ such that all n rows in $S_{|K}$ are still distinct?

Here, $S_{|K}$ is the $n \times |K|$ submatrix of S containing only the columns with indices in K. An example is given in Figure 4.1. In the above formulation, the input data is considered to be a matrix where the row vectors correspond to the *data points* and the columns represent *features* (dimensions). Thus, DISTINCT VECTORS constitutes the basic task to compress the data by discarding redundant or negligible dimensions without losing the essential information to tell apart all data points.

Related Work. DISTINCT VECTORS is a basic problem arising under several names in different areas. For instance, DISTINCT VECTORS is the problem of finding (candidate) keys in relational databases [Dat03], which is known to be

NP-complete [Bee+84]. It is also known as the Minimal Reduct problem in rough set theory [Paw91, SR92]. Later, Charikar et al. [Cha+00] investigated the computational complexity of several problems arising in the context of combinatorial feature selection, including Distinct Vectors. They showed that there exists a constant c such that Distinct Vectors is NP-hard to approximate in polynomial time within a factor of $c \log d$ on binary matrices. Moreover, Distinct Vectors is known [Fro+16a, Fro12] to be

- W[2]-hard with respect to k,
- W[1]-hard with respect to $d - k$,
- fixed-parameter tractable with respect to $(k, |\Sigma|)$ and also (k, H),
- NP-hard for $|\Sigma| = 2$ and $H \geq 4$,
- and polynomial-time solvable for $|\Sigma| = 2$ and $H \leq 3$.

Notably, Bläsius et al. [BFS17] studied the problem (among others) in the context of databases (therein called Unique) and also showed W[2]-hardness with respect to k.

Another combinatorial feature selection problem called Minimum Feature Set is a variant of Distinct Vectors where not all pairs of rows have to be distinguished but only all pairs of rows from two specified subsets. This problem is known to be NP-complete for binary input data [DR94]. In addition, Cotta and Moscato [CM03] investigated the parameterized complexity of Minimum Feature Set and proved W[2]-completeness with respect to the number of selected columns even for binary matrices.

Our Contributions. The already known NP-hardness for binary matrices with maximum pairwise row Hamming distance $H \geq 4$ [Fro12] serves as a starting point for our complexity analysis. With the aim of identifying tractable cases, we are particularly interested in the complexity of Distinct Vectors if the range of different Hamming distances between pairs of data points (that is, rows) is small as this special case implies a somewhat homogeneous structure of the input data. Hence, we also take into account the minimum pairwise row Hamming distance h. Our parameter is then defined as the gap $H - h$. We completely classify the classic complexity of Distinct Vectors with respect to all values of $H - h$ on binary input matrices (see Figure 4.2).

In Section 4.2.1, we show that binary Distinct Vectors is NP-hard if $H > 2\lceil h/2 \rceil + 1$ (Theorem 4.3) and in Section 4.2.2 we show polynomial-time solvability for $H \leq 2\lceil h/2 \rceil + 1$ (Theorem 4.9).

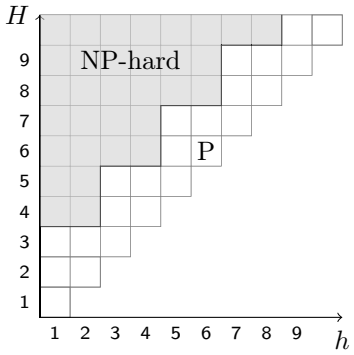


Figure 4.2: Overview of the computational complexity landscape of Distinct Vectors on binary matrices with respect to the maximum pairwise row Hamming distance H and the minimum pairwise row Hamming distance h. Gray cells correspond to NP-hard cases, whereas white cells correspond to polynomial-time solvable cases.

4.2 A Complexity Dichotomy for Binary Matrices

We focus on instances with a binary input alphabet, say, without loss of generality, $\Sigma = \{0, 1\}$. For a matrix $S \in \{0, 1\}^{n \times d}$ and a subset $J \subseteq [d]$ of column indices, we use the abbreviation $S_{|J} := S[[n], J]$ for the submatrix of S containing all the rows but only the columns with indices in J.

We consider instances where the Hamming distance of each pair of rows lies within a prespecified range. In other words, the number of columns in which a given pair of rows differs shall be bounded from below and above by some constants $\alpha, \beta \in \mathbb{N}$. We first give the formal definitions and then completely classify the classic complexity of Distinct Vectors with respect to the gap between α and β. The NP-hard cases are given in Section 4.2.1 and the polynomial-time solvable cases in Section 4.2.2. The formal definitions for our setup are the following.

Definition 4.1 (Weight)**.** For a vector $x \in \{0, 1\}^d$, we denote by $W_x := \{j \in [d] \mid (x)_j = 1\}$ the set of indices where x equals 1 and we call $w(x) := |W_x|$ the *weight* of x.

Definition 4.2 (Hamming Distance)**.** For vectors $x, y \in \Sigma^d$, let $D_{xy} := \{j \in [d] \mid (x)_j \neq (y)_j\}$ be the set of indices where x and y differ and let $\Delta(x, y) := |D_{xy}|$ denote the *Hamming distance* of x and y.

Note that, for $x, y \in \{0,1\}^d$, it holds $D_{xy} = (W_x \cup W_y) \setminus (W_x \cap W_y)$ and thus $\Delta(x, y) = w(x) + w(y) - 2|W_x \cap W_y|$.

For a Distinct Vectors instance $(S \in \Sigma^{n \times d}, k)$, we define the parameters

- *minimum pairwise row Hamming distance* $h := \min_{i \neq j \in [n]} \Delta(s_i, s_j)$ and
- *maximum pairwise row Hamming distance* $H := \max_{i \neq j \in [n]} \Delta(s_i, s_j)$.

To conveniently state our results, we define the following variant of Distinct Vectors:

Binary (α, β)-Distinct Vectors
Input: A matrix $S \in \{0,1\}^{n \times d}$ with n distinct rows such that $h = \alpha \leq \beta = H$, and $k \in \mathbb{N}$.
Question: Is there a subset $K \subseteq [d]$ of column indices with $|K| \leq k$ such that all rows in $S_{|K}$ are still distinct?

Intuitively, if the matrix consists of rows that are all "similar" to each other, one could hope to be able to solve the instance efficiently since there are at most β columns to choose from in order to distinguish two rows. The minimum pairwise row Hamming distance α plays a dual role in the sense that if α is large, then each pair of rows differs in many columns, which also could render the instance easy to solve. We perform a close inspection of the relation between the minimum and maximum pairwise row Hamming distance and show that it is possible to solve some cases in polynomial time for arbitrarily large constants α, β. These results are obtained by applying combinatorial arguments from extremal set theory revealing a certain structure of the input matrix if the values of α and β are close to each other, that is, the range $\beta - \alpha$ is small. Analyzing this structure, we can show how to find solutions in polynomial time. On the negative side, for all other cases NP-hardness still holds as we show in the next section.

4.2.1 NP-Hardness for Heterogeneous Data

As a starting point serves the NP-hardness of Binary $(2, 4)$-Distinct Vectors [Fro12, Theorem 4.4] ([Fro+16a, Theorem 1]). For the sake of being self-contained, we include the proof here.

Theorem 4.1 ([Fro+16a, Fro12]). Binary $(2,4)$-Distinct Vectors *is NP-hard.*

Proof. To prove NP-hardness, we give a polynomial-time many-one reduction from a special variant of the Independent Set problem in graphs, which is defined as follows.

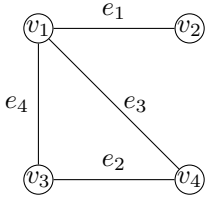
Distance-3 Independent Set
Input: An undirected graph $G = (V, E)$ and $k \in \mathbb{N}$.
Question: Is there a subset of vertices $I \subseteq V$ of size at least k such that each pair of vertices from I has distance at least three?

Here, the distance of two vertices is the number of edges contained in a shortest path between them. Distance-3 Independent Set is known to be NP-complete by a reduction from the NP-complete Induced Matching problem [BM11].

Our reduction works as follows: Let $(G = (V, E), k)$ with $|V| = n$ and $|E| = m$ be an instance of Distance-3 Independent Set and let $Z \in \{0,1\}^{m \times n}$ be the incidence matrix of G with rows corresponding to edges and columns to vertices, that is, $z_{ij} = 1$ means that the i-th edge contains the j-th vertex. We assume that G contains no isolated vertices since they are always contained in a maximum distance-3 independent set and can thus be removed. We further assume that G contains at least four edges of which at least two do not share an endpoint. Otherwise, G is either of constant size or a star, for which a maximum distance-3 independent set consists of only a single vertex. Hence, we can solve these cases in polynomial time and return a trivial yes- or no-instance.

The matrix $S \in \{0,1\}^{(m+1) \times n}$ of the Binary $(2,4)$-Distinct Vectors instance (S, k') is defined as follows: $s_i := z_i$ for all $i \in [m]$ and $s_{m+1} := \mathbf{0}$ (the length-n null vector). The desired solution size is set to $k' := n - k$. Note that each row in Z contains exactly two 1's and no two rows are equal since G contains no multiple edges. Moreover, by assumption, there exists a pair of rows with Hamming distance four since G contains a pair of edges without a common endpoint. Since matrix S additionally contains the null vector as a row, it follows that $h = 2$ and $H = 4$ (see Figure 4.3 for an example). The instance (S, k') can be computed in $O(nm)$ time.

The correctness of the reduction is due to the following argument. The instance (G, k) is a yes-instance if and only if there is a set $I \subseteq V$ of size exactly k such that every edge in G has at least one endpoint in $V \setminus I$ and no vertex in $V \setminus I$ has two neighbors in I. In other words, the latter condition



	v_1	v_2	v_3	v_4
e_1	1	1		
e_2			1	1
e_3	1			1
e_4	1		1	

Figure 4.3: Example of the construction in the proof of Theorem 4.1: An undirected graph G (left) and the corresponding binary matrix S with $h = 2$ and $H = 4$ (right). Empty cells in the matrix correspond to 0's.

says that no two edges with an endpoint in I share the same endpoint in $V \setminus I$. Equivalently, for the subset K of columns corresponding to the vertices in $V \setminus I$, it holds that all rows in $S_{|K}$ contain at least one 1 and no two rows contain only a single 1 in the same column. This is true if and only if K is a solution for (S, k') because s_{m+1} equals the null vector and thus two rows in $S_{|K}$ can only be identical if either they consist of 0's only or contain only a single 1 in the same column. Furthermore, $|K| = |V \setminus I| = n - k = k'$. □

We proceed with polynomial-time many-one reductions from Binary $(2, 4)$-Distinct Vectors to Binary (α, β)-Distinct Vectors for all combinations of α and β which are classified as NP-hard in Figure 4.2. The reductions will mainly build on some padding arguments, that is, starting from a given input matrix as constructed in the proof of Theorem 4.1, we expand it by adding new columns and rows such that we achieve the desired bounds on the Hamming distances without changing the actual answer to the original instance.

To start with, we define a type of column vectors which can be used for padding an input matrix without changing the answer to the original instance, that is, such "padding columns" are not contained in an optimal solution. Informally, a column j is *inessential* if all rows could still be distinguished by the same number of columns without selecting j. The formal definition is the following.

Definition 4.3. For a matrix $S \in \Sigma^{n \times d}$, a column $j \in [d]$ is called *inessential* if the following two conditions are fulfilled:

(1) There exists a row $i \in [n]$ such that column j *exactly distinguishes* row i from all other rows, that is, $s_{ij} \neq s_{lj}$ and $s_{lj} = s_{l'j}$ holds for all $l, l' \in [n] \setminus \{i\}$.

(2) All rows in $S_{|[d]\setminus\{j\}}$ are still distinct.

Note that for binary matrices, Condition (1) of Definition 4.3 can only be fulfilled by column vectors that contain either a single 1 or a single 0, that is, the column vectors of weight 1 or $n-1$. Next, we show that, for any inessential column in a given input matrix, we can assume that this column is not contained in a solution for the Distinct Vectors instance.

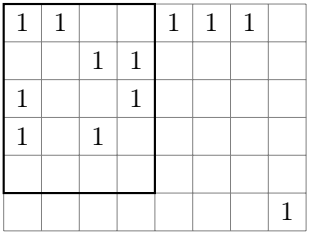
Lemma 4.2. *Let $(S \in \Sigma^{n\times d}, k)$ be a* Distinct Vectors *instance with an inessential column $j \in [d]$. Then (S, k) is a yes-instance if and only if $(S_{|[d]\setminus\{j\}}, k)$ is a yes-instance.*

Proof. It is clear that the "if" part of the statement holds; let us consider the "only if" part. To this end, assume that there is a solution set $K \subseteq [d]$ of columns for (S, k) with $j \in K$. Since column j exactly distinguishes row i from all other rows and no other pair of rows, it follows that $K' := K \setminus \{j\}$ is a solution for $(S[[n] \setminus \{i\}, [d] \setminus \{j\}], k-1)$. But then, there also exists a solution $K'' \subseteq [d] \setminus \{j\}$ for $(S[[n], [d] \setminus \{j\}], k)$. This is true because row i equals at most one other row l in $S[[n], K']$ since all rows in $S[[n] \setminus \{i\}, K']$ are distinct. Row i can thus be distinguished from row l by a column $j' \in [d] \setminus \{j\}$ with $s_{ij'} \neq s_{lj'}$, which exists because column j is inessential, and thus, by definition, all rows in $S[[n], [d] \setminus \{j\}]$ are distinct. Hence, $K'' := K' \cup \{j'\}$ is a solution for (S, k). □

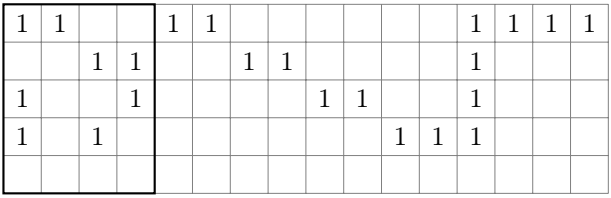
Note that, due to Lemma 4.2, adding inessential columns to a given input matrix yields an equivalent Distinct Vectors instance. Hence, for the binary case, any construction that only adds column vectors which either contain a single 1 or a single 0 to the input matrix yields an equivalent instance since these columns are clearly inessential. This allows us to increase the minimum and maximum pairwise row Hamming distances of a given binary matrix. Following this basic idea, we prove the following theorem.

Theorem 4.3. Binary (α, β)-Distinct Vectors *is NP-hard for all $\beta \geq \alpha+2$ if α is even, and for all $\beta \geq \alpha+3$ if α is odd.*

Proof. In the following, we give polynomial-time many-one reductions from Binary $(2, 4)$-Distinct Vectors. To this end, let $(S \in \{0,1\}^{n\times d}, k)$ be the Binary $(2, 4)$-Distinct Vectors instance as constructed in the proof of Theorem 4.1. Recall that this matrix S contains the null row vector, say $s_n = \mathbf{0}$, and all other rows have weight two, $w(s_i) = 2$ for all $i \in [n-1]$. Moreover,



(a) Case 1: $\alpha = 1$ and $\beta = 7$. By adding $b = 3$ copies of column vector $\mathbf{1}^5_{\{1\}}$, the maximum Hamming distance is increased to 7. Adding the bottom row and the last column yields a minimum Hamming distance of 1.



(b) Case 2: $\alpha = 5$ and $\beta = 9$, that is $a = b = 3$.

Figure 4.4: Examples of the padding construction. The submatrix framed by thick lines equals the example matrix in Figure 4.3.

there exists a pair of rows with Hamming distance four. Assume, without loss of generality, that the first two rows s_1 and s_2 have Hamming distance $\Delta(s_1, s_2) = 4$ (Figure 4.3 depicts an example of such a matrix). In the following, we will construct an equivalent Binary (α, β)-Distinct Vectors instance (S', k') in polynomial time.

Let $D_j \subseteq [n]$ denote the set of row indices where column vector s_{*j} equals 1. Further, for $i \in \mathbb{N}$ and $I \subseteq [i]$, let $\mathbf{1}^i_I \in \{0,1\}^i$ denote the size-i column vector that has 1-entries for all indices in I and 0-entries elsewhere.

We prove the theorem in two steps: First, the case $\alpha = 1$, $\beta = 4 + b$ for all $b \geq 0$, and second, the case $\alpha = 2 + a$, $\beta = 4 + 2\lceil a/2 \rceil + b$ for all $a, b \geq 0$. Note that these two cases together yield the statement of the theorem.

Case 1 ($\alpha = 1$, $\beta = 4 + b$, $b \geq 0$). We define the instance (S', k') as follows: The column vectors of the matrix $S' \in \{0,1\}^{(n+1)\times(d+b+1)}$ are

$$s'_{*j} := \begin{cases} \mathbf{1}^{n+1}_{D_j}, & j \in \{1, \ldots, d\}, \\ \mathbf{1}^{n+1}_{\{1\}}, & j \in \{d+1, \ldots, d+b\}, \\ \mathbf{1}^{n+1}_{\{n+1\}}, & j = d+b+1. \end{cases}$$

We set $k' := k + 1$. An example of the constructed instance is shown in Figure 4.4a. It is not hard to check that the rows of S' indeed fulfill the constraints on the Hamming distances:

$$h' := \min_{i \neq j \in [n+1]} \Delta(s_i, s_j) = \Delta(s'_n, s'_{n+1}) = 1 \text{ and}$$
$$H' := \max_{i \neq j \in [n+1]} \Delta(s_i, s_j) = \Delta(s'_1, s'_2) = \Delta(s_1, s_2) + b = 4 + b.$$

As regards correctness, observe first that every solution contains the column index $d + b + 1$ because the row vectors s'_n and s'_{n+1} only differ in this column. Since this column also distinguishes row s_{n+1} from all other rows and no other pair of rows in S', it follows that (S', k') is a yes-instance if and only if $(S'_{[[d+b]}, k)$ is a yes-instance. Due to Lemma 4.2, this is the case if and only if (S, k) is a yes-instance.

Case 2 ($\alpha = 2 + a$, $\beta = 4 + 2\lceil a/2 \rceil + b$, $a, b \geq 0$). We define the instance (S', k') as follows: Starting with $S' := S$, we add $\lceil a/2 \rceil$ copies of the column vector $\mathbf{1}^n_{\{i\}}$ for each $i \in [n-1]$ to S'. Moreover, we add $\lfloor a/2 \rfloor$ copies of the column vector $\mathbf{1}^n_{[n-1]}$ to S'. Finally, we add b copies of the column vector $\mathbf{1}^n_{\{1\}}$ to S' and set $k' = k$. Figure 4.4b shows an example of the construction. Indeed, we have the following Hamming distances:

$$\begin{aligned} \Delta(s'_n, s'_1) &= 2 + a + b, \\ \Delta(s'_n, s'_j) &= 2 + a, \\ \Delta(s'_1, s'_j) &= \Delta(s_1, s_j) + 2\lceil a/2 \rceil + b, \\ \Delta(s'_j, s'_{j'}) &= \Delta(s_j, s_{j'}) + 2\lceil a/2 \rceil, \end{aligned}$$

for all $j, j' \in \{2, \ldots, n-1\}$, $j \neq j'$. Thus, it holds $h' = \Delta(s'_n, s'_2) = 2 + a$ and $H' = \Delta(s'_1, s'_2) = 4 + 2\lceil a/2 \rceil + b$. Since all row vectors in S are distinct and since we only added columns which distinguish exactly one row from all others, the correctness follows due to Lemma 4.2. □

Theorem 4.3 implies that for a given instance with fixed minimum pairwise row Hamming distance α, it is possible to increase the maximum pairwise row Hamming distance β arbitrarily without changing the answer to the instance. On the contrary, however, it is not clear how to construct an equivalent instance where only the minimum pairwise row Hamming distance is increased. Indeed, in the following, we show polynomial-time solvability for the case $\alpha \geq 2\lfloor \beta/2 \rfloor - 1$.

4.2.2 Polynomial-Time Solvability for Homogeneous Data

The polynomial-time solvability for homogeneous data is based on the observation that, for small differences between maximum and minimum pairwise row Hamming distance, the input matrix is either highly structured or bounded in size by a constant depending only on the maximum pairwise row Hamming distance. This structure in turn guarantees that the instance is easily solvable. Before proving the theorem, we start with some helpful preliminary results.

First, we show that there is a linear-time preprocessing of a given input matrix such that the resulting matrix contains the null vector as a row and no two column vectors are identical.

Lemma 4.4. *For a given* Distinct Vectors *instance* $I = (S \in \{0,1\}^{n \times d}, k)$ *one can compute in* $O(nd)$ *time an equivalent* Distinct Vectors *instance* $I' := (S' \in \{0,1\}^{n \times d'}, k)$ *such that* S' *contains the null vector* $\mathbf{0} \in \{0\}^{d'}$ *as a row, the number* d' *of columns of* S' *is at most* d*, and no two column vectors of* S' *are identical (implying* $d' \leq 2^n$*).*

Proof. From an instance $I = (S, k)$, we compute S' as follows: First, in order to have the null vector $\mathbf{0}$ as a row, we consider an arbitrary row vector, say s_1, and iterate over all columns j. If $s_{1j} = 1$, then we exchange all 1's and 0's in column j. Then, we sort the columns of S lexicographically in $O(nd)$ time (using radix sort). We iterate over all columns again and check for any two successive column vectors whether they are identical and, if so, remove one of them. This ensures that all remaining column vectors are different, which implies that there are at most 2^n. Thus, in $O(nd)$ time, we end up with a matrix S' containing at most 2^n columns, where $s'_1 = \mathbf{0}$. Clearly, reordering columns, removing identical columns, as well as exchanging 1's and 0's in a column does not change the answer to the original instance. □

We henceforth assume all input instances to be already preprocessed according to Lemma 4.4. In fact, we can extend Lemma 4.4 by removing also inessential

columns (recall Definition 4.3), that is, we can use the following data reduction rule.

Reduction Rule 4.1. *Let (S, k) be a* Distinct Vectors *instance. If S contains an inessential column, then delete this column from S.*

Lemma 4.5. *Reduction Rule 4.1 is correct and can exhaustively be applied in $O(\min\{n, d\} \cdot nd)$ time.*

Proof. Correctness follows from Lemma 4.2. Exhaustive application of Reduction Rule 4.1 can be done as follows: First, we determine in $O(nd)$ time which columns fulfill Condition (1) of Definition 4.3. Recall that these are exactly the weight-1 and weight-$(n-1)$ columns of which there can be at most $\min\{2n, d\}$ after the preprocessing according to Lemma 4.4. For each of these candidate columns j, we check in $O(nd)$ time whether Condition (2) also holds, that is, whether all row vectors are still distinct without column j, by lexicographically sorting the rows of the matrix without column j. The overall running time is thus in $O(\min\{n, d\} \cdot nd)$. □

We now turn towards proving polynomial-time solvability of Binary (α, β)-Distinct Vectors for $\alpha \geq 2\lfloor \beta/2 \rfloor - 1$. The proof uses some results from extremal combinatorics concerning certain set systems. To start with, we introduce the necessary concepts and notation. Recall Definition 4.1, where we defined the set W_{s_i} of column indices where row i equals 1. In the following, for a given input matrix S and a given set of row indices I, we will consider the *column system* of I, that is, the system containing the sets W_{s_i} of column indices of all row vectors with indices in I.

Definition 4.4. For a matrix $S \in \{0,1\}^{n \times d}$ and a subset $I \subseteq [n]$ of row indices, let $\mathcal{W}(I) := \{W_{s_i} \mid i \in I\}$ denote the *column system* of I containing the sets W_{s_i} of column indices for all rows in I. For $\omega \in [d]$, let $I_\omega := \{i \in [n] \mid w(s_i) = \omega\}$ be the set of indices of the weight-ω rows and let $\mathcal{W}_\omega := \mathcal{W}(I_\omega)$ be the column system of the weight-ω rows.

Figure 4.5 illustrates Definition 4.4. Note that in order to distinguish all rows of weight ω from each other, we only have to consider those columns which appear in some of the sets contained in the column system $\mathcal{W}_\omega$ since the weight-ω rows only differ in these columns. Thus, in order to find subsolutions for the weight-ω rows, the structure of $\mathcal{W}_\omega$, especially the pairwise intersections of the contained sets, will be very important for us. Therefore, we make use of

1	2	3	4	5	6	7
1				1		
1			1			1
		1				
	1		1	1		
					1	1

$\mathcal{W}_1 = \{\{3\}\}$

$\mathcal{W}_2 = \{\{1,5\},\{6,7\}\}$

$\mathcal{W}_3 = \{\{1,4,7\},\{2,4,5\}\}$

Figure 4.5: An example of a binary matrix (left) containing rows of weight one, two, and three. The corresponding column systems are written on the right.

two general combinatorial concepts of set systems (see [Juk11, Chapter 6] for example), the first of which defines a system of sets that pairwise intersect in the same number of elements, whereas the second concept describes the even stronger condition that all pairwise intersections contain the same elements.

Definition 4.5 (Weak Δ-system)**.** A family $\mathcal{F} = \{S_1, \ldots, S_m\}$ of m different sets is called a *weak* Δ-system if there is some $\lambda \in \mathbb{N}$ such that $|S_i \cap S_j| = \lambda$ for all $i \neq j \in [m]$.

Definition 4.6 (Strong Δ-system)**.** A *strong* Δ-system (or *sunflower*) is a weak Δ-system $\{S_1, \ldots, S_m\}$ such that $S_i \cap S_j = C$ for all $i \neq j \in [m]$ and some set C called the *core*. The sets $\widetilde{S}_i := S_i \setminus C$ are called *petals*.

The following lemma illustrates the merit of the above definitions showing that any Distinct Vectors instance can easily be solved if the underlying column system of all non-zero-weight rows forms a sunflower.

Lemma 4.6. *Let $I := (S \in \{0,1\}^{n \times d}, k)$ be a* Distinct Vectors *instance such that $\mathcal{W} := \bigcup_{\omega \geq 1} \mathcal{W}_\omega$ forms a sunflower (note that $\mathcal{W}_0 = \emptyset \notin \mathcal{W}$). Then, I is a yes-instance if and only if $k \geq |\mathcal{W}|$. Moreover, any solution intersects at least $|\mathcal{W}| - 1$ of the petals of $\mathcal{W}$.*

Proof. Recall that we assume the instance I to be already preprocessed according to Lemma 4.4. Hence, we can assume without loss of generality that $s_n = \mathbf{0}$ and that no two column vectors of S are equal. Assume further that $\mathcal{W} = \{W_{s_1}, \ldots, W_{s_{n-1}}\}$ is a sunflower with core C. An example is depicted in Figure 4.6.

Recall that any solution K fulfills $K \cap D_{ij} \neq \emptyset$ for all $i \neq j \in [n]$, where D_{ij} is the set of column indices in which the row vectors s_i and s_j differ. Assume

1	2	3	4	5	6	7	8	9	10
1	1		1					1	
1	1					1			
1	1	1							1
1	1				1		1		
1	1			1					
1	1								

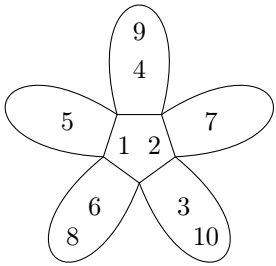


Figure 4.6: Example of a matrix where the set system $\mathcal{W}$ forms a sunflower with core $C = \{1, 2\}$ consisting of the first two columns. The six petals from top to bottom are $\{4, 9\}$, $\{7\}$, $\{3, 10\}$, $\{6, 8\}$, $\{5\}$ and $\emptyset$ (a schematic drawing of $\mathcal{W}$ is on the right). Framed by thick lines is a set of columns that distinguish all rows.

towards a contradiction that $K \subseteq [d]$ with $|K| < n-1$ is a solution. If $K \cap C = \emptyset$, then K only intersects the petals. Since the petals are pairwise disjoint, it follows that there exists an $i \in [n-1]$ such that $K \cap W_i = K \cap D_{in} = \emptyset$, which shows that K cannot be solution. If $K \cap C \neq \emptyset$, then K intersects at most $n-3$ of the $n-1$ petals in $\mathcal{W}$. Hence, there exist $i, j \in [n-1]$ with $i \neq j$ such that $K \cap (\widetilde{W}_i \cup \widetilde{W}_j) = K \cap D_{ij} = \emptyset$. Hence, K cannot be a solution. It remains to show that there is always a solution of size $|\mathcal{W}| = n-1$. To this end, let K contain an arbitrary element from each non-empty petal and, if there is an empty petal, also an arbitrary element from the core C. Clearly, K is a solution of size $n-1$. □

According to Lemma 4.6, identifying sunflower structures in a given input instance significantly simplifies our problem since they have easy solutions. To this end, the following result by Deza [Dez74] will serve as an important tool since it describes conditions under which a weak Δ-system actually becomes a strong one, that is, a sunflower (see also Jukna [Juk11, Chapter 6, Theorem 6.2]).

Lemma 4.7 (Deza [Dez74, Theorem 2])**.** *Let $\mathcal{F}$ be an s-uniform weak Δ-system, that is, each set contains s elements. If $|\mathcal{F}| \geq s^2 - s + 2$, then $\mathcal{F}$ is a sunflower.*

The basic scheme for proving polynomial-time solvability of Binary (α, β)-Distinct Vectors for $\alpha \geq 2\lfloor \beta/2 \rfloor - 1$ is the following: The bounds on the

minimum and maximum pairwise row Hamming distances imply that the column systems $\mathcal{W}_x$ for $x = \alpha, \ldots, \beta$ form x-uniform weak Δ-systems. Using Lemma 4.7, we then conclude that either the size of the instance is upper-bounded by a constant depending on β only, or that the $\mathcal{W}_x$ form sunflowers, which we can handle according to Lemma 4.6.

As a final prerequisite, we prove the following useful lemma concerning the intersections of a set with sets in a sunflower.

Lemma 4.8. *Let $\lambda \in \mathbb{N}$, let $\mathcal{F}$ be a sunflower with core C and let X be a set such that $|X \cap S| \geq \lambda$ for all $S \in \mathcal{F}$. If $|\mathcal{F}| > |X|$, then $|X \cap C| \geq \lambda$ (implying $\lambda \leq |C|$).*

Proof. Assume towards a contradiction that $|X \cap C| < \lambda$. Then X would intersect each of the $|\mathcal{F}| > |X|$ pairwise disjoint petals of $\mathcal{F}$, which is not possible. □

We are now ready to prove the following theorem.

Theorem 4.9. Binary (α, β)-Distinct Vectors *is solvable*

1.) in $O(\min\{n, d\} \cdot nd)$ time if $\beta \leq \alpha + 1$, and

2.) in $O(n^3 d)$ time if α is odd and $\beta = \alpha + 2$.

We prove both statements of Theorem 4.9 separately. As mentioned, the basic structures of both proofs are similar: We first partition the column system into uniform weak Δ-systems. Then, we consider for each system the cases of being a sunflower or being of bounded size (due to Lemma 4.7). Then, we make use of the preprocessing (Lemma 4.4 and Reduction Rule 4.1) and our knowledge about solutions for sunflowers (Lemma 4.6) to show that only a small number of candidate solutions are to be considered. That is, the instances are essentially solved by the preprocessing routines and we can try out all remaining candidates to find a solution in polynomial time. Showing that there are only few remaining candidate solutions seems to be more difficult for Statement (2.); thus, the proof of Statement (1.) can be seen as a "warm-up".

Proof (Theorem 4.9, Statement (1.)) In the following, let $I = (S \in \{0,1\}^{n \times d}, k)$ be an instance of Binary (α, β)-Distinct Vectors for $\alpha \leq \beta \leq \alpha + 1$. Recall that we assume I to be already preprocessed according to Lemma 4.4 and Reduction Rule 4.1 in $O(\min\{n, d\} \cdot nd)$ time, that is, S contains the null row vector, say $s_n = \mathbf{0}$, no two column vectors are equal, which implies $d \leq 2^n$, and there

are no inessential columns. We write $W_i := W_{s_i} = \{j \in [d] \mid s_{ij} = 1\}$ for the set of column indices j where row vector s_i equals 1, and we define $W_{ij} := W_i \cap W_j$. For $\omega \in [d]$, let $I_\omega := \{i \in [n] \mid w(s_i) = \omega\}$ denote the set of indices of the weight-ω rows and let $n_\omega := |I_\omega|$. For ease of presentation, we sometimes identify columns or rows and their corresponding indices.

For all $i \in [n-1]$, we have

$$\Delta(s_i, s_n) = \Delta(s_i, \mathbf{0}) = w(s_i) \in \{\alpha, \alpha+1\}.$$

Since also

$$\Delta(s_i, s_j) = w(s_i) + w(s_j) - 2|W_{ij}| \in \{\alpha, \alpha+1\}$$

holds for all $i \neq j \in [n-1]$, the following properties can be derived:

$$\forall i, j \in I_\alpha,\ i \neq j : |W_{ij}| = \lfloor \alpha/2 \rfloor, \tag{4.1}$$

$$\forall i, j \in I_{\alpha+1},\ i \neq j : |W_{ij}| = \lceil (\alpha+1)/2 \rceil, \text{ and} \tag{4.2}$$

$$\forall i \in I_\alpha,\ j \in I_{\alpha+1} : |W_{ij}| = \lfloor (\alpha+1)/2 \rfloor. \tag{4.3}$$

For example, let us prove Property (4.1). For $i, j \in I_\alpha$, we have $\Delta(s_i, s_j) = 2\alpha - 2|W_{ij}| \in \{\alpha, \alpha+1\}$. Since $|W_{ij}|$ is an integer, $2\alpha - 2|W_{ij}|$ is even. Therefore, if α is even, then $2\alpha - 2|W_{ij}| = \alpha$ implying $|W_{ij}| = \alpha/2 = \lfloor \alpha/2 \rfloor$. If α is odd, then $2\alpha - 2|W_{ij}| = \alpha + 1$ and, hence, $|W_{ij}| = (\alpha-1)/2 = \lfloor \alpha/2 \rfloor$. This proves Property (4.1). The proofs for the remaining properties are analogous.

Now, Property (4.1) implies that $\mathcal{W}_\alpha := \{W_i \mid i \in I_\alpha\}$ is an α-uniform weak Δ-system (every pair of sets intersects in $\lfloor \alpha/2 \rfloor$ elements) and Property (4.2) implies that $\mathcal{W}_{\alpha+1} := \{W_i \mid i \in I_{\alpha+1}\}$ is an $(\alpha+1)$-uniform weak Δ-system (every pair of sets intersects in $\lceil (\alpha+1)/2 \rceil$ elements). We define the constant

$$c := (\alpha+1)^2 - (\alpha+1) + 2$$

according to the bound in Lemma 4.7. We can assume that $\max\{n_\alpha, n_{\alpha+1}\} \geq c$ because otherwise $n = n_\alpha + n_{\alpha+1} + 1 \leq 2c$ is upper-bounded by a constant, and thus also $d \leq 2^n$ is upper-bounded by a constant (recall that we assume S to be preprocessed according to Lemma 4.4), which implies that the instance I is constant-time solvable.

If $n_{\alpha+1} \geq c$, then, by Lemma 4.7, it follows that $\mathcal{W}_{\alpha+1}$ is a sunflower with a core C of size $\lceil (\alpha+1)/2 \rceil$ and petals $\widetilde{W}_i$, $i \in I_{\alpha+1}$, of size $\alpha + 1 - |C| \geq 1$. For each $i \in I_{\alpha+1}$ and each $j \in I_\alpha$, Property (4.3) together with $n_{\alpha+1} \geq c > \alpha$ imply (due to Lemma 4.8) that $|W_j \cap C| = |W_{ij}|$. Thus, we have $W_{ij} \subseteq C$

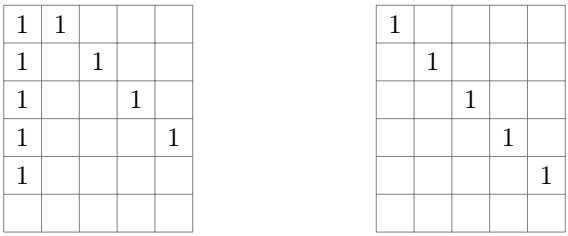


Figure 4.7: Examples of two possible instances for the case $\alpha = 1$, $\beta = 2$. In both cases the column system formed by all non-zero row vectors is a sunflower. A solution can thus be easily computed according to Lemma 4.6.

and $\widetilde{W}_i \cap W_j = \emptyset$. Hence, for $x \in \widetilde{W}_i$, the column vector s_{*x} contains exactly one 1 (namely in the i-th row), that is, $s_{*x} = \mathbf{1}^n_{\{i\}}$. Thus, column x exactly distinguishes row i from all other rows. For $\alpha \geq 2$, each pair of rows differs in at least two columns. Thus, all rows in $S_{|[d]\setminus\{x\}}$ are still distinct and column x is in fact inessential, which yields a contradiction. Hence, we can assume that $\alpha = 1$. Since for all $i \in I_1$ and $j \in I_2$, we then have $W_{ij} = W_i = C$, it follows that $n_1 = 1$. See Figure 4.7a for an illustrating example. By Lemma 4.6, I is a yes-instance if and only if $k \geq n_2 + n_1 = n - 1$ (which can be decided in $O(nd)$ time).

If $n_{\alpha+1} < c$, then $n_\alpha \geq c$ holds and Lemma 4.7 implies that $\mathcal{W}_\alpha$ is a sunflower with a core C of size $|C| = \lfloor \alpha/2 \rfloor$. If α is odd, then we have $\lfloor (\alpha+1)/2 \rfloor > |C|$, and thus, due to Property (4.3) and $n_\alpha \geq c > \alpha + 1$, Lemma 4.8 implies that there cannot exist a row vector of weight $\alpha + 1$, that is $n_{\alpha+1} = 0$ (see Figure 4.7b). Now, by Lemma 4.6, I is a yes-instance if and only if $k \geq n_\alpha = n-1$ (decidable in $O(nd)$ time). If α is even, then $|C| = \lfloor (\alpha+1)/2 \rfloor \geq 1$. Thus, Property (4.3) and Lemma 4.8 imply that $W_{ij} = C$ for all $i \in I_\alpha$, $j \in I_{\alpha+1}$. Note that $s_{*x} = \mathbf{1}^n_{[n-1]}$ for all $x \in C$, that is, column x exactly distinguishes row n from all others. Since α is even, hence $\alpha \geq 2$, it follows that column x is inessential, which again yields a contradiction. □

To sum up, the proof above shows that for $\beta \leq \alpha + 1$ the preprocessing (Lemma 4.4 and Reduction Rule 4.1) either yields an instance of constant size $O(\beta^2)$ or in fact yields an instance with $\alpha = 1$ and $\beta \leq 2$ which is structured

in such a way that it is easily solvable. Hence, the overall running time for this case is determined by the polynomial running time of the preprocessing.

Next, we show that Binary $(\alpha, \alpha+2)$-Distinct Vectors is solvable in $O(n^3 d)$ time if α is odd. We use the same notation as in the proof of Statement (1.) (also many arguments are completely analogous). Again, we assume the input instance $I = (S, k)$ to be already preprocessed according to Lemma 4.4 and Reduction Rule 4.1.

Proof (Theorem 4.9, Statement (2.)) Since

$$\Delta(s_i, s_j) = w(s_i) + w(s_j) - 2|W_{ij}| \in \{\alpha, \alpha+1, \alpha+2\}$$

holds for all $i \neq j \in [n]$, it follows that

$$\Delta(s_i, s_n) = w(s_i) \in \{\alpha, \alpha+1, \alpha+2\}$$

holds for all $i \in [n-1]$. By plugging in the respective values for $w(s_i)$ and $w(s_j)$ in the above formula for $\Delta(s_i, s_j)$, the following properties can be derived (in an analogous way as for Properties (4.1) to (4.3) in the proof of Statement (1)):

$$\forall i, j \in I_\alpha,\ i \neq j : |W_{ij}| = \lfloor \alpha/2 \rfloor, \tag{4.4}$$
$$\forall i \in I_\alpha,\ j \in I_{\alpha+1} : |W_{ij}| \in \{\lfloor \alpha/2 \rfloor, \lceil \alpha/2 \rceil\}, \tag{4.5}$$
$$\forall i \in I_\alpha,\ j \in I_{\alpha+2} : |W_{ij}| = \lceil \alpha/2 \rceil, \tag{4.6}$$
$$\forall i, j \in I_{\alpha+1},\ i \neq j : |W_{ij}| = \lceil \alpha/2 \rceil, \tag{4.7}$$
$$\forall i \in I_{\alpha+1},\ j \in I_{\alpha+2} : |W_{ij}| \in \{\lceil \alpha/2 \rceil, \lceil \alpha/2 \rceil + 1\}, \tag{4.8}$$
$$\forall i, j \in I_{\alpha+2},\ i \neq j : |W_{ij}| = \lceil \alpha/2 \rceil + 1. \tag{4.9}$$

Properties (4.4), (4.7), and (4.9) imply that $\mathcal{W}_\alpha$, $\mathcal{W}_{\alpha+1}$, and $\mathcal{W}_{\alpha+2}$ are α-, $(\alpha+1)$-, and $(\alpha+2)$-uniform weak Δ-systems, respectively.

In the following, we denote by $U_\omega := \bigcup_{i \in I_\omega} W_i$ the index set of the columns where at least one weight-ω row vector equals 1. Let $c := (\alpha+2)^2 - (\alpha+2) + 2$. For each $x \in \{\alpha, \alpha+1, \alpha+2\}$, we either have $n_x < c$ or $n_x \geq c$. Overall, this gives eight possible cases, each of which we now show how to solve:

Case 1 ($n_\alpha < c$, $n_{\alpha+1} < c$, $n_{\alpha+2} < c$). In this case, the number of rows in S is upper-bounded by a constant depending on α, and thus, I is of overall constant size.

Case 2 ($n_\alpha \geq c$, $n_{\alpha+1} < c$, $n_{\alpha+2} < c$). Due to Lemma 4.7, the system $\mathcal{W}_\alpha$ forms a sunflower. Let C with $|C| = \lfloor \alpha/2 \rfloor$ be the core of $\mathcal{W}_\alpha$. For $\alpha =$

1, clearly, any solution K contains all column indices from U_1 in order to distinguish the weight-1 rows from the null vector. Since $|U_2 \cup U_3| \leq 2n_2 + 3n_3$ is upper-bounded by a constant, the number of possible subsets $K' \subseteq U_2 \cup U_3$ is also upper-bounded by a constant. Thus, we only have to check a constant number of sets $K = U_1 \cup K'$. For $\alpha \geq 3$, the size of a petal $\widetilde{W}_i$, $i \in I_\alpha$, is $|\widetilde{W}_i| = |W_i| - |C| = \alpha - \lfloor \alpha/2 \rfloor = \lceil \alpha/2 \rceil \geq 2$. Since the petals are pairwise disjoint, it follows that, for each petal $\widetilde{W}_i$, there exists a $j \in I_{\alpha+1} \cup I_{\alpha+2}$ such that $\widetilde{W}_i \cap W_j \neq \emptyset$. Otherwise, the column vectors corresponding to the indices in petal $\widetilde{W}_i$ are all equal to $\mathbf{1}^n_{\{i\}}$, that is, at least one of them is inessential, which is a contradiction. Since $|U_{\alpha+1} \cup U_{\alpha+2}|$ is upper-bounded by a constant (depending on α), also the number n_α of petals in $\mathcal{W}_\alpha$ is upper-bounded by a constant, which yields an overall constant size of I.

Case 3 ($n_\alpha < c$, $n_{\alpha+1} \geq c$, $n_{\alpha+2} < c$). By Lemma 4.7, $\mathcal{W}_{\alpha+1}$ forms a sunflower with a core C of size $|C| = \lceil \alpha/2 \rceil$. The size of each petal $\widetilde{W}_i$, $i \in I_{\alpha+1}$, is thus $|\widetilde{W}_i| = \lceil \alpha/2 \rceil$. Hence, for $\alpha \geq 3$, analogous arguments as in Case 2 hold. For $\alpha = 1$, any solution K can be written as $K = C' \cup U_1 \cup K_2 \cup K_3$, where $C' \subseteq C$, $K_2 \subseteq U_2 \setminus C$ and $K_3 \subseteq U_3$. Note that $|C|$ and $|U_3|$ are upper-bounded by a constant. Hence, the number of different subsets C' and K_3 is also a constant. Since $|\widetilde{W}_i| = 1$ holds for all $i \in I_2$, we have $|U_2 \setminus C| = n_2$. From Lemma 4.6, it follows that $|K_2| \geq n_2 - 1$ holds in order to distinguish all weight-2 row vectors and the null vector from each other. Therefore, the overall number of possible choices for K_2, and thus for K, is in $O(n)$.

Case 4 ($n_\alpha < c$, $n_{\alpha+1} < c$, $n_{\alpha+2} \geq c$). By Lemma 4.7, $\mathcal{W}_{\alpha+2}$ forms a sunflower with a core C of size $|C| = \lceil \alpha/2 \rceil + 1$. The size of each petal $\widetilde{W}_i$, $i \in I_{\alpha+2}$, is thus $|\widetilde{W}_i| = \lceil \alpha/2 \rceil$. Hence, for $\alpha \geq 3$, analogous arguments as in Case 2 hold. For $\alpha = 1$, any solution K can be written as $K = C' \cup U_1 \cup K_2 \cup K_3$, where $C' \subseteq C$, $K_2 \subseteq U_2$ and $K_3 \subseteq U_3 \setminus C$. Note that $|C|$ and $|U_2|$ are upper-bounded by a constant. Hence, the number of different subsets C' and K_2 is also a constant. Lemma 4.6 implies that $|K_3| \geq n_3 - 1$. Since $|U_3 \setminus C| = n_3$, this yields an overall number of $O(n)$ possible choices for K.

Case 5 ($n_\alpha \geq c$, $n_{\alpha+1} \geq c$, $n_{\alpha+2} < c$). Due to Lemma 4.7, $\mathcal{W}_\alpha$ forms a sunflower with a core C of size $|C| = \lfloor \alpha/2 \rfloor$ and $\mathcal{W}_{\alpha+1}$ forms a sunflower with core C' of size $|C'| = \lceil \alpha/2 \rceil$. First, note that Property (4.6) implies $|W_{ij}| = \lceil \alpha/2 \rceil > |C|$ for all $i \in I_\alpha$, $j \in I_{\alpha+2}$, which is not possible due to Lemma 4.8. Thus, it follows that $n_{\alpha+2} = 0$. Moreover, since Property (4.5) implies $|W_{ij}| \geq |C|$ for all $i \in I_\alpha$ and $j \in I_{\alpha+1}$, Lemma 4.8 yields $C \subseteq W_{ij}$, and thus $C \subset C'$.

Hence, all column vectors in C equal $\mathbf{1}^n_{[n-1]}$, which yields a contradiction for $\alpha \geq 3$ because C then contains an inessential column. For $\alpha = 1$, any solution K can be written as $K = C'' \cup U_1 \cup K_2$, where $C'' \subseteq C'$ and $K_2 \subseteq U_2$. By Lemma 4.6, we know that $|K_2| \geq n_2 - 1$. Since $|C'| = 1$ and $|U_2 \setminus C'| = n_2$, there are $O(n)$ possible choices for K.

Case 6 ($n_\alpha \geq c$, $n_{\alpha+1} < c$, $n_{\alpha+2} \geq c$). This case is not possible since we showed in Case 5 that $n_\alpha \geq c$ implies $n_{\alpha+2} = 0$.

Case 7 ($n_\alpha \geq c$, $n_{\alpha+1} \geq c$, $n_{\alpha+2} \geq c$). This case is again not possible.

Case 8 ($n_\alpha < c$, $n_{\alpha+1} \geq c$, $n_{\alpha+2} \geq c$). From Lemma 4.7 it follows that $\mathcal{W}_{\alpha+1}$ forms a sunflower with a core C of size $|C| = \lceil \alpha/2 \rceil$ and $\mathcal{W}_{\alpha+2}$ forms a sunflower with core C' of size $|C'| = \lceil \alpha/2 \rceil + 1$. Moreover, analogously to Case 5, Property (4.8) and Lemma 4.8 imply $C \subset C'$.

If $\alpha = 1$, then any solution can be written as $K = U_1 \cup C'' \cup K_2 \cup K_3$, where $C'' \subseteq C'$, $K_2 \subseteq U_2 \setminus C$ and $K_3 \subseteq U_3 \setminus C'$. Since $|C'| = \lceil \alpha/2 \rceil + 1$, $|U_2 \setminus C| = n_2$, $|U_3 \setminus C'| = n_3$, and, by Lemma 4.6, $|K_2| \geq n_2 - 1$ and $|K_3| \geq n_3 - 1$, it follows that there are $O(n^2)$ possible choices for K.

For $\alpha \geq 3$, we show that the matrix S—recall that it is reduced with respect to Reduction Rule 4.1—has a specific structure, depicted in Figure 4.8. Namely, we claim that

(a) if $W_{ij} \setminus C' \neq \emptyset$ for $i \neq j \in [n-1]$, then $i \in I_{\alpha+1}$ and $j \in I_\alpha \cup I_{\alpha+2}$, and

(b) the one column vector s_{*z} with $z \in C \setminus C'$ equals $\mathbf{1}^n_{I_\alpha \cup I_{\alpha+2}}$.

Claim (a) implies that each column in $[d] \setminus C'$ contains at most two 1's (naturally, any column contains at least one 1). We will see that in fact all columns in $[d] \setminus C'$ contain exactly two 1's and therefore define the edges of a bipartite graph with vertex sets $I_{\alpha+1}$ and $I_\alpha \cup I_{\alpha+2}$. We find a matching that saturates $I_{\alpha+1}$ in this bipartite graph and show that the columns corresponding to the matching edges along with column z are an optimal solution.

To show Claim (a), observe that, if $i \neq j \in I_{\alpha+2}$, then $W_{ij} \setminus C' = \emptyset$ as $\mathcal{W}_{\alpha+2}$ is a sunflower with core C'. Likewise, if $i \neq j \in I_{\alpha+1}$, then $W_{ij} \setminus C' = \emptyset$ because $\mathcal{W}_{\alpha+1}$ is a sunflower with core C and $C \subset C'$. It hence suffices to show that $W_{ij} \setminus C' = \emptyset$ in the case that either both $i, j \in I_\alpha$ or $i \in I_\alpha$ and $j \in I_{\alpha+2}$. To see the latter, note that Property (4.6) and Lemma 4.8 imply $|W_i \cap C'| = \lceil \alpha/2 \rceil = |C'| - 1$, for all $i \in I_\alpha$, that is, we have $W_{ij} \subset C'$ for all $i \in I_\alpha$, $j \in I_{\alpha+2}$. Now, it only remains to show $W_{ij} \subseteq C'$ for $i, j \in I_\alpha$, $i \neq j$. We derived above that

$$|W_i \cap C'| = |W_j \cap C'| = \lceil \alpha/2 \rceil = |C'| - 1.$$

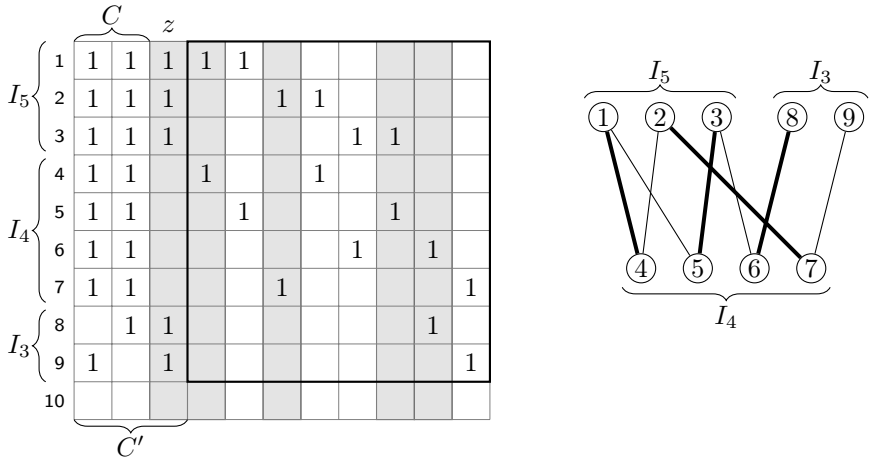


Figure 4.8: An instance for the case $\alpha = 3$, $\beta = 5$ (left). The submatrix framed by the thick rectangle is the incidence matrix of a bipartite graph (right). An optimal solution is highlighted in gray. Note that the columns in the solution (excluding column z) correspond to a maximum matching in the bipartite graph (thick edges).

Thus, $|(W_i \cap C') \cap (W_j \cap C')| = |W_{ij} \cap C'| \geq |C'| - 2 = \lfloor \alpha/2 \rfloor$. By Property (4.4), it holds $|W_{ij}| \leq \lfloor \alpha/2 \rfloor$, which implies $W_{ij} \subseteq C'$. Hence, $W_{ij} \setminus C' = \emptyset$, completing the proof of Claim (a).

Let us next prove Claim (b), that is, $s_{*z} = \mathbf{1}^n_{I_\alpha \cup I_{\alpha+2}}$, where z is the one column index in $C \setminus C'$. Assume the contrary, that is, either a row in I_α has a 0 at entry z or a row in $I_{\alpha+1}$ has a 1 at entry z. Let us first show that $s_{iz} = 0$, that is, $z \notin W_i$ is impossible for a row $i \in I_\alpha$. Using $|W_i \cap C'| = |C'| - 1$, it follows that $C \subset W_i$. Then, for all $j \in I_{\alpha+1} \cup I_{\alpha+2}$, it holds $W_{ij} \setminus C = \emptyset$ since otherwise either Property (4.5) or Property (4.6) is violated. Let us show that $W_{ij} \setminus C = \emptyset$ also holds for all $j \neq i \in I_\alpha$. Recall that $W_{ij} \setminus C' = \emptyset$, as shown in the proof of Claim (a) above. By assumption $z \notin W_{ij}$, yielding $W_{ij} \setminus C = W_{ij} \setminus C' = \emptyset$. But then, the columns in $W_i \setminus C$ equal $\mathbf{1}^n_{\{i\}}$. Note that $|W_i \setminus C| \geq \alpha - \lfloor \alpha/2 \rfloor \geq 2$ (recall that $\alpha \geq 3$). Hence, there is at least one inessential column, a contradiction.

Let us now show that $s_{iz} = 1$, that is $z \in W_i$ is not possible for a row $i \in I_{\alpha+1}$. If $z \in W_i$, then $C' \subseteq W_i$ and thus $|W_{ij}| \geq |C'| = \lceil \alpha/2 \rceil + 1$ for every $j \in I_{\alpha+2}$. Property (4.8) then implies $W_{ij} = C'$, that is $W_{ij} \setminus C' = \emptyset$

for every $j \in I_{\alpha+2}$. Moreover, consider a row $j \in I_\alpha$ and note that $|W_{ij} \cap C'| = 1 + |W_j \cap C|$ since $z \in W_j$ as shown before. By Property (4.5) and Lemma 4.8, we have $|W_j \cap C| \geq \lfloor \alpha/2 \rfloor$, which yields $|W_{ij} \cap C'| \geq \lceil \alpha/2 \rceil$. Now, Property (4.5) implies $|W_{ij} \cap C'| = \lceil \alpha/2 \rceil = |W_{ij}|$, which implies $W_{ij} \subseteq C'$, that is $W_{ij} \cap C' = \emptyset$. But then, the columns in $W_i \setminus C'$ are equal to $\mathbf{1}^n_{\{i\}}$ and there are at least $|W_i \setminus C'| = \alpha + 1 - \lceil \alpha/2 \rceil - 1 = \lfloor \alpha/2 \rfloor \geq 1$ such inessential columns, which yields a contradiction. Hence, for $z \in C' \setminus C$, it holds $s_{*z} = \mathbf{1}^n_{I_\alpha \cup I_{\alpha+2}}$, proving Claim (b). Note that column z distinguishes all rows in $I_{\alpha+1}$ from all rows in $I_\alpha \cup I_{\alpha+2}$.

To finish the proof of Case 8, we need one more observation about the number of weight-α rows, namely that $n_\alpha = |C| = \lceil \alpha/2 \rceil$. Since $|W_i \cap C'| = |C'| - 1 = |C|$ and $z \in W_i$ holds for all $i \in I_\alpha$, and since also $\{z\} = C' \setminus C$, we have $|W_i \cap C| = |C| - 1$. If $n_\alpha < |C|$, then there exists a column $x \in C$ such that $x \in W_i$ for every $i \in I_\alpha$. But then, $s_{*x} = \mathbf{1}^n_{[n-1]}$ is inessential, which is not possible. If $n_\alpha > |C|$, then there exists a pair $i \neq j \in I_\alpha$ such that $W_i \cap C = W_j \cap C$. This implies $|W_{ij}| \geq |C| - 1 + 1 = \lceil \alpha/2 \rceil$, which contradicts Property (4.4).

We now derive a solution from the abovementioned bipartite graph. Consider the columns in $[d] \setminus C'$. Clearly, if one of these columns contains only one 1, then this column is inessential, which yields a contradiction. Thus, each column contains at least two 1's. Using Claim (a), each of the columns also has at most two 1's. Also, after preprocessing, no two columns are equal. Thus, the submatrix $S[[n-1], [d] \setminus C']$ (framed by thick lines in Figure 4.8) is the incidence matrix of a bipartite graph G, where the rows correspond to the vertices (partitioned into $I_{\alpha+1}$ and $I_\alpha \cup I_{\alpha+2}$) and the columns define the edges. Moreover, each vertex $i \in I_{\alpha+2}$ has degree $|W_i \setminus C'| = \lceil \alpha/2 \rceil$. Also each vertex $i \in I_{\alpha+1}$ has degree $|W_i \setminus C'| = |W_i \setminus C| = \lceil \alpha/2 \rceil$ by Claim (b). Each row $i \in I_\alpha$ has $|W_i \cap C'| = \lceil \alpha/2 \rceil$ (as shown in the proof of Claim (a)), and, therefore, vertex i has degree $|W_i \setminus C'| = \lfloor \alpha/2 \rfloor$ in G. We can now use Hall's theorem [BG09][2], to show that there exists a matching in G that saturates $I_{\alpha+1}$, that is, a subset $M \subseteq [d] \setminus C'$ of $n_{\alpha+1}$ columns such that $|W_i \cap M| = 1$ for all $i \in I_{\alpha+1}$ and $|W_i \cap M| \leq 1$ for all $i \in I_\alpha \cup I_{\alpha+2}$. Indeed, taking any subset $T \subseteq I_{\alpha+1}$ of vertices, consider the set $N_G(T) \subseteq I_\alpha \cup I_{\alpha+2}$ of neighbors of T. Since the vertices in $N_G(T)$ have at most the degree of any vertex in T, we have $|N_G(T)| \geq |T|$. Hence, the condition of Hall's theorem is satisfied. Thus, matching M exists as claimed.

[2]Hall's theorem states that, for a bipartite graph $G = (X \dot\cup Y, E)$, there exists an X-saturating matching if and only if $|T| \leq |N_G(T)|$ holds for each subset $T \subseteq X$.

We now claim that $K := M \cup \{z\}$ with $|K| = n_{\alpha+1} + 1$ is an optimal solution (highlighted in gray in Figure 4.8). First, we show that all rows in $S_{|K}$ are pairwise distinct. Column z distinguishes each row in $I_\alpha \cup I_{\alpha+2}$ from each row in $I_{\alpha+1} \cup \{n\}$. Since M saturates $I_{\alpha+1}$, all rows in $I_{\alpha+1}$ are distinct from row n. It remains to show that all pairs of rows in $I_\alpha \cup I_{\alpha+2}$ are distinct. Recall that $W_{ij} \setminus C' = \emptyset$ for all $i \neq j \in I_\alpha \cup I_{\alpha+2}$ (by Claim (a)). Thus, every row $i \in I_\alpha \cup I_{\alpha+2}$ with $W_i \cap M \neq \emptyset$ is distinct from all other rows in $I_\alpha \cup I_{\alpha+2}$. Hence, since M is a matching, if $|M| \geq n_\alpha + n_{\alpha+2} - 1$, then all pairs of rows in $I_\alpha \cup I_{\alpha+2}$ are distinct. Since G is bipartite, we have

$$n_{\alpha+2}\lceil \alpha/2 \rceil + n_\alpha \lfloor \alpha/2 \rfloor = n_{\alpha+1}\lceil \alpha/2 \rceil.$$

Thus, using $n_\alpha = \lceil \alpha/2 \rceil$ (as observed above), we have

$$|M| = n_{\alpha+1} = n_{\alpha+2} + \lfloor \alpha/2 \rfloor = n_\alpha + n_{\alpha+2} - 1.$$

Consequently, all rows in $S_{|K}$ are distinct.

Regarding optimality, it remains to show that there is no solution of size $n_{\alpha+1}$. This can be seen as follows: Lemma 4.6 implies that any solution K intersects at least $n_{\alpha+1} - 1$ of the petals of $\mathcal{W}_{\alpha+1}$. If K intersects all $n_{\alpha+1}$ petals, then, as $n_{\alpha+1} = n_\alpha + n_{\alpha+2} - 1$, there exists a $j \in I_\alpha \cup I_{\alpha+2}$ such that $K \cap W_j = \emptyset$, which is not possible. Otherwise, if K intersects exactly $n_{\alpha+1} - 1$ of the petals, then there exists an $i \in I_{\alpha+1}$ and also $j \neq j' \in I_\alpha \cup I_{\alpha+2}$ such that $K \cap (W_i \setminus C') = \emptyset$, $K \cap (W_j \setminus C') = \emptyset$ and $K \cap (W_{j'} \setminus C') = \emptyset$. But it is not possible to pairwise distinguish all three rows i, j, and j' from each other with just one column. Hence, any optimal solution contains at least $n_{\alpha+1} + 1$ columns and, therefore, I is a yes-instance if and only if $k \geq n_{\alpha+1} + 1$. This finishes Case 8.

As regards the running time, observe that the maximum number of candidate solutions we have to test in any of the above cases is in $O(n^2)$. Checking whether a subset of column indices is a solution can be done in $O(nd)$ time via lexicographical sorting of the rows. This yields an overall running time in $O(n^3 d)$ which also subsumes the $O(\min\{n, d\} \cdot nd)$ time for the preprocessing. □

As a final remark, we mention that polynomial-time solvability actually holds for all instances that satisfy the conditions of Theorem 4.9 *after* the preprocessing (Lemma 4.4 and Reduction Rule 4.1). This yields an even stronger result since in practice also instances with a larger difference between maximum and minimum pairwise row Hamming distance could be efficiently solvable.

4.3 Conclusion

Based on a combined parameterization by minimum and maximum pairwise row Hamming distances, we proved a complexity dichotomy for Distinct Vectors on binary matrices. We observed that for small differences between the maximum and minimum pairwise row Hamming distance, the rows of the input matrix correspond to set systems having a certain "sunflower" structure. The polynomial-time solvable cases are obtained by a precise analysis of these underlying set systems using combinatorial results from extremal set theory and matching theory. It is a major open question whether the polynomial-time solvability can be generalized to matrices on non-binary alphabets. For practical purposes, it is an interesting question whether one can even obtain linear time for the tractable cases.

We conclude with two further challenges for future research. The concepts of weak and strong Δ-systems might prove useful to obtain further algorithmic results and also to gain insight into the underlying combinatorial structure of other problems. It is interesting to search for problems involving set systems where the structure allows to apply results from extremal set theory.

From a combinatorial point of view, the study of structural parameterizations for matrix problems in general seems to be a fertile but rather little researched area for parameterized complexity studies which should be extended in the future (another example of a parameterized complexity analysis for a vector problem deals with the explanation of integer vectors by few homogenous segments [Bre+15a]). This also includes the task of defining appropriate parameters in the first place. For example, instead of using the Hamming distance of rows one could also consider other distance measures (for instance, the weighted edit distance). Other possible parameter candidates could be the number of different rows (or columns), the frequency of an input value in the matrix (that is, how many times this value appears in the matrix), or the number of non-zero entries (measuring the sparsity of the matrix).

Chapter 5

Co-Clustering

Co-clustering, that is, partitioning a numerical matrix into "homogeneous" submatrices, has many applications ranging from bioinformatics to election analysis. We focus on the basic variant of co-clustering where the homogeneity of a submatrix is measured in terms of minimizing the maximum distance between two entries. In contrast to Chapter 4, where we combined two parameters, we now conduct a thorough multivariate complexity analysis involving several combinations of parameters such as the number of clusters, the number of rows, or the number of different values in the input matrix. As in Chapter 4, we spot several NP-hard as well as a number of polynomial-time solvable special cases, thus charting the border of tractability for this challenging data mining problem (however, we do not obtain a complete dichotomy). Moreover, we obtain some fixed-parameter tractability results by developing a reduction to SAT solving.

5.1 Introduction

Co-clustering, also known as *biclustering* [MO04], performs a simultaneous clustering of the rows and columns of a data matrix (hence, the term "co-clustering"). Roughly speaking, the task is, given a numerical input matrix $\mathcal{A}$, to partition the rows and columns of $\mathcal{A}$ into subsets minimizing a given *cost* function (measuring "homogeneity"). For a given subset I of rows and a subset J of columns, the corresponding *cluster* consists of all entries a_{ij} with $i \in I$ and $j \in J$. The cost function usually defines homogeneity in terms of distances (measured by some norm $\mathcal{L}$) between the entries of each cluster. An illustration is given in Figure 5.1. Note that the variant where clusters are allowed to "overlap", meaning that some rows and columns are contained in multiple clusters, has also been studied [MO04]. We focus on the non-overlapping variant which can be stated as follows.

$$\left[\begin{array}{cc|cc|cccc} 1 & 1 & 3 & 3 & 5 & 5 & 6 & 6 \\ 2 & 2 & 3 & 2 & 5 & 5 & 5 & 5 \\ 1 & 1 & 3 & 3 & 6 & 6 & 5 & 5 \\ \hline 3 & 3 & 0 & 1 & 4 & 4 & 4 & 4 \\ 3 & 2 & 0 & 1 & 4 & 4 & 4 & 4 \\ 2 & 2 & 0 & 1 & 4 & 4 & 4 & 4 \\ \hline 0 & 0 & 1 & 2 & 8 & 8 & 8 & 8 \\ 0 & 1 & 2 & 2 & 8 & 7 & 8 & 7 \end{array}\right]$$

Figure 5.1: An example of a co-clustering of an 8×8 matrix into $k = 3$ row subsets and $\ell = 3$ column subsets resulting in 9 clusters (submatrices). Within each cluster the maximum difference between any two values (this corresponds to the maximum norm L_∞) is at most 1.

Co-Clustering$_\mathcal{L}$
Input: A matrix $\mathcal{A} \in \mathbb{Q}^{m \times n}$ and two positive integers $k, \ell \in \mathbb{N}$.
Task: Find a partition of $\mathcal{A}$'s rows into k subsets and a partition of $\mathcal{A}$'s columns into ℓ subsets such that a given cost function (defined with respect to some norm $\mathcal{L}$) is minimized for the corresponding clustering.

Co-clustering is a fundamental method for unsupervised data analysis. Its applications range from microarrays and bioinformatics over recommender systems to election analysis [ADK12, Ban+07, MO04, TSS05]. Due to its enormous practical significance, there is a vast amount of literature discussing various variants; however, due to the observed NP-hardness of "almost all interesting variants" [MO04, Section 2.2], most of the literature deals with heuristic, typically empirically validated algorithms. Indeed, there has been very active research on co-clustering in terms of heuristic algorithms while there is little substantial theoretical work. Motivated by an effort towards a deeper theoretical understanding, we refine the theoretical analysis of the computational complexity of a natural special case of Co-Clustering$_\mathcal{L}$, namely we study the case of $\mathcal{L}$ being the maximum norm L_∞, where the problem comes down to minimizing the maximum distance between any two entries of a cluster. This cost function might be a reasonable choice in practice due to its outlier robustness. Also, by focussing on Co-Clustering$_\infty$, we investigate a scenario that is combinatorially easier to grasp. In particular, our exact and combinatorial

polynomial-time algorithms exploit structural properties of the input matrix. We also study a more restricted clustering version, where the row and column partitions have to contain subsets of consecutive indices. This version (see Section 5.3.3) subsumes the problem of feature discretization, which is used as a preprocessing technique in data mining [CN98, Ngu06, NS95].

Related Work. Our main point of reference is the work of Anagnostopoulos et al. [ADK12] dealing with Co-Clustering$_{\mathcal{L}}$ for L_p-norms ($p \geq 1$). The focus of their purely theoretical study is on polynomial-time approximation algorithms (Jegelka et al. [JSB09] further generalized the results to higher dimensions), but they also provide computational hardness results. In particular, they point to challenging open questions concerning the cases $k = \ell = 2$, $k = 1$, or binary input matrices. Within our more restricted setting using the maximum norm, we can resolve parts of these questions.

The survey of Madeira and Oliveira [MO04] provides an excellent overview on the many variations of Co-Clustering$_{\mathcal{L}}$, there called biclustering, and discusses many applications in bioinformatics and beyond. In particular, they also discuss Hartigan's [Har72] special case where the goal is to partition into uniform clusters (that is, each cluster has only one entry value). Our studies indeed generalize this very puristic scenario by not demanding completely uniform clusters (which would correspond to clusters with maximum entry difference 0) but allowing some variation between maximum and minimum cluster entries. In an experimental work, Califano et al. [CST00] aimed at clusterings where in each submatrix the distance between any two entries of a column is upper-bounded. This is a slightly weaker definition of cluster homogeneity than ours. Wulff et al. [WUB13] considered a so-called "monochromatic" biclustering where the cost for each submatrix is defined as the number of minority entries. For binary data, this clustering task coincides with Co-Clustering$_{L_1}$ as defined by Anagnostopoulos et al. [ADK12]. Wulff et al. [WUB13] showed NP-hardness of Monochromatic Biclustering for binary data with an additional third value denoting missing entries (which are not considered in their cost function) and give a randomized polynomial-time approximation scheme (PTAS). They also developed a heuristic approach which they evaluated in experiments. Feige [Fei14] gave a proof of NP-hardness of Hypercube 2-Segmentation which is equivalent to Co-Clustering$_{L_1}$ with $k = 2$ and $\ell = n$ for binary matrices (the reduction is from Max Cut and makes heavy use of so-called Hadamard codes). Note that in contrast to the L_1-version, we observe that Co-Clustering$_{L_\infty}$ is

easily solvable on binary matrices. Cheng et al. [Che+16] developed a greedy heuristic for hierarchical co-clustering based on mutual information.

Our Contributions. In terms of defining "cluster homogeneity", we focus on minimizing the maximum distance between two entries within a cluster (maximum norm). Figure 5.2 summarizes our results. Our main conceptual contribution is to provide a seemingly first study on the exact complexity of a natural special case of Co-Clustering$_{\mathcal{L}}$.

Our main technical contributions are as follows. We prove several computational intractability results even for strongly restricted cases (that is, for constant parameter values). Notably, we reveal non-obvious connections to other fields such as geometric covering and partitioning problems. Moreover, we demonstrate that the NP-hardness of Co-Clustering$_{L_\infty}$ does not stem from the inherent permutation combinatorics (of rows and columns): the problem remains NP-hard when all clusters consist of consecutive rows or columns. This is a strong constraint (the search space size is tremendously reduced—basically from $k^m \cdot \ell^n$ to $\binom{m}{k} \cdot \binom{n}{\ell}$) which directly gives a polynomial-time algorithm if k and ℓ are constants. Note that in the general case we have NP-hardness for $k = \ell = 3$. Concerning the algorithmic results, we develop a novel reduction to SAT solving. This reduction reveals interesting connections between co-clustering tasks and SAT solving, which yields polynomial-time algorithms and fixed-parameter tractability for some special cases.

Section 5.2 introduces the formal problem definition and some first observations. We start with the various NP-hardness results in Section 5.3. In Section 5.4, we describe the SAT solving approach and the obtained algorithmic results.

5.2 Problem Definition and First Observations

We follow the terminology of Anagnostopoulos et al. [ADK12]. For a matrix $\mathcal{A} \in \mathbb{Q}^{m \times n}$, a (k, ℓ)-*co-clustering* is a pair $(\mathcal{I}, \mathcal{J})$ consisting of a k-*partition* $\mathcal{I} = \{I_1, \ldots, I_k\}$ of the row indices $[m]$ of $\mathcal{A}$ (that is, $I_i \subseteq [m]$ for all $1 \leq i \leq k$, $I_i \cap I_j = \emptyset$ for all $1 \leq i < j \leq k$, and $\bigcup_{i=1}^{k} I_i = [m]$) and an ℓ-partition $\mathcal{J} = \{J_1, \ldots, J_\ell\}$ of the column indices $[n]$ of $\mathcal{A}$. We call the elements of $\mathcal{I}$ *row blocks* and the elements of $\mathcal{J}$ *column blocks*. Additionally, we require $\mathcal{I}$ and $\mathcal{J}$ to not contain empty sets. For $(r, s) \in [k] \times [\ell]$, the set $\mathcal{A}_{rs} := \{a_{ij} \in \mathcal{A} \mid (i, j) \in I_r \times J_s\}$ is called a *cluster*.

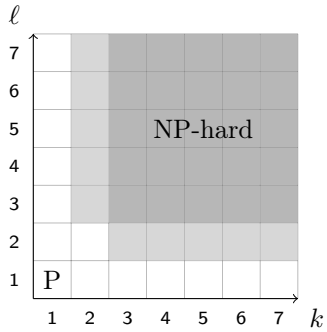


Figure 5.2: Overview of the computational complexity landscape of Co-Clustering$_\infty$ with respect to the number k of row clusters and the number ℓ of column clusters. White cells are polynomial-time solvable cases (Theorems 5.8 and 5.9). The dark gray cells are NP-hard even with an alphabet size $|\Sigma| = 3$ and a maximum allowed cost $c = 1$ (Theorem 5.3). For the light gray cells, we show the following: Co-Clustering$_\infty$ with $k = 2$ is polynomial-time solvable if $|\Sigma| \leq 3$ (Theorem 5.10), but NP-hard if $|\Sigma|$ is unbounded (Theorem 5.5). However, for $k = 2$ and $c = 1$, Co-Clustering$_\infty$ is fixed-parameter tractable with respect to ℓ and with respect to $|\Sigma|$ (Corollary 5.16).

The cost of a co-clustering (under maximum norm, which is the only norm we consider here) is defined as the maximum difference between any two entries in any cluster, formally

$$\text{cost}_\infty(\mathcal{I}, \mathcal{J}) := \max_{(r,s)\in[k]\times[\ell]} (\max \mathcal{A}_{rs} - \min \mathcal{A}_{rs}).$$

Herein, $\max \mathcal{A}_{rs}$ ($\min \mathcal{A}_{rs}$) denotes the maximum (minimum) entry in $\mathcal{A}_{rs}$.

The decision variant of Co-Clustering$_\mathcal{L}$ with maximum norm is as follows.

Co-Clustering$_\infty$
Input: A matrix $\mathcal{A} \in \mathbb{Q}^{m\times n}$, integers $k, \ell \in \mathbb{N}$, and a cost $c \geq 0$.
Question: Is there a (k, ℓ)-co-clustering $(\mathcal{I}, \mathcal{J})$ of $\mathcal{A}$ with $\text{cost}_\infty(\mathcal{I}, \mathcal{J}) \leq c$?

See Figure 5.3 for an introductory example. We define $\Sigma := \{a_{ij} \in \mathcal{A} \mid (i, j) \in [m] \times [n]\}$ to be the *alphabet* of the input matrix $\mathcal{A}$ (consisting of the numerical values that occur in $\mathcal{A}$). Note that $|\Sigma| \leq mn$. We use the abbreviation (k, ℓ)-Co-Clustering$_\infty$ to refer to Co-Clustering$_\infty$ with constants $k, \ell \in \mathbb{N}$, and

$$\mathcal{A} = \begin{bmatrix} 1 & 3 & 4 & 1 \\ 2 & 2 & 1 & 3 \\ 0 & 4 & 3 & 0 \end{bmatrix}$$

$$\begin{array}{c} \\ I_1 \\ I_2 \end{array} \begin{array}{c} \begin{array}{cccc} J_1 & & & J_2 \end{array} \\ \left[\begin{array}{ccc|c} 1 & 4 & 1 & 3 \\ \hline 2 & 1 & 3 & 2 \\ 0 & 3 & 0 & 4 \end{array}\right] \end{array} \qquad \begin{array}{c} \\ I_1 \\ I_2 \end{array} \begin{array}{c} \begin{array}{cccc} J_1 & & J_2 & \end{array} \\ \left[\begin{array}{cc|cc} 2 & 3 & 2 & 1 \\ \hline 1 & 1 & 3 & 4 \\ 0 & 0 & 4 & 3 \end{array}\right] \end{array}$$

$I_1 = \{1\}$, $I_2 = \{2, 3\}$ $\quad$ $I_1 = \{2\}$, $I_2 = \{1, 3\}$
$J_1 = \{1, 3, 4\}$, $J_2 = \{2\}$ $\quad$ $J_1 = \{1, 4\}$, $J_2 = \{2, 3\}$

Figure 5.3: The example shows two $(2, 2)$-co-clusterings (middle and right) of the same matrix $\mathcal{A}$ (left-hand side). It demonstrates that by sorting rows and columns according to the co-clustering, the clusters can be illustrated as submatrices of this (permuted) input matrix. The cost of the $(2, 2)$-co-clustering in the middle is three and the cost of the $(2, 2)$-co-clustering on the right-hand side is one.

by $(k, *)$-Co-Clustering$_\infty$ we refer to the case where only k is constant and ℓ is part of the input. Clearly, Co-Clustering$_\infty$ is symmetric with respect to k and ℓ in the sense that any (k, ℓ)-co-clustering of a matrix $\mathcal{A}$ is equivalent to an (ℓ, k)-co-clustering of the transposed matrix $\mathcal{A}^T$. Hence, we always assume that $k \leq \ell$.

We next collect some simple observations. First, determining whether there is a cost-zero (perfect) co-clustering is easy. Moreover, since, for a binary alphabet, the only interesting case is a perfect co-clustering, we get the following.

Observation 5.1. Co-Clustering$_\infty$ *is solvable in* $O(mn)$ *time for cost zero and also for any size-two alphabet.*

Proof. Let $(\mathcal{A}, k, \ell, 0)$ be a Co-Clustering$_\infty$ input instance. For a (k, ℓ)-co-clustering with cost 0, it holds that all entries of a cluster are equal. This is only possible if there are at most k different rows and at most ℓ different columns in $\mathcal{A}$ since otherwise there will be a cluster containing two different entries. Thus, the case $c = 0$ can be solved by lexicographically sorting the rows and columns of $\mathcal{A}$ in $O(mn)$ time (e.g. using radix sort). □

We further observe that the input matrix can, without loss of generality, be assumed to contain only integer values (by some rescaling arguments preserving the distance relations between elements).

Observation 5.2. *For any* Co-Clustering$_\infty$*-instance with arbitrary alphabet* $\Sigma \subset \mathbb{Q}$*, one can find in* $O(|\Sigma|^2)$ *time an equivalent instance with alphabet* $\Sigma' \subset \mathbb{Z}$ *and cost value* $c' \in \mathbb{N}$*.*

Proof. We show that for any instance with arbitrary alphabet $\Sigma \subset \mathbb{Q}$ and cost $c \geq 0$, there exists an equivalent instance with $\Sigma' \subset \mathbb{Z}$ and $c' \in \mathbb{N}$. Let σ_i be the i-th element of Σ with respect to any fixed ordering. The idea is that the cost value c determines which elements of Σ are allowed to appear together in a cluster of a cost-c co-clustering. Namely, in any cost-c co-clustering two elements $\sigma_i \neq \sigma_j$ can occur in the same cluster if and only if $|\sigma_i - \sigma_j| \leq c$. These constraints can be encoded in an undirected graph $G_c := (\Sigma, E)$ with $E := \{\{\sigma_i, \sigma_j\} \mid \sigma_i \neq \sigma_j \in \Sigma, |\sigma_i - \sigma_j| \leq c\}$, where each vertex corresponds to an element of Σ, and there is an edge between two vertices if and only if the corresponding elements can occur in the same cluster of a cost-c co-clustering.

Now, observe that G_c is a so-called *unit interval graph*, that is, each vertex σ_i can be represented by the length-c interval $[\sigma_i, \sigma_i + c]$ such that it holds $\{\sigma_i, \sigma_j\} \in E \Leftrightarrow [\sigma_i, \sigma_i + c] \cap [\sigma_j, \sigma_j + c] \neq \emptyset$ (here, we assume all intervals to contain real values). Hence, one can find in $O(|\Sigma|^2)$ time [Dur+15] an interval representation of G_c where the vertices σ_i are represented by length-c' intervals $[\sigma'_i, \sigma'_i + c']$ of equal integer length $c' \in \mathbb{N}$ with integer starting points $\sigma'_i \in \mathbb{Z}$ such that $0 \leq \sigma'_i \leq |\Sigma|^2$, $c' \leq |\Sigma|$, and $|\sigma'_i - \sigma'_j| \leq c' \Leftrightarrow |\sigma_i - \sigma_j| \leq c$. Hence, replacing the elements σ_i by σ'_i in the input matrix yields a matrix that has a cost-c' co-clustering if and only if the original input matrix has a cost-c co-clustering. Thus, for any instance with alphabet Σ and cost c, there is an equivalent instance with alphabet $\Sigma' \subseteq \{0, \ldots, |\Sigma|^2\}$ and cost $c' \in \{0, \ldots, |\Sigma|\}$. Consequently, we can upper-bound the values in Σ' by $|\Sigma|^2 \leq (mn)^2$. □

Due to Observation 5.2, we henceforth assume for the rest of the chapter that the input matrix contains integers, that is, $\mathcal{A} \in \mathbb{Z}^{m \times n}$.

5.3 Hardness Results

In the previous section, we observed that Co-Clustering$_\infty$ is easy to solve for binary input matrices (Observation 5.1). In contrast to this, we show in this section that its computational complexity significantly changes as soon as the input matrix contains at least three different values. In fact, even for very

restricted special cases we show NP-hardness. These special cases comprise co-clusterings with a constant number of clusters (Section 5.3.1) or input matrices with only two rows (Section 5.3.2). We also show NP-hardness of finding co-clusterings where the row and column partitions are only allowed to contain consecutive blocks (Section 5.3.3).

5.3.1 Constant Number of Clusters

We start by showing that for input matrices containing three different entries, Co-Clustering$_\infty$ is NP-hard even if the co-clustering consists only of nine clusters.

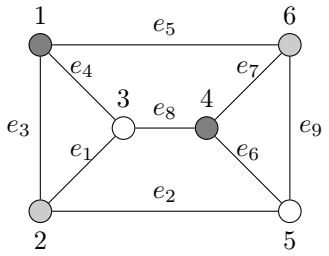
Theorem 5.3. *(k, ℓ)-Co-Clustering$_\infty$ is NP-hard for all $\ell \geq k \geq 3$ over alphabet $\Sigma = \{0, 1, 2\}$ and with maximum allowed cost $c = 1$.*

Proof. We prove NP-hardness with a polynomial-time many-one reduction from the NP-complete k-Coloring [GJ79].

k-Coloring
Input: An undirected graph $G = (V, E)$.
Question: Is there a partition of the vertices V into k subsets $V_1, \ldots, V_k$ such that the induced subgraph $G[V_i]$ is an independent set for each $i \in [k]$?

Let $G = (V, E)$ be a k-Coloring instance with $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$. We construct a (k, ℓ)-Co-Clustering$_\infty$ instance $(\mathcal{A} \in \{0, 1, 2\}^{n \times m}, k, \ell, c := 1)$ as follows. The rows of $\mathcal{A}$ correspond to the vertices V and the columns correspond to the edges E. For every edge $e_j = \{v_h, v_i\} \in E$ with $h < i$, we set $a_{hj} := 0$ and $a_{ij} := 2$. All other matrix entries are set to 1. Hence, each column corresponding to an edge $\{v_h, v_i\}$ consists of 1-entries except for the rows h and i, which contain 0 and 2 (see Figure 5.4). Thus, every co-clustering of $\mathcal{A}$ with cost at most $c = 1$ puts row h and row i into different row blocks. We next prove that there is a (k, ℓ)-co-clustering of $\mathcal{A}$ with cost at most $c = 1$ if and only if G admits a k-coloring.

First, assume that $V_1, \ldots, V_k$ is a partition of the vertex set V into k independent sets. We define a (k, ℓ)-co-clustering $(\mathcal{I}, \mathcal{J})$ of $\mathcal{A}$ as follows: The row partition $\mathcal{I} := \{I_1, \ldots, I_k\}$ one-to-one corresponds to the k sets $V_1, \ldots, V_k$, that is, $I_s := \{i \mid v_i \in V_s\}$ for all $s \in [k]$. By the construction above, each column has exactly two non-1-entries being 0 and 2. The column partition $\mathcal{J} := \{J_1, \ldots, J_\ell\}$ is defined as follows: Column block J_s, $s \in [\ell]$, contains all columns which have



	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9
2	0	0	2	1	1	1	1	1	1
6	1	1	1	1	2	1	2	1	2
1	1	1	0	0	0	1	1	1	1
4	1	1	1	1	1	0	0	2	1
3	2	1	1	2	1	1	1	0	1
5	1	2	1	1	1	2	1	1	0

Figure 5.4: An illustration of the polynomial-time many-one reduction from 3-Coloring. Left: An undirected graph with a proper 3-coloring of the vertices such that no two neighboring vertices have the same color. Right: The corresponding matrix where the columns are labeled by vertices and the rows by edges with a $(3, 3)$-co-clustering of cost 1. The coloring of the vertices determines the row partition into three row blocks, whereas the column blocks are generated by the following simple scheme: Columns corresponding to edges where the vertex with smaller index is light gray/dark gray/white are in the first/second/third column block.

the 0-entry in row block I_s. If $\ell > k$, then we simply put an arbitrary column into each column cluster J_s with $s > k$. Clearly, it holds that the non-1-entries in any cluster of $(\mathcal{I}, \mathcal{J})$ are either all 0 or all 2, implying that $\text{cost}_\infty(\mathcal{I}, \mathcal{J}) \leq 1$.

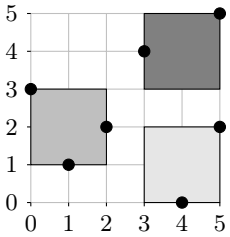
Next, assume that $(\{I_1, \ldots, I_k\}, \mathcal{J})$ is a (k, ℓ)-co-clustering of $\mathcal{A}$ with cost at most 1. The vertex subsets $V_1, \ldots, V_k$, where V_s, $s \in [k]$, contains the vertices corresponding to the rows in I_s, form k independent sets: If there is an edge between two vertices in V_s, then the corresponding column would have the 0-entry and the 2-entry in the same row block I_s, yielding a cost of 2, which is a contradiction. □

It is an easy folklore result that k-Coloring is also NP-hard if all k vertex subsets are required to have the same size (by adding enough vertices to the input graph). Hence, we obtain the following corollary of Theorem 5.3.

Corollary 5.4. (k, ℓ)-Co-Clustering$_\infty$ *is NP-hard for all* $\ell \geq k \geq 3$ *with* $\Sigma = \{0, 1, 2\}$ *and* $c = 1$ *if the row blocks are forced to have equal sizes.*

5.3.2 Constant Number of Rows

The reduction in the proof of Theorem 5.3 generates matrices containing only three different values with an unbounded number of rows (and columns). We



0	1	2	3	5	4	5
3	1	2	4	5	0	2

Figure 5.5: Example of a Box Cover instance with seven points (left) and the corresponding Co-Clustering$_\infty$ matrix containing the coordinates of the points as columns (right). Indicated is a $(2, 3)$-co-clustering of cost 2 where the column blocks are colored according to the three squares (of side length 2) that cover all points.

now show that also the "dual restriction" is NP-hard, that is, the input matrix only has a constant number of rows (two) but contains an unbounded number of different values. Interestingly, this special case is closely related to a two-dimensional variant of geometric set covering.

Theorem 5.5. Co-Clustering$_\infty$ *is NP-hard for* $k = m = 2$ *and unbounded alphabet size* $|\Sigma|$.

Proof. We give a polynomial-time many-one reduction from the NP-complete Box Cover problem [FPT81].

Box Cover
Input: A set $P \subseteq \mathbb{Z}^2$ of n points in the plane and $\ell \in \mathbb{N}$.
Question: Are there ℓ squares $S_1, \ldots, S_\ell$, each with side length 2, that cover P, that is, $P \subseteq \bigcup_{1 \leq i \leq \ell} S_i$?

Let $I = (P, \ell)$ be a Box Cover instance. We define the instance $I' := (\mathcal{A}, k, \ell', c)$ as follows: The matrix $\mathcal{A} \in \mathbb{Z}^{2 \times n}$ has the points $p_1, \ldots, p_n$ in P as columns. Further, we set $k := 2$, $\ell' := \ell$, $c := 2$. See Figure 5.5 for a small example.

The correctness can be seen as follows: Assume that I is a yes-instance, that is, there are ℓ squares $S_1, \ldots, S_\ell$ covering all points in P. We define $J_1 := \{j \mid p_j \in P \cap S_1\}$ and $J_s := \{j \mid p_j \in P \cap S_s \setminus (\bigcup_{1 \leq l < s} S_l)\}$ for all $2 \leq s \leq \ell$. Note that $(\mathcal{I} := \{\{1\}, \{2\}\}, \mathcal{J} := \{J_1, \ldots, J_\ell\})$ is a $(2, \ell)$-co-clustering of $\mathcal{A}$. Moreover, since all points with indices in J_s lie inside a square with side length 2, it holds

that each pair of entries in $\mathcal{A}_{1s}$ as well as in $\mathcal{A}_{2s}$ has distance at most 2, implying $\text{cost}_\infty(\mathcal{I}, \mathcal{J}) \leq 2$.

Conversely, if I' is a yes-instance, then let $(\{\{1\}, \{2\}\}, \mathcal{J})$ be the $(2, \ell)$-co-clustering of cost at most 2. For any $J_s \in \mathcal{J}$, it holds that all points corresponding to the columns in J_s have pairwise distance at most 2 in both coordinates. Thus, there exists a square of side length 2 covering all of them. □

5.3.3 Clustering into Consecutive Clusters

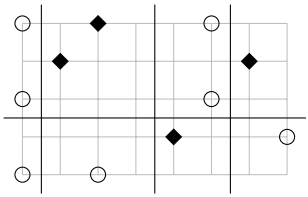
One is tempted to assume that the hardness of the previous special cases of Co-Clustering$_\infty$ is rooted in the fact that we are allowed to choose arbitrary subsets for the corresponding row and column partitions since the problem remains hard even for a constant number of clusters even with equal cluster sizes. Hence, in this section, we consider a restricted version of Co-Clustering$_\infty$, where the row and the column partition has to consist of consecutive blocks.

Formally, for row indices $R = \{r_1, \ldots, r_{k-1}\}$ with $1 < r_1 < \ldots < r_{k-1} \leq m$ and column indices $C = \{c_1, \ldots, c_{\ell-1}\}$ with $1 < c_1 < \ldots < c_{\ell-1} \leq n$, the corresponding *consecutive* (k, ℓ)-co-clustering $(\mathcal{I}_R, \mathcal{J}_C)$ is defined as

$$\mathcal{I}_R := \{\{1, \ldots, r_1 - 1\}, \{r_1, \ldots, r_2 - 1\}, \ldots, \{r_{k-1}, \ldots, m\}\},$$
$$\mathcal{J}_C := \{\{1, \ldots, c_1 - 1\}, \{c_1, \ldots, c_2 - 1\}, \ldots, \{c_{\ell-1}, \ldots, n\}\}.$$

The Consecutive Co-Clustering$_\infty$ problem now is to find a consecutive (k, ℓ)-co-clustering of a given input matrix with a given cost.

As it turns out, also this restriction is not sufficient to overcome the inherent intractability of co-clustering, that is, we prove it to be NP-hard. Similarly to Section 5.3.2, we encounter a close relation between consecutive co-clustering and a geometric problem, namely to find an optimal discretization of the plane; a preprocessing problem with applications in data mining [CN98, Ngu06, NS95]. The NP-hard Optimal Discretization problem [CN98] is the following: Given a set of points in the plane, where each point is either colored black or white, the task is to find a consistent set of axis-parallel lines, that is, the vertical and horizontal lines partition the plane into rectangular regions such that no region contains two points of different colors (see Figure 5.6 for an example).



$$\left[\begin{array}{cccccccc}
0 & 1 & 2 & 1 & 1 & 0 & 1 & 1 \\
1 & 2 & 1 & 1 & 1 & 1 & 2 & 1 \\
0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\
\hline
1 & 1 & 1 & 1 & 2 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 1 & 1
\end{array}\right]$$

Figure 5.6: Example instance of Optimal Discretization (left) and the corresponding instance of Consecutive Co-Clustering$_\infty$ (right). The point set consists of white (circles) and black (diamonds) points. A solution for the corresponding Consecutive Co-Clustering$_\infty$ instance (shaded clusters) naturally translates into a consistent set of lines.

Optimal Discretization
Input: A set $S = \{p_1 = (x_1, y_1), \ldots, p_s = (x_s, y_s)\} \subseteq \mathbb{Q}^2$ of s different points in the plane, partitioned into two disjoint subsets B, W and integers $k, \ell \in \mathbb{N}$.
Question: Is there a consistent set of k horizontal and ℓ vertical lines?

Here, a vertical (horizontal) line is a simple number denoting its x-coordinate (y-coordinate).

Theorem 5.6. Consecutive Co-Clustering$_\infty$ *is NP-hard for an alphabet* $\Sigma = \{0, 1, 2\}$ *and maximum allowed cost* $c = 1$.

Proof. We give a polynomial-time many-one reduction from Optimal Discretization. Let (S, k, ℓ) be an Optimal Discretization instance and let $X := \{x_1^*, \ldots, x_n^*\}$ be the set of different x-coordinates and let $Y := \{y_1^*, \ldots, y_m^*\}$ be the set of different y-coordinates of the points in S. Note that n and m can be smaller than $|S|$ since two points can have the same x- or y-coordinate. Furthermore, assume that $x_1^* < \ldots < x_n^*$ and $y_1^* < \ldots < y_m^*$. We now define the Consecutive Co-Clustering$_\infty$ instance $(\mathcal{A}, k+1, \ell+1, c)$ as follows: The matrix $\mathcal{A} \in \{0,1,2\}^{m \times n}$ has columns labeled with $x_1^*, \ldots, x_n^*$ and rows labeled with $y_1^*, \ldots, y_m^*$. For $(x, y) \in X \times Y$, the entry a_{xy} is defined as 0 if $(x, y) \in W$, 2 if $(x, y) \in B$, and otherwise 1. The cost is set to $c := 1$. Clearly, this instance can be constructed in polynomial time.

To verify the correctness of the reduction, assume first that I is a yes-instance, that is, there is a set $H = \{x_1, \ldots, x_k\}$ of k horizontal lines and a set $V = \{y_1, \ldots, y_\ell\}$ of ℓ vertical lines partitioning the plane consistently. We

define row indices $R := \{r_1, \ldots, r_k\}$, $r_i := \max\{x^* \in X \mid x^* \leq x_i\}$, $i = 1, \ldots, k$, and column indices $C := \{c_1, \ldots, c_\ell\}$, $c_j := \max\{y^* \in Y \mid y^* \leq y_j\}$, $j = 1, \ldots, \ell$. For the corresponding consecutive $(k+1, \ell+1)$-co-clustering $(\mathcal{I}_R, \mathcal{J}_C)$, it holds that no cluster contains both values 0 and 2, since otherwise the corresponding partition of the plane defined by H and V contains a region with two points of different colors, which contradicts consistency. Thus, we have $\text{cost}_\infty(\mathcal{I}_R, \mathcal{J}_C) \leq 1$, implying that I' is a yes-instance.

Conversely, if I' is a yes-instance, then there exists a consecutive $(k+1, \ell+1)$-co-clustering $(\mathcal{I}_R, \mathcal{J}_C)$ with cost at most 1, that is, no cluster contains both values 0 and 2. Clearly, then the k horizontal lines $x_i := \min I_{i+1}$, $i = 1, \ldots, k$, and the ℓ vertical lines $y_j := \min J_{j+1}$, $j = 1, \ldots, \ell$, are consistent. Hence, I is a yes-instance. □

Note that even though Consecutive Co-Clustering$_\infty$ is NP-hard, there still is some difference in its computational complexity compared to the general version. In contrast to Co-Clustering$_\infty$, the consecutive version is polynomial-time solvable for constants k and ℓ by simply trying out all $O(m^k n^\ell)$ consecutive partitions of the rows and columns (that is, it is contained in XP with respect to (k, ℓ)).

5.4 Tractability Results

In Section 5.3, we showed that Co-Clustering$_\infty$ is NP-hard for all constants $\ell \geq k \geq 3$ and also for $k = 2$ in case of unbounded ℓ and $|\Sigma|$. In contrast to these hardness results, we now investigate which parameter combinations yield tractable cases. It turns out that the problem is polynomial-time solvable for $k = \ell = 2$ and for $k = 1$ (Section 5.4.2). Moreover, we can solve the case $k = 2$ and $\ell \geq 3$ for $|\Sigma| = 3$ in polynomial time by showing that this case is in fact equivalent to the case $k = \ell = 2$. We further show fixed-parameter tractability (for $k = 2$ and $c = 1$) for the parameters size $|\Sigma|$ of the alphabet and the number ℓ of column blocks (Section 5.4.3).

In Section 5.4.1, we start with a reduction from Co-Clustering$_\infty$ to CNF-SAT (the satisfiability problem for Boolean formulas in conjunctive normal form). Later on, it will be used in some special cases (see Theorem 5.9 and Theorem 5.11) because there the corresponding formula—or an equivalent formula—only consists of clauses containing two literals, thus being a polynomial-time solvable 2-SAT instance.

$$\mathcal{U} = \begin{bmatrix} 0 & 2 & 4 \\ 1 & 0 & 4 \\ 0 & 1 & 7 \end{bmatrix}$$

$$\mathcal{A} = \left[\begin{array}{cc|cc|cccc} 1 & 1 & 4 & 3 & 5 & 5 & 6 & 6 \\ 2 & 2 & 3 & 2 & 5 & 5 & 4 & 5 \\ 0 & 1 & 3 & 3 & 6 & 6 & 4 & 5 \\ \hline 3 & 3 & 0 & 1 & 4 & 6 & 4 & 4 \\ 3 & 2 & 0 & 1 & 5 & 4 & 4 & 4 \\ 2 & 2 & 0 & 1 & 4 & 4 & 6 & 4 \\ \hline 2 & 0 & 1 & 2 & 8 & 8 & 8 & 8 \\ 0 & 2 & 2 & 2 & 8 & 7 & 8 & 7 \end{array}\right]$$

Figure 5.7: A 3×3 cluster boundary $\mathcal{U}$ (left) and an 8×8 matrix $\mathcal{A}$ (right). The indicated $(3,3)$-co-clustering of $\mathcal{A}$ satisfies the cluster boundary $\mathcal{U}$ for $c = 2$, since for all $(r,s) \in [3] \times [3]$, the cluster $\mathcal{A}_{rs}$ contains only values from the interval $[u_{rs}, u_{rs}+2]$.

5.4.1 Reduction to CNF-SAT Solving

In this section we describe an approach to solve Co-Clustering$_\infty$ via CNF-SAT which leads to some polynomial-time solvable and fixed-parameter tractable special cases. More precisely, our approach requires to solve $O(|\Sigma|^{k\ell})$ many CNF-SAT instances with clauses of size at most $\max\{k, \ell, 2\}$ (if k and ℓ are constants, then there are only polynomially many CNF-SAT instances to solve, and for $k \leq \ell \leq 2$, we obtain polynomial-time solvable 2-SAT instances).

To this end, we introduce the concept of *cluster boundaries*, which impose a certain structure on the co-clustering. Basically, a cluster boundary defines lower (and upper) bounds for the values in each cluster of a co-clustering (see Figure 5.7).

Definition 5.1. Given an alphabet $\Sigma \subseteq \mathbb{Z}$ and two integers k, ℓ, we define a cluster boundary to be a matrix $\mathcal{U} = (u_{rs}) \in \Sigma^{k \times \ell}$. For a matrix $\mathcal{A} \in \Sigma^{m \times n}$ and a cost c, we say that a given (k,ℓ)-co-clustering of $\mathcal{A}$ *satisfies* a cluster boundary $\mathcal{U}$ if $\mathcal{A}_{rs} \subseteq [u_{rs}, u_{rs} + c]$ for all $(r,s) \in [k] \times [\ell]$.

It is easy to see that a given (k,ℓ)-co-clustering has cost at most c if and only if it satisfies at least one cluster boundary (u_{rs}), namely, the one with $u_{rs} = \min \mathcal{A}_{rs}$ for all $(r,s) \in [k] \times [\ell]$.

The following "subtask" of Co-Clustering$_\infty$ can be reduced to a certain CNF-SAT instance: Given a cluster boundary $\mathcal{U}$ and a Co-Clustering$_\infty$

instance I, find a co-clustering for I that satisfies $\mathcal{U}$. The polynomial-time reduction provided by the following central lemma can be used to obtain exact Co-Clustering$_\infty$ solutions with the help of SAT solvers and we use it in our subsequent algorithms.

Lemma 5.7. *Given a* Co-Clustering$_\infty$*-instance* $(\mathcal{A}, k, \ell, c)$ *and a cluster boundary* $\mathcal{U} \in \Sigma^{k\times\ell}$*, one can construct in* $O(mnk\ell)$ *time a* CNF-SAT *instance* $\phi_{\mathcal{A},\mathcal{U}}$ *with at most* $\max\{k, \ell, 2\}$ *variables per clause such that* $\phi_{\mathcal{A},\mathcal{U}}$ *is satisfiable if and only if there is a* (k, ℓ)*-co-clustering of* $\mathcal{A}$ *which satisfies* $\mathcal{U}$*.*

Proof. Given an instance $(\mathcal{A}, k, \ell, c)$ of Co-Clustering$_\infty$ and a cluster boundary $\mathcal{U} = (u_{rs}) \in \Sigma^{k\times\ell}$, we define the following Boolean variables: For each $(i, r) \in [m] \times [k]$, the variable $x_{i,r}$ represents the expression "row i could be put into row block I_r". Similarly, for each $(j, s) \in [n] \times [\ell]$, the variable $y_{j,s}$ represents that "column j could be put into column block J_s".

We now define a Boolean CNF formula $\phi_{\mathcal{A},\mathcal{U}}$ containing the following clauses: A clause $R_i := (x_{i,1} \vee x_{i,2} \vee \ldots \vee x_{i,k})$ for each row $i \in [m]$ and a clause $C_j := (y_{j,1} \vee y_{j,2} \vee \ldots \vee y_{j,\ell})$ for each column $j \in [n]$. Additionally, for each $(i, j) \in [m] \times [n]$ and each $(r, s) \in [k] \times [\ell]$ such that element a_{ij} does not fit into the cluster boundary at coordinate (r, s), that is, $a_{ij} \notin [u_{rs}, u_{rs} + c]$, there is a clause $B_{ijrs} := (\neg x_{i,r} \vee \neg y_{j,s})$. Note that the clauses R_i and C_j ensure that row i and column j are put into some row and some column block, respectively. The clause B_{ijrs} expresses that it is impossible to have both row i in block I_r and column j in block J_s if a_{ij} does not satisfy $u_{rs} \leq a_{ij} \leq u_{rs} + c$. Clearly, $\phi_{\mathcal{A},\mathcal{U}}$ is satisfiable if and only if there exists a (k, ℓ)-co-clustering of $\mathcal{A}$ satisfying the cluster boundary $\mathcal{U}$. Note that $\phi_{\mathcal{A},\mathcal{U}}$ consists of $km + \ell n$ variables and $O(mnk\ell)$ clauses and is computable in $O(mnkl)$ time. □

Using Lemma 5.7, we can solve Co-Clustering$_\infty$ by solving $O(|\Sigma|^{k\ell})$ many CNF-SAT instances (one for each possible cluster boundary) with $km + \ell n$ variables and $O(mnk\ell)$ clauses of size at most $\max\{k, \ell, 2\}$.

5.4.2 Polynomial-Time Solvability

We first present a simple and efficient algorithm for $(1, *)$-Co-Clustering$_\infty$, that is, the variant where all rows belong to one row block.

Theorem 5.8. $(1, *)$-Co-Clustering$_\infty$ *is solvable in* $O(n(m + \log n))$ *time.*

Algorithm 5.1: Algorithm for $(1,*)$-Co-Clustering$_\infty$

Input: $\mathcal{A} \in \mathbb{Z}^{m \times n}$, $\ell \geq 1$, $c \geq 0$.
Output: A partition of $[n]$ into at most ℓ blocks yielding a cost of at most c, or `no` if no such partition exists.

```
1  for j ← 1 to n do                    // compute minimum and maximum of each column
2      α_j ← min{a_ij | 1 ≤ i ≤ m}
3      β_j ← max{a_ij | 1 ≤ i ≤ m}
4      if β_j − α_j > c then             // check if solution is possible
5          return no;
6  N ← [n]
7  for s ← 1 to ℓ do                    // fill column blocks with columns
8      Let j_s ∈ N be the index such that α_{j_s} is minimal
9      J_s ← {j ∈ N | β_j − α_{j_s} ≤ c}
10     N ← N \ J_s
11     if N = ∅ then                    // if no columns remain, then return solution
12         return (J_1, ..., J_s)
13 return no
```

Proof. We show that Algorithm 5.1 solves $(1,*)$-Co-Clustering$_\infty$. In fact, it even computes the minimum ℓ' such that $\mathcal{A}$ has a $(1,\ell')$-co-clustering of cost c. The overall idea is that with only one row block, all entries of a column j are contained in a cluster in any solution, and thus, it suffices to consider only the minimum α_j and the maximum β_j value in column j. More precisely, for a column block $J \subseteq [n]$ of a solution it follows that $\max\{\beta_j \mid j \in J\} - \min\{\alpha_j \mid j \in J\} \leq c$. The algorithm starts with the column j_1 that contains the overall minimum value α_{j_1} of the input matrix, that is, $\alpha_{j_1} = \min\{\alpha_j \mid j \in [n]\}$. Clearly, j_1 has to be contained in some column block, say J_1. The algorithm then adds all other columns j to J_1 where $\beta_j \leq \alpha_{j_1} + c$, removes the columns J_1 from the matrix, and recursively proceeds with the column containing the minimum value of the remaining matrix.

We continue with the correctness of the described procedure. If Algorithm 5.1 returns $(J_1, \ldots, J_{\ell'})$ at Line 12, then this is a column partition into $\ell' \leq \ell$ blocks satisfying the cost constraint. First, it is a partition by construction: The sets J_s are successively removed from $\mathcal{N}$ until it is empty. Now, let $s \in [\ell']$.

Then, for all $j \in J_s$, it holds $\alpha_j \geq \alpha_{j_s}$ (by definition of j_s) and $\beta_j \leq \alpha_{j_s} + c$ (by definition of J_s). Thus, $\mathcal{A}_{1s} \subseteq [\alpha_{j_s}, \alpha_{j_s} + c]$ holds for all $s \in [\ell']$, which yields $\mathrm{cost}_\infty(\{[m]\}, \{J_1, \ldots, J_{\ell'}\}) \leq c$.

Otherwise, if Algorithm 5.1 returns `no` in Line 5, then it is clearly a no-instance since the difference between the maximum and the minimum value in a column is larger than c. If `no` is returned in Line 13, then the algorithm has computed column indices j_s and column blocks J_s for each $s \in [\ell]$, and there still exists at least one index $j_{\ell+1}$ in $\mathcal{N}$ when the algorithm terminates. We claim that the columns $j_1, \ldots, j_{\ell+1}$ all have to be in different blocks in any solution. To see this, consider any $s, s' \in [\ell + 1]$ with $s < s'$. By construction, $j_{s'} \notin J_s$. Therefore, $\beta_{j_{s'}} > \alpha_{j_s} + c$ holds, and columns j_s and $j_{s'}$ contain elements with distance more than c. Thus, in any co-clustering with cost at most c, the columns $j_1, \ldots, j_{\ell+1}$ must be in different blocks, which is impossible with only ℓ blocks. Hence, we indeed have a no-instance.

The time complexity is seen as follows. The first loop examines in $O(mn)$ time all elements of the matrix. The second loop can be performed in $O(n)$ time if the α_j and the β_j are sorted beforehand, requiring $O(n \log n)$ time. Overall, the running time is in $O(n(m + \log n))$. □

From now on, we focus on the $k = 2$ case, that is, we need to partition the rows into two blocks. We first consider the simplest case, where also $\ell = 2$.

Theorem 5.9. $(2, 2)$-Co-Clustering$_\infty$ *is solvable in* $O(|\Sigma|^2 mn)$ *time.*

Proof. Let $(\mathcal{A} \in \mathbb{Z}^{m \times n}, k = 2, \ell = 2, c)$ be a $(2, 2)$-Co-Clustering$_\infty$ instance. We use the reduction to CNF-SAT provided by Lemma 5.7. First, note that a cluster boundary $\mathcal{U} \in \Sigma^{2\times 2}$ can only be satisfied if it contains the elements $\min \Sigma$ and $\min\{a \in \Sigma \mid a \geq \max \Sigma - c\}$. The algorithm enumerates all $O(|\Sigma|^2)$ of these cluster boundaries. For a fixed $\mathcal{U}$, we construct the Boolean formula $\phi_{\mathcal{A},\mathcal{U}}$ in $O(mn)$ time. Observe that this formula is in 2-CNF form: The formula consists of k-clauses, ℓ-clauses, and 2-clauses, and we have $k = \ell = 2$. Hence, we can determine whether it is satisfiable in linear time [APT79] (note that the size of the formula is in $O(mn)$). Overall, the input is a yes-instance if and only if $\phi_{\mathcal{A},\mathcal{U}}$ is satisfiable for some cluster boundary $\mathcal{U}$. □

Finally, we show that it is possible to extend the above result to any number of column blocks for size-three alphabets.

Theorem 5.10. $(2,*)$-Co-Clustering$_\infty$ *is* $O(mn)$*-time solvable for* $|\Sigma| = 3$.

Proof. Let $I = (\mathcal{A} \in \{\alpha, \beta, \gamma\}^{m \times n}, k = 2, \ell, c)$ be a $(2,*)$-Co-Clustering$_\infty$ instance. We assume without loss of generality that $\alpha < \beta < \gamma$. The case $\ell \leq 2$ is solvable in $O(mn)$ time by Theorem 5.9. Hence, it remains to consider the case $\ell \geq 3$. As $|\Sigma| = 3$, there are four potential values for a minimum-cost $(2,\ell)$-co-clustering. Namely, cost 0 (all cluster entries are equal), cost $\beta - \alpha$, cost $\gamma - \beta$, and cost $\gamma - \alpha$. Since any $(2,\ell)$-co-clustering is of cost at most $\gamma - \alpha$ and because it can be checked in $O(mn)$ time whether there is a $(2,\ell)$-co-clustering of cost 0 (Observation 5.1), there are only two interesting cases left, that is, $c \in \{\beta - \alpha, \gamma - \beta\}$.

Avoiding a pair $(x, y) \in \{\alpha, \beta, \gamma\}^2$ means to find a co-clustering without a cluster containing x and y. If $c = \max\{\beta - \alpha, \gamma - \beta\}$ (Case 1), then the problem comes down to finding a $(2,\ell)$-co-clustering avoiding the pair (α, γ). Otherwise (Case 2), the problem is to find a $(2,\ell)$-co-clustering avoiding the pair (α, γ) and, additionally, either (α, β) or (β, γ).

Case 1. Finding a $(2,\ell)$-co-clustering avoiding (α, γ):
In this case, we substitute $\alpha := 0$, $\beta := 1$, and $\gamma := 2$. We describe an algorithm for finding a $(2,\ell)$-co-clustering of cost 1 (avoiding $(0, 2)$). We assume that there is no $(2,\ell-1)$-co-clustering of cost 1 (iterating over all values from 2 to ℓ). Consider a $(2,\ell)$-co-clustering $(\mathcal{I}, \mathcal{J} = \{J_1, \ldots, J_\ell\})$ of cost 1, that is, for all $(r, s) \in [2] \times [\ell]$, it holds $\mathcal{A}_{rs} \subseteq \{0, 1\}$ or $\mathcal{A}_{rs} \subseteq \{1, 2\}$. For $s \neq t \in [\ell]$, let $(\mathcal{I}, \mathcal{J}_{st} := \mathcal{J} \setminus \{J_s, J_t\} \cup \{J_s \cup J_t\})$ denote the $(2,\ell-1)$-co-clustering where the column blocks J_s and J_t are merged. By assumption, for all $s \neq t \in [\ell]$, it holds that $\text{cost}_\infty(\mathcal{I}, \mathcal{J}_{st}) > 1$ since otherwise we have found a $(2,\ell-1)$-co-clustering of cost 1. It follows that $\{0, 2\} \subseteq \mathcal{A}_{1s} \cup \mathcal{A}_{1t}$ or $\{0, 2\} \subseteq \mathcal{A}_{2s} \cup \mathcal{A}_{2t}$ holds for all $s \neq t \in [\ell]$. This can only be true for $\ell = 2$.

This proves that there is a $(2,\ell)$-co-clustering of cost 1 if and only if there is a $(2, 2)$-co-clustering of cost 1. Hence, Theorem 5.9 shows that this case is $O(mn)$-time solvable.

Case 2. Finding a $(2,\ell)$-co-clustering avoiding (α, γ) and (α, β) (or (β, γ)):
In this case, we substitute $\alpha := 0$, $\gamma := 1$, and $\beta := 1$ if (α, β) has to be avoided, or $\beta := 0$ if (β, γ) has to be avoided. It remains to determine whether there is a $(2,\ell)$-co-clustering with cost 0, which can be done in $O(mn)$ time due to Observation 5.1. □

It is open whether polynomial-time solvability of $(2,*)$-Co-Clustering$_\infty$ also holds for larger (constant-size) alphabets. In the following section, however, we

show fixed-parameter tractability with respect to the combined parameter $(\ell, |\Sigma|)$ if $c = 1$.

5.4.3 Fixed-Parameter Tractability

The hardness results from Section 5.3 show that Co-Clustering$_\infty$ is computationally hard to solve even from a parameterized view (for each of the five parameters we considered, NP-hardness holds for a small constant). This indicates that fixed-parameter tractability might be a challenging goal to achieve. Based on our reduction to CNF-SAT (Lemma 5.7), however, we show fixed-parameter tractability with respect to some parameter combinations for the special case $k = 2$ and $c = 1$.

We develop an algorithm solving $(2, *)$-Co-Clustering$_\infty$ for $c = 1$. The main idea is, given matrix $\mathcal{A}$ and cluster boundary $\mathcal{U}$, to simplify the Boolean formula $\phi_{\mathcal{A},\mathcal{U}}$ into a 2-CNF formula which can be solved efficiently. This is made possible by the constraint $c = 1$, which imposes a very specific structure on the cluster boundary. This approach requires to enumerate all (exponentially many) possible cluster boundaries, but yields fixed-parameter tractability for the combined parameter $(\ell, |\Sigma|)$.

Theorem 5.11. $(2, *)$-Co-Clustering$_\infty$ *is* $O(|\Sigma|^{3\ell} n^2 m^2)$*-time solvable for* $c = 1$.

In the following, we prove Theorem 5.11 in several steps.

A first subresult for the proof of Theorem 5.11 is the following lemma, which we use to solve the case where the number 2^m of possible row partitions is less than $|\Sigma|^\ell$.

Lemma 5.12. *For a fixed row partition* $\mathcal{I}$ *of size* k*, one can solve* $(k, *)$-Co-Clustering$_\infty$ *in* $O(|\Sigma|^{k\ell} mn\ell)$ *time. Moreover,* Co-Clustering$_\infty$ *is fixed-parameter tractable with respect to the combined parameter* (m, k, ℓ, c)*.*

Proof. Given a fixed size-k row partition $\mathcal{I}$, the algorithm enumerates all $|\Sigma|^{k\ell}$ different cluster boundaries $\mathcal{U} = (u_{rs})$. We say that a given column j *fits* in column block J_s if, for each $r \in [k]$ and $i \in I_r$, we have $a_{ij} \in [u_{rs}, u_{rs} + c]$ (this can be decided in $O(m)$ time for any pair (j, s)). The input is a yes-instance if and only if for some cluster boundary $\mathcal{U}$, every column fits in at least one column block.

Fixed-parameter tractability with respect to (m, k, ℓ, c) is obtained from two simple further observations. First, all possible row partitions can be enumerated

in $O(k^m)$ time. Second, since each of the $k\ell$ clusters contains at most $c+1$ different values, the alphabet size $|\Sigma|$ for yes-instances is upper-bounded by $(c+1)k\ell$. Together, this yields a running time in $O(k^m((c+1)k\ell)^{k\ell}mn\ell)$. □

The following lemma, also used for the proof of Theorem 5.11, yields that for specially structured cluster boundaries, there is no need to consider more than two column clusters to which any column can be assigned.

Lemma 5.13. *Let $I = (\mathcal{A} \in \mathbb{Z}^{m\times n}, k = 2, \ell, c = 1)$ be an instance of $(2,*)$-Co-Clustering$_\infty$, h_1 be an integer with $0 < h_1 < m$, and $\mathcal{U} = (u_{rs}) \in \Sigma^{2\times\ell}$ be a cluster boundary with pairwise different columns such that $|u_{1s} - u_{2s}| = 1$ for all $s \in [\ell]$. Then, in $O(n(m+\ell))$ time, one can compute two indices $s_j \in [\ell]$ and $t_j \in [\ell]$ for each column $j \in [n]$, such that the following holds:*

If $\mathcal{A}$ has a $(2,\ell)$-co-clustering $(\{I_1, I_2\}, \{J_1, \ldots, J_\ell\})$ satisfying $\mathcal{U}$ with $|I_1| = h_1$, then either $j \in J_{s_j}$ or $j \in J_{t_j}$ holds.

Proof. To start with, let us define some notation. Given a column $j \in [n]$ and any element $a \in \Sigma$, we write $w_a(j)$ for the number of a-entries in column j. For $s \in [\ell]$, we define $U_{1s} := \{u_{1s}, u_{1s}+1\}$, $U_{2s} := \{u_{2s}, u_{2s}+1\}$ and let α be the integer in $U_{1s} \setminus U_{2s}$, β be the integer in $U_{1s} \cap U_{2s}$, and γ be the integer in $U_{2s} \setminus U_{1s}$ (note that all the three sets contain exactly one integer since $|u_{1s} - u_{2s}| = 1$). We say that a column $j \in [n]$ *fits* boundary column s if the following three conditions hold:

(i) $w_x(j) = 0$ for any $x \in \Sigma \setminus \{\alpha, \beta, \gamma\}$,

(ii) $w_\alpha(j) \leq h_1$, and

(iii) $w_\gamma(j) \leq h_2 := m - h_1$.

Note that if Condition (i) is violated, then the column contains an element which is neither in U_{1s} nor in U_{2s}. If Condition (ii) (respectively (iii)) is violated, then there are more than h_1 (respectively h_2) rows that have to be in row block I_1 (respectively I_2). Thus, if j does not fit any boundary column s, then there is no $(2,\ell)$-co-clustering $(\{I_1, I_2\}, \{J_1, \ldots, J_\ell\})$ satisfying $\mathcal{U}$ with $|I_1| = h_1$ and $j \in J_s$. Hence, in order to find a solution, we need to find out, for each column in $\mathcal{A}$, to which fitting boundary column in $\mathcal{U}$ it should be assigned.

Intuitively, we now prove that in most cases a column has at most two fitting boundary columns, and, in the remaining cases, at most two pairs of "equivalent" boundary columns.

Consider a given column $j \in [n]$. Let $a := \min\{a_{ij} \mid i \in [m]\}$ and $b := \max\{a_{ij} \mid i \in [m]\}$ (computable in $O(m)$ time). If $b \geq a + 3$, then Condition (i) is always violated, that is, column j does not fit any boundary column, and the instance is a no-instance.

If $b = a + 2$, then, again by Condition (i), column j can only fit a boundary column s where $\{u_{1s}, u_{2s}\} = \{a, a + 1\}$. There are at most two such boundary columns in $\mathcal{U}$ since all columns in $\mathcal{U}$ are distinct. Let s_j and t_j denote their indices (computable in $O(\ell)$ time).

Case $b = a$ is also easy: all values in column j are equal to a. If j fits boundary column s, then, with Conditions (ii) and (iii), $a \in U_{1s} \cap U_{2s}$, and s is one of at most two columns in $\mathcal{U}$ for which $\{u_{1s}, u_{2s}\} = \{a - 1, a\}$. Again, let s_j and t_j denote their indices (computable in $O(\ell)$ time).

Finally, if $b = a + 1$, then let $s \in [\ell]$ be such that j fits boundary column s. Hence, by Condition (i), it holds $(u_{1s}, u_{2s}) \in \{(a - 1, a), (a, a - 1), (a, b), (b, a)\}$. Let $s_1, \ldots, s_4$ be the four column indices (computable in $O(\ell)$ time) in $\mathcal{U}$ (if they exist) corresponding to these four cases. We define $s_j := s_1$ if j fits boundary column s_1 (decidable in $O(m)$ time), and $s_j := s_3$ otherwise. Similarly, we define $t_j := s_2$ if j fits boundary column s_2, and $t_j := s_4$ otherwise ($O(m)$ time). Now, consider a $(2, \ell)$-co-clustering $(\{I_1, I_2\}, \{J_1, \ldots, J_\ell\})$ satisfying $\mathcal{U}$ with $|I_1| = h_1$ such that $j \in J_{s^*}$ for $s^* \in \{s_1, s_3\}$ with $s^* \neq s_j$. Since j must fit boundary column s^*, the only possibility is that $s^* = s_3$ and $s_j = s_1$. Thus, j fits both boundary columns s_1 and s_3, so Conditions (ii) and (iii) imply $w_a(j) \leq h_1$ and $w_b(j) \leq h_2$. Since $w_a(j) + w_b(j) = m = h_1 + h_2$, we have $w_a(j) = h_1$ and $w_b(j) = h_2$. Thus, placing j in either of the two column blocks J_{s_1}, J_{s_3} yields the same row partition, namely $I_1 = \{i \mid a_{ij} = a\}$ and $I_2 = \{i \mid a_{ij} = b\}$. Hence, column j can also be added to J_{s_1} instead of J_{s_3}.

Similarly with s_2 and s_4, any solution with $j \in J_{s_2}$ or $j \in J_{s_4}$ implies that column j can also be added to J_{t_j} (without any other modification). Thus, since column j has to be contained in one of the blocks $J_{s_1}, \ldots, J_{s_4}$, we can assume that it is contained in one of J_{s_j}, J_{t_j} instead.

As regards the running time, for each of the n columns we can find the two indices within at most $O(m + \ell + m)$ time. Hence, the overall running time is in $O(n(m + \ell))$. □

Having all prerequisites at hand, we now present the proof of Theorem 5.11.

Proof of Theorem 5.11. Let $I = (\mathcal{A} \in \mathbb{Z}^{m \times n}, k = 2, \ell, c = 1)$ be a $(2, *)$-Co-Clustering$_\infty$ instance. The proof is by induction on ℓ. For $\ell = 1$, the problem

is solvable in $O(n(m + \log n))$ time (Theorem 5.8). We now consider general values of ℓ. Note that if ℓ is large compared to m (that is, $2^m < |\Sigma|^\ell$), then one can directly guess the row partition and run the algorithm of Lemma 5.12. Thus, we henceforth assume that $\ell < m$.

For a cluster boundary $\mathcal{U} = (u_{rs}) \in \Sigma^{2\times\ell}$, let $U_{rs} := \{u_{rs}, u_{rs} + 1\}$ for each $(r, s) \in [2] \times [\ell]$. We say that the boundary column $s \in [\ell]$ has

- *equal* bounds if $U_{1s} = U_{2s}$ (that is, $u_{1s} = u_{2s}$),
- *non-overlapping* bounds if $U_{1s} \cap U_{2s} = \emptyset$ (that is $|u_{1s} - u_{2s}| > 1$),
- *properly overlapping* bounds otherwise (that is $|u_{1s} - u_{2s}| = 1$).

We first show that instances having a $(2, \ell)$-co-clustering satisfying a cluster boundary that contains at least one column with equal or non-overlapping bounds can be easily solved.

Claim 5.14. *If there exists a $(2, \ell)$-co-clustering that satisfies a cluster boundary with equal bounds, then it can be computed in $O(|\Sigma|^{2\ell}n^2m^2)$ time.*

Proof. We assume without loss of generality that boundary column ℓ has equal bounds. We try out all possible values $u_{1\ell} = u_{2\ell} = u \in \Sigma$. Note that column block J_ℓ imposes no restrictions on the row partition. Hence, any column of $\mathcal{A}$ with all values in $\{u, u + 1\}$ can be put into block J_ℓ, and all other columns have to end up in the $\ell - 1$ other blocks, thus forming an instance of $(2, \ell - 1)$-Co-Clustering$_\infty$. By induction (the case $\ell = 1$ is easily solvable by Theorem 5.8), each of these cases can be solved in $O(|\Sigma|^{2(\ell-1)}n^2m^2)$ time. This procedure finds a $(2, \ell)$-co-clustering with a column block having equal bounds in $O(|\Sigma| \cdot (mn + |\Sigma|^{2(\ell-1)}n^2m^2)) \subseteq O(|\Sigma|^{2\ell}n^2m^2)$ time. □

Claim 5.15. *If there exists a $(2, \ell)$-co-clustering that satisfies a cluster boundary with non-overlapping bounds, then it can be computed in $O(|\Sigma|^{2\ell}n^2m^2)$ time.*

Proof. Let s be the index of the boundary column with non-overlapping bounds and assume that, without loss of generality, $u_{1s}+1 < u_{2s}$. We try out all possible values $u_{2s} = u \in \Sigma$, and we check for each column j in $\mathcal{A}$, whether $j \in J_s$ is possible. In order to have $j \in J_s$, it is necessary that all entries in j are contained in $\{u_{1s}, u_{1s} + 1, u, u + 1\}$ for some $u_{1s} < u - 1$. Note that if column j fulfills the above condition, then assuming that $j \in J_s$ determines the boundary column s and also the entire row partition, that is, if $j \in J_s$, then $I_1 = \{i \mid a_{ij} < u\}$ and $I_2 = \{i \mid a_{ij} \geq u\}$. Using the algorithm described in Lemma 5.12, we can deduce

the column partition in $O(|\Sigma|^{2(\ell-1)}nm\ell)$ time (note that there are only $|\Sigma|^{2(\ell-1)}$ possible cluster boundaries to enumerate since column s is already fixed). The overall running time is thus in $O(|\Sigma|\cdot n\cdot(m+|\Sigma|^{2(\ell-1)}nm\ell)) \subseteq O(|\Sigma|^{2\ell}n^2m^2)$. □

We now show how to find a $(2,\ell)$-co-clustering that satisfies a cluster boundary with only properly overlapping bounds. We enumerate all such cluster boundaries $\mathcal{U} = (u_{rs})$. Note that, for each $s \in [\ell]$, we only need to consider the cases where $|u_{1s} - u_{2s}| = 1$. Also note that we can assume all columns in $\mathcal{U}$ to be distinct since two identical columns could be merged. We then enumerate all possible values $h_1 \in \{1,\dots,m-1\}$ (the *height* of the first row block) and define $h_2 := m - h_1 > 0$. Overall, there are at most $(2|\Sigma|)^\ell m$ cases to consider.

Using Lemma 5.13, we compute in $O(mn)$ time two indices s_j, t_j for each column j in $\mathcal{A}$ such that for any $(2,\ell)$-co-clustering $(\{I_1, I_2\}, \{J_1,\dots,J_\ell\})$ satisfying cluster boundary $\mathcal{U}$ with $|I_1| = h_1$, either $j \in J_{s_j}$ or $j \in J_{t_j}$ holds.

We now introduce a 2-CNF formula allowing us to simultaneously assign the rows and columns to the possible blocks (similar to the formula created in Lemma 5.7). Let $\phi'_{\mathcal{A},\mathcal{U},h_1}$ be a Boolean formula over the same Boolean variables as in Lemma 5.7 containing the following clauses:

- For each row $i \in [m]$, there is a clause $R'_i := (x_{i,1} \vee x_{i,2})$.
- For each column $j \in [n]$, there is a clause $C'_j := (y_{j,s_j} \vee y_{j,t_j})$.
- For each $(i,j) \in [m] \times [n]$ and each $(r,s) \in [2] \times \{s_j, t_j\}$ such that $a_{ij} \notin [u_{rs}, u_{rs}+1]$, there is a clause $B'_{ijrs} := (\neg x_{i,r} \vee \neg y_{j,s})$.

Note that $\phi'_{\mathcal{A},\mathcal{U},h_1}$ is a 2-CNF formula with $2m+2n$ variables and $O(mn)$ clauses. Clearly, if $\phi'_{\mathcal{A},\mathcal{U},h_1}$ is satisfiable, then $\mathcal{A}$ admits a $(2,\ell)$-co-clustering satisfying $\mathcal{U}$. Conversely, if $\mathcal{A}$ admits a $(2,\ell)$-co-clustering satisfying $\mathcal{U}$ with $|I_1| = h_1$, then, by the discussion above, there exists a $(2,\ell)$-co-clustering where each column j is in one of the column blocks J_{s_j} or J_{t_j}. For the corresponding Boolean variable assignment, each clause of $\phi'_{\mathcal{A},\mathcal{U},h_1}$ is satisfied. Hence, $\phi'_{\mathcal{A},\mathcal{U},h_1}$ is satisfiable.

Overall, for each cluster boundary $\mathcal{U}$ and each h_1, we construct and solve the formula $\phi'_{\mathcal{A},\mathcal{U},h_1}$ defined above. The matrix $\mathcal{A}$ admits a $(2,\ell)$-co-clustering of cost 1 if and only if $\phi'_{\mathcal{A},\mathcal{U},h_1}$ is satisfiable for some $\mathcal{U}$ and h_1. The running time for constructing and solving the formula $\phi'_{\mathcal{A},\mathcal{U},h_1}$, for fixed $\mathcal{U}$ and h_1 is in $O(mn)$, which yields a running time in $O((2|\Sigma|)^\ell m \cdot (mn+mn)) \subseteq O((2|\Sigma|)^\ell nm^2)$ for this last part.

The final running time is thus in $O(2|\Sigma|^{2\ell}n^2m^2 + (2|\Sigma|)^\ell nm^2) \subseteq O(|\Sigma|^{2\ell}n^2m^2)$. □

Finally, by Theorem 5.11, we obtain the following simple corollary.

Corollary 5.16. $(2,*)$-Co-Clustering$_\infty$ *with* $c = 1$ *is fixed-parameter tractable with respect to the alphabet size* $|\Sigma|$ *and with respect to the number* ℓ *of column blocks.*

Proof. Theorem 5.11 presents an FPT-algorithm with respect to the combined parameter $(\ell, |\Sigma|)$. For $(2,*)$-Co-Clustering$_\infty$ with $c = 1$, both parameters can be polynomially upper-bounded by each other. Indeed, $\ell \leq |\Sigma|^2$ (otherwise there are two column blocks with identical cluster boundaries, which could be merged) and $|\Sigma| \leq 2(c+1)\ell = 4\ell$ (all columns within a column block can only contain values from two intervals, each covering at most $c+1$ elements). Hence, the algorithm from Theorem 5.11 yields fixed-parameter tractability for each of the two parameters alone. □

5.5 Conclusion

In this chapter, we started to closely investigate the multivariate time complexity of exactly solving the NP-hard Co-Clustering$_\infty$ problem, contributing a detailed view on the computational complexity landscape of a prominent data mining problem. We developed a reduction that allows to solve the problem using SAT-solvers. In some first experiments, however, we observed that this approach is not efficient enough for practical purposes yet. The bottleneck is the number of possible cluster boundaries, which grows extremely fast. While a single CNF-SAT instance can be solved quickly, generating all possible cluster boundaries together with the corresponding CNF formulas becomes quite expensive, such that we could only solve instances with very small values of $|\Sigma| \leq 4$ and $k \leq \ell \leq 5$ [Bul+16a]. Thus, further improvement is required to efficiently compute exact solutions in practice. For example, parallel computing could bring a significant speed-up since the SAT instances for different cluster boundaries can be solved independently. It might also be worth to consider integer linear programming in the context of co-clustering.

Several open questions derive from our results. Perhaps the most pressing open question is whether the case $k = 2$ and $\ell \geq 3$ is polynomial-time solvable or NP-hard in general. So far, we only know that $(2,*)$-Co-Clustering$_\infty$ is polynomial-time solvable for ternary matrices (Theorem 5.10). Another open question is whether Optimal Discretization [CN98] (that is, Consecutive Co-Clustering$_\infty$ with $\Sigma = \{0,1,2\}$ and $c = 1$) is fixed-parameter

tractable with respect to the combined parameter (k, ℓ). Note that this restricted problem is a special case of the Rectangle Stabbing problem which is known to be W[1]-hard with respect to the number of stabbing lines (that is, clusters) [DFR12]. Optimal Discretization is also known under the name Red-Blue Separation and was recently shown to be W[1]-hard when parameterized by the number of separating lines if the lines can be chosen arbitrarily (non-axis parallel) and conjectured to be fixed-parameter tractable for axis-parallel lines [BGL17]. Last but not least, the computational complexity of higher-dimensional co-clustering versions, e.g. on three-dimensional tensors as input (the most basic case here corresponds to (2,2,2)-Co-Clustering$_\infty$, that is, partitioning each dimension into two subsets) is also open.

We conclude with the following more abstract vision for future research: Note that for the maximum norm, the cost value c defines a "conflict relation" on the values occurring in the input matrix. That is, for any two numbers $\sigma, \sigma' \in \Sigma$ with $|\sigma - \sigma'| > c$, we know that they must end up in different clusters. These conflict pairs completely determine all constraints of a solution since all other pairs can be grouped arbitrarily. This observation can be generalized to a graph model. Given a "conflict relation" $R \subseteq \binom{\Sigma}{2}$ determining which pairs are not allowed to be put together into a cluster, we can define a "conflict graph" (Σ, R). Studying co-clusterings in the context of such conflict graphs and their structural properties could be a promising and fruitful direction for future research.

Chapter 6

F-Free Editing

In this chapter, we study the parameterized complexity of a general graph modification problem called F-free Editing: Given a graph G and a natural number k, is it possible to modify at most k edges in G so that the resulting graph contains no induced subgraph isomorphic to F? This problem comprises some well-known special cases with applications in machine learning and network analysis. For example, the case when F is a path on three vertices is the graph clustering problem Cluster Editing.

In contrast to the previous chapters, where we analyzed problems whose complexity has not been studied in detail before, this chapter considers a problem whose parameterized complexity has already been studied. However, earlier works on F-free Editing mostly considered the standard parameter, that is, the number k of edge modifications. We consider instead a parameterization by the number ℓ of edge modifications above a given lower bound (provided by a packing of induced subgraphs of the input graph), which is a stronger (that is, smaller) parameter. Hence, this parameter allows a sharpening of the complexity picture by identifying further tractable cases.

We develop a framework to show fixed-parameter tractability for F-free Editing with respect to ℓ by creating a *win-win* situation where we can either apply a polynomial-time data reduction or upper-bound the number k of edge modifications in ℓ. We show how to apply the framework for three well-known problems: Triangle Deletion, Feedback Arc Set in Tournaments, and Cluster Editing. Moreover, we explore the limits of this parameterization approach by proving NP-hardness for several problem variants (for example, for edge-disjoint subgraph packings or vertex deletions) with constant parameter value $\ell = 0$.

6.1 Introduction

Graph modification problems are a core topic of algorithmic research [Cai96, LY80, Yan81]. Given a graph G, the aim in these problems is to transform G by a minimum number of modifications (like vertex deletions, edge deletions, or edge insertions) into another graph G' fulfilling certain properties. Particularly well-studied are *hereditary* graph properties, which are closed under vertex deletions and which are characterized by *minimal forbidden induced subgraphs*: a graph fulfills a hereditary property Π if and only if it does not contain a graph F from a property-specific family $\mathcal{F}$ of graphs as an induced subgraph. All nontrivial vertex deletion problems and many edge modification and deletion problems for establishing hereditary graph properties are NP-complete [Alo06, ASS16, KM86, LY80, SST04, Yan81]. If the desired graph property has a finite forbidden induced subgraph characterization, then the corresponding vertex deletion, edge deletion, and edge modification problems are known to be fixed-parameter tractable with respect to the number k of modifications [Cai96]. All vertex deletion problems for establishing graph properties characterized by a finite number of forbidden induced subgraphs have a problem kernel of size polynomial in the number k of allowed vertex deletions [Kra12]. In contrast, many variants of F-free Editing do not admit a problem kernel whose size is polynomial in k [CC15, Gui+13, KW13].

Parameterization Above Lower Bounds. When combined with data reduction and pruning rules, search-tree based fixed-parameter algorithms for the parameter k of allowed modifications are competitive in some cases [HH15, MNS12]. Nevertheless, the number k of modifications is often too large and smaller parameters are desirable.

A natural approach to obtain smaller parameters is "parameterization above guaranteed values" [Cyg+13, GP16, Lok+14, MR99]. The idea is to use a lower bound h on the solution size and to use $\ell := k - h$ as parameter instead of k. This idea has been applied successfully to Vertex Cover, that is, K_2-Free Vertex Deletion. Since the size of a smallest vertex cover is large in many input graphs, parameterizations above the lower bounds "size of a maximum matching M in the input graph" and "optimum value L of the LP relaxation of the standard ILP-formulation of Vertex Cover" have been considered. After a series of improvements [Cyg+13, GP16, Lok+14, RO09], the currently best running time is $3^\ell \cdot n^{O(1)}$, where $\ell := k - (2 \cdot L - |M|)$ and $|M|$ and L denote the lower bounds mentioned above [GP16].

We extend this approach to edge modification problems, where the number k of modifications tends to be even larger than for vertex deletion problems. For example, in the case of Cluster Editing, which asks to destroy induced paths on three vertices by edge modifications, the number of modifications is often larger than the number of vertices in the input graph [Böc+09]. Hence, parameterization above lower bounds seems natural and even more relevant for edge modification problems. Somewhat surprisingly, this approach has not been considered so far. We thus initiate research on parameterization above lower bounds in this context. As a starting point, we focus on edge modification problems for graph properties that are characterized by *one* forbidden induced subgraph F:

F-free Editing
Input: A graph $G = (V, E)$ and a natural number k.
Question: Is there an *F-free editing set* $S \subseteq \binom{V}{2}$ of size at most k, that is, an edge modification set such that $G \triangle S := (V, (E \setminus S) \cup (S \setminus E))$ does not contain F as induced subgraph?

In the context of a concrete variant of F-free Editing, we refer to an F-free editing set as *solution* and call a solution *optimal* if it has minimum size.

Lower Bounds from Packings of Induced Subgraphs. Following the approach of parameterizing Vertex Cover above the size of a maximum matching, we can parameterize F-free Editing above a lower bound obtained from packings of arbitrary induced non-F-free subgraphs.

Definition 6.1. A *vertex-disjoint (or edge-disjoint) packing* of induced subgraphs of a graph G is a set $\mathcal{H} = \{H_1, \ldots, H_z\}$ such that each H_i is an induced subgraph of G and such that the vertex sets of the H_i are mutually disjoint (or intersect in at most one vertex).

Though Definition 6.1 does not require $\mathcal{H}$ to be maximal (we still call it a packing), it is of course natural in practice to use a maximal packing of vertex-disjoint subgraphs (for example, computed by some greedy strategy). Also, while it is at first sight natural to consider packings of induced subgraphs that are isomorphic to F in order to obtain a lower bound on the solution size, a packing of other graphs that contain F as induced subgraph might yield even better lower bounds and thus a smaller parameter above this lower bound. For example, a K_4 contains four K_3s and two edge deletions are necessary to

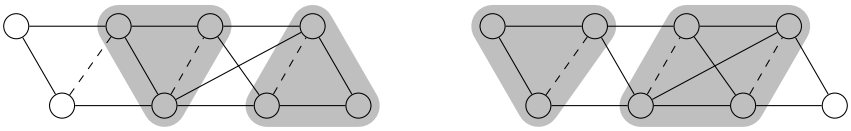


Figure 6.1: Two instances of K_3-Free Editing with Cost-2 Packing where the maximum number of edge modifications is $k = 3$. The packing graphs have gray background. Deleting the three dashed edges is a solution. Left: A vertex-disjoint packing of two K_3s giving excess $\ell = 1$. Right: A vertex-disjoint packing of a K_3 and a K_4 giving $\ell = 0$.

make it K_3-free. Thus, if a graph G has a vertex-disjoint packing of h_3 K_3s and h_4 K_4s, then at least $h_3 + 2 \cdot h_4$ edge deletions are necessary to make it triangle-free. Moreover, when allowing arbitrary graphs for the packing, the lower bounds provided by vertex-disjoint packings can be better than the lower bounds provided by edge-disjoint packings of F. A disjoint union of h K_4s, for example, has h edge-disjoint K_3s but also h vertex-disjoint K_4s. Hence, the lower bound provided by packing vertex-disjoint K_4s is twice as large as the one provided by packing edge-disjoint K_3s in this graph.

Motivated by this benefit of vertex-disjoint packings of arbitrary graphs, we mainly consider lower bounds obtained from vertex-disjoint packings, which we assume to receive as input. Thus, we arrive at the following problem, where $\tau(G)$ denotes the minimum size of an F-free editing set for a graph G:

F-free Editing with Cost-t Packing

Input: A graph $G = (V, E)$, a vertex-disjoint packing $\mathcal{H}$ of induced subgraphs of G such that $1 \leq \tau(H) \leq t$ for each $H \in \mathcal{H}$, and a natural number k.

Question: Is there an *F-free editing set* $S \subseteq \binom{V}{2}$ of size at most k such that $G \triangle S := (V, (E \setminus S) \cup (S \setminus E))$ does not contain F as induced subgraph?

The special case of F-free Editing with Cost-t Packing where only subgraphs isomorphic to F are allowed in the packing is called F-free Editing with F-Packing.

From the packing $\mathcal{H}$, we obtain the lower bound $h(\mathcal{H}) := \sum_{H \in \mathcal{H}} \tau(H)$ on the size of an F-free editing set, which allows us to use the excess $\ell := k - h(\mathcal{H})$ over this lower bound as parameter, as illustrated in Figure 6.1. Since F is a fixed graph, we can compute the bound $h(\mathcal{H})$ in $|\mathcal{H}| \cdot f(t) \cdot |G|^{O(1)}$ time using

the generic algorithm [Cai96] mentioned in the introduction for each $H \in \mathcal{H}$. In the same time we can also verify whether the cost-t property is fulfilled.

Packings of forbidden induced subgraphs have been used in implementations of fixed-parameter algorithms to prune the corresponding search trees tremendously [HH15]. By showing fixed-parameter algorithms for parameters above these lower bounds, we hope to explain the fact that these packings help in obtaining fast algorithms.

Our Contributions. We investigate three variants of F-free Editing with Cost-t Packing for small graphs F, that is, on three vertices (the smallest nontrivial number of vertices).

In the first (introductory) variant, the graph F is a *triangle*, that is, a K_3 (complete graph on three vertices). This problem is known as Triangle Deletion (note that adding edges is never optimal since it cannot remove triangles) and is related to network cluster analysis.

Second, we consider a directed variant in which the input is a tournament graph and F is a directed cycle on three vertices. This is known as Feedback Arc Set in Tournaments and has applications in rank aggregation.

The third variant is a clustering problem on undirected graphs called Cluster Editing, where F is a P_3, that is, a path on three vertices. This problem has various applications in machine learning and bioinformatics.

Our positive results are fixed-parameter algorithms and problem kernels. Using a general approach described in Section 6.2, we obtain fixed-parameter algorithms for the above variants of F-free Editing with Cost-t Packing parameterized by t and ℓ (mainly by developing appropriate data reduction rules). This implies fixed-parameter tractability for F-free Editing with F-Packing parameterized by ℓ. Specifically, we obtain the following results (for the kernelization results, we need to assume that $t \in O(\log n)$ to guarantee polynomial running time of the data reduction):

- For Triangle Deletion, we show an $O((2t+3)^{\ell} \cdot (nm + n \cdot 2.076^t))$-time algorithm and an $O(t \cdot \ell)$-vertex problem kernel for cost-t packings (Corollary 6.6).

- For Feedback Arc Set in Tournaments, we show a $2^{O\left(\sqrt{(2t+1)\ell}\right)} \cdot n^{O(1)}$-time algorithm and an $O(t \cdot \ell)$-vertex problem kernel for cost-t packings (Corollary 6.11).

- For Cluster Editing, we show an $O(1.62^{(2t+1)\ell} + nm + n \cdot 1.62^t)$-time algorithm and an $O(t \cdot \ell)$-vertex kernel for cost-t packings (Corollary 6.16), and also give a $4^\ell \cdot n^{O(1)}$-time algorithm for P_3-packings [BFK18, Theorem 6.13].

To the best of our knowledge, the above results are the first fixed-parameter tractability results for edge modification problems parameterized above lower bounds obtained from induced subgraph packings. Moreover, since the parameter ℓ is potentially significantly smaller than the number k of edge modifications, our algorithms might be fast in practice.

On the negative side, we show several NP-hardness results which point out the limitations of parameterizing graph modification problems above lower bounds obtained from subgraph packings and justify to a certain extent the focus on lower bounds obtained from vertex-disjoint packings. In particular, we show the following:

- K_6-free Deletion with K_6-Packing is NP-hard for $\ell = 0$ (Theorem 6.18). This implies that a general fixed-parameter tractability result as it is known for the parameter k [Cai96] cannot be expected.

- If F is a K_3 and $\mathcal{H}$ is an *edge-disjoint* packing of h K_3s in a graph G, then it is NP-hard to decide whether G has a K_3-free deletion set of size h, that is, $\ell = 0$ (Theorem 6.21). Thus, parameterization by ℓ is hopeless for edge-disjoint packing lower bounds.

- For all $q \geq 3$, P_q-free Vertex Deletion with P_q-Packing is NP-hard even if $\ell = 0$ (Theorem 6.23).

Organization. In Section 6.2, we present the general approach used in our algorithmic results. In Section 6.3, we present the results for Triangle Deletion, in Section 6.4 those for Feedback Arc Set in Tournaments, and in Section 6.5 the results for Cluster Editing. Section 6.6 shows edge and vertex deletion problems that remain NP-hard for $\ell = 0$, where ℓ is the number of modifications that are allowed in addition to a lower bound based on different types (edge- and vertex-disjoint) of subgraph packings. We conclude with some open questions in Section 6.7.

6.2 General Approach

In this section, we describe the general approach of our fixed-parameter algorithms. Recall that $\tau(H)$ is the minimum number of edge modifications required to transform a graph H into an F-free graph. The idea behind the algorithms is to arrive at a classic win-win scenario [Fel03, FNN16, Har+15] where we can either apply polynomial-time data reduction or show that the packing size $|\mathcal{H}|$ is upper-bounded in ℓ. This upper bound will allow us to upper-bound k in $t \cdot \ell$ for yes-instances and, thus, to apply known fixed-parameter algorithms for the parameter k to obtain fixed-parameter tractability results for (t, ℓ).

More precisely, we show that, for each induced subgraph H of G in a given packing $\mathcal{H}$, we face essentially two situations. If there is an optimal solution for H that is a subset of an optimal solution for G, then we can apply a data reduction rule. Otherwise, we find a "certificate", that is, a set of vertex pairs witnessing that either H itself needs to be solved suboptimally or that a vertex pair containing exactly one vertex from H needs to be modified. We use the following terminology for these pairs.

Definition 6.2 (External vertex pairs and edges)**.** A vertex pair $\{u, v\}$ is an *external pair for a packing graph* $H \in \mathcal{H}$ if exactly one of u or v is in $V(H)$. An edge in G is an *external edge for* $H \in \mathcal{H}$ if exactly one of its endpoints is in H.

Observe that every vertex pair (every edge) is an external pair (edge) for at most two packing graphs since the packing graphs are vertex-disjoint. Consequently, the modification of an external vertex pair can destroy at most two certificates. This is the main fact used in the proof of the following bound on k.

Lemma 6.1. *Let $(G, \mathcal{H}, k)$ be an instance of* F-free Editing with Cost-t Packing *and let S be a size-k solution that contains, for each $H \in \mathcal{H}$, either*

(a) at least $\tau(H) + 1$ vertex pairs from $\binom{V(H)}{2}$, or

(b) at least one external vertex pair $\{v, w\}$ for H.

Then, $|\mathcal{H}| \leq 2\ell$ and thus, $k \leq (2t + 1)\ell$.

Proof. Denote by $\mathcal{H}_a \subseteq \mathcal{H}$ the set of all graphs in $\mathcal{H}$ that fulfill Property (a) and let $p_a := |\mathcal{H}_a|$. Let $\mathcal{H}_b := \mathcal{H} \setminus \mathcal{H}_a$ denote the set containing the remaining packing graphs (fulfilling Property (b)) and let $p_b := |\mathcal{H}_b|$. Thus, $|\mathcal{H}| = p_a + p_b$. Furthermore, let $h_a := \sum_{H \in \mathcal{H}_a} \tau(H)$ denote the lower bound obtained from

the graphs in $\mathcal{H}_a$ and let $h_b := h(\mathcal{H}) - h_a$ denote the part of the lower bound obtained by the remaining graphs.

The packing graphs in $\mathcal{H}_a$ require at least $h_a + p_a$ edge modifications inside of them. Similarly, the packing graphs in $\mathcal{H}_b$ cause at least h_b edge modifications inside of them, and each packing graph $H \in \mathcal{H}_b$ additionally causes a modification of at least one external vertex pair for H. Since every vertex pair is an external pair for at most two different packing graphs, at least $h_b + p_b/2$ edge modifications are caused by the graphs in $\mathcal{H}_b$. This implies that

$$\begin{aligned} & k \geq h_a + h_b + p_a + p_b/2 \\ \Leftrightarrow \quad & k - h(\mathcal{H}) \geq p_a + p_b/2 \\ \Leftrightarrow \quad & 2\ell \geq 2p_a + p_b \geq |\mathcal{H}|. \end{aligned}$$

Consequently, $k = \ell + h(\mathcal{H}) \leq \ell + t \cdot |\mathcal{H}| \leq \ell + t \cdot 2\ell = (2t+1)\ell$. □

Using Lemma 6.1, we can derive fixed-parameter tractability of F-free Editing with Cost-t Packing with respect to (ℓ, t) from the fixed-parameter tractability with respect to k. Hence, for a specific problem at hand, we have to find suitable data reduction rules that shrink a given instance in such a way that Lemma 6.1 becomes applicable. We give examples of how to apply Lemma 6.1 in the following three sections.

6.3 Triangle Deletion

In this section, we study Triangle Deletion, that is, the problem of destroying all *triangles* (K_3s) in a graph by at most k edge deletions. Note that this problem equals K_3-free Editing since adding edges can never help to destroy an induced K_3.

The number of triangles in a graph is an important graph statistic with many applications, for example in complex network analysis, spam detection, or bioinformatics (we refer to the work by Tsourakakis et al. [TKM11] for a detailed discussion). For example, in network analysis the *clustering coefficient* (or *transitivity*) is based on the number of triangles and measures how strong a network "clusters together" [New03]. In this context, Triangle Deletion can be motivated by measuring how robust a network is clustered together, that is, how many edges have to be deleted in order to obtain a network with a clustering coefficient of zero. We apply our framework from Section 6.2 to show

that Triangle Deletion is fixed-parameter tractable parameterized above the lower bound given by a cost-t packing.

Before presenting our fixed-parameter tractability results for Triangle Deletion, let us first summarize the known results concerning the (parameterized) complexity of Triangle Deletion. Triangle Deletion is NP-complete [Yan81]. It allows for a trivial reduction to 3-Hitting Set since edge deletions do not create new triangles [Gra+04]. Combining this approach with the currently fastest known algorithms for 3-Hitting Set [Wah07] gives an algorithm for Triangle Deletion with running time $O(2.076^k + nm)$. Finally, Triangle Deletion admits a problem kernel with at most $6k$ vertices [BKM09].

We show that Triangle Deletion with Cost-t Packing is fixed-parameter tractable with respect to the combination of t and $\ell := k - h(\mathcal{H})$. More precisely, we obtain a kernelization and a search tree algorithm. Both make crucial use of the following generic data reduction rule for Triangle Deletion with Cost-t Packing, which is designed in such a way that we can apply Lemma 6.1 to instances where the rule does not apply. The intuition behind the rule is that if a packing subgraph has a solution (that is, an edge deletion set) which is also "locally optimal" for the whole graph G, then we can delete those edges since they are always contained in a global solution for G.

Reduction Rule 6.1. *If there is an induced subgraph $H \in \mathcal{H}$ and a set $T \subseteq E(H)$ of $\tau(H)$ edges such that deleting T destroys all triangles of G that contain edges of H, then delete T from G, H from $\mathcal{H}$, and decrease k by $\tau(H)$.*

Lemma 6.2. *Reduction Rule 6.1 is correct.*

Proof. Let $(G, \mathcal{H}, k)$ be the instance to which Reduction Rule 6.1 is applied and let $(G', \mathcal{H} \setminus \{H\}, k - \tau(H))$ with $G' := G - T$ be the result. We show that $(G, \mathcal{H}, k)$ is a yes-instance if and only if $(G', \mathcal{H} \setminus \{H\}, k - \tau(H))$ is.

First, let $S \subseteq E(G)$ be a solution of size at most k for $(G, \mathcal{H}, k)$. Let $S_H := S \cap E(H)$ denote the subset of edge deletions that destroy all triangles in H. By definition, $|S_H| \geq \tau(H)$. Since $S_H \subseteq E(H)$, only triangles containing at least one edge of H are destroyed by deleting S_H. It follows that the set of triangles destroyed by S_H is a subset of the triangles destroyed by T. Hence, $(S \setminus S_H) \cup T$ has size at most k and clearly is a solution for $(G, \mathcal{H}, k)$. Thus, $(S \setminus S_H)$ is a solution of size at most $k - \tau(H)$ for G' and $(G', \mathcal{H} \setminus \{H\}, k - \tau(H))$ is a yes-instance.

For the converse direction, let S' be a solution of size at most $k - \tau(H)$ for $(G', \mathcal{H} \setminus \{H\}, k - \tau(H))$. Since $T \subseteq E(H)$, it holds that every triangle

contained in G that does not contain any edge of H is also a triangle in G'. Thus, S' is a set of edges whose deletion in G destroys all triangles that do not contain any edge of H. Since T destroys all triangles containing an edge of H, we have that $T \cup S'$ is a solution for G of size at most k. □

We now show that if Reduction Rule 6.1 is not applicable to H, then we can find a *certificate* for this inapplicability, which will allow us later to branch efficiently on the destruction of triangles:

Definition 6.3 (Certificate)**.** A *certificate* for inapplicability of Reduction Rule 6.1 to an induced subgraph $H \in \mathcal{H}$ is a set $\mathcal{T}$ of triangles in G, each containing exactly one distinct edge of H, such that either

(a) $|\mathcal{T}| = \tau(H) + 1$ or

(b) $|\mathcal{T}| \leq \tau(H)$ and $\tau(H') > \tau(H) - |\mathcal{T}|$, where H' is the subgraph obtained from H by deleting, for each triangle in $\mathcal{T}$, its edge shared with H.

Clearly, if we find a set $\mathcal{T}$ of triangles as defined in Definition 6.3, then Reduction Rule 6.1 is not applicable to H since either deleting $\tau(H)$ edges in H leaves a triangle in G with an edge in H (a), or deleting all triangles $\mathcal{T}$ and all triangles in H requires more than $\tau(H)$ edge deletions in H (b).

In the following, let $\Gamma(G, k)$ be the fastest running time to compute a triangle-free deletion set of size at most k in a graph G if it exists. We assume that Γ is monotonically nondecreasing in the size of G and in k. As mentioned above, currently $O(2.076^k + |V(G)| \cdot |E(G)|)$ is the best known upper bound for $\Gamma(G, k)$.

Lemma 6.3. *We can check for each $H \in \mathcal{H}$ whether Reduction Rule 6.1 applies and output a certificate $\mathcal{T}_H$ if it does not apply within an overall time of $O(nm + \sum_{H \in \mathcal{H}} \Gamma(H, t))$.*

Proof. First, in $O(nm)$ time, we compute for all $H \in \mathcal{H}$ a set $\mathcal{T}_H$ of triangles in G that contain exactly one edge $e \in E(H)$. These edges are labeled in each $H \in \mathcal{H}$. Then, for each $H \in \mathcal{H}$, in $\Gamma(H, t)$ time we determine the size $\tau(H)$ of an optimal triangle-free deletion set for H. Let t' denote the number of labeled edges in H.

If $t' > \tau(H)$, then we return as certificate $\tau(H) + 1$ triangles of $\mathcal{T}_H$, each containing one distinct edge of $\tau(H) + 1$ arbitrary labeled edges in H.

If $t' \leq \tau(H)$, then let H' denote the graph obtained from H by deleting the labeled edges. Each triangle in G that contains at least one edge of H

either contains a labeled edge of H or it is a subgraph of H'. Thus, we now determine in $\Gamma(H', \tau(H) - t')$ time whether H' can be made triangle-free by $\tau(H) - t'$ edge deletions. If this is the case, then Reduction Rule 6.1 applies and the set T consists of the solution for H' plus the t' deleted labeled edges. Otherwise, destroying all triangles in $\mathcal{T}_H$ leads to a solution for H which needs more than $\tau(H)$ edge deletions and thus Reduction Rule 6.1 does not apply. In this case, we return the certificate $\mathcal{T}_H$ for this $H \in \mathcal{H}$.

The overall running time now follows from the monotonicity of Γ and from the fact that one pass over $\mathcal{H}$ is sufficient since deleting edges in each H does not produce new triangles and does not destroy triangles in any $H' \in \mathcal{H}$ with $H' \neq H$. □

Observe that Reduction Rule 6.1 never increases the parameter ℓ since we decrease both k as well as the lower bound $h(\mathcal{H})$ by $\tau(H)$. After application of Reduction Rule 6.1, we can upper-bound the solution size k in terms of t and ℓ, which allows us to transfer parameterized complexity results for the parameter k to the combined parameter (t, ℓ).

Lemma 6.4. *Let $(G, \mathcal{H}, k)$ be a yes-instance of* Triangle Deletion with Cost-t Packing *such that Reduction Rule 6.1 is inapplicable. Then, $k \leq (2t + 1)\ell$.*

Proof. Since $(G, \mathcal{H}, k)$ is reduced with respect to Reduction Rule 6.1, for each graph $H \in \mathcal{H}$, there is a set of edges between $V(H)$ and $V(G) \setminus V(H)$ witnessing that every optimal solution for H does not destroy all triangles in G containing at least one edge from H. Consider an optimal solution S for G. For each graph $H \in \mathcal{H}$, there are two possibilities:

(a) at least $\tau(H) + 1$ edges inside H are deleted by S, or

(b) at least one external edge of H is deleted by S.

Hence, S fulfills the condition of Lemma 6.1 and thus $k \leq (2t + 1)\ell$. □

Using the upper bound on k, we prove the following fixed-parameter tractability results.

Theorem 6.5. *Let $\Gamma(G, k)$ be the fastest running time for computing a triangle-free deletion set of size at most k in a graph G if it exists. Then,* Triangle Deletion with Cost-t Packing

(i) can be solved in $O((2t+3)^\ell \cdot (nm + \sum_{H\in\mathcal{H}} \Gamma(H,t)))$ *time, and*

(ii) admits a problem kernel with at most $(12t+6)\ell$ *vertices that can be computed in* $O(nm + \sum_{H\in\mathcal{H}} \Gamma(H,t))$ *time.*

Proof. We first prove (ii). To this end, let $(G = (V,E), \mathcal{H}, k)$ be the input instance. First, compute in $O(nm + \sum_{H\in\mathcal{H}} \Gamma(H,t))$ time an instance that is reduced with respect to Reduction Rule 6.1 (Lemma 6.3). Afterwards, by Lemma 6.4, we can return a trivial constant-size no-instance if $k > (2t+1)\ell$. Otherwise, we apply the known kernelization algorithm for Triangle Deletion to the instance (G,k) (that is, without $\mathcal{H}$). This kernelization produces in $O(m\sqrt{m}) \subseteq O(nm)$ time a problem kernel (G',k') with at most $6k \leq (12t+6)\ell$ vertices and with $k' \leq k$ [BKM09]. Adding an empty packing gives an equivalent instance $(G',\emptyset,k')$ with parameter $\ell' = k' \leq (2t+1)\ell$ of Triangle Deletion with Cost-t Packing.

It remains to prove (i). To this end, we give a search tree algorithm. If $\ell < 0$, then we can clearly reject the instance. Otherwise, apply Reduction Rule 6.1 exhaustively in $O(nm + \sum_{H\in\mathcal{H}} \Gamma(H,t))$ time. Now, consider the following two cases for the reduced instance.

Case 1: $\mathcal{H} = \emptyset$. If G is triangle-free, then we are done. Otherwise, pick an arbitrary triangle in G and add it to $\mathcal{H}$.

Case 2: $\mathcal{H}$ contains a graph H. Since Reduction Rule 6.1 does not apply to H, there is a certificate $\mathcal{T}$ of $t' \leq \tau(H)+1$ triangles, each containing exactly one distinct edge of H such that deleting the edges of these triangles contained in H produces a subgraph H' of H that cannot be made triangle-free by $\tau(H)-t'$ edge deletions. Thus, branch into the following $(2t'+1) \leq (2t+3)$ cases: First, for each triangle $K \in \mathcal{T}$, create two cases, in each deleting a different one of the two edges of K that are not in H. In the remaining case, delete the t' edges of H and replace H by H' in $\mathcal{H}$ if $\tau(H') > 0$.

It remains to show the running time by bounding the search tree size. In Case 1, no branching is performed and the parameter ℓ is decreased by one. In Case 2, the parameter ℓ is decreased by one in each branch: in the first $2t'$ cases, an edge that is not contained in any packing graph is deleted (that is, k decreases by one while $h(\mathcal{H})$ remains unchanged). In the final case, the value of k decreases by t'. However, $\tau(H') \geq \tau(H) - t' + 1$. Hence, the lower bound $h(\mathcal{H})$ decreases by $\tau(H) - \tau(H') \leq t' - 1$ and thus the parameter ℓ decreases by at least one. Note that applying Reduction Rule 6.1 never increases the parameter. Hence, the depth of the search tree is at most ℓ. □

Corollary 6.6. Triangle Deletion with Cost-t Packing

(i) can be solved in $O((2t+3)^\ell \cdot (nm + n \cdot 2.076^t))$ *time, and*

(ii) admits a problem kernel with at most $(12t+6)\ell$ *vertices that can be computed in* $O(nm + n \cdot 2.076^t)$ *time.*

For the natural special case $t = 1$, that is, for vertex-disjoint triangle packings, Theorem 6.5(i) immediately yields the following running time.

Corollary 6.7. Triangle Deletion with Triangle Packing *is solvable in* $O(5^\ell nm)$ *time.*

Notably, in Section 6.6.2, we prove NP-hardness for $\ell = 0$ in case of edge-disjoint triangle packings which means that we cannot expect similar fixed-parameter tractability results if we pack triangles edge-disjointly (since this allows to obtain a tight lower bound as shown in the proof of Theorem 6.21). This can be seen as a theoretical justification to use vertex-disjoint packings as we did in this section.

6.4 Feedback Arc Set in Tournaments

In this section, we present a fixed-parameter algorithm and a problem kernel for Feedback Arc Set in Tournaments parameterized above lower bounds of cost-t packings.

In Feedback Arc Set in Tournaments, we are given a directed tournament graph G (every pair of vertices having exactly one arc between them) as input and want to delete a minimum number of arcs to make the graph acyclic, that is, to destroy all directed cycles in G. FAST is a well-studied problem with applications in rank aggregation tasks, for example, in machine learning and information retrieval [CSS99, Dwo+01].

It is a well-known observation that Feedback Arc Set in Tournaments can also be viewed as an arc *reversal* problem: After deleting a minimum set of arcs to make the graph acyclic, adding the arc (u, v) for every deleted arc (v, u) does not create any cycle (see Figure 6.2 for a small example). Since in tournaments, every pair of vertices is connected by exactly one arc, it follows that destroying cycles by arc deletions is equivalent to destroying them by arc reversals. Altogether, we arrive at the following problem definition.

Figure 6.2: Example of FAST. A tournament on four vertices containing several cycles (left). Deleting the dashed arc leaves an acyclic directed graph. Reversing the dashed arc also yields an acyclic tournament (right).

Feedback Arc Set in Tournaments (FAST)
Input: An n-vertex tournament $G = (V, A)$ and a natural number k.
Question: Does G have a *feedback arc set* $S \subseteq A$, that is, a set S such that reversing all arcs in S yields an acyclic tournament of size at most k?

FAST is known to be NP-complete [Alo06] and fixed-parameter tractable with respect to k [ALS09, Dom+10, Fei09, FP13, KS10, RS06]. The running time of the asymptotically best fixed-parameter algorithm currently is $2^{c \cdot \sqrt{k}} + n^{O(1)}$, where $c \leq 5.24$ [FP13]. Moreover, a problem kernel with $(2+\epsilon)k$ vertices for each *constant* $\epsilon > 0$ is known [Bes+11] as well as a simpler $4k$-vertex kernel [PPT16]. It is well-known that a tournament is acyclic if and only if it does not contain a *directed triangle* (a cycle on 3 vertices). Hence, the problem is to find a set of arcs whose reversal leaves no directed triangle in the tournament.

We show fixed-parameter tractability of Feedback Arc Set in Tournaments with Cost-t Packing parameterized by the combination of t and $\ell := k - h(\mathcal{H})$. Recall that $h(\mathcal{H}) := \sum_{H \in \mathcal{H}} \tau(H) \geq |\mathcal{H}|$, where $\tau(G)$ is the size of a minimum feedback arc set for a directed graph G. The approach is to upper-bound the solution size k in t and ℓ (using Lemma 6.1) and apply the fixed-parameter algorithm for k [KS10]. Observe in this context that Lemma 6.1 is also correct if the input graphs are directed and if a solution contains arc reversals, since we observed arc reversals and deletions to be equivalent in the context of FAST.[1] We use the following data reduction rule, which is similar to Reduction Rule 6.1 for Triangle Deletion.

Reduction Rule 6.2. *If there is a subtournament $H \in \mathcal{H}$ and a feedback arc set $T \subseteq A(H)$ of H of size $\tau(H)$ such that reversing the arcs in T leaves no*

[1] For directed input graphs, we use the term external *arc* instead of external *edge*.

directed triangles in G containing any arcs of H, then reverse the arcs in T, remove H from $\mathcal{H}$, and decrease k by $\tau(H)$.

Although Reduction Rule 6.2 is strikingly similar to Reduction Rule 6.1 , its correctness proof is significantly more involved. In the following, let $\Gamma(G,k)$ be the fastest running time for computing a feedback arc set of size at most k for a given tournament G if it exists. We assume that Γ is monotonically nondecreasing in both the size of G and in k. As mentioned earlier, $2^{5.24\sqrt{k}} + |V(G)|^{O(1)}$ is the currently best known upper bound for $\Gamma(G,k)$ [FP13].

Lemma 6.8. *Reduction Rule 6.2 is correct and, given the tournaments G and H it can be applied in $O\left(\binom{q}{2}(n-q) + \Gamma(H,t)\right)$ time, where $q := |V(H)|$.*

Proof. We first show correctness. Let $I' := (G', \mathcal{H} \setminus \{H\}, k - \tau(H))$ be the instance created by Reduction Rule 6.2 from $I := (G, \mathcal{H}, k)$ by reversing a subset T of arcs of a subtournament $H \in \mathcal{H}$ of G. If I' is a yes-instance, then so is I since G' is the graph G with the $\tau(H)$ arcs in T reversed and, thus, adding these arcs to a feedback arc set of size $k - \tau(H)$ for G' gives a feedback arc set of size k for G. It remains to prove that if I is a yes-instance, then so is I'. To this end, we show that there is a minimum-size feedback arc set S for G with $T \subseteq S$.

Let S be a minimum-size feedback arc set for $G = (V, A)$. This implies the existence of a linear ordering $\sigma_S = v_1, \ldots, v_n$ of the vertices V such that there are $|S|$ *backward arcs*, that is, arcs $(v_i, v_j) \in A$ such that $i > j$. Now, let $\sigma_T = w_1, \ldots, w_{|W|}$ be the ordering of the vertices of $H = (W, B)$ corresponding to the local solution T for H with $\tau(H)$ backward arcs. Let $N^+(w) := \{(w, v) \in A \mid v \in V \setminus W\}$ denote the *out-neighbors in $V \setminus W$* of a vertex $w \in W$. Analogously, $N^-(w) := \{(v, w) \in A \mid v \in V \setminus W\}$ denotes the set of *in-neighbors*. By the assumption of Reduction Rule 6.2, for all $i < j$,

$$N^+(w_j) \subseteq N^+(w_i) \text{ and } N^-(w_i) \subseteq N^-(w_j)$$

holds since otherwise, after reversing the arcs in T, there exists a directed triangle containing the arc (w_i, w_j) of H.

If the vertices in W appear in the same relative order in σ_S as in σ_T, then we have $T \subseteq S$ and we are done. Otherwise, we show that we can swap the positions of vertices of W in σ_S so that their relative order is the same as in σ_T without increasing the number of backward arcs.

First, note that the number of backward arcs between vertices in $V \setminus W$ does not change when only swapping positions of vertices in W. Also, by

assumption, the number of backward arcs between vertices in W in any ordering is at least $\tau(H)$, whereas it is exactly $\tau(H)$ when ordering them according to σ_T. Thus, it remains to show that the number of backward arcs between vertices in W and $V \setminus W$ is not increased. To this end, consider a series of *swaps* of pairs of vertices w_j and w_i such that $i < j$, where w_j appears before w_i in σ_S, reordering the vertices in W according to σ_T. Let Y denote the set of all vertices that lie between w_j and w_i in σ_S (see Figure 6.3 for an example) and note that swapping w_j and w_i can only introduce backward arcs between $\{w_j, w_i\}$ and vertices in Y. Now, swapping w_j and w_i removes the backward arcs from w_i to the vertices in $N^+(w_i) \cap Y$ and the backward arcs from vertices in $N^-(w_j) \cap Y$ to w_j, whereas it introduces new backward arcs from w_j to $(N^+(w_j) \cap Y) \subseteq (N^+(w_i) \cap Y)$ and from $(N^-(w_i) \cap Y) \subseteq (N^-(w_j) \cap Y)$ to w_i. It follows that the overall number of backward arcs does not increase in each swap. Hence, the overall number of backward arcs is not increased by repositioning the vertices in W according to σ_T. It follows that there is an optimal solution containing T.

It remains to show the running time. First, in $\Gamma(H, t)$ time, we compute the size $\tau(H)$ of an optimal feedback arc set for $H = (W, B)$. Now, for each arc $(u, v) \in B$, we check whether there is a vertex $w \in V \setminus W$ that forms a directed triangle with u and v. If such a vertex exists, then we reverse the arc (u, v). If this arc reversal introduces a new directed triangle with another vertex from $V \setminus W$, then the rule does not apply. Overall, this procedure requires $O\big(|B| \cdot (n - |W|)\big)$ time. Let T^* denote the set of arcs that are reversed in this process. Clearly, if $|T^*| > \tau(H)$, then the rule does not apply. Otherwise, let H' denote the graph obtained from H by reversing the arcs in T^* and observe that each remaining directed triangle of G that contains at least one arc of H' is contained in H'. Thus, we now compute whether H' has a feedback arc set T' of size $\tau(H) - |T^*|$ in $\Gamma(H', \tau(H) - |T^*|)$ time. If this is the case, then the rule applies and we set $T := T' \cup T^*$ (note that $T' \cap T^* = \emptyset$, since otherwise $|T| < \tau(H)$, which is not possible by definition of $\tau(H)$). Otherwise, removing all directed triangles in G that contain at least one arc from H requires more than $\tau(H)$ arc reversals and thus the rule does not apply. □

Exhaustive application of Reduction Rule 6.2 allows us to show that $k \leq (2t+1)\ell$ holds for any yes-instance.

Lemma 6.9. *Let $(G, \mathcal{H}, k)$ be a yes-instance of* Feedback Arc Set in Tournaments with Cost-t Packing *such that Reduction Rule 6.2 cannot be applied to any tournament in $\mathcal{H}$. Then, $k \leq (2t + 1)\ell$.*

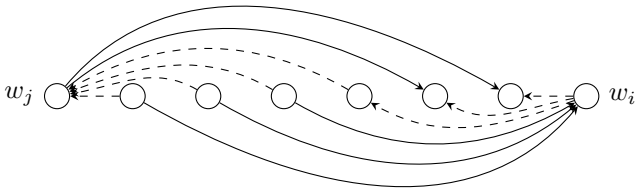


Figure 6.3: Example of two vertices w_j, w_i with $i < j$ in an ordering σ_S. The set Y of vertices between w_j and w_i in σ_S contains six vertices. By swapping w_i and w_j, all backward arcs (dashed) are reversed (become solid) whereas all forward arcs (solid) become new backward arcs (become dashed). Note that the number of backward arcs does not increase since the number of backward arcs is at least the number of forward arcs.

Proof. Since Reduction Rule 6.2 cannot be applied to any tournament in $\mathcal{H}$, for each tournament H in $\mathcal{H}$, there is a set of arcs between $V(H)$ and $V(G) \setminus V(H)$ that witness that no optimal feedback arc set for H removes all directed triangles containing at least one arc in H.

Now, for any size-k solution S, there are two possibilities for each packing tournament $H \in \mathcal{H}$:

(a) at least $\tau(H) + 1$ arcs in H are reversed, or

(b) at least one external arc of H is reversed.

Therefore, S fulfills the condition of Lemma 6.1 and, thus, $k \leq 2(t+1)\ell$. □

Theorem 6.10. *Let $\Gamma(G, k)$ be the fastest running time for computing a feedback arc set of size at most k for a given tournament G if it exists. Then,* Feedback Arc Set in Tournaments with Cost-t Packing

(i) is solvable in $\Gamma(G, (2t+1)\ell) + n^{O(1)} + \sum_{H \in \mathcal{H}} \Gamma(H, t)$ time, and

(ii) admits a problem kernel with at most $(8t+4)\ell$ vertices computable in $n^{O(1)} + \sum_{H \in \mathcal{H}} \Gamma(H, t)$ time.

Proof. (i) Given $(G, \mathcal{H}, k)$, we first apply Reduction Rule 6.2 for each $H \in \mathcal{H}$. This application can be performed in $O(n^4 + \sum_{H \in \mathcal{H}} \Gamma(H, t))$ time by Lemma 6.8

since $|\mathcal{H}| \leq n$. One pass of this rule over all $H \in \mathcal{H}$ is sufficient to obtain an instance $(G', \mathcal{H}', k')$ that is reduced since reversing arcs in some $H \in \mathcal{H}$ does not remove any directed triangles containing arcs of any other $H' \in \mathcal{H}$ with $H' \neq H$. By Lemma 6.9, we can then reject the instance if $k' > (2t+1)\ell$. Otherwise, we can find a solution in $\Gamma(G', (2t+1)\ell)$ time.

(ii) First, we apply Reduction Rule 6.2 once for each $H \in \mathcal{H}$ in $O(n^4 + \sum_{H \in \mathcal{H}} \Gamma(H, t))$ time. After one pass of this rule, the resulting instance $(G', \mathcal{H}', k')$ is reduced since reversing arcs in some $H \in \mathcal{H}$ does not remove any directed triangles containing arcs of any other $H' \in \mathcal{H}$ with $H' \neq H$. By Lemma 6.9, we can reject if $k' > (2t+1)\ell$. Otherwise, we apply the kernelization algorithm for FAST by Paul et al. [PPT16] to the instance (G', k') to obtain an equivalent instance (G'', k'') with at most $4k' \leq (8t+4)\ell$ vertices and a solution size $k'' \leq k'$ in polynomial time. Hence, $(G'', \emptyset, k'')$ is our problem kernel with parameter $\ell'' = k'' \leq (2t+1)\ell$ of Feedback Arc Set in Tournaments with Cost-t Packing. □

Corollary 6.11. Feedback Arc Set in Tournaments with Cost-t Packing

(i) can be solved in $2^{O\left(\sqrt{(2t+1)\ell}\right)} + n^{O(1)} + n2^{O(\sqrt{t})}$ *time, and*

(ii) admits a problem kernel with at most $(8t+4)\ell$ *vertices computable in* $n^{O(1)} + n2^{O(\sqrt{t})}$ *time.*

6.5 Cluster Editing

We finally apply our framework from Section 6.2 to Cluster Editing, a well-studied edge modification problem in parameterized complexity [Böc12, CM12, Fom+14, Gra+05, KU12]. The goal is to modify at most k edges in a given graph such that the resulting graph is a *cluster graph*, that is, a disjoint union of cliques. For brevity, we refer to the connected components of a cluster graph (which are cliques) and to their vertex sets as *clusters*. A graph is a cluster graph if and only if it is P_3-free [SST04]. Thus, Cluster Editing is the problem P_3-free Editing. This graph clustering problem (also known as Correlation Clustering) has a wide range of applications in the context of machine learning and bioinformatics [BB13, BBC04, CDK14, Deh+06, SST04].

The currently fastest algorithm for Cluster Editing parameterized by the solution size k runs in $O(1.62^k + n + m)$ time [Böc12]. Assuming the Exponential

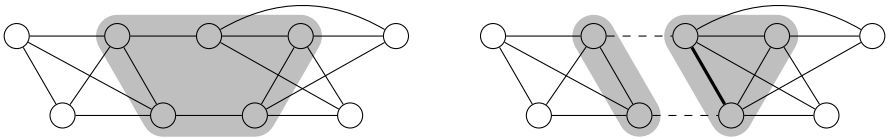


Figure 6.4: An illustration of Reduction Rule 6.3. Left: An induced subgraph $H = (W, D)$ (highlighted by the gray background) fulfilling the conditions of Reduction Rule 6.3. Right: The result of applying Reduction Rule 6.3. The two clusters in $G[W]$ produced by the optimal solution for H are highlighted by a gray background.

Time Hypothesis [IPZ01] (see also Chapter 3), Cluster Editing cannot be solved in $2^{o(k)} \cdot n^{O(1)}$ time [Fom+14, KU12]. Cluster Editing admits a problem kernel with at most $2k$ vertices [CM12].

We present a fixed-parameter algorithm and a problem kernel for Cluster Editing parameterized above lower bounds given by cost-t packings. Several kernelizations for Cluster Editing are based on the following observation: If G contains a clique such that all vertices in this clique have the same closed neighborhood, then there is an optimal solution that leaves these vertices in the same cluster [CM12, Guo09, PSS09]. This implies that the edges of this clique are never deleted. The following rule is based on a generalization of this observation.

Reduction Rule 6.3. *If $G = (V, E)$ contains an induced subgraph $H = (W, D) \in \mathcal{H}$ having an optimal solution S of size $\tau(H)$ such that, for all vertices $u, v \in W$,*

- *$N_G(v) \setminus W = N_G(u) \setminus W$ if u and v are in the same cluster of $H \triangle S$, and*
- *$N_G(v) \cap N_G(u) \subseteq W$ otherwise,*

then replace G by $G \triangle S$, remove H from $\mathcal{H}$, and decrease k by $\tau(H)$.

An example of Reduction Rule 6.3 is presented in Figure 6.4. Note that a necessary condition for Reduction Rule 6.3 to apply is that for every pair of vertices in H, their neighborhoods outside of H are either equal or disjoint.

Lemma 6.12. *Reduction Rule 6.3 is correct.*

Proof. Let $I' := (G \triangle S, \mathcal{H}', k - \tau(H))$ be the instance obtained by applying Reduction Rule 6.3 to $I := (G, \mathcal{H}, k)$ for some induced subgraph $H = (W, D) \in$

$\mathcal{H}$ of $G = (V, E)$. If I' is a yes-instance, then so is I: If there is a solution S' of size at most $k - \tau(H)$ for $G \triangle S$, then $S^* := (S \setminus S') \cup (S' \setminus S)$ is a solution of size at most k for G since $G \triangle S^* = (G \triangle S) \triangle S'$. It remains to prove that if I is a yes-instance, then so is I'. To this end, we show that $(G, \mathcal{H}, k)$ has an optimal solution S' such that $S \subseteq S'$.

Let S^* be an optimal solution for G, and denote by $G^* := G \triangle S^*$ the cluster graph produced by S^*. We show how to transform S^* into an optimal solution S' containing S. For convenience, let $X := V \setminus W$ and consider the partition $S^* = S^*_X \uplus S^*_{XW} \uplus S^*_W$, where

- $S^*_X := \{\{u, v\} \in S^* \mid u, v \in X\}$ contains all edge modifications outside of H,
- $S^*_{XW} := \{\{u, v\} \in S^* \mid u \in X \wedge v \in W\}$ contains the edge modifications between H and the rest of G, and
- $S^*_W := \{\{u, v\} \in S^* \mid u, v \in W\}$ contains the edge modifications inside of H.

The new solution S' will also contain the edge modifications S^*_X. Moreover, the edge modifications S^*_W will be replaced by S. It remains to replace S^*_{XW} by a suitable set of edge modifications between H and the rest of G. The idea is to treat all vertices of H with the same neighborhood in X equally.

Let W_{ext} be the vertices of W that have at least one neighbor in X and let $W_{\text{int}} := W \setminus W_{\text{ext}}$. We define the following equivalence relation $\sim \subseteq W_{\text{ext}} \times W_{\text{ext}}$: Two vertices $u, v \in W_{\text{ext}}$ are equivalent with respect to $\sim$ (denoted $u \sim v$) if and only if $N_G(u) \cap X = N_G(v) \cap X$. For $u \in W_{\text{ext}}$, let $[u] := \{v \in W_{\text{ext}} \mid u \sim v\}$ denote the *equivalence class* of u with respect to $\sim$. Fix within each equivalence class of $\sim$ an arbitrary vertex that is incident to a minimum number of edge modifications in S^*_{XW} and, for each vertex $u \in W_{\text{ext}}$, denote the fixed vertex of $[u]$ by $\tilde{u}$. Furthermore, for each cluster K of G^* containing vertices of X and of W_{ext}, fix an arbitrary vertex of $K \cap W_{\text{ext}}$ that has in G a maximum number of neighbors in $K \cap X$; denote this vertex by u_K. Finally, call a vertex $u \in W_{\text{ext}}$ *good* if there exists a cluster K of G^* such that $[u] = [u_K]$, that is, $u_K \in [u]$ (note that if u is good, then all vertices in $[u]$ are good and have the maximum number of neighbors in $K \cap X$).

Now, consider the edge modification set $S' := S^*_X \cup \tilde{S} \cup S$, where

$$\tilde{S} := \Big\{\{u,v\} \mid u \in W_{\text{ext}} \text{ is good and } \{\tilde{u},v\} \in S^*_{XW}\Big\} \cup \Big\{\{u,v\} \mid u \in W_{\text{ext}} \text{ is not good and } v \in (N_G(u) \cap X)\Big\}.$$

Intuitively, the modifications in $\tilde{S}$ are such that $\tilde{u}$ determines how to treat all vertices in the equivalence class $[\tilde{u}]$. If $\tilde{u}$ is good, then all vertices in $[\tilde{u}]$ are treated like $\tilde{u}$ in S^*_{XW}. Otherwise, all edges between vertices in $[\tilde{u}]$ and X in G are deleted (that is, are contained in $\tilde{S}$).

We first show that S' is a solution, that is, $G' := G \triangle S'$ is a cluster graph. Clearly, $G'[X] = G[X] \triangle S^*_X = G^*[X]$ is a cluster graph and $G'[W] = H \triangle S$ is a cluster graph. It remains to show that every connected component of G' that contains vertices of W and X is a clique. We first show that such a connected component does not contain any vertex of W_{int}. To see this, note that there are no edges between W_{int} and X in G. Also, no edges between W_{int} and X are added by $\tilde{S}$. By the first condition on S in Reduction Rule 6.3, no cluster of $H \triangle S$ contains vertices from W_{int} and from W_{ext}. This implies that the connected component of each vertex $u \in W_{\text{int}}$ in G' is completely contained in W_{int}. Hence, it remains to consider connected components of G' containing a vertex $u \in W_{\text{ext}}$ that is connected to some vertices in X. By the conditions of Reduction Rule 6.3, the equivalence class $[u]$ is a cluster in $H \triangle S$. If u is not good, then u is not adjacent to any vertex of X in G'. Thus, u is good (that is, all vertices in $[u]$ are good). Then, all vertices of $[u]$ are adjacent to all vertices of exactly one cluster in $G'[X]$ only and not to any other vertex in X. Hence, the connected component of u in G' is a clique. Altogether this shows that G' is a cluster graph.

It remains to show that $|S'| \leq |S^*|$. First, S^*_X is a subset of both S^* and S'. Second, $|S| \leq |S^*_W|$ since S is an optimal solution for H. Thus, it remains to show $|\tilde{S}| \leq |S^*_{XW}|$. Since the vertices of W_{int} are not incident to any edge modifications in $\tilde{S}$, it suffices to prove this inequality by showing, for each vertex $u \in W_{\text{ext}}$, that

$$|\{e \in \tilde{S} \mid u \in e\}| \leq |\{e \in S^*_{XW} \mid u \in e\}|.$$

If u is good, then the number of edge modifications incident with u in $\tilde{S}$ is the same as the number of edge modifications incident with $\tilde{u}$ in S^*_{XW}. By the choice of $\tilde{u}$, this number is at most the number of edge modifications incident with u in S^*_{XW} and the above inequality holds.

Otherwise, all edges between u and X are deleted by $\tilde{S}$, that is, u is incident to $|N_G(u) \cap X|$ edge deletions in $\tilde{S}$. Let K denote the cluster in G^* containing u. If K contains only vertices in W_{ext}, then u is also incident to $|N_G(u) \cap X|$ edge deletions in S^*_{XW}. Otherwise, since u is not good, there is a vertex $w \in (K \cap W_{\text{ext}})$ that is not in $[u]$ such that w has at least as many neighbors in $K \cap X$ as u. Since $w \notin [u]$ (that is, $w \not\sim u$), it follows that $N_G(u) \cap X \neq N_G(w) \cap X$. By the conditions of Reduction Rule 6.3, it follows that $(N_G(u) \cap X) \cap (N_G(w) \cap X) = \emptyset$. Since w has at least as many neighbors in $K \cap X$ as u, this means that

$$|N_G(u) \cap K \cap X| \leq |K \cap X|/2.$$

Now, observe that S^*_{XW} contains an edge insertion between u and each vertex in $(K \cap X) \setminus N_G(u)$. Thus, at least $|K \cap X|/2$ edges between u and $K \cap X$ are inserted by S^*_{XW}. Moreover, S^*_{XW} contains an edge deletion between u and each vertex in $N_G(u) \cap (X \setminus K)$. Altogether, this implies

$$\begin{aligned} |\{e \in S^*_{Xw} \mid u \in e\}| &\geq |K \cap X|/2 + |N_G(u) \cap (X \setminus K)| \\ &\geq |N_G(u) \cap K \cap X| + |N_G(u) \cap (X \setminus K)| \\ &= |N_G(u) \cap X| = |\{e \in \tilde{S} \mid u \in e\}|. \end{aligned}$$

□

It remains to analyze the running time for applying Reduction Rule 6.3. Let $\Gamma(G, k)$ be the fastest running time for computing a P_3-free editing set of size at most k in a graph G if it exists. Here, we assume that Γ is monotonically nondecreasing in k and polynomial in the size of G. Currently, $O(1.62^k + |V(G)| + |E(G)|)$ is the best known upper bound for $\Gamma(G, k)$ [Böc12, Theorem 2].

Lemma 6.13. *In $O(m + \sum_{H \in \mathcal{H}} \Gamma(H, t))$ time, we can apply Reduction Rule 6.3 to all graphs in $\mathcal{H}$.*

Proof. We first show that the rule can be applied in $O(|W| + \sum_{w \in W} \deg_G(w) + \Gamma(H, t))$ time to an arbitrary graph $H = (W, D) \in \mathcal{H}$. For convenience, let $X := V \setminus W$.

First, observe that a necessary condition for the rule is that, for each pair of vertices u and v in H, their neighborhoods in X are the same or disjoint. This can be checked in $O(|W| + \sum_{w \in W} \deg_G(w))$ time as follows. First, build the bipartite graph

$$B := (W \cup (N_G(W) \setminus W), \{\{u, v\} \in E(G) \mid u \in W, v \in (N_G(W) \setminus W)\}).$$

The graph B is a disjoint union of complete bipartite graphs if and only if the above necessary condition is fulfilled. Thus, we check in $O(|W| + \sum_{w \in W} \deg_G(w))$ time whether B is a disjoint union of complete bipartite graphs. If not, then the rule does not apply. Otherwise, we compute in $O(|W| + \sum_{w \in W} \deg_G(w))$ time the groups of vertices of W whose neighborhood is the same in B. Afterwards, we can check in $O(1)$ time whether u and v have the same neighborhood in X in G by checking whether they belong to the same group. We now compute the set S' of edge modifications that is already determined by the conditions of Reduction Rule 6.3: If v and w are nonadjacent and have the same nonempty neighborhood in X, then the edge $\{v, w\}$ needs to be inserted and is thus added to S'. Similarly, if v and w are adjacent and have different neighborhoods in X, then $\{v, w\}$ needs to be deleted and is thus added to S'. Observe that if S' is a subset of an optimal solution S for H, then $|S'| \leq |D|$. Hence, at most $|D|$ edges are added to S'. Since we already computed the groups of vertices that have the same or disjoint neighborhoods, we can thus compute S' in $O(|W| + |D|)$ time.

Let W_{ext} denote the vertices of W that have at least one neighbor in X and let $W_{\text{int}} := W \setminus W_{\text{ext}}$. Note that after applying the edge modifications in S', that is, in $H \triangle S'$, the induced subgraph $(H \triangle S')[W_{\text{ext}}]$ is a cluster graph and there are no edges between W_{ext} and W_{int} in $H \triangle S'$. Thus, to determine whether S' can be extended to an optimal solution S for H that fulfills the conditions of Reduction Rule 6.3 we compute an optimal solution of $H[W_{\text{int}}]$ in $\Gamma(H[W_{\text{int}}], \tau(H) - |S'|)$ time. Since $|H[W_{\text{int}}]| \leq |H|$ and $\tau(H) - |S'| \leq t$, this can be done in $\Gamma(H, t)$ time. The size of the resulting solution S is compared with the size of an optimal solution of H, which can also be computed in $\Gamma(H, t)$ time.

It remains to show that Reduction Rule 6.3 can be applied to all packing subgraphs within the claimed running time. For each graph $H = (W, D) \in \mathcal{H}$, we can check in $O(|W| + \sum_{w \in W} \deg_G(w) + \Gamma(H, t))$ time whether Reduction Rule 6.3 applies. If it does, then we can apply the rule in $O(|D|)$ time by modifying at most $|D|$ edges. Summing up over all graphs in $\mathcal{H}$ gives the claimed running time. □

Observe that since k is decreased by $\tau(H)$, the parameter ℓ does not increase when Reduction Rule 6.3 is applied. As for the previous problems, applying the rule to each $H \in \mathcal{H}$ is sufficient for upper-bounding k in terms of t and ℓ and thus, for transferring known fixed-parameter tractability results for the parameter k to the combined parameter (t, ℓ).

Lemma 6.14. *Let $(G, \mathcal{H}, k)$ be a yes-instance of* Cluster Editing with Cost-t Packing *such that Reduction Rule 6.3 does not apply to any $H \in \mathcal{H}$. Then, $k \leq (2t+1)\ell$.*

Proof. Since the instance is reduced with respect to Reduction Rule 6.3, for each $H = (W, D)$ in $\mathcal{H}$ and each size-$\tau(H)$ solution S for H, either the vertices of some cluster have different neighborhoods in $V \setminus W$ or two vertices of two distinct clusters have a common neighbor outside of W.

Now, fix an arbitrary optimal solution S for G. By the observation above, there are the following two possibilities for how S modifies each $H \in \mathcal{H}$:

(a) more than $\tau(H)$ vertex pairs of H are modified by S, or

(b) at least one external vertex pair for H is modified.

Therefore, S fulfills the condition of Lemma 6.1 and thus $k \leq (2t+1)\ell$. □

Theorem 6.15. *Let $\Gamma(G, k)$ be the fastest running time for computing a P_3-free editing set of size at most k in a graph G if it exists. Then,* Cluster Editing with Cost-t Packing

(i) is solvable in $O(\Gamma(G, (2t+1)\ell) + nm + \sum_{H \in \mathcal{H}} \Gamma(H, t))$ time and

(ii) admits a problem kernel with at most $(4t+2)\ell$ vertices, which can be computed in $O(nm + \sum_{H \in \mathcal{H}} \Gamma(H, t))$ time.

Proof. (ii) First, apply Reduction Rule 6.3 exhaustively in $O(m + \sum_{H \in \mathcal{H}} \Gamma(H, t))$ time. Then, by Lemma 6.14, we can either return a trivial constant-size no-instance or have $k \leq (2t+1)\ell$. In the latter case, we apply a kernelization algorithm for Cluster Editing to the instance (G, k) (that is, without $\mathcal{H}$), which produces, in $O(nm)$ time, a problem kernel (G', k') with at most $2k \leq (4t+2)\ell$ vertices and with $k' \leq k$ [CM12]. Adding an empty packing gives an equivalent instance $(G', \emptyset, k')$ with parameter $\ell' = k'$ of Cluster Editing with Cost-t Packing.

(i) First, apply the kernelization. Then, by Lemma 6.14, we can either return "no" or have $k \leq (2t+1)\ell$. We can now apply the algorithm for Cluster Editing that runs in $\Gamma(G, (2t+1)\ell)$ time. □

By plugging in the best known bound for $\Gamma(G, k)$, we obtain the following.

Corollary 6.16. Cluster Editing with Cost-t Packing

(i) can be solved in $O(1.62^{(2t+1)\cdot\ell} + nm + n \cdot 1.62^t)$ *time, and*

(ii) admits a problem kernel with at most $(4t+2)\ell$ *vertices that can be computed in* $O(nm + n \cdot 1.62^t)$ *time.*

For Cluster Editing with P_3-Packing, the above generic algorithm with $t = 1$ leads to a running time of $O(4.26^\ell + nm)$. This running time can be improved using two special cases of Reduction Rule 6.3 and two further reduction rules combined within a branching algorithm.

Theorem 6.17 ([BFK18])**.** Cluster Editing with P_3-Packing *can be solved in* $O(4^\ell \cdot \ell^3 + m + n)$ *time.*

6.6 NP-Hardness Results

In the previous three sections we demonstrated fixed-parameter tractability for three versions of F-free Editing with respect to our parameterization above lower bounds based on subgraph packings. It is natural to ask how far we can get with the parameter ℓ in terms of problems that are fixed-parameter tractable with respect to ℓ. In this section, we partially answer this question and show various NP-hardness results even for small forbidden induced subgraphs F and constant parameter $\ell = 0$. The results imply that, unless P = NP, the considered problems cannot be fixed-parameter tractable with respect to ℓ. Similar hardness results have also been shown for several other problems with respect to parameters above guaranteed values [MRS09].

6.6.1 Hard Edge Deletion Problems

Our first result shows that fixed-parameter tractability of F-free Editing with Cost-t Packing with respect to ℓ is unlikely to hold for all graphs F (in contrast to the parameter k for which this is the case [Cai96]).

Theorem 6.18. *For every fixed* $q \geq 6$, K_q-free Deletion with K_q-Packing *is NP-hard for* $\ell = 0$.

We give a polynomial-time many-one reduction from 3-SAT.

3-SAT

Input: A Boolean formula $\phi = C_1 \wedge \ldots \wedge C_m$ in conjunctive normal form over variables $x_1, \ldots, x_n$ with at most three literals per clause.

Question: Does ϕ have a satisfying truth assignment?

Construction 6.19. Let ϕ be a Boolean CNF formula with variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$. We assume without loss of generality that each clause C_j contains exactly three pairwise distinct variables. We create a graph G and a vertex-disjoint K_q-packing $\mathcal{H}$ as follows (see Figure 6.5 for an example).

For each variable x_i, add a clique X_i on q vertices $v_{i1}, \ldots, v_{iq}$ to G that has two distinguished disjoint edges $e_i^{\mathrm{F}} := \{v_{i1}, v_{i2}\}$ and $e_i^{\mathrm{T}} := \{v_{i3}, v_{i4}\}$. For each clause $C_j = (l_1 \wedge l_2 \wedge l_3)$ with literals l_1, l_2, and l_3, add $q - 6$ vertices $U_j = \{u_{j1}, \ldots, u_{j(q-6)}\}$ to G. Now, for each $t \in [3]$, if $l_t = x_i$, then connect the endpoints of e_i^{T} with all vertices in U_j and if $l_t = \neg x_i$, then connect the endpoints of e_i^{F} with all vertices in U_j. Finally, connect all six neighbors of U_j with each other such that the vertices in U_j together with their neighbors induce a K_q and denote this subgraph by Y_j. The packing $\mathcal{H}$ consists of all X_i introduced for the variables x_i of ϕ and we set $k := |\mathcal{H}|$.

Before proving Theorem 6.18, we make the following simple observation about induced K_qs in the graph obtained from Construction 6.19.

Observation 6.20. *Let G be the graph output by Construction 6.19 and let H be an induced K_q in G. Then, H is either one of the cliques X_i or one of the cliques Y_j.*

Proof. First, note that the X_i are pairwise vertex-disjoint. For any X_i and Y_j, the vertices in $V(X_i) \setminus V(Y_j)$ are nonadjacent to those in $V(Y_j) \setminus V(X_i) = U_j$. Similarly, for Y_i and Y_j, the vertices in U_i are nonadjacent to those in U_j for $i \neq j$. Thus, every clique in G is entirely contained in one of the X_i or Y_j. □

Using Observation 6.20, we now prove Theorem 6.18.

Proof of Theorem 6.18. Let ϕ be a 3-SAT instance with variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$ and let $(G, \mathcal{H}, k = |\mathcal{H}|)$ be the instance returned by Construction 6.19. We show that ϕ is satisfiable if and only if G can be made K_q-free by $k = |\mathcal{H}|$ edge deletions (that is, $\ell = 0$).

First, assume that there is an assignment that satisfies ϕ. We construct a K_q-free deletion set S for G as follows: If the variable x_i is set to true, then

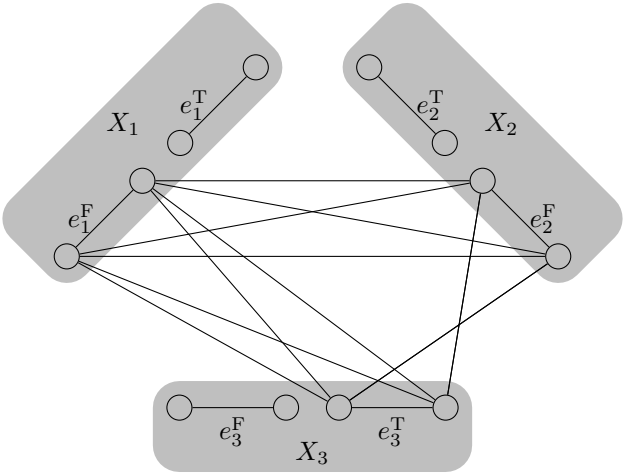


Figure 6.5: An illustration of Construction 6.19 for $q = 6$. The gray rectangles depict the vertex-disjoint packing K_6s X_1, X_2, and X_3 (for simplicity only two edges are shown inside each packing K_6). The K_6 containing the edges e_1^{F}, e_2^{F}, and e_3^{T} is denoted Y_j and encodes the clause $C_j = (\neg x_1 \vee \neg x_2 \vee x_3)$.

add the edge e_i^{T} to S. If the variable x_i is set to false, then add e_i^{F} to S. Thus, for each X_i, we add exactly one edge to S. Since $\mathcal{H}$ consists of the X_i, we have $|S| = |\mathcal{H}|$. Moreover, since each clause C_j contains a true literal, at least one edge of each Y_j is contained in S. Thus, $G \setminus S$ is K_q-free, since, by Observation 6.20, the only K_qs in G are the X_i and Y_j and, for each of them, S contains at least one edge.

Now, assume that G can be made K_q-free by deleting a set S of $|\mathcal{H}|$ edges. Then, S deletes exactly one edge of each X_i and at least one edge of each Y_j. We can assume without loss of generality that S contains for each X_i either the edge e_i^{T} or the edge e_i^{F} since deleting one of these edges instead of another edge in X_i always yields a solution by construction. Thus, the edge deletion set S corresponds to a truth assignment that satisfies ϕ. □

Let us briefly discuss why it is not possible to apply our framework (that is, use Lemma 6.1 to derive $k \leq (2t+1)\ell$) for the case $F = K_6$. In fact, it is indeed possible to formulate a data reduction rule for K_6-free Deletion with K_6-Packing analogous to Reduction Rule 6.1 for the triangle case. However,

it is not possible to use Lemma 6.1 since the conditions of the lemma do not necessarily hold for reduced yes-instances. This is due to the fact that a certificate for solving a packing K_6 suboptimally (that is, a set of external edges) might not exist since we could delete an edge contained in another K_6 from the packing (Construction 6.19 is based on exactly this property). This is possible since a K_6 contains vertex-disjoint edges.

6.6.2 Hardness for Edge-Disjoint Packings

We complement the fixed-parameter tractability of Triangle Deletion with Cost-t Packing with respect to ℓ (Theorem 6.5) by the following hardness result for the case of edge-disjoint K_3-packings:

Theorem 6.21. Triangle Deletion *is NP-hard even for* $\ell := k - |\mathcal{H}| = 0$ *if* $\mathcal{H}$ *is an* edge-disjoint *packing of* K_3*s.*

Theorem 6.21 shows that Triangle Deletion remains intractable when parameterized above a lower bound given by edge-disjoint packings. The reason is that edge-disjoint triangles can be packed more densely, which allows to construct hard instances with a tight lower bound on the number of required edge deletions as the following polynomial-time many-one reduction from 3-SAT shows.

Construction 6.22. Let ϕ be a Boolean 3-CNF formula with n variables and m clauses. We assume that each clause of ϕ contains exactly three pairwise distinct variables. We create a graph G and an edge-disjoint packing $\mathcal{H}$ of K_3s as follows (the construction is illustrated in Figure 6.6).

For each variable x_i of ϕ, create a triangle X_i on the vertex set $\{x_i^1, x_i^2, x_i^3\}$ with two distinguished edges $e_i^{\mathrm{T}} := \{x_i^1, x_i^2\}$ and $e_i^{\mathrm{F}} := \{x_i^2, x_i^3\}$ and add X_i to $\mathcal{H}$. For each clause $C_j = (l_1, l_2, l_3)$ of ϕ, create a triangle Y_j on the vertex set $\{c_j^1, c_j^2, c_j^3\}$ with three edges $c_j^{l_1}$, $c_j^{l_2}$, and $c_j^{l_3}$. Connect the clause gadget Y_j to the variable gadgets X_i as follows: Consider each literal l_t of C_j. Let $\{u, v\} := c_j^{l_t}$. If $l_t = x_i$, then add the edges $\{u, x_i^1\}$, $\{v, x_i^1\}$ and $\{v, x_i^2\}$. Note that the vertices $\{u, v, x_i^1\}$ induce a K_3. We call this subgraph A_{ij} and add it to $\mathcal{H}$. Note that also the vertices $\{v, x_i^1, x_i^2\}$ induce a K_3 (called B_{ij}). If $l_t = \neg x_i$, then add the edges $\{u, x_i^2\}$, $\{v, x_i^2\}$ and $\{v, x_i^3\}$. In this case the vertices $\{u, v, x_i^2\}$ induce a K_3, called A_{ij}, which is added to $\mathcal{H}$. Also, the vertices $\{v, x_i^2, x_i^3\}$ induce a K_3 (called B_{ij}). Clearly, we set $k := |\mathcal{H}|$.

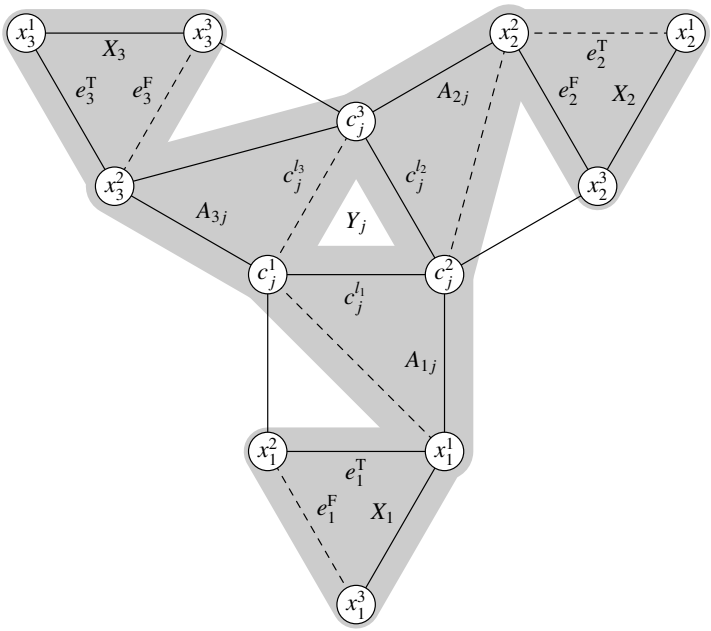


Figure 6.6: Example of constructed gadgets for a clause $C_j = (x_1 \vee \neg x_2 \vee \neg x_3)$. The triangles on a gray background are contained in the mutually edge-disjoint triangle packing $\mathcal{H}$. Deleting the dashed edges corresponds to setting x_1 and x_3 to false and x_2 to true, thus satisfying C_j. Note that it is impossible to destroy triangle Y_j by six edge deletions if we delete the edges e_1^{F}, e_2^{T}, and e_3^{T}, which corresponds to the fact that clause C_j is not satisfied by this truth assignment.

Note that the induced K_3s in $\mathcal{H}$ are indeed pairwise edge-disjoint since each pair intersects in at most one vertex.

Proof of Theorem 6.21. Let ϕ be a 3-SAT instance with variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$ and let $(G = (V, E), \mathcal{H}, k = |\mathcal{H}|)$ be the instance obtained by Construction 6.22. First, observe that Construction 6.22 introduces no edges between distinct clause gadgets or distinct variable gadgets. Thus, under the assumption that each clause contains each variable at most once, the only K_3s in the constructed graph are the X_i for all variables x_i, the Y_j for all clauses C_j, and the A_{ij} and B_{ij} for all variables x_i contained in clauses C_j.

Now, assume that ϕ allows for a satisfying truth assignment. We construct a set S of edges of size $|\mathcal{H}|$ such that $G' := (V, E \setminus S)$ is K_3-free. For each variable x_i that is true, add e_i^{T} to S. For each variable x_i that is false, add e_i^{F} to S. By this choice, each X_i is destroyed in G'. Additionally, for each clause C_j and its *true* literals $l \in \{x_i, \neg x_i\}$, B_{ij} is destroyed. To destroy A_{ij}, we add to S the edge of A_{ij} shared with Y_j, which also destroys the triangle Y_j. For each *false* literal $l \in \{x_i, \neg x_i\}$, we destroy both B_{ij} and A_{ij} by adding to S the shared edge of A_{ij} and B_{ij}.

Conversely, assume that there is a set S of size $|\mathcal{H}|$ such that $G' = (V, E \setminus S)$ is K_3-free. We construct a satisfying truth assignment for ϕ. First, observe that, since the K_3s in $\mathcal{H}$ are pairwise edge-disjoint, S contains exactly one edge of each K_3 in $\mathcal{H}$. Thus, for each X_i, at most one of the two edges e_i^{F} and e_i^{T} is contained in S. The set S contains at least one edge e of each Y_j. This edge is shared with an A_{ij}. Since $A_{ij} \in \mathcal{H}$ and S already contains the edge e of A_{ij}, it follows that S does not contain the edge shared by A_{ij} and B_{ij}. Since $B_{ij} \notin \mathcal{H}$, the set S has to contain an edge of B_{ij} shared with another K_3 in $\mathcal{H}$. If the clause C_j contains x_i, then the only such edge is e_i^{T} and we set x_i to true. If the clause C_j contains $\neg e_i$, then the only such edge is e_i^{F} and we set x_i to false. In both cases, clause C_j is satisfied. Since at most one of e_i^{T} and e_i^{F} is in S, the value of each variable x_i is well-defined. □

6.6.3 Hard Vertex Deletion Problems

In contrast to the previous sections, we consider vertex deletions in this section. We show that the vertex deletion variant (that is, F-Free Vertex Deletion) is an even harder problem than the edge editing variant. Concretely, we prove NP-hardness of the problem of destroying all induced paths P_q on $q \geq 3$ vertices by at most $|\mathcal{H}|$ vertex deletions if a packing $\mathcal{H}$ of vertex-disjoint induced P_qs in the input graph G is provided as input.

P_q-free Vertex Deletion with P_q-Packing
Input: A graph $G = (V, E)$, a vertex-disjoint packing $\mathcal{H}$ of induced P_qs, and a natural number k.
Question: Is there a vertex subset $S \subseteq V$ of size at most k such that $G[V \setminus S]$ does not contain a P_q as an induced subgraph?

The following theorem implies that P_3-Free Vertex Deletion with P_3-Packing parameterized by $\ell := k - |\mathcal{H}|$ is presumably not in XP which is

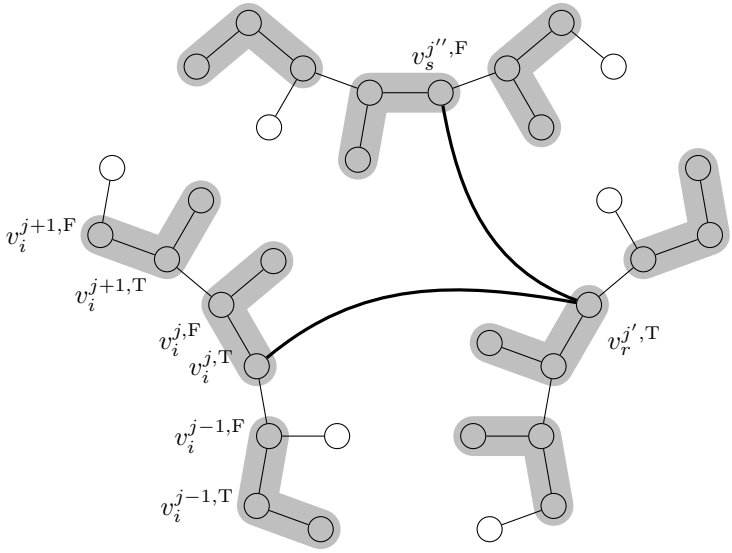


Figure 6.7: An illustration of Construction 6.24 for $q = 3$. The figure shows parts of the variable cycles for three variables x_i, x_r, x_s that occur in the clause $C_t = (x_i \vee x_r \vee \neg x_s)$. The packing P_3s are highlighted by a gray background.

in contrast to the fixed-paramater tractability of Cluster Editing with P_3-Packing (see Section 6.5).

Theorem 6.23. *For every fixed $q \geq 3$, P_q-free Vertex Deletion with P_q-Packing is NP-hard even if $\ell = k - |\mathcal{H}| = 0$.*

The polynomial-time many-one reduction is from q-SAT.

Construction 6.24. Let ϕ be a Boolean q-CNF formula with clauses $C_1, \ldots, C_m$ and variables $x_1, \ldots, x_n$. We assume that each clause C_j contains exactly q pairwise distinct variables. We construct a graph G and a vertex-disjoint packing $\mathcal{H}$ of P_qs as follows; an illustration of the construction is given in Figure 6.7.

First, we introduce variable gadgets, which will ensure that a P_q-free vertex deletion set corresponds to a truth assignment of ϕ. In the following, let $\alpha(i)$ denote the number of occurrences of the literals of variable x_i (that is, x_i and $\neg x_i$) in clauses of ϕ. For each variable x_i, add $4\alpha(i)$ vertices: $v_i^{j,\mathrm{T}}$ and $v_i^{j,\mathrm{F}}$, for each $j \in [2\alpha(i)]$. Call each $v_i^{j,\mathrm{T}}$ a *true vertex* and each $v_i^{j,\mathrm{F}}$ a *false vertex*.

Create an induced cycle on the true and false vertices of variable x_i by adding the edge set

$$E_i := \left\{\{v_i^{j,\mathrm{T}}, v_i^{j,\mathrm{F}}\} \mid j \in [2\alpha(i)]\right\} \cup \left\{\{v_i^{j,\mathrm{F}}, v_i^{j+1,\mathrm{T}}\} \mid j \in [2\alpha(i)-1]\right\} \cup \left\{\{v_i^{2\alpha(i),\mathrm{F}}, v_i^{1,\mathrm{T}}\}\right\}.$$

We call this cycle the *variable cycle* of x_i.

Then, for each even $j \in [2\alpha(i)]$, attach to $v_i^{j,\mathrm{T}}$ and $v_i^{j,\mathrm{F}}$ an induced P_{q-2} each. These paths are called the *attachment paths* of the j-th segment of the variable cycle of x_i.

Now, for each variable x_i, assign to each clause C_t containing x_i or $\neg x_i$ a unique number $p \in [\alpha(i)]$. Consider the number $j = 2p - 1$. We will use vertex $v_i^{j,\mathrm{T}}$ or $v_i^{j,\mathrm{F}}$ to build the clause gadget for clause C_t. If C_t contains the literal x_i, then attach an induced P_{q-2} to $v_i^{j,\mathrm{F}}$. Otherwise, attach an induced P_{q-2} to $v_i^{j,\mathrm{T}}$. As before, call the path the *attachment path* of the j-th segment of the cycle. Now, let $\gamma_i^t := v_i^{j,\mathrm{T}}$ if C_t contains x_i, and let $\gamma_i^t := v_i^{j,\mathrm{F}}$ if C_t contains $\neg x_i$. Call these vertices the *literal vertices* of clause C_t and denote their set by Π_t. The construction of G is completed as follows. For each Π_t, add an arbitrary set of edges to G such that $G[\Pi_t]$ is an induced P_q. The P_q-packing $\mathcal{H}$ contains for each variable x_i an induced P_q for every segment of the variable cycle of x_i: for each $j \in [2\alpha(i)]$, choose an (arbitrary) attachment path (a P_{q-2}) plus the two segment vertices $v_i^{j,\mathrm{F}}$ and $v_i^{j,\mathrm{T}}$. Note that the packing P_qs are clearly vertex-disjoint by construction. We set $k := |\mathcal{H}|$.

Proof of Theorem 6.23. Let ϕ be a q-SAT instance with variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$ and let $(G = (V, E), \mathcal{H}, k = |\mathcal{H}|)$ be the instance obtained by Construction 6.24. We show that ϕ has a satisfying truth assignment if and only if G can be made P_q-free by exactly $|\mathcal{H}|$ vertex deletions (that is, $\ell = 0$).

Assume that ϕ has a satisfying truth assignment. For each true variable x_i in this assignment, delete all true vertices in its variable gadget, that is, $v_i^{j,\mathrm{T}}$ for $j \in [2\alpha(i)]$. For each false variable x_i in this assignment, delete all false vertices in its variable gadget, that is, $v_i^{j,\mathrm{F}}$ for $j \in [2\alpha(i)]$. Denote this vertex set by S and observe that $|S| = |\mathcal{H}|$. Moreover, observe that each vertex on the variable cycle for x_i is either deleted or both of its neighbors on the cycle are deleted. Every induced P_q in G contains at least one vertex from a variable cycle as the attachment paths are too short to induce P_qs. Thus, to show P_q-freeness

of $G[V \setminus S]$ it is sufficient to show that no vertex from a variable cycle is in an induced P_q.

Consider an undeleted vertex in the variable cycle for x_i. Assume, without loss of generality, that this is a true vertex $v_i^{j,\mathrm{T}}$. If j is even, then $v_i^{j,\mathrm{T}}$ is not in an induced P_q as its neighbors on the cycle (which are false vertices) are deleted and its only other neighbor is in an attachment path with only $q-2$ vertices. If j is odd and the clause C_t corresponding to the j-th segment of the cycle contains $\neg x_i$, then the only neighbor of $v_i^{j,\mathrm{T}}$ in $G[V \setminus S]$ is in an attachment path. It remains to show that $v_i^{j,\mathrm{T}}$ is not in an induced P_q if C_t contains x_i. The only neighbors of $v_i^{j,\mathrm{T}}$ in $G[V \setminus S]$ are in Π_t. Observe that $G[\Pi_t]$ is an induced P_q and that, in $G[V \setminus S]$, every vertex on this path is deleted or its neighbors in $V \setminus \Pi_t$ are deleted. Hence, the connected component of $G[V \setminus S]$ containing $v_i^{j,\mathrm{T}}$ is an induced subgraph of $G[\Pi_t]$. Since the assignment is satisfying, at least one vertex of Π_t is deleted. Thus, this connected component has at most $q-1$ vertices and does not contain a P_q.

Conversely, let $S \subseteq V$ be a size-$|\mathcal{H}|$ vertex set such that $G[V \setminus S]$ is P_q-free. First, observe that, without loss of generality, for each variable cycle either all true or all false vertices are deleted: No vertex in an attachment path P is deleted since it is also a solution to delete the vertex in the variable cycle that has a neighbor in P instead. Hence, at least one vertex of each segment is deleted since, otherwise, one of the P_qs in $\mathcal{H}$ is not destroyed. This already requires $|\mathcal{H}|$ vertex deletions and thus *exactly* one vertex for each segment of each variable cycle is deleted. Finally, by construction, every adjacent pair of vertices in the variable cycle forms a P_q with some attachment path. Consequently, one of the two vertices is deleted, which implies that either every even or every odd vertex of the cycle is deleted.

Hence, the vertex deletions in the variable cycle define a truth assignment for $x_1, \ldots, x_n$: If all true vertices of the variable cycle of x_i are deleted, then set x_i to true; otherwise, set x_i to false. This assignment is satisfying: Since $G[V \setminus S]$ is P_q-free, for each clause C_t, at least one vertex γ_i^t of Π_t is deleted. Without loss of generality, let $\gamma_i^t = v_i^{j,\mathrm{T}}$, that is, C_t contains the literal x_i. Then, x_i is true and clause C_t is satisfied. □

Theorem 6.23 easily implies an NP-hardness result for the generalization of Vertex Cover to d-uniform hypergraphs:

d-Uniform Hitting Set with Packing
Input: A hypergraph $H = (V, E)$ with $|e| = d$ for all $e \in E$, a set $\mathcal{H} \subseteq E$ of pairwise vertex-disjoint hyperedges, and an integer k.
Question: Is there a vertex set $V' \subseteq V$ of size at most k such that $\forall e \in E : V' \cap e \neq \emptyset$?

An instance $(G, \mathcal{H}, k)$ of P_q-free Vertex Deletion with P_q-Packing can easily be transformed in polynomial time into an equivalent instance $(H, \mathcal{H}, k)$ of q-Uniform Hitting Set with Packing by taking the hypergraph H on the same vertex set as G having a hyperedge e if and only if $G[e]$ is a P_q. The packing $\mathcal{H}$ and the integer k remain unchanged, and so does ℓ. Thus, we obtain the following corollary:

Corollary 6.25. *For every $d \geq 3$, d-Uniform Hitting Set with Packing is NP-hard even if $\ell = 0$.*

Corollary 6.25 shows that the known above-guarantee fixed-parameter algorithms for Vertex Cover [Cyg+13, GP16, Lok+14, RO09] do not generalize to d-Uniform Hitting Set.

6.7 Conclusion

In this chapter, we introduced a novel parameterization of F-free Editing above a given lower bound derived from a packing of subgraphs and developed a framework for showing fixed-parameter tractability. The basic idea is to apply polynomial-time data reduction to the packing subgraphs such that for the resulting instance the number of edge modifications is upper-bounded by the parameter. We showed that the framework can be applied to three data-science related variants of F-free Editing. Our obtained algorithms are potentially fast in practice due to a smaller parameter value than the overall number of edge modifications. Moreover, since our framework allows to use the fastest known algorithms with respect to the number of edge modifications, existing implementations can be used.

There are several open questions to discuss. First of all, the running times of our algorithms (especially the search tree algorithms for Cluster Editing and

Triangle Deletion) could be improved. Maybe more importantly, it is open to determine the complexity of Cluster Editing and Feedback Arc Set in Tournaments parameterized above a lower bound of *edge-disjoint* packings of forbidden induced subgraphs.

It is open to extend our framework to further edge modification problems. The most natural candidates appear to be F-free Editing variants where the forbidden induced subgraph F has four vertices. Examples are Cograph Editing [Liu+12] which is the problem of destroying all induced P_4s, K_4-free Editing, Claw-free Editing, and Diamond-free Deletion [Fel+11, SS15]. Finding other editing problems on directed graphs for our framework is also interesting.

Furthermore, it would be nice to obtain more general theorems separating the tractable from the hard cases of forbidden subgraphs F for this parameterization. For example, the reduction from 3-SAT in Theorem 6.18 essentially relies on the fact that the graph F contains a matching of size three. Maybe it is possible to prove a complexity dichotomy based on the matching number of F.

On the practical side, our framework offers an interesting tradeoff between running time and power of generic data reduction rules (this is similar to a kernelization scheme by Hüffner et al. [HKN15]). Exploring such tradeoffs (using the maximum cost t of the packing subgraphs as adjusting screw) might be a rewarding topic for future research. For example, an interesting question to answer empirically is how much the value of ℓ can be decreased by increasing t in practice. Our generic data reduction rules are suited for implementation and subsequent experiments to evaluate their effectiveness. Such a project includes further challenges, for example, computing an appropriate packing of induced subgraphs efficiently. A concrete subproblem could be formulated as follows: Given t, find an induced subgraph with editing cost t such that a data reduction rule is applicable. The question whether this problem can be solved efficiently is interesting from a practical as well as from a theoretical perspective.

Finally, parameterizing above a lower bound is a promising approach to obtain efficient algorithms which should also be explored for non-graph problems, for example, feature selection as studied in Chapter 4.

Chapter 7

Dynamic Time Warping

In previous chapters we already studied clustering problems on matrices (Chapter 5) and on graphs (Chapter 6). In this chapter, we study a problem from time series analysis which is fundamental for clustering time series. In particular, we consider the problem of computing a mean series of a given sample of series in dynamic time warping spaces. Dynamic time warping is a distance measure between time series and constitutes a major tool for analyzing time series. Averaging time series under dynamic warping distance is a challenging computational problem, so far solved by several heuristics. Our contributions comprise a (seemingly first) exact exponential-time algorithm for computing a mean (showing that the problem is in XP with respect to the number of input time series, that is, for a constant number of input time series, the problem is polynomial-time solvable). Moreover, we give an exact polynomial-time algorithm for the special case of (arbitrarily many) binary time series.

7.1 Introduction

The mean of a random variable is a fundamental statistical concept with many applications in pattern recognition and data mining. The mean is well understood in Euclidean spaces, but can become obscure in non-metric spaces. An important class of non-metric spaces are dynamic time warping spaces. Time series such as acoustic signals, electrocardiograms, and internet traffic data are time-dependent observations that vary in length and temporal dynamics. Given a sample of time series, to filter out such variations, a major direction of time series averaging is based on using *dynamic time warping* (dtw) as distance measure between time series. Computing a mean of time series with respect to the dynamic time warping distance is a fundamental problem arising

in the context of nearest neighbor classifiers and centroid-based clustering algorithms [HNF08, Mor+18, Pet+16, SDG16].

One way to pose time series averaging as an optimization problem is as follows [CB17, HNF08, PKG11, SDG16, SJ18]: Suppose that $\mathcal{X} = \left(x^{(1)}, \ldots, x^{(k)}\right)$ is a sample of k time series (numerical sequences of arbitrary finite lengths). Then a (Fréchet) *mean* (with respect to dynamic time warping) is any time series z that minimizes the *Fréchet function* [Fré48] (of the sample $\mathcal{X}$)

$$F(z) = \frac{1}{k} \sum_{i=1}^{k} \left(\mathrm{dtw}\left(z, x^{(i)}\right) \right)^2,$$

where $\mathrm{dtw}(x, y)$ denotes the *dynamic time warping distance* between the time series x and y. We refer to the problem of minimizing F over the set of all time series of finite length as the DTW-Mean problem. Note that if we consider only time series from the Euclidean space $\mathbb{R}^n$ and we replace the dtw-distance by the Euclidean norm $\|\cdot\|_2$ in the definition of the Fréchet function, then the solution z simply is the Euclidean mean. For arbitrary time series (of different lengths), however, one cannot apply the Euclidean distance.

Dynamic time warping has been introduced to deal with this scenario (see Section 7.2 for a formal definition). Intuitively, in order to measure the similarity between two time series of different lengths, dynamic time warping allows to stretch the time series non-uniformly such that the resulting "warped" series align well in terms of some cost function. Hence, a mean of a sample of time series with respect to dynamic time warping is a time series that can be aligned well to each sample time series (on average). Figure 7.1 shows an illustrating example.

Related Work. It is known that solutions for DTW-Mean are guaranteed to exist, but are not unique in general [SJ18]. Polynomial-time algorithms for DTW-Mean are unknown. Until recently [BFN18], it was not known to be NP-hard (though NP-hardness of related problems is mentioned in the literature [HNF08, Pet+16, PG17]). Some exponential-time algorithms have been proposed to optimally solve DTW-Mean [HNF08, PG12, PKG11], however, without proving their correctness or running times. Several heuristics have been devised to approximately solve DTW-Mean [CB17, PKG11, SJ18] (without any guarantee on the quality of the solution). They are based on prespecifying the length of feasible solutions without any knowledge about whether an optimal solution of this length exists. In summary, both the computational complexity of

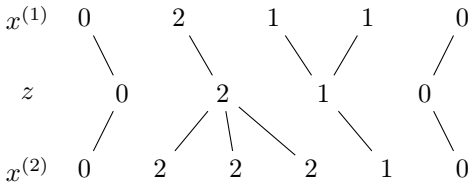


Figure 7.1: A sample of two time series $x^{(1)} = (0, 2, 1, 1, 0)$, $x^{(2)} = (0, 2, 2, 2, 1, 0)$ with a mean $z = (0, 2, 1, 0)$. The lines indicate the corresponding warpings between z and the sample time series. For example, the mean z is stretched at the second position in order to align the 2 with the three 2's in $x^{(2)}$. In this example, the mean is perfectly aligned with both time series.

DTW-Mean and the development of nontrivial exact algorithms with provable running time guarantees are to be considered widely open.

Our Contributions. To start with, Section 7.2 introduces the formal definition of dynamic time warping. In Section 7.3, we briefly discuss several problematic statements in the literature concerning the computational complexity of exact algorithms for DTW-Mean. We refute (with a counterexample) some false claims from the literature and clarify the known complexity status of DTW-Mean. In Section 7.4, we develop a dynamic program solving DTW-Mean optimally (Theorem 7.2). The time complexity of our dynamic program is in $O(n^{2k+1} \cdot k2^k)$, where k is the number of time series in the sample and n is the maximum length of a sample time series. Hence, DTW-Mean is in XP with respect to k. Moreover, as a further contribution, we show in Section 7.5 that in case of binary time series (where both the input series and the mean are over a binary alphabet) a mean can be computed in $O(kn^3)$ time (Theorem 7.3).

7.2 Preliminaries

Throughout this chapter, we consider only finite time series with rational elements. A univariate *time series* of length n is a sequence $x = (x_1, \ldots, x_n) \in \mathbb{Q}^n$. We denote the set of all univariate rational time series of length n by $\mathcal{T}_n$. Furthermore, $\mathcal{T} = \bigcup_{n \in \mathbb{N}} \mathcal{T}_n$ denotes the set of all univariate rational time series of finite length. For simplicity, we neglect running times of arithmetical operations on rational numbers throughout this chapter.

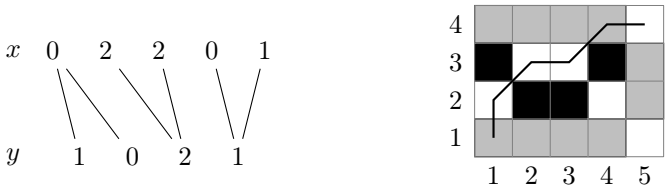


Figure 7.2: Example of two time series $x = (0, 2, 2, 0, 1)$ and $y = (1, 0, 2, 1)$. Left: The lines indicate the alignment according to the (optimal) warping path $p = ((1,1),(1,2),(2,3),(3,3),(4,4),(5,4)) \in \mathcal{P}_{5,4}$ of length six. The corresponding aligned time series are $x' = (0, 0, 2, 2, 0, 1)$ and $y' = (1, 0, 2, 2, 1, 1)$. Right: The warping path p (thick line) can be illustrated in a 5×4 matrix where the columns correspond to indices of x and the rows to indices of y. The color of cell (i, j) indicates the value $(x_i - y_j)^2$ (darker means larger). The cost is $C_p(x, y) = \|x' - y'\|_2^2 = (0-1)^2 + (0-0)^2 + (2-2)^2 + (2-2)^2 + (0-1)^2 + (1-1)^2 = 2$, which is optimal for this example, hence $\mathrm{dtw}(x, y) = 2$.

We proceed with introducing dynamic time warping. The next definition is fundamental for the central computational problem of this chapter (see Figure 7.2 for an illustrating example).

Definition 7.1 (Warping Path)**.** A *warping path of order* $m \times n$ is a sequence $p = (p_1, \ldots, p_L)$ of pairs $p_\ell = (i_\ell, j_\ell) \in [m] \times [n]$ for all $\ell \in [L]$, $L \geq 1$, such that

(i) $p_1 = (1, 1)$,
(ii) $p_L = (m, n)$, and
(iii) $p_{\ell+1} - p_\ell := (i_{\ell+1} - i_\ell, j_{\ell+1} - j_\ell) \in \{(1,0),(0,1),(1,1)\}$ for all $\ell \in [L-1]$.

Clearly, $\max\{m, n\} \leq L \leq m + n$. We denote the set of all warping paths of order $m \times n$ by $\mathcal{P}_{m,n}$. For two time series $x = (x_1, \ldots, x_m)$ and $y = (y_1, \ldots, y_n)$, a warping path $p = (p_1, \ldots, p_L) \in \mathcal{P}_{m,n}$ defines an *alignment* of x and y as follows: Each pair $p_\ell = (i_\ell, j_\ell)$, $\ell \in [L]$, of p aligns element x_{i_ℓ} with y_{j_ℓ}. Aligning x and y along a warping path p yields two *aligned* time series $(x_{i_1}, \ldots, x_{i_L})$ and $(y_{j_1}, \ldots, y_{j_L})$ of length L. The *cost* $C_p(x, y)$ for aligning x and y along a warping path p is defined as

$$C_p(x, y) := \sum_{\ell=1}^{L} (x_{i_\ell} - y_{j_\ell})^2.$$

Now, the *dtw-distance* between x and y is defined as

$$\mathrm{dtw}(x,y) := \min_{p \in \mathcal{P}_{m,n}} \left\{ \sqrt{C_p(x,y)} \right\}.$$

A warping path p with $C_p(x,y) = (\mathrm{dtw}(x,y))^2$ is called an *optimal* warping path for x and y.

It is well-known that the dtw-distance of two time series of length n can be computed in $O(n^2)$ time by a simple dynamic program [SC78]. Recently, the running time has been improved to subquadratic time $O(n^2 \log\log\log n / \log\log n)$ [GS17]. There is no strongly subquadratic-time algorithm (that is, $O(n^{2-\epsilon})$ for some $\epsilon > 0$) unless the Strong Exponential Time Hypothesis fails [BK15].

Our central problem DTW-Mean is defined as follows.

DTW-Mean
Input: Sample $\mathcal{X} = \left(x^{(1)}, \ldots, x^{(k)}\right)$ of k univariate rational time series.
Task: Find a univariate rational time series z that minimizes the Fréchet function $F(z)$.

It is known that a mean always exists [JS16, Proposition 2.10] and that there always exists a mean of length at most $\sum_{i=1}^{k} n_i - 2(k-1)$ [JS16, Theorem 2.7], where $n_i \geq 2$ denotes the length of x_i. Also, a mean of rational time series is itself rational [SJ18, Theorem 3.3].

7.3 Problematic Statements in the Literature

In this section, we discuss some misleading and wrong claims in the literature concerning the computational complexity of DTW-Mean.

NP-Hardness. The DTW-Mean problem is often described as being closely related to the Steiner String (STS) problem [ASW15, HNF08, Pet+16, PG12, PG17, PKG11]. A *Steiner string* [Gus97] (or *Steiner sequence*) for a set S of strings is a string s^* that minimizes $\sum_{s \in S} D(s^*, s)$, where D is a distance measure between two strings, for example the *weighted edit distance* (often assumed to be a metric). Computing a Steiner string is related to solving the Multiple Sequence Alignment (MSA) problem in bioinformatics [Gus97]. Both, STS and MSA, are known to be NP-hard for several (metric) distance measures even on binary alphabets [BD01, NR05]. Some papers mention the NP-hardness of MSA or STS in the context of DTW-Mean [HNF08, Pet+16, PG17], probably

Figure 7.3: Example of two time series $x_1 = (1, 1, 1, 1, 1)$, $x_2 = (0, 0, 0, 0, 0)$ showing that the multiple alignment approach does not yield a mean (warping paths are indicated by straight lines). Left: The unique optimal warping path between x_1 and x_2 clearly is $p = ((1, 1), (2, 2), \ldots, (5, 5))$, which yields a column-wise average series $\bar{x} = (0.5, 0.5, 0.5, 0.5, 0.5)$ with $F(\bar{x}) = \frac{1}{2} \cdot 10 \cdot 0.5^2 = 1.25$. Right: A mean of x_1 and x_2 is $z = (0.8, 0.2)$ with $F(z) = \frac{1}{2}(2 \cdot 0.8^2 + 8 \cdot 0.2^2) = 0.8$.

as a justification to use heuristic approaches for DTW-Mean. However, it is not clear (and not shown in the literature) how to reduce from MSA (or STS) to DTW-Mean since the involved distance measures are significantly different (for example, the dtw-distance lacks the metric properties of the triangle inequality and also the identity of indiscernibles). Interestingly, as we show in Theorem 7.3, DTW-Mean is solvable in polynomial time for binary time series, which contrasts the NP-hardness of STS and MSA on binary data. A formal proof of NP-hardness for DTW-Mean has recently been given by Bulteau et al. [BFN18].

Computation of Exact Solutions. It is claimed that DTW-Mean can be solved by averaging a (global) multiple alignment of the k input series [HNF08, Pet+16, PG12, PKG11]. A multiple alignment of k time series in the context of dynamic time warping is described as computing a k-dimensional warping path in a k-dimensional matrix. This approach may have derived from the (misleading) claimed relation to MSA. Concerning the running time, it is claimed that computing a multiple alignment requires $\Theta(n^k)$ time [PG12, PKG11] ($O(n^k)$ time is claimed by Petitjean et al. [Pet+16]), where n is the maximum length of an input time series. Neither the upper bound of $O(n^k)$ nor the lower bound of $\Omega(n^k)$ on the running time are formally proven. Given a multiple alignment, it is claimed that averaging the k resulting aligned time series column-wise yields a mean [PG12]. This approach is not correct even for two time series. Note that for two time series, the multiple alignment is obtained by aligning along an optimal warping path. However, the column-wise average of

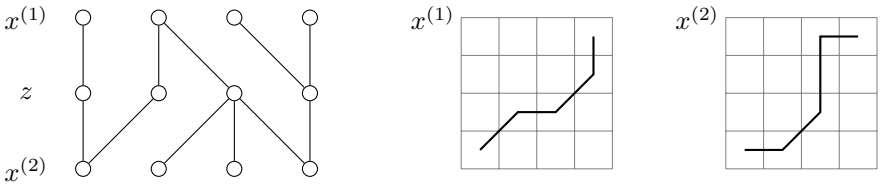


Figure 7.4: Illustration of three length-4 time series $z, x^{(1)}, x^{(2)}$ with warping paths between z and $x^{(1)}$ and between z and $x^{(2)}$ indicated by straight lines (left) and by the two matrices (right), where z corresponds to the columns. The elements z_2 and z_3 are redundant. For example, z_3 is aligned to $x_2^{(1)}$ which is also aligned to z_2, and z_3 is aligned to $x_4^{(2)}$ which is also aligned to z_4 (in both matrices the warping path contains a horizontal segment including the third column). Note that removing z_2 (and the corresponding entries in the warping pahts) yields a shorter mean without increasing the value of the Fréchet function.

two aligned time series obtained from an optimal warping path is not always an optimal solution as a simple example in Figure 7.3 shows.

7.4 An XP Algorithm for the Number of Input Series

We develop a nontrivial exponential-time algorithm showing that DTW-Mean parameterized by the number k of input time series is in XP. The key is to observe a certain structure of the alignments between a mean and the corresponding input time series. To this end, we define *redundant* elements in a mean. Note that this concept was already used by Jain and Schultz [JS16] in order to prove the existence of a mean of bounded length [JS16, Theorem 2.7] (though [JS16, Definition 3.20] is slightly different).

Definition 7.2. Let $x^{(1)}, \ldots, x^{(k)}$ and z be time series and let $p^{(j)}$, $j \in [k]$ denote a warping path between $x^{(j)}$ and z. We call an element z_i of z *redundant* (with respect to the warping paths $p^{(j)}$) if in every time series $x^{(j)}$ there exists an element that is aligned by $p^{(j)}$ with z_i and with another element of z.

Figure 7.4 illustrates Definition 7.2. The next lemma states that there always exists a mean without redundant elements. The idea is that a redundant mean

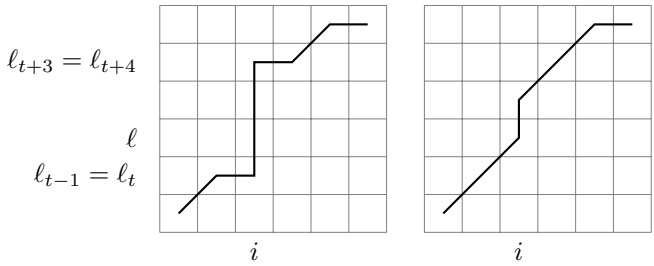


Figure 7.5: Left: Example of a warping path containing the pairs $(i-1, \ell_{t-1}), (i, \ell_t), \ldots, (i, \ell_{t+3}), (i+1, \ell_{t+4})$, that is, $\alpha = 3$. The vertical segment in column i contains the four indices $\ell_t, \ldots, \ell_{t+3}$ that are aligned to i. Note that ℓ_{t+1} and ℓ_{t+2} are only aligned to i, whereas ℓ_t and ℓ_{t+3} are also aligned to other indices (horizontal segments). Hence, element z_i is redundant. Right: Removing the pairs (i, ℓ_t) and (i, ℓ_{t+3}) yields a warping path such that z_i is not redundant without increasing the cost.

element can either be completely removed or that the warping paths can be modified such that the costs do not increase. The proof is implicitly contained in the proof of [JS16, Theorem 2.7]. For the sake of clarity, however, we give an explicit proof here.

Lemma 7.1. *There exist a mean z for time series $x^{(1)}, \ldots, x^{(k)}$ and optimal warping paths $p^{(j)}$ between z and $x^{(j)}$ for each $j \in [k]$ such that z contains no redundant element.*

Proof. Let $z = (z_1, \ldots, z_q)$ be a mean of $x^{(1)}, \ldots, x^{(k)}$ with optimal warping paths $p^{(j)}$, $j \in [k]$, such that the element z_i is redundant. Note that by [JS16, Proposition 2.10] a mean z always exists. We show that there also exists a mean z' with optimal warping paths $p^{(j)'}$ such that no element in z' is redundant.

Case 1. There exists a $j \in [k]$ such that the element z_i is aligned by $p^{(j)}$ with at least one element $x^{(j)}_\ell$ in $x^{(j)}$ that is not aligned with any other element in z. Then, $p^{(j)}$ is of the form

$$p^{(j)} = (p_1, \ldots, (i-1, \ell_{t-1}), (i, \ell_t), \ldots, (i, \ell_{t+\alpha}), (i+1, \ell_{t+\alpha+1}), \ldots, p_{L_j})$$

for some $\ell_t \leq \ell \leq \ell_{t+\alpha}$ with $t \in [L_j - \alpha]$, $\alpha \geq 1$. Since z_i is redundant, it follows that $\ell_{t-1} = \ell_t$ or $\ell_{t+\alpha} = \ell_{t+\alpha+1}$ holds (see Figure 7.5). If $\ell_{t-1} = \ell_t$, then we remove the pair (i, ℓ_t) from $p^{(j)}$. Also, if $\ell_{t+\alpha} = \ell_{t+\alpha+1}$, then we remove the

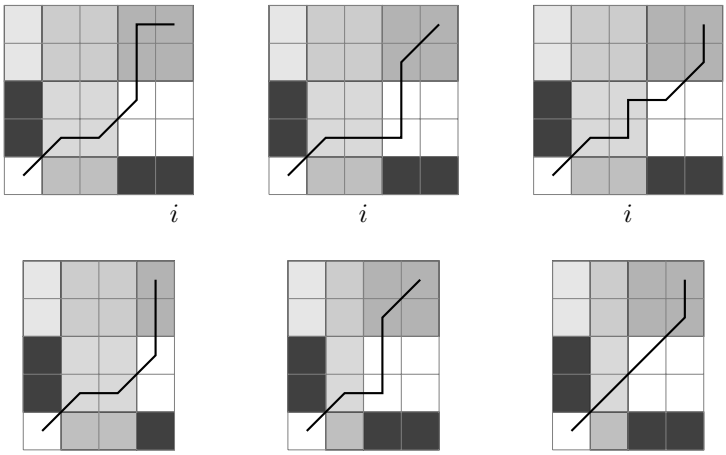


Figure 7.6: Examples of warping paths between z (corresponding to columns) and $x^{(j)}$ (corresponding to rows), where element z_i is redundant (top row) and the resulting warping paths $p^{(j)'}$ for z', where z_i is removed (bottom row). The costs do not increase. The case $i = q$ is shown left, the case $i < q$ and $i_{t+1} = i + 1$ is shown in the middle, and the case $i < q$ and $i_{t+1} = i$ is shown right.

pair $(i, \ell_{t+\alpha})$ from $p^{(j)}$. Note that this yields a warping path $p^{(j)'}$ between z and $x^{(j)}$ since even if we removed both pairs (i, ℓ_t) and $(i, \ell_{t+\alpha})$, then we know by assumption that there still exists the pair (i, ℓ) with $\ell_t < \ell < \ell_{t+\alpha}$ in $p^{(j)'}$ since z_i is aligned with $x_\ell^{(j)}$ which is not aligned with another element in z. Since we only removed pairs from $p^{(j)}$, it holds $C_{p^{(j)'}}(z, x^{(j)}) \leq C_{p^{(j)}}(z, x^{(j)})$. Moreover, z_i is not redundant anymore.

Case 2. For all $j \in [k]$, z_i is aligned only with elements in $x^{(j)} = (x_1^{(j)}, \ldots, x_{n_j}^{(j)})$ which are also aligned with another element of z by $p^{(j)}$.

Let $z' = (z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_q)$ denote the time series obtained by deleting the element z_i from z. See Figure 7.6 for some examples illustrating the following cases.

If $i = q$, then each $p^{(j)}$ has the form $((1,1), \ldots, (q-1, n_j), (q, n_j))$. We define the new warping path $p^{(j)'} = ((1,1), \ldots, (q-1, n_j))$. Clearly, this is a valid warping path between z' and $x^{(j)}$ and we have

$$C_{p^{(j)'}}(z', x^{(j)}) = C_{p^{(j)}}(z, x^{(j)}) - (z_q - x_{n_j}^{(j)})^2 \leq C_{p^{(j)}}(z, x^{(j)}).$$

Hence, z' is a mean with less redundant elements.

If $i \in [q-1]$, then consider a warping path $p^{(j)} = (p_1, \ldots, p_{L_j})$ and let $p_t = (i_t, \ell_t)$, $t \in [L_j - 1]$ be the first pair such that $i_t = i$.

If $i_{t+1} = i + 1$, then $\ell_t = \ell_{t-1}$ or $\ell_t = \ell_{t+1}$ holds (or both). We define the new warping path $p^{(j)'} = (p1, \ldots, p_{t-1}, (i_{t+1} - 1, \ell_{t+1}), \ldots, (i_{L_j} - 1, \ell_{L_j}))$. Note that $p^{(j)'}$ is a warping path between z' and $x^{(j)}$ since $i_{t+1} - 1 - i_{t-1} = i - (i-1) = 1$ and $\ell_{t+1} - \ell_{t-1} \leq 1$. The cost is

$$C_{p^{(j)'}}(z', x^{(j)}) = C_{p^{(j)}}(z, x^{(j)}) - (z_i - x^{(j)}_{\ell_t})^2 \leq C_{p^{(j)}}(z, x^{(j)}).$$

Hence, z' is a mean with less redundant elements.

If $i_{t+1} = i$, then it follows $1 < t < L_j - 1$ and $\ell_{t-1} = \ell_t = \ell_{t+1} - 1 = \ell_{t+2} - 1$ and $i_{t+2} = i + 1$. We define the warping path $p^{(j)'} = (p1, \ldots, p_{t-1}, (i_{t+2} - 1, \ell_{t+2}), \ldots, (i_{L_j} - 1, \ell_{L_j}))$. Clearly, $p^{(j)'}$ is a warping path between z' and $x^{(j)}$ since $(i_{t+2} - 1, \ell_{t+2}) - (i_{t-1}, \ell_{t-1}) = (1, 1)$. The cost is

$$C_{p^{(j)'}}(z', x^{(j)}) = C_{p^{(j)}}(z, x^{(j)}) - (z_i - x^{(j)}_{\ell_t})^2 - (z_i - x^{(j)}_{\ell_{t+1}})^2 \leq C_{p^{(j)}}(z, x^{(j)}),$$

that is, z' is a mean with less redundant elements.

In both cases above, we reduced the number of redundant elements. In Case 1, we found another warping path (by removing horizontal segments in the picture) such that z_i is not redundant. In Case 2, we removed element z_i and constructed new warping paths (by cutting out horizontal segments in the picture). Hence, we can repeat the above arguments until we obtain a mean z' without redundant elements. □

Lemma 7.1 allows us to devise a dynamic program to compute a mean. The general scheme behind the dynamic program is to test all possibilities to align the last mean element to elements from the input time series while recursively adding an optimal solution for the remaining non-aligned elements in the input time series. The fact that we can assume the last mean element not to be redundant is crucial for this recursive approach, which is described in the proof of the following theorem.

Theorem 7.2. DTW-Mean *for k input time series is solvable in $O(n^{2k+1}2^k k)$ time, where n is the maximum length of all input time series.*

Proof. Assume for simplicity that all time series have length n (the general case can be solved analogously). We find a mean using a dynamic programming

approach. Let C be a k-dimensional table, where for all $(i_1, \ldots, i_k) \in [n]^k$, we define

$$C[i_1, \ldots, i_k] = \min_{z \in \mathcal{T}} \left(\frac{1}{k} \sum_{j=1}^{k} \left(\mathrm{dtw}(z, (x_1^{(j)}, \ldots, x_{i_j}^{(j)})) \right)^2 \right),$$

that is, $C[i_1, \ldots, i_k]$ is the value $F(z)$ of the Fréchet function of a mean z for the subseries $(x_1^{(1)}, \ldots, x_{i_1}^{(1)}), \ldots, (x_1^{(k)}, \ldots, x_{i_k}^{(k)})$. Clearly, $C[n, \ldots, n]$ yields the optimal value $F(z)$ of the input instance.

For $i_1 = i_2 = \ldots = i_k = 1$, it is clear that there exists a mean z containing just one element and each optimal warping path between z and $(x_1^{(j)})$ trivially equals $((1,1))$. Since it can be shown [SJ18, Theorem 3.3] that, given optimal warping paths, each element of a mean equals the arithmetic mean of all elements in all time series that are aligned to this element, we initialize

$$\mu = 1/k \sum_{j=1}^{k} x_1^{(j)}, \qquad C[1, \ldots, 1] = \frac{1}{k} \sum_{j=1}^{k} (x_1^{(j)} - \mu)^2.$$

Note that the corresponding mean is $z = (\mu)$.

For the case that $i_j > 1$ holds for at least one $j \in [k]$, we can assume by Lemma 7.1 that there exist a mean z for $(x_1^{(1)}, \ldots, x_{i_1}^{(1)}), \ldots, (x_1^{(k)}, \ldots, x_{i_k}^{(k)})$ and optimal warping paths $p^{(j)}$ between z and $(x_1^{(j)}, \ldots, x_{i_j}^{(j)})$ such that z contains no redundant elements. Let z_q be the last element of z. Then, for each $j \in [k]$, z_q is aligned by $p^{(j)}$ with some elements $x_{\ell_j}^{(j)}, \ldots, x_{i_j}^{(j)}$ for $\ell_j \in [i_j]$. Again, z_q is the arithmetic mean of all elements in all time series with which it is aligned [SJ18, Theorem 3.3]. Hence, the contribution of z_q to $F(z)$ can easily be determined (see the formula for $\sigma(\ell_1, \ldots, \ell_k)$ below).

Now, consider the case that there exists another element z_{q-1} in z. Clearly, for each $j \in [k]$, z_{q-1} is aligned only to elements with indices at most ℓ_j since otherwise the warping path conditions are violated. Hence, the remaining part of $F(z)$ can be obtained recursively from a mean of the time series $(x_1^{(1)}, \ldots, x_{\ell_1}^{(1)}), \ldots, (x_1^{(k)}, \ldots, x_{\ell_k}^{(k)})$. Recall, however, that we assumed z not to contain any redundant element. It follows that z_{q-1} cannot be aligned with $x_{\ell_j}^{(j)}$ for all $j \in [k]$ since z_q is already aligned with each $x_{\ell_j}^{(j)}$. Therefore, we add the minimum value $C[\ell'_1, \ldots, \ell'_k]$ over all $\ell'_j \in \{\max(1, \ell_j - 1), \ell_j\}$ such that $\ell'_j = \ell_j - 1$ holds for at least one $j \in [k]$.

Hence, the following recursion holds:

$$C[i_1,\ldots,i_k] = \min\{c^*(\ell_1,\ldots,\ell_k) + \sigma(\ell_1,\ldots,\ell_k) \mid \ell_1 \in [i_1],\ldots,\ell_k \in [i_k]\},$$

where

$$\sigma(\ell_1,\ldots,\ell_k) = \frac{1}{k}\sum_{j=1}^{k}\sum_{t=\ell_j}^{i_j}(x_t^{(j)} - \mu)^2, \qquad \mu = \frac{\sum_{j=1}^{k}\sum_{t=\ell_j}^{i_j} x_t^{(j)}}{\sum_{j=1}^{k}(i_j - \ell_j + 1)},$$

and we define

$$c^*(\ell_1,\ldots,\ell_k) = \min\{C[\ell'_1,\ldots,\ell'_k] \mid \ell'_j \in \{\max(1,\ell_j-1),\ell_j\}, \sum_{j=1}^{k}(\ell_j - \ell'_j) > 0\}$$

if $\ell_j > 1$ holds for some $j \in [k]$. We set $c^*(1,\ldots,1) := 0$.

For the value $C[i_1,\ldots,i_k]$, the minimum is computed over all possible choices $\ell_j \in [i_j]$, $j \in [k]$. For each choice, the arithmetic mean μ corresponds to the element of the mean and $\sigma(\ell_1,\ldots,\ell_k)$ equals the induced cost for aligning this element with $x_{\ell_j}^{(j)},\ldots,x_{i_j}^{(j)}$ for each $j \in [k]$. The value $c^*(\ell_1,\ldots,\ell_k)$ recursively yields the value $F(z')$ of a mean z' for the remaining subseries $(x_1^{(j)},\ldots,x_{\ell'_j}^{(j)})$ over all $\ell'_j \in \{\max(1,\ell_j-1),\ell_j\}$ such that $\sum_{j=1}^{k}(\ell_j - \ell'_j) > 0$, which implies that $\ell'_j = \ell_j - 1$ holds for at least one $j \in [k]$. This condition guarantees that we only find warping paths such that z_q is not redundant (which we can assume due to Lemma 7.1). Note that $\ell_j = 1$ implies that $\ell'_j = 1$ (since index 0 does not exist in $x^{(j)}$).

The dynamic programming table C can be filled iteratively along the dimensions starting from $C[1,\ldots,1]$. The overall number of entries is n^k. For each table entry, the minimum of a set containing $O(n^k)$ elements is computed. Computing an element requires the computation of $\sigma(\ell_1,\ldots,\ell_k)$ which can be done in $O(kn)$ time plus the computation of $c^*(\ell_1,\ldots,\ell_k)$ which is the minimum of a set of size at most 2^k whose elements can be obtained by constant-time table look-ups. Thus, the table C can be filled in $O(n^k \cdot n^k \cdot 2^k \cdot kn)$ time. A mean can be obtained by storing the values μ for which the minimum in the above recursion is attained (Algorithm 7.1 contains the pseudocode). □

We close with some remarks on Theorem 7.2.

Algorithm 7.1: Dynamic Program solving DTW-MEAN

Input: Time series $x^{(1)}, \ldots, x^{(k)}$ of lengths $n_1, \ldots, n_k$.
Output: Mean z and $F(z)$.

Initialize C // k-dimensional DP table storing F-values
Initialize Z // k-dimensional table storing means
foreach $(i_1, \ldots, i_k) \in [n_1] \times \ldots \times [n_k]$ **do** // fill tables iteratively
  $C[i_1, \ldots, i_k] := \infty$
  $Z[i_1, \ldots, i_k] := ()$
  foreach $(\ell_1, \ldots, \ell_k) \in [i_1] \times \ldots \times [i_k]$ **do** // compute $C[i_1, \ldots, i_k]$
    $\mu := (\sum_{j=1}^{k} \sum_{t=\ell_j}^{i_j} x_t^{(j)}) / \sum_{j=1}^{k} (i_j - \ell_j + 1)$
    $\sigma := \frac{1}{k} \sum_{j=1}^{k} \sum_{t=\ell_j}^{i_j} (x_t^{(j)} - \mu)^2$
    $c^* := \infty$
    $z := ()$
    if $\ell_1 = \ell_2 = \ldots = \ell_k = 1$ **then** // $c^*(1, \ldots, 1)$ is defined as 0
      $c^* := 0$
    else // compute $c^*(\ell_1, \ldots, \ell_k)$ based on table look-ups
      foreach $(\ell'_1, \ldots, \ell'_k) \in \{\ell_1 - 1, \ell_1\} \times \ldots \times \{\ell_k - 1, \ell_k\}$ **do**
        if $\forall j \in [k] : \ell'_j \geq 1$ **and** $\exists j \in [k] : \ell'_j < \ell_j$ **then**
          if $C[\ell'_1, \ldots, \ell'_k] < c^*$ **then**
            $c^* := C[\ell'_1, \ldots, \ell'_k]$
            $z := Z[\ell'_1, \ldots, \ell'_k]$
    if $c^* + \sigma < C[i_1, \ldots, i_k]$ **then** // update mean and F-value
      $C[i_1, \ldots, i_k] := c^* + \sigma$
      $Z[i_1, \ldots, i_k] := \texttt{append}(z, \mu)$
return $(Z[n_1, \ldots, n_k], C[n_1, \ldots, n_k])$

- Our dynamic programming approach also allows to compute all means of finite length (without identical successive elements) by storing all possible values for which the minimum in the recursion is attained. Since a mean does not have to be unique in general, this is interesting in practice in order to select the most suitable mean for the task at hand.
- It is possible to incorporate a maximum mean length q into the dynamic program such that it outputs only optimal solutions among those of length

at most q. The running time increases by a factor of q^2. This is useful in practice, for example, when a mean is desired not to be longer than any input time series (which might be unnatural).
- Many applications in medicine, finance, or multimedia involve multivariate time series (that is, measuring several signals over time) [GŁ15]. The algorithm can easily be extended to multivariate time series with elements in $\mathbb{Q}^d$ with a running time increase by a factor of d.

7.5 Polynomial-Time Solvability for Binary Data

In this section, we consider the special case of DTW-Mean where the input time series are binary (that is, containing only 0's and 1's). This case naturally occurs in applications involving status-based information such as *on/off*, *active/inactive*, or *open/closed* [Mue+16]. In this scenario, a mean might be desired to also contain only 0's and 1's since arbitrary rational values might not always be meaningful in the respective context. Hence, we define the following problem.

Binary DTW-Mean
Input: Sample $\mathcal{X} = \left(x^{(1)}, \ldots, x^{(k)}\right)$ of k time series with elements in $\{0, 1\}$.
Task: Find a time series $z \in \{0, 1\}^*$ that minimizes $F(z)$.

We prove that Binary DTW-Mean is polynomial-time solvable.

Theorem 7.3. Binary DTW-Mean *for k input time series is solvable in $O(kn^3)$ time, where n is the maximum length of all input time series.*

To show polynomial-time solvability of Binary DTW-Mean, we first prove some auxiliary results about the dtw-distance of binary time series and properties of a binary mean. More specifically, we show that a mean can always be assumed to be an alternating sequence of 0's and 1's of length at most $n + 1$. Hence, we only have to check $O(n)$ possible candidate time series.

We start with the following general definition.

Definition 7.3. A time series $x = (x_1, \ldots, x_n)$ is *condensed* if no two consecutive elements are equal, that is, $x_i \neq x_{i+1}$ holds for all $i \in [n-1]$. We denote the *condensation* of a time series x by $\tilde{x}$ and define it to be the time series obtained by repeatedly removing one of two equal consecutive elements in x until the remaining series is condensed.

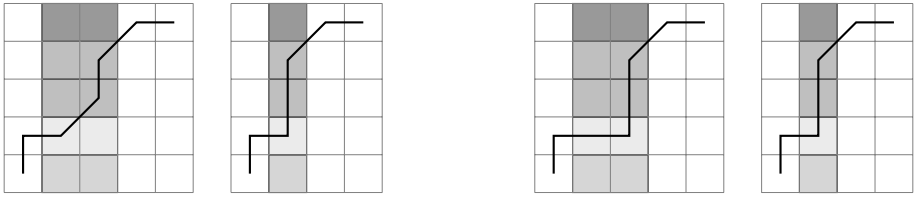


Figure 7.7: Illustration of two warping paths (a) and (c) between two time series x (corresponding to columns) and y (corresponding to rows), where x is not condensed. The second and third element of x are equal as indicated by the same colors in the corresponding columns. In both cases it is possible to delete the third element from x and to find a warping path without increasing the cost. The warping path (b) has the same length and cost as (a), whereas warping path (d) is shorter than (c) and might have decreased cost.

The following proposition states that the dtw-distance of a time series x to any other time series is at least the dtw-distance of the condensation $\tilde{x}$ to that time series. This implies that a mean can always be assumed to be condensed, since the Fréchet function of the condensation of a mean does not increase. Note that this holds for arbitrary time series (not only for the binary case) and might thus also be useful in designing better heuristics for the general case.

Proposition 7.4. *Let x be a time series and let $\tilde{x}$ denote its condensation. Then, for every time series y, it holds that* $\mathrm{dtw}(\tilde{x}, y) \leq \mathrm{dtw}(x, y)$.

Proof. Let y have length m and assume that $x = (x_1, \ldots, x_n)$ is not condensed. Then, $x_i = x_{i+1}$ holds for some $i \in [n-1]$. Let $p = ((i_1, j_1), \ldots, (i_L, j_L))$ be an optimal warping path for x and y. Now, consider the time series $x' = (x_1, \ldots, x_i, x_{i+2}, \ldots, x_n)$ that is obtained from x by deleting element x_{i+1}. We construct a warping path p' for x' and y such that $C_{p'}(x', y) \leq C_p(x, y)$ (see Figure 7.7 for examples).

To this end, let $p_a = (i_a, j_a)$, $2 \leq a \leq L$, be the first index pair in p where $i_a = i + 1$ (hence, $i_{a-1} = i$). Now, we consider two cases.

Case 1. If $j_a = j_{a-1} + 1$ (see illustration in Figure 7.7 (a)), then we define the order-$((n-1) \times m)$ warping path

$$p' := ((i_1, j_1), \ldots, (i_{a-1}, j_{a-1}), (i_a - 1, j_a), (i_{a+1} - 1, j_{a+1}), \ldots, (i_L - 1, j_L))$$

of length L. Thus, each element of y that was aligned to x_{i+1} in p is now aligned to x_i instead (see illustration in Figure 7.7 (b)). To check that p' is a valid warping path, note first that $(i_1, j_1) = (1, 1)$ and $(i_L - 1, j_L) = (n - 1, m)$ holds since p is a warping path. For all $\ell \in [a-2]$, it holds $(i_{\ell+1}, j_{\ell+1}) - (i_\ell, j_\ell) \in \{(1,0), (0,1), (1,1)\}$ since p is a warping path. Moreover, $(i_a - 1, j_a) - (i_{a-1}, j_{a-1}) = (0, 1)$. Finally, for all ℓ with $a \leq \ell \leq L-1$, we have

$$(i_{\ell+1} - 1, j_{\ell+1}) - (i_\ell - 1, j_\ell) = (i_{\ell+1} - i_\ell, j_{\ell+1} - j_\ell) \in \{(1,0), (0,1), (1,1)\}$$

since p is a warping path. The cost of p' is

$$\begin{aligned} C_{p'}(x', y) &= \sum_{\ell=1}^{a-1} (x'_{i_\ell} - y_{j_\ell})^2 + \sum_{\ell=a}^{L} (x'_{(i_\ell - 1)} - y_{j_\ell})^2 \\ &= \sum_{\ell=1}^{a-1} (x_{i_\ell} - y_{j_\ell})^2 + \sum_{\ell=a}^{L} (x_{i_\ell} - y_{j_\ell})^2 = C_p(x, y). \end{aligned}$$

Case 2. If $j_a = j_{a-1}$ (Figure 7.7 (c)), then we define the order-$((n-1) \times m)$ warping path

$$p' := ((i_1, j_1), \ldots, (i_{a-1}, j_{a-1}), (i_{a+1} - 1, j_{a+1}), \ldots, (i_L - 1, j_L))$$

of length $L-1$, where the pair (i_a, j_a) is removed and all elements of y that were aligned to x_{i+1} are now aligned to x_i (Figure 7.7 (d)). Again, $(i_1, j_1) = (1, 1)$ and $(i_L - 1, j_L) = (n - 1, m)$ holds since p is a warping path. Clearly, for each $\ell \in [a-2]$, it holds $(i_{\ell+1}, j_{\ell+1}) - (i_\ell, j_\ell) \in \{(1,0), (0,1), (1,1)\}$ since p is warping path. Further, we have

$$(i_{a+1} - 1, j_{a+1}) - (i_{a-1}, j_{a-1}) = (i_{a+1} - i_a, j_{a+1} - j_a) \in \{(1,0), (0,1), (1,1)\}$$

since p is a warping path. Finally, for all ℓ with $a+1 \leq \ell \leq L-1$, it holds

$$(i_{\ell+1} - 1, j_{\ell+1}) - (i_\ell - 1, j_\ell) = (i_{\ell+1} - i_\ell, j_{\ell+1} - j_\ell) \in \{(1,0), (0,1), (1,1)\}$$

since p is a warping path. Thus, p' is a valid warping path and its cost is

$$\begin{aligned} C_{p'}(x', y) &= \sum_{\ell=1}^{a-1} (x'_{i_\ell} - y_{j_\ell})^2 + \sum_{\ell=a+1}^{L} (x'_{(i_\ell - 1)} - y_{j_\ell})^2 \\ &= \sum_{\ell=1}^{a-1} (x_{i_\ell} - y_{j_\ell})^2 + \sum_{\ell=a+1}^{L} (x_{i_\ell} - y_{j_\ell})^2 \\ &= C_p(x, y) - (x_{i_a} - y_{j_a})^2 \leq C_p(x, y). \end{aligned}$$

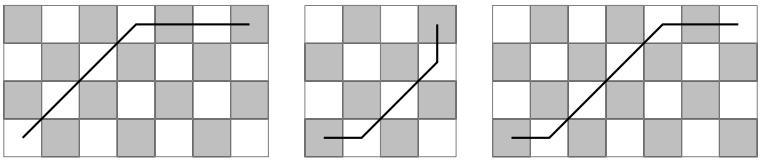


Figure 7.8: Examples of warping paths having the cost claimed in Lemma 7.5 for the three cases of binary condensed time series x and y. Gray cells in a matrix indicate index pairs (i, j) with $(x_i - y_j)^2 = 1$. The case $x_1 = y_1$ is depicted left. The other examples depict the case that $x_1 \neq y_1$. The case that x and y have equal length is shown in the middle. Note that the shown warping paths are optimal since in each case every warping path contains at least two gray cells.

Since in both cases above, the cost does not increase, we obtain

$$\mathrm{dtw}(x', y) \leq C_{p'}(x', y) \leq C_p(x, y) = \mathrm{dtw}(x, y).$$

Repeating this argument until x' is condensed finishes the proof. □

Proposition 7.4 implies that we can assume a mean to be condensed. Next, we are going to prove an upper bound of $n + 1$ on the length of a binary mean. To this end, we analyze the dtw-distances of binary time series. Note that a binary condensed time series is completely determined by its first element and its length. We use this property to give a closed expression for the dtw-distance of two condensed binary time series.

Lemma 7.5. *Let $x = (x_1, \ldots, x_n)$, $y = (y_1, \ldots, y_m) \in \{0, 1\}^*$ be two condensed binary time series with $n \geq m$. Then, it holds that*

$$\mathrm{dtw}(x, y)^2 = \begin{cases} \lceil (n-m)/2 \rceil, & x_1 = y_1 \\ 2, & x_1 \neq y_1 \wedge n = m \\ 1 + \lfloor (n-m)/2 \rfloor, & x_1 \neq y_1 \wedge n > m \end{cases}.$$

Proof. We prove the statement by first giving a warping path that has the claimed cost and second proving that every warping path has at least the claimed cost.

"$\leq$": We show that there exists a warping path p between x and y that has the claimed cost (see Figure 7.8 for examples). The warping path p is defined as follows:

If $x_1 = y_1$, then we have $x_i = y_i$ for all $i \in [m]$ (since x and y are condensed and binary) and we set

$$p := ((1,1),(2,2),\ldots,(m,m),(m+1,m),\ldots,(n,m)).$$

This warping path has cost $C_p(x,y) = \sum_{i=m+1}^{n}(x_i - y_m)^2 = \lceil (n-m)/2 \rceil$.

If $x_1 \neq y_1$, then we have $x_i = y_{i-1}$ for all $2 \leq i \leq m$. Thus, for $n = m$, the warping path

$$p := ((1,1),(2,1),(3,2),\ldots,(n,m-1),(n,m))$$

has cost $C_p(x,y) = (x_1 - y_1)^2 + (x_n - y_m)^2 = 2$, Finally, for $n > m$, the warping path

$$p := ((1,1),(2,1),(3,2),\ldots,(m+1,m),(m+2,m),\ldots,(n,m))$$

yields cost

$$\begin{aligned} C_p(x,y) &= (x_1 - y_1)^2 + \sum_{i=2}^{m+1}(x_i - y_{i-1})^2 + \sum_{i=m+2}^{n}(x_i - y_m)^2 \\ &= 1 + \sum_{i=m+2}^{n}(x_i - y_m)^2 = 1 + \lceil (n-m-1)/2 \rceil = 1 + \lfloor (n-m)/2 \rfloor. \end{aligned}$$

"$\geq$": We show that every warping path has at least the cost claimed above. Consider an optimal warping path $p = (p_1,\ldots,p_L)$ for x and y and note that, since $n \geq m$, there are at least $n - m$ different indices $\ell_1,\ldots,\ell_{n-m}$ with

$$1 \leq \ell_1 < \ldots < \ell_{n-m} \leq L - 1$$

such that $p_{\ell_t+1} - p_{\ell_t} = (1,0)$ for every $t \in [n-m]$. For such an index ℓ_t, let $p_{\ell_t+1} = (i,j)$ and $p_{\ell_t} = (i',j')$ with $i = i' + 1$ and $j = j'$. Then, we have

$$(x_i - y_j)^2 + (x_{i'} - y_{j'})^2 = (x_i - y_j)^2 + (x_{i'} - y_j)^2 = 1$$

since $x_i \neq x_{i'}$ (recall that x is condensed and binary). Hence, for each $t \in \{1,\ldots,\lceil (n-m)/2 \rceil\}$, the index ℓ_{2t-1} contributes a cost of 1 to the sum of $C_p(x,y)$. Thus, $\mathrm{dtw}(x,y)^2 \geq \lceil (n-m)/2 \rceil$.

If $x_1 = y_1$, then this lower bound matches the claimed cost. If $x_1 \neq y_1$ and $n = m$, then also $x_n \neq y_m$ and hence, every warping path has cost at

least 2. Finally, consider the case that $x_1 \neq y_1$ and $n > m$. If $n - m$ is odd, then $\lceil (n-m)/2 \rceil = 1 + \lfloor (n-m)/2 \rfloor$ and hence, the above lower bound matches the claimed cost. If $n - m$ is even, then $x_n \neq y_m$. Consider the indices $\ell_1, \ldots, \ell_{n-m}$ from above implying the lower bound of $\lceil (n-m)/2 \rceil$. Note that if $\ell_1 > 1$ or $\ell_{n-m} < L - 1$ holds, then $C_p(x, y) \geq \lceil (n-m)/2 \rceil + 1$ holds since $(x_1 - y_1)^2 = (x_n - y_m)^2 = 1$ (which contributes at least one further 1 to the sum $C_p(x, y)$). This lower bound matches the claimed costs. Otherwise, if $1 = \ell_1 < \ell_2 < \ldots < \ell_{n-m} = L - 1$, then each of the indices $\ell_1, \ell_3, \ell_5, \ldots, \ell_{n-m-1}, \ell_{n-m}$ contributes a cost of 1 to $C_p(x, y)$. Hence, the lower bound is $\lceil (n-m)/2 \rceil + 1$ as claimed. This finishes the proof. □

Note that according to Lemma 7.5, for a fixed condensed binary time series y of length m, the value $\mathrm{dtw}(x, y)^2$ is monotonically increasing in the length of x for all condensed binary time series x of length $n \geq m+1$. We use this property later in the proof of Lemma 7.7 where we derive an upper bound on the length of a binary mean. In order to prove Lemma 7.7, we also need the following lemma concerning the dtw-distances between condensed and non-condensed time series.

Lemma 7.6. *Let $x = (x_1, \ldots, x_n)$ be a condensed binary time series and let $y = (y_1, \ldots, y_m) \in \{0, 1\}^*$ with $n \geq m$. Then, for the condensation $\tilde{y}$ of y it holds* $\mathrm{dtw}(x, y)^2 = \mathrm{dtw}(x, \tilde{y})^2$.

Proof. Assume that y is not condensed. Then, y consists of $\ell \in [m]$ *blocks*, where a block is a maximal subsequence of consecutive 0's or consecutive 1's in y. Let $m_1, \ldots, m_\ell$ denote the lengths of these blocks where $m_1 + \ldots + m_\ell = m$. Note also that $\tilde{y}$ has length ℓ with $\ell < m \leq n$. We define a warping path p between x and y such that $C_p(x, y) = \mathrm{dtw}(x, \tilde{y})^2$. Note that, by Lemma 7.5, we have

$$\mathrm{dtw}(x, \tilde{y})^2 = \begin{cases} \lceil (n-\ell)/2 \rceil, & x_1 = \tilde{y}_1 \\ 1 + \lfloor (n-\ell)/2 \rfloor, & x_1 \neq \tilde{y}_1 \end{cases}.$$

The idea is, to extend the optimal warping path between x and $\tilde{y}$ as given in the proof of Lemma 7.5 (see Figure 7.9).

If $x_1 = y_1$, then we set

$$p := ((1,1), \ldots, (1, m_1), (2, m_1 + 1), \ldots, (2, m_1 + m_2), \ldots, \\ (\ell, m - m_\ell + 1), \ldots, (\ell, m), (\ell + 1, m), \ldots, (n, m))$$

and obtain cost $C_p(x, y) = \sum_{i=\ell+1}^{n} (x_i - y_m)^2 = \lceil (n-\ell)/2 \rceil$.

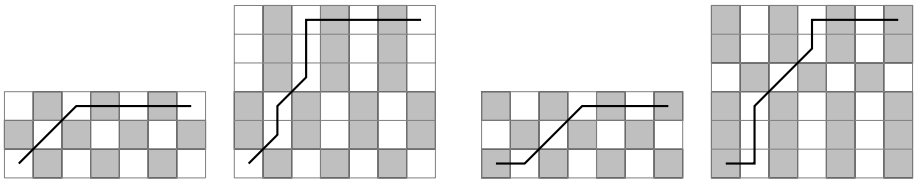


Figure 7.9: Examples of optimal warping paths between a condensed time series $x = (0, 1, 0, 1, 0, 1, 0)$ and a non-condensed time series $y = (0, 1, 1, 0, 0, 0)$ (b) and $y = (1, 1, 1, 0, 1, 1)$ (d). The matrices (a) and (c) depict optimal warping paths between x and the respective condensation $\tilde{y}$.

If $x_1 \neq y_1$, then we set

$$p := ((1,1),(2,1),\ldots,(2,m_1),(3,m_1+1),\ldots,\\ (3,m_1+m_2),\ldots,(\ell+1,m),(\ell+2,m),\ldots,(n,m))$$

and obtain cost

$$C_p(x,y) = 1 + \sum_{i=\ell+2}^{n} (x_i - y_m)^2 = 1 + \lfloor (n-\ell)/2 \rfloor.$$

□

We now have all ingredients to show that there always exists a binary mean of length at most one larger than the maximum length of any input time series.

Lemma 7.7. *For binary input time series $x^{(1)}, \ldots, x^{(k)} \in \{0,1\}^*$ of maximum length n, there exists a binary mean $z \in \{0,1\}^*$ of length at most $n+1$.*

Proof. Assume that $z = (z_1, \ldots, z_m) \in \{0,1\}^*$ is a mean of length $m > n+1$. By Proposition 7.4, we can assume that z is condensed, that is, $z_i \neq z_{i+1}$ for all $i \in [m-1]$. We claim that $z' := (z_1, \ldots, z_{n+1})$ is also a mean. We prove this claim by showing that $\mathrm{dtw}(z', x^{(i)})^2 \leq \mathrm{dtw}(z, x^{(i)})^2$ holds for all $i \in [k]$. By Lemmas 7.5 and 7.6, we have

$$\mathrm{dtw}(z', x^{(i)})^2 = \mathrm{dtw}(z', \tilde{x}^{(i)})^2 \leq \mathrm{dtw}(z, \tilde{x}^{(i)})^2 = \mathrm{dtw}(z, x^{(i)})^2,$$

where the inequality follows from Lemma 7.5 since z' is of length $n+1 < m$ and the dtw-distance is monotonically increasing. □

Having established that a binary mean is always condensed and of bounded length, we now show that it can be found in polynomial time.

Proof of Theorem 7.3. By Proposition 7.4 and Lemma 7.7, we can assume the desired mean z to be a condensed series of length at most $n+1$. Thus, there are at most $2n+2$ many possible candidates for z. For each candidate z, we can compute the value $F(z)$ in $O(kn^2)$ time and select the one with the smallest value. This yields an overall running time in $O(kn^3)$. □

The proof of Theorem 7.3 relies on the fact that there are only $O(n)$ different binary condensed candidate time series of length at most $n+1$. Note that, for an alphabet with three elements, the number of condensed time series of bounded length is already exponential in the length. It is therefore not clear how to compute a mean over larger alphabets in polynomial time even if we assume an upper bound on its length.

7.6 Conclusion

We studied the complexity of computing an exact mean in dynamic time warping spaces and developed a first provably correct algorithm (running in exponential time). Moreover, we proved that the special case of binary data is polynomial-time solvable. Our results are obtained by analyzing the structure of warping paths for optimal solutions. In this chapter we took a first step towards a theoretically profound (parameterized) complexity analysis of a fundamental problem in time series analysis; an endeavor which might also prove beneficial in practice at some point. Note that DTW-Mean has recently been shown to be NP-hard, W[1]-hard with respect to the number k of input time series, and not solvable in time $n^{o(k)} \cdot \rho(k)$ for any computable function ρ (assuming the Exponential Time Hypothesis), where n is the maximum length of any input time series [BFN18] (it is conjectured that these hardness results even hold for binary input time series).

In preliminary experimental work, we used our exact dynamic program to evaluate the performance of state-of-the-art heuristics. The results indicate a poor worst-case performance of heuristic approaches with considerable space for improvements [Bri+18].

We conclude with some challenges for future research. From an algorithmic point of view, improving the exponential running time part (for example, reducing n^{2k+1} to n^k) might be an interesting challenge. Another natural

parameter is the maximum length n of the input series. So far, it is even open whether DTW-Mean is in XP with respect to n. Concerning other parameterizations, it might be interesting to search for data-driven parameters that are motivated by the structure of existing real-world data sets. For instance, such parameters could measure the sparsity of the time series (that is, the number of non-zero entries) or the maximum Euclidean distance between any two time series (in case of equal-length time series).

It is further interesting to investigate whether one can extend the polynomial-time solvability of Binary DTW-Mean to larger alphabet sizes; already the case of alphabet size three is open. Improving the $O(kn^3)$-time algorithm for Binary DTW-Mean might also be worth in order to conduct experiments.

Finally, we wonder whether there are other (practically relevant) restrictions of DTW-Mean that make the problem more tractable. For example, one could restrict the length or the shape of a warping path, or one could optimize over a restricted set of candidate means. Another idea is to define a maximum number of elements from each input time series that are allowed to be aligned to a single mean element in order to avoid degenerate solutions.

Chapter 8

Outlook

We shed some light on the border of computational tractability for some practically motivated problems related to data science questions. Most of the investigated problems are computationally hard (NP-hard) to solve in general. However, by adopting a parameterized view, we showed that choosing the "right" parameters allows to detect tractable (sometimes polynomial-time solvable and sometimes fixed-parameter tractable) cases for each problem. Our theoretical results serve as a first step towards exactly solving problems efficiently in practice and raise some hope not to be forced to always rely on heuristics. Moreover, our results carry the potential to obtain a better understanding of heuristic approaches (cf. [KNN16]) by determining and explaining the cases in which they might not perform well.

We suggested concrete directions for future research specific to each considered problem in the conclusion of the corresponding chapter. This chapter concludes the thesis with some more general ideas for future research directions.

First of all, the presented results are clearly theoretical in nature and constitute some initial points of reference for further investigations. An obvious question is how well the developed algorithms perform in practice. Answering this question profoundly[1] should be done by an algorithm engineering approach including implementation, optimization and thorough experimental evaluation on real-world data (which should easily be available for most of the considered problems).

A good candidate problem for evaluation certainly is Cluster Editing which is practically relevant and empirically well-studied. Our fine-tunable data reduction rules developed in Chapter 6 carry the potential to speed up state-of-the-art algorithms. It is interesting to investigate to which extent this

[1] Note that we already conducted some preliminary experiments for Co-Clustering$_\infty$ (see Section 5.5) and DTW-Mean (see Section 7.6).

potential can be exploited on real-world instances. It is also promising to implement the search tree algorithm which could also be enhanced by better branching and data reduction rules. Another candidate is DTW-Mean from Chapter 7. Here, the dynamic program can be used to compute optimal means of real-world data to gain new insights on some characteristics of a mean (for example, uniqueness) that help to improve existing heuristics. It might also be interesting to implement the polynomial-time algorithm for binary time series and subsequently compare its performance in terms of time series classification and clustering tasks with existing heuristics. We point out that our algorithm and its running time analysis are fairly simple. We believe that it might be possible to further improve the running time to become more practical. Moreover, it might be possible to use some of our insights to tweak existing heuristics in order to increase their solution quality. Lastly, we believe that solving co-clustering tasks using SAT solvers as described in Chapter 5 is a reasonable approach which might become competitive in practice with some effort towards more sophisticated reductions to SAT instances.

On the theoretical side, our results encourage to apply the method of a fine-grained multivariate complexity analysis to other interesting and practically relevant problems (also see the survey by Niedermeier [Nie10]). Besides the problem domains we touched in this thesis, one might, for example, study problems in other fields such as statistics, social network analysis, artificial intelligence, computer vision, robotics, or big data (see the survey by Mnich [Mni17]). The search itself might include the hurdle of finding an appropriate problem formalization in order for a concrete problem to become tangible for complexity studies. This process of abstraction also has the potential benefit of showing interesting connections between different research areas such as graph theory, combinatorics, geometry and others.

Regarding the study of complexity borders, we remark that there are two different types of borders to study, a *problem-oriented* and a *parameter-oriented* border. The problem-oriented border describes the computational complexity of a specific problem with respect to different parameters (as studied in Chapters 3 to 5), whereas the parameter-oriented border is defined for a specific parameter and separates the tractable from the intractable problems with respect to this parameter (we studied this kind of border in Chapter 6). Certainly, parameter-oriented complexity borders are more challenging to investigate which is due to the fact that problems are very different in nature. Nevertheless, studying

such borders might lead to a better theoretical understanding of the algorithmic utility of a parameter.

Concerning parameterization, it is natural to seek other useful parameters. Problems arising in machine learning, for example, appear to be rich of interesting "non-standard" parameters such as the number of features, the number or the size of clusters, the number of classes, or the VC dimension. Of particular interest for practical purposes are data-driven parameters which can be measured on given data sets beforehand in order to determine whether the parameter is actually small. Such parameters could, for example, measure the density, the similarity or the entropy of the input data in some way.

Finally, we remark that the "art of parameterization" (cf. [Nie06, Chapter 5]) is itself a data-scientific problem in which the goal is to extract some crucial information from the input data that can be used to efficiently solve a problem. This raises the question whether the process of choosing "the right" parameter can be automated to some extent. For example, one could imagine to use data mining techniques to discover new and interesting parameters in data sets. Moreover, it is conceivable that machine learning methods will at some point allow a program to learn which parameter is "the best" to consider for a given instance. The goal would be to devise an algorithm that first predicts the parameterized algorithm from a given suite of parameterized algorithms that solves a given instance most efficiently (a problem related to the *Algorithm Selection Problem* [Kot16, Ric76]) and then runs it on the instance (note that similar ideas are used in the *Programming by Optimization* paradigm [HH15, Hoo12, Ley+14]). This might pave the way for a successful interplay between algorithm theory and practice.

Bibliography

[AB09] S. Arora and B. Barak. *Computational Complexity—A Modern Approach*. Cambridge University Press, 2009 (cited on p. 7).

[ADK12] A. Anagnostopoulos, A. Dasgupta, and R. Kumar. "A constant-factor approximation algorithm for co-clustering". In: *Theory of Computing* 8 (2012), pp. 597–622 (cited on pp. 64–66).

[AK00] P. Alimonti and V. Kann. "Some APX-completeness results for cubic graphs". In: *Theoretical Computer Science* 237.1–2 (2000), pp. 123–134 (cited on p. 24).

[Alo06] N. Alon. "Ranking tournaments". In: *SIAM Journal on Discrete Mathematics* 20.1 (2006), pp. 137–142 (cited on pp. 90, 102).

[ALS09] N. Alon, D. Lokshtanov, and S. Saurabh. "Fast FAST". In: *Proceedings of the 36th International Colloquium on Automata, Languages, and Programming (ICALP '09)*. Vol. 5555. LNCS. Springer, 2009, pp. 49–58 (cited on p. 102).

[APT79] B. Aspvall, M. F. Plass, and R. E. Tarjan. "A linear-time algorithm for testing the truth of certain quantified boolean formulas". In: *Information Processing Letters* 8.3 (1979), pp. 121–123 (cited on p. 79).

[ASS16] N. R. Aravind, R. B. Sandeep, and N. Sivadasan. "Parameterized lower bounds and dichotomy results for the NP-completeness of H-free edge modification problems". In: *Proceedings of the 12th Latin American Symposium on Theoretical Informatics, (LATIN '16)*. Vol. 9644. 2016, pp. 82–95 (cited on p. 90).

[ASW15] S. Aghabozorgi, A. S. Shirkhorshidi, and T. Y. Wah. "Time-series clustering – A decade review". In: *Information Systems* 53 (2015), pp. 16–38 (cited on p. 129).

[Ban+07] A. Banerjee, I. S. Dhillon, J. Ghosh, S. Merugu, and D. S. Modha. "A generalized maximum entropy approach to Bregman co-clustering and matrix approximation". In: *Journal of Machine Learning Research* 8 (2007), pp. 1919–1986 (cited on p. 64).

[Bar+17] L. Barba, J. Cardinal, J. Iacono, S. Langerman, A. Ooms, and N. Solomon. "Subquadratic algorithms for algebraic generalizations of 3SUM". In: *Proceedings of the 33rd International Symposium on Computational Geome-*

try (SoCG '17). Vol. 77. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, 13:1–13:15 (cited on pp. 13, 16).

[BB13] S. Böcker and J. Baumbach. "Cluster editing". In: *The Nature of Computation. Logic, Algorithms, Applications: Proceedings of the 9th Conference on Computability in Europe (CiE '13)*. Vol. 7921. LNCS. Springer, 2013, pp. 33–44 (cited on p. 106).

[BBC04] N. Bansal, A. Blum, and S. Chawla. "Correlation clustering". In: *Machine Learning* 56.1 (2004), pp. 89–113 (cited on p. 106).

[BD01] P. Bonizzoni and G. Della Vedova. "The complexity of multiple sequence alignment with SP-score that is a metric". In: *Theoretical Computer Science* 259.1–2 (2001), pp. 63–79 (cited on p. 129).

[Bee+84] C. Beeri, M. Dowd, R. Fagin, and R. Statman. "On the structure of Armstrong relations for functional dependencies". In: *Journal of the ACM* 31.1 (1984), pp. 30–46 (cited on p. 41).

[Bes+11] S. Bessy, F. V. Fomin, S. Gaspers, C. Paul, A. Perez, S. Saurabh, and S. Thomassé. "Kernels for feedback arc set in tournaments". In: *Journal of Computer and System Sciences* 77.6 (2011), pp. 1071–1078 (cited on p. 102).

[Bev+15] R. van Bevern, R. Bredereck, J. Chen, V. Froese, R. Niedermeier, and G. J. Woeginger. "Network-based vertex dissolution". In: *SIAM Journal on Discrete Mathematics* 29.2 (2015), pp. 888–914 (cited on p. xi).

[Bev+17] R. van Bevern, R. Bredereck, L. Bulteau, J. Chen, V. Froese, R. Niedermeier, and G. J. Woeginger. "Partitioning perfect graphs into stars". In: *Journal of Graph Theory* 85.2 (2017), pp. 297–335 (cited on p. xi).

[Bev14] R. van Bevern. "Towards optimal and expressive kernelization for d-Hitting Set". In: *Algorithmica* 70.1 (2014), pp. 129–147 (cited on p. 30).

[BFK16] R. van Bevern, V. Froese, and C. Komusiewicz. "Parameterizing edge modification problems above lower bounds". In: *Proceedings of the 11th International Computer Science Symposium in Russia (CSR '16)*. Vol. 9691. LNCS. Springer, 2016, pp. 57–72 (cited on p. x).

[BFK18] R. van Bevern, V. Froese, and C. Komusiewicz. "Parameterizing edge modification problems above lower bounds". In: *Theory of Computing Systems* 62.3 (2018), pp. 739–770 (cited on pp. x, 94, 113).

[BFN18] L. Bulteau, V. Froese, and R. Niedermeier. "Hardness of consensus problems for circular strings and time series averaging". In: *CoRR* abs/1804.02854 (2018). Submitted to FOCS. (cited on pp. 4, 126, 130, 145).

[BFS17] T. Bläsius, T. Friedrich, and M. Schirneck. "The parameterized complexity of dependency detection in relational databases". In: *Proceedings of*

the 11th International Symposium on Parameterized and Exact Computation (IPEC '16). Vol. 63. LIPIcs. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, 6:1–6:13 (cited on p. 41).

[BFT16] L. Bulteau, V. Froese, and N. Talmon. "Multi-player diffusion games on graph classes". In: *Internet Mathematics* 12.6 (2016), pp. 363–380 (cited on p. xi).

[BG09] J. Bang-Jensen and G. Gutin. *Digraphs: Theory, Algorithms and Applications*. Springer, 2009 (cited on p. 60).

[BGL17] É. Bonnet, P. Giannopoulos, and M. Lampis. "On the parameterized complexity of red-blue points separation". In: *Proceedings of the 12th International Symposium on Parameterized and Exact Computation (IPEC '17)*. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017, 8:1–8:13 (cited on p. 87).

[BJK14] H. L. Bodlaender, B. M. P. Jansen, and S. Kratsch. "Kernelization lower bounds by cross-composition". In: *SIAM Journal on Discrete Mathematics* 28.1 (2014), pp. 277–305 (cited on p. 10).

[BK15] K. Bringmann and M. Künnemann. "Quadratic conditional lower bounds for string problems and dynamic time warping". In: *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS '15)*. IEEE, 2015, pp. 79–97 (cited on p. 129).

[BKM09] D. Brügmann, C. Komusiewicz, and H. Moser. "On generating triangle-free graphs". In: *Proceedings of the DIMAP Workshop on Algorithmic Graph Theory (AGT '09)*. Vol. 32. Electronic Notes in Discrete Mathematics. Elsevier, 2009, pp. 51–58 (cited on pp. 97, 100).

[BL97] A. Blum and P. Langley. "Selection of relevant features and examples in machine learning". In: *Artificial Intelligence* 97.1-2 (1997), pp. 245–271 (cited on p. 40).

[BM11] A. Brandstädt and R. Mosca. "On distance-3 matchings and induced matchings". In: *Discrete Applied Mathematics* 159.7 (2011), pp. 509–520 (cited on p. 44).

[Böc+09] S. Böcker, S. Briesemeister, Q. B. A. Bui, and A. Truß. "Going weighted: parameterized algorithms for cluster editing". In: *Theoretical Computer Science* 410.52 (2009), pp. 5467–5480 (cited on p. 91).

[Böc12] S. Böcker. "A golden ratio parameterized algorithm for cluster editing". In: *Journal of Discrete Algorithms* 16 (2012), pp. 79–89 (cited on pp. 106, 110).

[Bod+09] H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin. "On problems without polynomial kernels". In: *Journal of Computer and System Sciences* 75.8 (2009), pp. 423–434 (cited on p. 10).

[Boi+17] J.-D. Boissonnat, K. Dutta, A. Ghosh, and S. Kolay. "Kernelization of the subset general position problem in geometry". In: *Proceedings of the 42nd International Symposium on Mathematical Foundations of Computer Science (MFCS '17)*. LIPIcs. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, 25:1–25:13 (cited on pp. 36, 37).

[Bre+15a] R. Bredereck, J. Chen, S. Hartung, C. Komusiewicz, R. Niedermeier, and O. Suchý. "On explaining integer vectors by few homogenous segments". In: *Journal of Computer and System Sciences* 81.4 (2015), pp. 766–782 (cited on p. 62).

[Bre+15b] R. Bredereck, V. Froese, S. Hartung, A. Nichterlein, R. Niedermeier, and N. Talmon. "The complexity of degree anonymization by vertex addition". In: *Theoretical Computer Science* 607.1 (2015), pp. 16–34 (cited on p. xi).

[Bre+18] R. Bredereck, V. Froese, M. Koseler, M. G. Millani, A. Nichterlein, and R. Niedermeier. "A parameterized algorithmics framework for digraph degree sequence completion problems". In: *Algorithmica* (2018). Accepted for publication. URL: https://arxiv.org/abs/1604.06302 (cited on p. xi).

[Bri+18] M. Brill, T. Fluschnik, V. Froese, B. Jain, R. Niedermeier, and D. Schultz. "Exact mean computation in dynamic time warping spaces". In: *Proceedings of the SIAM International Conference on Data Mining (SDM '18)*. SIAM, 2018, pp. 540–548 (cited on pp. xi, 145).

[BTY11] H. L. Bodlaender, S. Thomassé, and A. Yeo. "Kernel bounds for disjoint cycles and disjoint paths". In: *Theoretical Computer Science* 412.35 (2011), pp. 4570–4578 (cited on p. 10).

[Bul+14] L. Bulteau, V. Froese, S. Hartung, and R. Niedermeier. "Co-clustering under the maximum norm". In: *Proceedings of the 25th International Symposium on Algorithms and Computation (ISAAC '14)*. LNCS 8889. Springer, 2014, pp. 298–309 (cited on p. x).

[Bul+16a] L. Bulteau, V. Froese, S. Hartung, and R. Niedermeier. "Co-clustering under the maximum norm". In: *Algorithms* 9.1 (2016). Art. No. 17 (cited on pp. x, 86).

[Bul+16b] L. Bulteau, V. Froese, K. Kutzkov, and R. Pagh. "Triangle counting in dynamic graph streams". In: *Algorithmica* 76.1 (2016), pp. 259–278 (cited on p. xi).

[Bul+17] L. Bulteau, S. Fafianie, V. Froese, R. Niedermeier, and N. Talmon. "The complexity of finding effectors". In: *Theory of Computing Systems* 60.2 (2017), pp. 253–279 (cited on p. xi).

[Cai+97] L. Cai, J. Chen, R. G. Downey, and M. R. Fellows. "Advice classes of parameterized tractability". In: *Annals of Pure and Applied Logic* 84.1 (1997), pp. 119–138 (cited on p. 9).

[Cai96] L. Cai. "Fixed-parameter tractability of graph modification problems for hereditary properties". In: *Information Processing Letters* 58.4 (1996), pp. 171–176 (cited on pp. 90, 93, 94, 113).

[Cao12] C. Cao. "Study on Two Optimization Problems: Line Cover and Maximum Genus Embedding". MA thesis. Texas A&M University, May 2012 (cited on pp. 15, 24–26, 28).

[CB17] M. Cuturi and M. Blondel. "Soft-DTW: a differentiable loss function for time-series". In: *Proceedings of the 34th International Conference on Machine Learning (ICML '17)*. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 894–903 (cited on p. 126).

[CC15] L. Cai and Y. Cai. "Incompressibility of H-free edge modification problems". In: *Algorithmica* 71.3 (2015), pp. 731–757 (cited on p. 90).

[CDK14] F. Chierichetti, N. Dalvi, and R. Kumar. "Correlation clustering in MapReduce". In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14)*. ACM, 2014, pp. 641–650 (cited on p. 106).

[Cha+00] M. Charikar, V. Guruswami, R. Kumar, S. Rajagopalan, and A. Sahai. "Combinatorial feature selection problems". In: *Proceedings of the 41st IEEE Annual Symposium on Foundations of Computer Science (FOCS '00)*. 2000, pp. 631–640 (cited on pp. 39–41).

[Che+16] W. Cheng, X. Zhang, F. Pan, and W. Wang. "HICC: an entropy splitting-based framework for hierarchical co-clustering". In: *Knowledge and Information Systems* 46.2 (2016), pp. 343–367 (cited on p. 66).

[CM03] C. Cotta and P. Moscato. "The k-feature set problem is W[2]-complete". In: *Journal of Computer and System Sciences* 67.4 (2003), pp. 686–690 (cited on p. 41).

[CM12] J. Chen and J. Meng. "A $2k$ kernel for the cluster editing problem". In: *Journal of Computer and System Sciences* 78.1 (2012), pp. 211–220 (cited on pp. 106, 107, 112).

[CN98] B. S. Chlebus and S. H. Nguyen. "On finding optimal discretizations for two attributes". In: *Proceedings of the First International Conference on Rough Sets and Current Trends in Computing (RSCTC '98)*. Vol. 1424. LNCS. Springer, 1998, pp. 537–544 (cited on pp. 65, 73, 86).

[Cov65] T. M. Cover. “Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition”. In: *IEEE Transactions on Electronic Computers* EC-14.3 (1965), pp. 326–334 (cited on p. 13).

[CS14] G. Chandrashekar and F. Sahin. “A survey on feature selection methods”. In: *Computers & Electrical Engineering* 40.1 (2014), pp. 16–28 (cited on p. 40).

[CSS99] W. W. Cohen, R. E. Schapire, and Y. Singer. “Learning to order things”. In: *Journal of Artificial Intelligence Research* 10 (1999), pp. 243–270 (cited on p. 101).

[CST00] A. Califano, G. Stolovitzky, and Y. Tu. “Analysis of gene expression microarrays for phenotype classification”. In: *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology (ISMB'00)*. AAAI, 2000, pp. 75–85 (cited on p. 65).

[Cyg+13] M. Cygan, M. Pilipczuk, M. Pilipczuk, and J. O. Wojtaszczyk. “On multiway cut parameterized above lower bounds”. In: *ACM Transactions on Computation Theory* 5.1 (2013), p. 3 (cited on pp. 90, 122).

[Cyg+15] M. Cygan, F. V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015 (cited on p. 8).

[Das+07] A. Dasgupta, P. Drineas, B. Harb, V. Josifovski, and M. W. Mahoney. “Feature selection methods for text classification”. In: *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '07)*. ACM, 2007, pp. 230–239 (cited on p. 40).

[Dat03] C. J. Date. *An Introduction to Database Systems*. Pearson, 2003 (cited on p. 40).

[Deh+06] F. Dehne, M. A. Langston, X. Luo, S. Pitre, P. Shaw, and Y. Zhang. “The cluster editing problem: implementations and experiments”. In: *Proceedings of the Second International Workshop on Parameterized and Exact Computation (IWPEC '06)*. Vol. 4169. LNCS. Springer, 2006, pp. 13–24 (cited on p. 106).

[Deĭ+06] V. G. Deĭneko, M. Hoffmann, Y. Okamoto, and G. J. Woeginger. “The traveling salesman problem with few inner points”. In: *Operations Research Letters* 34.1 (2006), pp. 106–110 (cited on p. 33).

[Dez74] M. Deza. “Solution d'un problème de Erdős-Lovász”. In: *Journal of Combinatorial Theory, Series B* 16.2 (1974), pp. 166–167 (cited on p. 52).

[DF13] R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Springer, 2013 (cited on pp. 8, 20).

[DF99] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999 (cited on p. 8).

[DFR12] M. Dom, M. R. Fellows, and F. A. Rosamond. "The parameterized complexity of stabbing rectangles". In: *Algorithmica* 62.1–2 (2012), pp. 564–594 (cited on p. 87).

[DK08] A. Das and D. Kempe. "Algorithms for subset selection in linear regression". In: *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC '08)*. ACM, 2008, pp. 45–54 (cited on p. 40).

[DM14] H. Dell and D. van Melkebeek. "Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses". In: *Journal of the ACM* 61.4 (2014), 23:1–23:27 (cited on pp. 10, 33, 34, 36).

[Dom+10] M. Dom, J. Guo, F. Hüffner, R. Niedermeier, and A. Truß. "Fixed-parameter tractability results for feedback set problems in tournaments". In: *Journal of Discrete Algorithms* 8.1 (2010), pp. 76–86 (cited on p. 102).

[DR94] S. Davies and S. Russell. "NP-completeness of searches for smallest possible feature sets". In: *AAAI Symposium on Intelligent Relevance*. 1994, pp. 37–39 (cited on p. 41).

[Dru15] A. Drucker. "New limits to classical and quantum instance compression". In: *SIAM Journal on Computing* 44.5 (2015), pp. 1443–1479 (cited on p. 10).

[Dur+15] G. Durán, F. F. Slezak, L. Grippo, F. de S. Oliveira, and J. Szwarcfiter. "On unit interval graphs with integer endpoints". In: *Electronic Notes in Discrete Mathematics* 50.Supplement C (2015), pp. 445–450 (cited on p. 69).

[Dwo+01] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. "Rank aggregation methods for the web". In: *Proceedings of the 10th International Conference on World Wide Webseries (WWW '01)*. ACM, 2001, pp. 613–622 (cited on p. 101).

[EOS86] H. Edelsbrunner, J. O'Rourke, and R. Seidel. "Constructing arrangements of lines and hyperplanes with applications". In: *SIAM Journal on Computing* 15.2 (1986), pp. 341–363 (cited on p. 32).

[Erd86] P. Erdős. "On some metric and combinatorial geometric problems". In: *Discrete Mathematics* 60 (1986), pp. 147–153 (cited on p. 15).

[Est+05] V. Estivill-Castro, M. Fellows, M. A. Langston, and F. Rosamond. "FPT is P-time extremal structure I (fixed-parameter tractability is polynomial-time extremal structure theory)". In: *Algorithms and Complexity (Texts in Algorithmics 4)*. King's College Publications, 2005, pp. 1–41 (cited on p. 2).

[Fei09] U. Feige. "Faster FAST (feedback arc set in tournaments)". In: *CoRR* abs/0911.5094 (2009). URL: http://arxiv.org/abs/0911.5094 (cited on p. 102).

[Fei14] U. Feige. "NP-hardness of hypercube 2-segmentation". In: *CoRR* abs/1411.0821 (2014). URL: https://arxiv.org/abs/1411.0821 (cited on p. 65).

[Fel+11] M. R. Fellows, J. Guo, C. Komusiewicz, R. Niedermeier, and J. Uhlmann. "Graph-based data clustering with overlaps". In: *Discrete Optimization* 8.1 (2011), pp. 2–17 (cited on p. 123).

[Fel03] M. R. Fellows. "Blow-ups, win/win's, and crown rules: some new directions in FPT". In: *Proceedings of the 29th International Workshop on Graph-Theoretic Concepts in Computer Science (WG '03)*. Vol. 2880. LNCS. 2003, pp. 1–12 (cited on p. 95).

[FG06] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006 (cited on pp. 3, 8, 39).

[FKV14] S. Felsner, M. Kaufmann, and P. Valtr. "Bend-optimal orthogonal graph drawing in the general position model". In: *Computational Geometry* 47.3, Part B (2014), pp. 460–468 (cited on p. 13).

[FNN16] V. Froese, A. Nichterlein, and R. Niedermeier. "Win-win kernelization for degree sequence completion problems". In: *Journal of Computer and System Sciences* 82.6 (2016), pp. 1100–1111 (cited on pp. xi, 95).

[Fom+14] F. V. Fomin, S. Kratsch, M. Pilipczuk, M. Pilipczuk, and Y. Villanger. "Tight bounds for parameterized complexity of cluster editing with a small number of clusters". In: *Journal of Computer and System Sciences* 80.7 (2014), pp. 1430–1447 (cited on pp. 106, 107).

[For03] G. Forman. "An extensive empirical study of feature selection metrics for text classification". In: *Journal of Machine Learning Research* 3 (2003), pp. 1289–1305 (cited on p. 40).

[FP13] F. V. Fomin and M. Pilipczuk. "Subexponential parameterized algorithm for computing the cutwidth of a semi-complete digraph". In: *Proceedings of the 21st Annual European Symposium (ESA '13)*. Vol. 8125. LNCS. Springer, 2013, pp. 505–516 (cited on pp. 102, 103).

[FPT81] R. J. Fowler, M. S. Paterson, and S. L. Tanimoto. "Optimal packing and covering in the plane are NP-complete". In: *Information Processing Letters* 12.3 (1981), pp. 133–137 (cited on p. 72).

[Fré48] M. Fréchet. "Les éléments aléatoires de nature quelconque dans un espace distancié". In: *Annales de l'institut Henri Poincaré* (1948), pp. 215–310 (cited on p. 126).

[Fro+13] V. Froese, R. van Bevern, R. Niedermeier, and M. Sorge. "A parameterized complexity analysis of combinatorial feature selection problems". In: *Proceedings of the 38th International Symposium on Mathematical Foundations of Computer Science (MFCS '13)*. Vol. 8087. LNCS. Springer, 2013, pp. 445–456 (cited on p. x).

[Fro+16a] V. Froese, R. van Bevern, R. Niedermeier, and M. Sorge. "Exploiting hidden structure in selecting dimensions that distinguish vectors". In: *Journal of Computer and System Sciences* 82.3 (2016), pp. 521–535 (cited on pp. x, 41, 43, 44).

[Fro+16b] V. Froese, I. Kanj, A. Nichterlein, and R. Niedermeier. "Finding points in general position". In: *Proceedings of the 28th Canadian Conference on Computational Geometry (CCCG '16)*. 2016, pp. 7–14 (cited on p. ix).

[Fro+17] V. Froese, I. Kanj, A. Nichterlein, and R. Niedermeier. "Finding points in general position". In: *International Journal of Computational Geometry & Applications* 27.4 (2017), pp. 277–296 (cited on p. ix).

[Fro12] V. Froese. "Combinatorial Feature Selection: Parameterized Algorithms and Complexity". MA thesis. TU Berlin, 2012 (cited on pp. ix, 41, 43, 44).

[Für91] Z. Füredi. "Maximal independent subsets in Steiner systems and in planar sets". In: *SIAM Journal on Discrete Mathematics* 4.2 (1991), pp. 196–199 (cited on p. 29).

[GE03] I. Guyon and A. Elisseeff. "An introduction to variable and feature selection". In: *Journal of Machine Learning Research* 3 (2003), pp. 1157–1182 (cited on p. 40).

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979 (cited on pp. 7, 24, 70).

[GK68] R. K. Guy and P. A. Kelly. "The no-three-in-line problem". In: *Canadian Mathematical Bulletin* 11 (1968), pp. 527–531 (cited on p. 14).

[GŁ15] T. Górecki and M. Łuczak. "Multivariate time series classification with parametric derivative dynamic time warping". In: *Expert Systems with Applications* 42.5 (2015), pp. 2305–2312 (cited on p. 138).

[GOR96] L. J. Guibas, M. H. Overmars, and J. Robert. "The exact fitting problem in higher dimensions". In: *Computational Geometry: Theory and Applications* 6.4 (1996), pp. 215–230 (cited on pp. 26, 28).

[GP16] S. Garg and G. Philip. "Raising the bar for vertex cover: fixed-parameter tractability above a higher guarantee". In: *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '16)*. SIAM, 2016, pp. 1152–1166 (cited on pp. 90, 122).

[GR15] S. K. Ghosh and B. Roy. "Some results on point visibility graphs". In: *Theoretical Computer Science* 575 (2015), pp. 17–32 (cited on p. 16).

[Gra+04] J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier. "Automated generation of search tree algorithms for hard graph modification problems". In: *Algorithmica* 39.4 (2004), pp. 321–347 (cited on p. 97).

[Gra+05] J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier. "Graph-modeled data clustering: exact algorithms for clique generation". In: *Theory of Computing Systems* 38.4 (2005), pp. 373–392 (cited on p. 106).

[GRT97] F. Gómez, S. Ramaswami, and G. T. Toussaint. "On removing non-degeneracy assumptions in computational geometry". In: *Proceedings of the 3rd Italian Conference on Algorithms and Complexity (CIAC '97)*. Vol. 1203. LNCS. Springer, 1997, pp. 86–99 (cited on p. 32).

[GS17] O. Gold and M. Sharir. "Dynamic time warping and geometric edit distance: breaking the quadratic barrier". In: *Proceedings of the 44th International Colloquium on Automata, Languages and Programming (ICALP '17)*. Vol. 80. LIPIcs. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, 25:1–25:14 (cited on p. 129).

[Gui+13] S. Guillemot, F. Havet, C. Paul, and A. Perez. "On the (non-)existence of polynomial kernels for P_ℓ-free edge modification problems". In: *Algorithmica* 65.4 (2013), pp. 900–926 (cited on p. 90).

[Guo09] J. Guo. "A more effective linear kernelization for cluster editing". In: *Theoretical Computer Science* 410.8-10 (2009), pp. 718–726 (cited on p. 107).

[Gus97] D. Gusfield. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, 1997 (cited on p. 129).

[Hal+75] R. Hall, T. Jackson, A. Sudbery, and K. Wild. "Some advances in the no-three-in-line problem". In: *Journal of Combinatorial Theory, Series A* 18.3 (1975), pp. 336–341 (cited on p. 14).

[Har+15] S. Hartung, A. Nichterlein, R. Niedermeier, and O. Suchý. "A refined complexity analysis of degree anonymization in graphs". In: *Information and Computation* 243 (2015), pp. 249–262 (cited on p. 95).

[Har72] J. A. Hartigan. "Direct clustering of a data matrix". In: *Journal of the American Statistical Association* 67.337 (1972), pp. 123–129 (cited on p. 65).

[HH15] S. Hartung and Holger H. Hoos. "Programming by optimisation meets parameterised algorithmics: a case study for cluster editing". In: *Proceedings of the 9th International Conference on Learning and Intelligent Optimization (LION '15)*. Vol. 8994. LNCS. Springer, 2015, pp. 43–58 (cited on pp. 90, 93, 149).

[Him+18] A.-S. Himmel, C. Hoffmann, P. Kunz, V. Froese, and M. Sorge. "Computational complexity aspects of point visibility graphs". In: *Discrete Applied Mathematics* (2018). Accepted for publication. URL: https://arxiv.org/abs/1711.01811 (cited on p. xi).

[HKN15] F. Hüffner, C. Komusiewicz, and A. Nichterlein. "Editing graphs into few cliques: complexity, approximation, and kernelization schemes". In:

Proceedings of the Algorithms and Data Structures Symposium (WADS'15). Vol. 9214. LNCS. Springer, 2015, pp. 410–421 (cited on p. 123).

[HNF08] V. Hautamaki, P. Nykanen, and P. Franti. "Time-series clustering by approximate prototypes". In: *Proceedings of the 19th International Conference on Pattern Recognition (ICPR '08)*. IEEE, 2008, pp. 1–4 (cited on pp. 126, 129, 130).

[HO06] M. Hoffmann and Y. Okamoto. "The minimum weight triangulation problem with few inner points". In: *Computational Geometry: Theory and Applications* 34.3 (2006), pp. 149–158 (cited on p. 33).

[Hoo12] H. H. Hoos. "Programming by optimization". In: *Communications of the ACM* 55.2 (2012), pp. 70–80 (cited on p. 149).

[IP01] R. Impagliazzo and R. Paturi. "On the complexity of k-SAT". In: *Journal of Computer and System Sciences* 62.2 (2001), pp. 367–375 (cited on p. 8).

[IPZ01] R. Impagliazzo, R. Paturi, and F. Zane. "Which problems have strongly exponential complexity?" In: *Journal of Computer and System Sciences* 63.4 (2001), pp. 512–530 (cited on pp. 23, 24, 107).

[Joh14] H. R. Johnson. "Some new maximum VC classes". In: *Information Processing Letters* 114.6 (2014), pp. 294–298 (cited on p. 13).

[JS16] B. Jain and D. Schultz. "A reduction theorem for the sample mean in dynamic time warping spaces". In: *CoRR* abs/1610.04460 (2016) (cited on pp. 129, 131, 132).

[JS99] D. Johnson and M. Szegedy. "What are the least tractable instances of maximum independent set?" In: *Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms (SODA '99)*. 1999, pp. 927–928 (cited on p. 24).

[JSB09] S. Jegelka, S. Sra, and A. Banerjee. "Approximation algorithms for tensor clustering". In: *Proceedings of the 20th International Conference of Algorithmic Learning Theory (ALT'09)*. Vol. 5809. LNCS. Springer, 2009, pp. 368–383 (cited on p. 65).

[Juk11] S. Jukna. *Extremal Combinatorics*. Springer, 2011 (cited on pp. 51, 52).

[KKW15] C. Knauer, S. König, and D. Werner. "Fixed-parameter complexity and approximability of norm maximization". In: *Discrete & Computational Geometry* 53.2 (2015), pp. 276–295 (cited on p. 37).

[KM86] M. Křivánek and J. Morávek. "NP-hard problems in hierarchical-tree clustering". In: *Acta Informatica* 23.3 (1986), pp. 311–323 (cited on p. 90).

[KNN16] C. Komusiewicz, A. Nichterlein, and R. Niedermeier. "Parameterized algorithmics for graph modification problems: on interactions with heuristics". In: *Proceedings of the 41st International Workshop on Graph-Theoretic*

Concepts in Computer Science (WG '15). Vol. 9224. LNCS. Springer, 2016, pp. 3–15 (cited on p. 147).

[Kot16] L. Kotthoff. "Algorithm selection for combinatorial search problems: a survey". In: *Data Mining and Constraint Programming: Foundations of a Cross-Disciplinary Approach*. Ed. by C. Bessiere, L. De Raedt, L. Kotthoff, S. Nijssen, B. O'Sullivan, and D. Pedreschi. Springer, 2016, pp. 149–190 (cited on p. 149).

[KPR16] S. Kratsch, G. Philip, and S. Ray. "Point line cover: the easy kernel is essentially tight". In: *ACM Transactions on Algorithms* 12.3 (2016), 40:1–40:16 (cited on pp. 16, 34–36).

[Kra12] S. Kratsch. "Polynomial kernelizations for MIN $F^{+}\Pi_1$ and MAX NP". In: *Algorithmica* 63.1-2 (2012), pp. 532–550 (cited on p. 90).

[KS10] M. Karpinski and W. Schudy. "Faster algorithms for feedback arc set tournament, Kemeny rank aggregation and betweenness tournament". In: *Proceedings of the 21st International Symposium on Algorithms and Computation (ISAAC '10)*. Vol. 6506. LNCS. Springer, 2010, pp. 3–14 (cited on p. 102).

[KS96] D. Koller and M. Sahami. "Towards optimal feature selection". In: *Proceedings of the 13th International Conference on Machine Learning (ICML 1996)*. 1996, pp. 284–292 (cited on p. 39).

[KU12] C. Komusiewicz and J. Uhlmann. "Cluster editing with locally bounded modifications". In: *Discrete Applied Mathematics* 160.15 (2012), pp. 2259–2270 (cited on pp. 106, 107).

[KW13] S. Kratsch and M. Wahlström. "Two edge modification problems without polynomial kernels". In: *Discrete Optimization* 10.3 (2013), pp. 193–199 (cited on p. 90).

[Ley+14] K. Leyton-Brown, H. H. Hoos, F. Hutter, and L. Xu. "Understanding the empirical hardness of NP-complete problems". In: *Communications of the ACM* 57.5 (2014), pp. 98–107 (cited on p. 149).

[Liu+12] Y. Liu, J. Wang, J. Guo, and J. Chen. "Complexity and parameterized algorithms for Cograph Editing". In: *Theoretical Computer Science* 461 (2012), pp. 45–54 (cited on p. 123).

[LM05] S. Langerman and P. Morin. "Covering things with things". In: *Discrete & Computational Geometry* 33.4 (2005), pp. 717–729 (cited on p. 26).

[Lok+14] D. Lokshtanov, N. S. Narayanaswamy, V. Raman, M. S. Ramanujan, and S. Saurabh. "Faster parameterized algorithms using linear programming". In: *ACM Transactions on Algorithms* 11.2 (2014), 15:1–15:31 (cited on pp. 90, 122).

[LY80] J. M. Lewis and M. Yannakakis. “The node-deletion problem for hereditary properties is NP-complete”. In: *Journal of Computer and System Sciences* 20.2 (1980), pp. 219–230 (cited on p. 90).

[Mni17] M. Mnich. “Big data algorithms beyond machine learning”. In: *KI - Künstliche Intelligenz* (2017), pp. 1–9 (cited on p. 148).

[MNS12] H. Moser, R. Niedermeier, and M. Sorge. “Exact combinatorial algorithms and experiments for finding maximum k-plexes”. In: *Journal of Combinatorial Optimization* 24.3 (2012), pp. 347–373 (cited on p. 90).

[MO04] S. C. Madeira and A. L. Oliveira. “Biclustering algorithms for biological data analysis: a survey”. In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 1.1 (2004), pp. 24–45 (cited on pp. 63–65).

[Mor+18] M. Morel, C. Achard, R. Kulpa, and S. Dubuisson. “Time-series averaging using constrained dynamic time warping with tolerance”. In: *Pattern Recognition* 74 (2018), pp. 77–89 (cited on p. 126).

[MR15] P. Moradi and M. Rostami. “A graph theoretic approach for unsupervised feature selection”. In: *Engineering Applications of Artificial Intelligence* 44.Supplement C (2015), pp. 33–45 (cited on p. 40).

[MR99] M. Mahajan and V. Raman. “Parameterizing above guaranteed values: MaxSat and MaxCut”. In: *Journal of Algorithms* 31.2 (1999), pp. 335–354 (cited on p. 90).

[MRS09] M. Mahajan, V. Raman, and S. Sikdar. “Parameterizing above or below guaranteed values”. In: *Journal of Computer and System Sciences* 75.2 (2009), pp. 137–153 (cited on p. 113).

[Mue+16] A. Mueen, N. Chavoshi, N. Abu-El-Rub, H. Hamooni, and A. Minnich. “AWarp: fast warping distance for sparse time series”. In: *Proceedings of the 16th IEEE International Conference on Data Mining (ICDM ’16)*. IEEE, 2016, pp. 350–359 (cited on p. 138).

[New03] M. E. J. Newman. “The structure and function of complex networks”. In: *SIAM Review* 45.2 (2003), pp. 167–256 (cited on p. 96).

[Ngu06] H. S. Nguyen. “Approximate boolean reasoning: foundations and applications in data mining”. In: *Transactions on Rough Sets V*. Vol. 4100. LNCS. Springer, 2006, pp. 334–506 (cited on pp. 65, 73).

[Nie06] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006 (cited on pp. 8, 149).

[Nie10] R. Niedermeier. “Reflections on multivariate algorithmics and problem parameterization”. In: *Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science (STACS ’10)*. Vol. 5. LIPIcs.

Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010, pp. 17–32 (cited on p. 148).

[NR05] F. Nicolas and E. Rivals. "Hardness results for the center and median string problems under the weighted and unweighted edit distances". In: *Journal of Discrete Algorithms* 3.2 (2005), pp. 390–415 (cited on p. 129).

[NS95] S. H. Nguyen and A. Skowron. "Quantization of real value attributes - rough set and boolean reasoning approach". In: *Proc. Second Joint Annual Conference on Information Sciences*. 1995, pp. 34–37 (cited on pp. 65, 73).

[Pap94] C. H. Papadimitriou. *Computational Complexity*. Addison Wesley, 1994 (cited on p. 7).

[Paw91] Z. Pawlak. *Rough Sets: Theoretical Aspects of Reasoning about Data*. Kluwer Academic, 1991 (cited on p. 41).

[Pet+16] F. Petitjean, G. Forestier, G. I. Webb, A. E. Nicholson, Y. Chen, and E. Keogh. "Faster and more accurate classification of time series by exploiting a novel dynamic time warping averaging algorithm". In: *Knowledge and Information Systems* 47.1 (2016), pp. 1–26 (cited on pp. 126, 129, 130).

[PG12] F. Petitjean and P. Gançarski. "Summarizing a set of time series by averaging: from Steiner sequence to compact multiple alignment". In: *Theoretical Computer Science* 414.1 (2012), pp. 76–91 (cited on pp. 126, 129, 130).

[PG17] J. Paparrizos and L. Gravano. "Fast and accurate time-series clustering". In: *ACM Transactions on Database Systems* 42.2 (2017), 8:1–8:49 (cited on pp. 126, 129).

[PKG11] F. Petitjean, A. Ketterlin, and P. Gançarski. "A global averaging method for dynamic time warping, with applications to clustering". In: *Pattern Recognition* 44.3 (2011), pp. 678–693 (cited on pp. 126, 129, 130).

[PPT16] C. Paul, A. Perez, and S. Thomassé. "Linear kernel for rooted triplet inconsistency and other problems based on conflict packing technique". In: *Journal of Computer and System Sciences* 82.2 (2016), pp. 366–379 (cited on pp. 102, 106).

[PSS09] F. Protti, M. D. da Silva, and J. L. Szwarcfiter. "Applying modular decomposition to parameterized cluster editing problems". In: *Theory of Computing Systems* 44.1 (2009), pp. 91–104 (cited on p. 107).

[Put+15] N. Puthiyedth, C. Riveros, R. Berretta, and P. Moscato. "A new combinatorial optimization approach for integrated feature selection using different datasets: a prostate cancer transcriptomic study". In: *PLoS ONE* 10.6 (2015), e0127702 (cited on p. 40).

[PW13] M. S. Payne and D. R. Wood. “On the general position subset selection problem”. In: *SIAM Journal on Discrete Mathematics* 27.4 (2013), pp. 1727–1733 (cited on pp. 15, 26, 27).

[Ric76] J. R. Rice. “The algorithm selection problem”. In: ed. by M. Rubinoff and M. C. Yovits. Vol. 15. Advances in Computers Supplement C. Elsevier, 1976, pp. 65–118 (cited on p. 149).

[RO09] I. Razgon and B. O’Sullivan. “Almost 2-SAT is fixed-parameter tractable”. In: *Journal of Computer and System Sciences* 75.8 (2009), pp. 435–450 (cited on pp. 90, 122).

[Rob55] H. Robbins. “A remark on Stirling’s formula”. In: *The American Mathematical Monthly* 62.1 (1955), pp. 26–29 (cited on p. 28).

[Rot51] K. F. Roth. “On a problem of Heilbronn”. In: *Journal of the London Mathematical Society* 1.3 (1951), pp. 198–204 (cited on p. 14).

[RS06] V. Raman and S. Saurabh. “Parameterized algorithms for feedback set problems and their duals in tournaments”. In: *Theoretical Computer Science* 351.3 (2006), pp. 446–458 (cited on p. 102).

[Rud17] A. G. Rudi. “An improved lower bound for general position subset selection”. In: *International Journal of Computing Science and Mathematics* 8.6 (2017), pp. 562–569 (cited on p. 15).

[SC78] H. Sakoe and S. Chiba. “Dynamic programming algorithm optimization for spoken word recognition”. In: *IEEE Transactions on Acoustics, Speech and Signal Processing* 26.1 (1978), pp. 43–49 (cited on p. 129).

[SDG16] S. Soheily-Khah, A. Douzal-Chouakria, and E. Gaussier. “Generalized k-means-based clustering for temporal data under weighted and kernel time warp”. In: *Pattern Recognition Letters* 75 (2016), pp. 63–69 (cited on p. 126).

[SJ18] D. Schultz and B. Jain. “Nonsmooth analysis and subgradient methods for averaging in dynamic time warping spaces”. In: *Pattern Recognition* 74.Supplement C (2018), pp. 340–358 (cited on pp. 126, 129, 135).

[SR92] A. Skowron and C. Rauszer. “The discernibility matrices and functions in information systems”. In: *Intelligent Decision Support—Handbook of Applications and Advances of the Rough Sets Theory*. Ed. by R. Slowinski. Kluwer Academic, 1992, pp. 331–362 (cited on p. 41).

[SS15] R. B. Sandeep and N. Sivadasan. “Parameterized lower bound and improved kernel for diamond-free edge deletion”. In: *Proceedings of the 10th International Symposium on Parameterized and Exact Computation (IPEC ’15)*. Vol. 43. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015, pp. 365–376 (cited on p. 123).

[SST04] R. Shamir, R. Sharan, and D. Tsur. “Cluster graph modification problems”. In: *Discrete Applied Mathematics* 144.1 (2004), pp. 173–182 (cited on pp. 90, 106).

[Tan96] L. Tan. “The group of rational points on the unit circle”. In: *Mathematics Magazine* 96.3 (1996), pp. 163–171 (cited on p. 18).

[TKM11] C. E. Tsourakakis, M. N. Kolountzakis, and G. L. Miller. “Triangle sparsifiers”. In: *Journal of Graph Algorithms and Applications* 15.6 (2011), pp. 703–726 (cited on p. 96).

[TSS05] A. Tanay, R. Sharan, and R. Shamir. “Biclustering algorithms: a survey”. In: *Handbook of Computational Molecular Biology*. Chapman & Hall/CRC, 2005 (cited on p. 64).

[Vaz01] V. V. Vazirani. *Approximation Algorithms*. Springer, 2001 (cited on p. 11).

[Wah07] M. Wahlström. “Algorithms, Measures and Upper Bounds for Satisfiability and Related Problems”. PhD thesis. Linköping University, Mar. 2007 (cited on pp. 30, 97).

[WS11] D. P. Williamson and D. B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011 (cited on p. 11).

[WUB13] S. Wulff, R. Urner, and S. Ben-David. “Monochromatic bi-clustering”. In: *Proceedings of the 30th International Conference on Machine Learning (ICML’13)*. Vol. 28 (2). JMLR Workshop and Conference Proceedings, 2013, pp. 145–153 (cited on p. 65).

[Yan81] M. Yannakakis. “Edge-deletion problems”. In: *SIAM Journal on Computing* 10.2 (1981), pp. 297–309 (cited on pp. 90, 97).

[Yap83] C. K. Yap. “Some consequences of non-uniform conditions on uniform classes”. In: *Theoretical Computer Science* 26.3 (1983), pp. 287–300 (cited on p. 10).

Schriftenreihe **Foundations of computing**
Hrsg.: Prof. Dr. Stephan Kreutzer, Prof. Dr. Uwe Nestmann, Prof. Dr. Rolf Niedermeier

ISSN 2199-5249 (print)
ISSN 2199-5257 (online)

01: Bevern, René van: Fixed-Parameter Linear-Time Algorithms for NP-hard Graph and Hypergraph Problems Arising in Industrial Applications. - 2014. - 225 S.
ISBN **978-3-7983-2705-4** (print) EUR **12,00**
ISBN **978-3-7983-2706-1** (online)

02: Nichterlein, André: Degree-Constrained Editing of Small-Degree Graphs. - 2015. - xiv, 225 S.
ISBN **978-3-7983-2705-4** (print) EUR **12,00**
ISBN **978-3-7983-2706-1** (online)

03: Bredereck, Robert: Multivariate Complexity Analysis of Team Management Problems. - 2015. - xix, 228 S.
ISBN **978-3-7983-2764-1** (print) EUR **12,00**
ISBN **978-3-7983-2765-8** (online)

04: Talmon, Nimrod: Algorithmic Aspects of Manipulation and Anonymization in Social Choice and Social Networks. - 2016. - xiv, 275 S.
ISBN **978-3-7983-2804-4** (print) EUR **13,00**
ISBN **978-3-7983-2805-1** (online)

05: Siebertz, Sebastian: Nowhere Dense Classes of Graphs. Characterisations and Algorithmic Meta-Theorems. - 2016. - xxii, 149 S.
ISBN **978-3-7983-2818-1** (print) EUR **11,00**
ISBN **978-3-7983-2819-8** (online)

06: Chen, Jiehua: Exploiting Structure in Computationally Hard Voting Problems. - 2016. - xxi, 255 S.
ISBN **978-3-7983-2825-9** (print) EUR **13,00**
ISBN **978-3-7983-2826-6** (online)

07: Arbach, Youssef: On the Foundations of dynamic coalitions. Modeling changes and evolution of workflows in healthcare scenarios - 2016. - xv, 171 S.
ISBN **978-3-7983-2856-3** (print) EUR **12,00**
ISBN **978-3-7983-2857-0** (online)

08: Sorge, Manuel: Be sparse! Be dense! Be robust! Elements of parameterized algorithmics. - 2017. - xvi, 251 S.
ISBN 978-3-7983-2885-3 (print) EUR **13,00**
ISBN 978-3-7983-2886-0 (online)

09: Dittmann, Christoph: Parity games, separations, and the modal μ-calculus. - 2017. - x, 274 S.
ISBN **978-3-7983-2887-7** (print) EUR **13,00**
ISBN **978-3-7983-2888-4** (online)

10: noch nicht erschienen

11: noch nicht erschienen

Universitätsverlag der TU Berlin

Fine-Grained Complexity Analysis of Some Combinatorial Data Science Problems

This thesis is concerned with analyzing the computational complexity of combinatorial problems arising in data science. The considered problems include selecting points in general position, combinatorial feature selection, co-clustering, graph clustering, and time series averaging. Adopting a parameterized viewpoint, a multivariate complexity analysis is conducted to chart the border between tractability and intractability for these practically motivated problems.
Intractability results include NP- and W-hardness, approximation hardness, as well as kernelization and running time lower bounds. Tractability is achieved by developing parameterized algorithms which solve specific problem instances in polynomial time for constant parameter values. The developed algorithms utilize various parameters such as the number of clusters, the Hamming distance of data points, or the solution value above a guarantee. The algorithms involve tools from combinatorics, data reduction, SAT-solving, and dynamic programming. While most of the studied problems turned out to be computationally hard to solve in general, this thesis reveals meaningful tractable special cases for each of the considered problems.

ISBN 978-3-7983-3003-0 (print)
ISBN 978-3-7983-3004-7 (online)

ISBN 978-3-7983-3003-0

http://verlag.tu-berlin.de