The Computational Hardness and Tractability of Restricted Seriation Problems on Inaccurate Data

René van Bevern

Diplomarbeit

Zur Erlangung des akademischen Grades
Diplom-Informatiker

Friedrich-Schiller-Universität Jena
Fakultät für Mathematik und Informatik
Theoretische Informatik I / Komplexitätstheorie

Eingereicht von René van Bevern
geb. am 16.11.1985 in Saalfeld

Betreuer: Dipl.-Bioinf. Christian Komusiewicz,
Dr. Hannes Moser, Prof. Dr. Rolf Niedermeier

Jena, April 2010

Revised version. Chapter 5 partly appeared at 36th International Workshop on Graph Theoretic Concepts in Computer Science (WG'10), Zarós, Crete, Greece, June 2010 [BKMN10].

Zusammenfassung. Seriation ist die Suche nach einer linearen Anordnung von Objekten, deren paarweise Ähnlichkeiten gegeben sind, sodass sich einander ähnliche Objekte in der Anordnung näher stehen als einander unähnliche. Seriation hat ihren Ursprung in der relativen Datierung archäologischer Artefakte und kommt darüber hinaus in den Sozialwissenschaften, in der Psychologie, zum Zwecke technischer Diagnosen und in der Ökologie zum Einsatz.

Existiert eine Anordnung von Objekten, sodass zwei beliebige Objekte sich nicht ähnlicher sind als alle dazwischen angeordneten Objektpaare, so heißt das gegebene Ähnlichkeitsmaß *robinsonsch* – nach W. S. Robinson, der mittels Seriation archäologische Artefakte datierte. Die zugehörige Ordnung heißt *kompatibel* mit dem gegebenen Ähnlichkeitsmaß. Falls ein Ähnlichkeitsmaß eine kompatible Anordnung seiner Objekte erlaubt – also robinsonsch ist – so ist diese in polynomieller Rechenzeit konstruierbar. Auf Grund von beispielsweise Messfehlern existiert jedoch nicht für jedes Ähnlichkeitsmaß eine kompatible Anordnung. Auf diesen nicht-robinsonschen Ähnlichkeitsmaßen liegt das Hauptaugenmerk dieser Arbeit. Wir untersuchen zwei Ansätze, um das Seriationsproblem für solche Ähnlichkeitsmaße zu lösen.

Der erste Ansatz besteht in der Suche nach Ausreißern in der gegebenen Objektmenge, ohne die die verbleibenden Objekte eine kompatible Anordnung erlauben. Wir zeigen, dass dieses Problem – genannt PARTIAL ROBINSONIAN SIMILARITY – NP-vollständig ist und dass für das Problem parametrisierte Algorithmen existieren (wobei die Anzahl der zu entfernenden Ausreißer als Parameter dient) wenn die Ähnlichkeiten je zweier Objekte nur Binärwerte annehmen. Dazu entwickeln wir einen parametrisierten Algorithmus für das eng verwandte Problem UNIT INTERVAL VERTEX DELETION, der insbesondere in den Sozialwissenschaften anwendbar ist.

Der zweite Ansatz ist die Suche nach einem robinsonschen Ähnlichkeitsmaß, das Werte aus einer Menge M annimmt und einem gegebenen Ähnlichkeitsmaß am nächsten ist. Indem man für dieses eine kompatible Anordnung bestimmt, kann man das Seriationsproblem für die ursprünglichen Eingabedaten lösen. Hierbei wird Nähe durch L_p-Normen modelliert und wir nennen das Problem ROBINSON (L_p, M)-FITTING. Wir zeigen die NP-Vollständigkeit des Problems im Falle $M = \mathbb{N}$ und $M = \{0, 1\}$. Wir zeigen eine Formulierung des Problems als gemischt-ganzzahliges lineares Optimierungsproblem.

Eine Variante des zweiten Ansatzes ist die Suche eines (Werte einer Menge M annehmenden) Ähnlichkeitsmaßes, welches einem gegebenen Maß am nächsten und kompatibel zu einer *gegebenen* Ordnung ist. Algorithmen für dieses RESTRICTED ROBINSON (L_p, M)-FITTING getaufte Problem lassen sich für Seriationsprobleme anwenden, bei denen Teile der gesuchten Ordnung bereits bekannt sind. Mittels konvexer Optimierung zeigen wir, dass ROBINSON $(L_p, \mathbb{R})$-FITTING innerhalb beliebiger Genauigkeit in Polynomzeit lösbar ist. Für RESTRICTED ROBINSON (L_∞, M)-FITTING mit $M \in \{\mathbb{R}, \mathbb{N}\}$ und für ROBINSON $(L_1, \{0, 1\})$-FITTING zeigen wir Linearzeitalgorithmen.

Abstract. Seriation is the task of finding a linear order of objects, whose pairwise similarities are given, such that similar objects are placed more closely in the order than dissimilar ones. The seriation problem originates from the task of relatively dating archaeological artifacts. Moreover, it is applied in the social sciences, in psychology, and technical diagnosis. Seriation is a special case of *ordination* as applied in the ecological sciences.

If there is an order of the objects such that two objects are not more similar to each other than any object pair ordered in between them, then the similarity associated with the objects is called *Robinsonian*—named after W. S. Robinson, who applied seriation to date archaeological artifacts. The corresponding order is called *compatible* with the given similarity. If a similarity is Robinsonian, that is, if it allows for a compatible order, then the order is computable in polynomial time. However, not all similarities allow for a compatible order. This, for example, can be due to inaccuracies in the input data. This work focuses on similarities that are not Robinsonian. We study two approaches to solve the seriation task in this case.

The first approach tries to identify a small subset of outliers of the given object set, so that the remaining objects allow for a compatible order. We call the problem Partial Robinsonian Similarity and show that it is NP-complete. By developing a fixed-parameter algorithm for the closely related Unit Interval Vertex Deletion problem, it is shown that Partial Robinsonian Similarity is fixed-parameter tractable (parameterized by the number of outliers) in case that the similarities between objects obtain only binary values. This case has applications in measuring indifference in the social sciences.

The second approach is searching for a Robinsonian similarity (obtaining values from a set M) that comes closest to a given similarity. For the resulting similarity, a compatible order can be constructed to solve the seriation problem for the original input data. Here, we measure closeness using L_p-norms and call the problem Robinson (L_p, M)-Fitting. We show that the problem is NP-complete for $M = \mathbb{N}$ and $M = \{0, 1\}$. We present a mixed integer program formulation of the problem.

As a variant of the second approach, we study the Restricted Robinson (L_p, M)-Fitting problem, which searches for a similarity (obtaining values from M) that comes closest to the given input similarity and that is compatible with a given order. Algorithms solving this problem are applicable to seriation problems where parts of the sought order are already known. Exploiting convex programming, we obtain that Restricted Robinson $(L_p, \mathbb{R})$-Fitting is polynomial-time solvable within arbitrary precision. Moreover, we present linear-time algorithms for Restricted Robinson (L_∞, M)-Fitting for $M \in \{\mathbb{N}, \mathbb{R}\}$, and for Restricted Robinson $(L_1, \{0, 1\})$-Fitting.

Contents

1 Introduction

Seriation is a data analysis problem that, given a set of objects and a similarity associated with each pair of objects, searches for an order of the given objects such that similar objects appear close in the order, while less similar objects get assigned distant positions in the order. The result could, for example, resemble a chronological order underlying the given objects.

Archaeological origin. The roots of the seriation problem lie in archaeology, where seriation is applied to chronologically order archaeological artifacts [Rob51]. As an introductory example, consider ancient graves found at a dig site. The task is to find the chronological order underlying the graves. Here, the similarity between two graves can be measured by the number of common objects found in them. Using seriation, the graves can be linearly ordered so that graves containing many common objects appear close in the order. As the frequency of specific materials' usage changes over time, it is likely that the age of graves containing many, for example, bronze artifacts is similar. As a consequence, the order found through seriation is likely to resemble the chronological order of the found graves. According to Kendall [Ken69], the archaeological roots of the seriation problem even date back to the archaeologist Flinders Petrie [Pet99], who already applied seriation in the year 1899.

Measuring indifference. Another application of seriation arises in the social sciences [Luc56, Rob78]. Here, for example, a person may express his or her preference of an object x over an object y. If neither object is preferred, the person is indifferent between x and y. The task is to assign a merit value $f(x)$ to each object x that expresses the person's preferences. That is, $f(x) < f(y)$ shall hold if and only if the person prefers x over y. However, modeling indifference between two objects x and y by choosing equal merit values $f(x) = f(y)$ yields a very restrictive model: if the person is indifferent between x and y and between y and z, we set $f(x) = f(y)$ and $f(y) = f(z)$ and therefore imply that the person is also indifferent between x and z. This makes the model arguable: a person might prefer x over z even if the difference between x and y and between y and z is not noticeable.

An approach to circumvent this problem has been described by Luce [Luc56]:

d	x_1	x_2	x_3	x_4
x_1	3	2	1	0
x_2	2	3	2	1
x_3	1	2	3	2
x_4	0	1	2	3

Figure 1.1: A Robinsonian similarity with rows and columns arranged according to a compatible order. The entry in the i-th row and k-th column displays $d(x_i, x_k)$.

instead of letting indifference correspond to equal merit values, he measures indifference by *close* merit values. Now, the task is, given a set of objects and a *just noticeable difference* $\delta \geq 0$, to associate with each object x a merit value $f(x)$ such that $|f(x) - f(y)| \leq \delta$ holds if and only if a person expresses indifference between x and y. The function f can be thought of as an indifference measure. Observe how similar this task is to the seriation task: in the order $x \preceq y$ induced by $f(x) \leq f(y)$, two objects x and y appear close to each other if and only if the person is indifferent between x and y. Otherwise, they get assigned distant positions. The difference to the general seriation problem lies in the measure of similarity between two objects: while in general, the similarity between two objects may obtain any value, the indifference between two objects either holds or not.

Besides archaeology and the social sciences, seriation has been used in psychology [Hub74] and technical diagnosis [Nök91]. In ecology, seriation is applied under the name *ordination* [Gau82]. The practical importance of seriation raised various packages providing heuristics and exact algorithms for seriation [HHB08] for the R programming language[1], which is frequently used for statistical programming and data mining.

Modeling Seriation. One way to model seriation is based on the seriation method that Robinson [Rob51] employed to date archaeological artifacts: let X be a set of objects and let $d: X \times X \to \mathbb{R}$ be the *similarity* between two objects, where $d(x,x) \geq d(x,y) = d(y,x) \geq 0$ holds for every $x, y \in X$. In this setting, the seriation task is finding a linear order $\preceq$ on X such that $x \preceq y \preceq z$ implies $d(x,z) \leq d(x,y)$ and $d(x,z) \leq d(y,z)$. If such an order $\preceq$ exists, we call d Robinsonian and say that $\preceq$ is *compatible* with d. It is possible to construct a compatible order for a similarity in $O(|X|^3)$ time [CF97], if one exists. To give an intuition of Robinsonian similarities, consider a similarity $d: X \times X \to \mathbb{R}$ that allows for a compatible order $x_1 \preceq x_2 \preceq \ldots \preceq x_n$ on X. Then, the matrix $A := (a_{ij})$ with $a_{ij} = d(x_i, x_j)$ is a symmetric matrix

[1]http://www.r-project.org/

whose entries never increase when moving away from the main diagonal; this is because $i \leq j \leq k$ implies $a_{ik} \leq \min\{a_{ij}, a_{jk}\}$. Figure 1.1 shows a Robinsonian similarity in this form.

Section 1.1 describes ways to transform general similarities into Robinsonian similarities. These constitute the focus of this work. Section 1.2 presents relations between the considered similarity modification problems and known graph modification problems—relations that we will frequently exploit. Section 1.3 shows an application to the Consecutive Ones Submatrix problem.

1.1 Seriation on Inaccurate Data

As Roberts [Rob79] points out, real-world data rarely fits a model perfectly, that is, we might measure similarities between objects such that the resulting similarity does not allow for a compatible order. We investigate two ways to transform general similarities into Robinsonian similarities by a minimum of modifications, so that the seriation problem is efficiently solvable on the resulting Robinsonian similarities [MR84, CF97, ABH99]. This is the main objective of Chapter 4.

Suppose we are given a similarity $d\colon X \times X \to \mathbb{R}$ that is not allowing for a compatible order. Under the assumption that some plausible order should underlie the objects in X, the nonexistence of a compatible order for d is likely due to inaccuracies in the input data. On the one hand, the inaccuracies in the input data might be outliers in the object set X. On the other hand, they might come from inexact measurement of the similarities between objects [Rob79].

If we find a small set of outliers, we can exclude them from the seriation task in order to solve it, or we might reconsider their similarities to other objects and correct the similarity data. Thus, one approach to make similarities Robinsonian is trying to find a small subset S of outliers in X such that d allows for a compatible order if we remove the outliers in S from X:

Partial Robinsonian Similarity
Input: A similarity $d\colon X \times X \to \mathbb{R}$ and a natural number k.
Question: Is $d|_{X\setminus S}$ Robinsonian for some subset $S \subseteq X$ with $|S| \leq k$?

Here, $d|_{X\setminus S}\colon X\setminus S \times X\setminus S \to \mathbb{R}$ is the similarity with $d|_{X\setminus S}(x,y) = d(x,y)$ for all $x, y \in X\setminus S$. If we require the input to be a $\{0,1\}$-*similarity* $d\colon X \times X \to \{0,1\}$, then we call the problem Partial Robinsonian $\{0,1\}$-Similarity.

Another approach to transform a similarity $d\colon X \times X \to \mathbb{R}$ into a Robinsonian similarity is to search for a Robinsonian similarity that comes "closest" to d by minimally changing the similarity between objects. Here, the distance between two

d	x_1	x_2	x_3	x_4
x_1	3	2	1	0
x_2	2	3	2	1
x_3	1	2	3	2
x_4	0	1	2	3

d'	x_1	x_2	x_3	x_4
x_1	3	0	0	0
x_2	0	3	2	1
x_3	0	2	3	2
x_4	0	1	2	3

Figure 1.2: The order $x_1 \preceq x_2 \preceq x_3 \preceq x_4$ is compatible with the similarities d and d'; the values never increase when moving away from the main diagonals. Here, $\|d-d'\|_1$ is the sum of the distances between $d(x,y)$ and $d'(x,y)$ for objects above the main diagonal; one has $\|d-d'\|_1 = 3$. Similarly, $\|d-d'\|_2^2 = 5$ and $\|d-d'\|_\infty = 2$.

similarities $d, d' : X \times X \to \mathbb{R}$ is measured using the L_p-norm for $p \geq 1$ (see Figure 1.2 for examples), which we define as

$$\|d-d'\|_p := \left(\frac{1}{2} \sum_{\substack{x \neq y \\ x,y \in X}} |d(x,y) - d'(x,y)|^p \right)^{\frac{1}{p}}.$$

The sum is multiplied with 1/2 because it counts $|d(x,y) - d'(x,y)|^p$ twice for each $x, y \in X$. This is due to the symmetry of d. As a special case, we define the L_∞-norm as $\|d-d'\|_\infty := \max_{x,y \in X} |d(x,y) - d'(x,y)|$. We write $\|d-d'\|_p^p$ instead of $(\|d-d'\|_p)^p$. Observe that for two similarities $d, d' : X \times X \to \mathbb{R}$, if we consider d as fixed, then d' is a global minimum for the (always nonnegative) distance $\|d-d'\|_p$ if and only if it is a global minimum for the (also nonnegative) distance $\|d-d'\|_p^p$. Thus, we can safely choose to minimize the easier computable $\|d-d'\|_p^p$ instead of $\|d-d'\|_p$. We arrive at the following decision problem:

Robinson (L_p, M)-Fitting
Input: A similarity $d : X \times X \to M$ and a number $\varepsilon > 0$.
Question: Is there a Robinsonian similarity $d' : X \times X \to M$ with $\|d-d'\|_p^p \leq \varepsilon$ for $p < \infty$ and $\|d-d'\|_\infty \leq \varepsilon$ for $p = \infty$?

1.2 Seriation and Unit Interval Graphs

This section presents connections between graph modification problems and the similarity modification problems described in Section 1.1. These connections are mainly based on connections between graph theory and seriation that have been established by Roberts [Rob69, Rob78]. We will frequently exploit them in proofs of hardness and tractability.

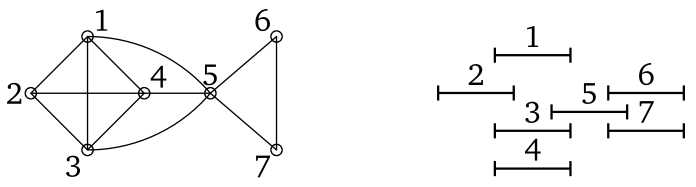


Figure 1.3: A unit interval graph and its interval representation.

Reconsider the introductory example of measuring indifference: for an object set X, a person may express indifference between objects in X. Let the set I contain sets of the form $\{x, y\}$ if a person is indifferent between two objects x and y in X. A task described by Luce [Luc56] is to assign a merit value $f(x)$ to each object $x \in X$ such that $|f(x) - f(y)| \leq \delta$ if and only if $\{x, y\} \in I$. Here, δ denotes the *just noticeable difference* between any two objects. If such an assignment of merit values exists, then the graph (X, I) is called *indifference graph*. Roberts [Rob69, Rob78] has shown that indifference graphs are equivalent to *proper interval graphs* and *unit interval graphs*. A unit interval graph is a graph whose vertices can be mapped to unit-length intervals on the real line such that an edge exists between two vertices if and only if the corresponding intervals intersect. Figure 1.3 shows a unit interval graph with its interval representation.

Roberts [Rob79] pointed out that the indifference expressed by a person might not fit Luce's model [Luc56]. In this case, (X, I) is not a unit interval graph. As suggested in Section 1.1, one way to solve this problem is searching for a minimum set S of outliers in X such that there is a merit function that, at least for the objects $x, y \in X \setminus S$, satisfies $|f(x) - f(y)| \leq \delta$ if and only if $\{x, y\} \in I$. This is equivalent to the following graph modification problem:

Unit Interval Vertex Deletion
Input: A graph $G = (V, E)$ and a natural number k.
Question: Is $G - S$ a unit interval graph for some vertex set $S \subseteq V$ with $|S| \leq k$?

Here, $G - S$ denotes the graph G with the vertices in S removed. In Chapter 5, we develop a fixed-parameter algorithm for Unit Interval Vertex Deletion parameterized by k, showing one possible way to make empirical indifference data compatible with Luce's [Luc56] mathematical indifference model.

Roberts [Rob69, Rob78] has shown that unit interval graphs can also be used to characterize Robinsonian similarities. To this end, the concept of compatible orders is extended to graphs: a graph has a compatible order $\preceq$ of its vertices if, for all vertices u, v, w with $u \preceq v \preceq w$, the existence of the edge $\{u, w\}$ implies the existence of the edges $\{u, v\}$ and $\{v, w\}$. A similarity $d : X \times X \to \mathbb{R}$ is Robinsonian if and only if there is an order $\preceq$ that is compatible with all graphs G_ε, where G_ε is a graph with the vertex set X such that $\{x, y\}$ is an edge of G_ε if and only if $d(x, y) \geq \varepsilon$ [Rob78]. Observe that the graphs G_ε used in this characterization have a loop at each vertex, that is, for each vertex v, there is an edge between v and itself. According to this characterization, a $\{0,1\}$-similarity $d: X \times X \to \{0,1\}$ is Robinsonian if and only if there is an order compatible with the corresponding graphs G_0, G_1, and G_2. Observe that G_0 has all possible edges and G_2 has no edges: all orders are compatible with G_0 and G_2. As a consequence, d is Robinsonian if and only if G_1 allows for a compatible order, which, as shown by Roberts [Rob69, Rob78], is equivalent to G_1 (with the loops at each vertex removed) being a unit interval graph. This shows:

Proposition 1.1. *A similarity $d: X \times X \to \{0,1\}$ is Robinsonian if and only if the graph $G = (X, E)$ is a unit interval graph, where $E := \{\{x, y\} \mid x, y \in X, x \neq y, \text{ and } d(x, y) = 1\}$.*

From Proposition 1.1, it is easily seen that Unit Interval Vertex Deletion is equivalent to Partial Robinsonian $\{0,1\}$-Similarity. Moreover, Section 4.2 will show that Robinson $(L_1, \{0,1\})$-Fitting is equivalent to the following problem:

Unit Interval Editing
Input: A graph $G = (V, E)$ and a natural number k.
Question: Is there a unit interval graph $G' = (V, E')$ with $|E \Delta E'| \leq k$?

Chapter 4 uses these relations to show that Partial Robinsonian Similarity and Robinson $(L_p, \mathbb{N})$-Fitting are NP-complete and that Partial Robinsonian $\{0,1\}$-Similarity is fixed-parameter tractable parameterized by the number of sought outliers.

1.3 Relation to Consecutive Ones Submatrix

Chapter 5 presents a fixed-parameter algorithm for Unit Interval Vertex Deletion parameterized by the allowed number of vertex deletions. This can be exploited to show fixed-parameter tractability of a problem related to the Consecutive Ones Submatrix problem, which is explained in the following.

A matrix has the consecutive ones property (for columns) if its rows can be permuted such that in each column the ones appear consecutively. The consecutive ones property has applications in scheduling, information retrieval, computational biology, and railway optimization. Refer to Dom [Dom08, Dom09] for examples of applications. The NP-complete [HG02] Consecutive Ones Submatrix problem is

Consecutive Ones Submatrix
Input: A binary matrix A and a natural number k
Question: Can A be transformed into a matrix having the consecutive ones property by at most k row deletions?

While the Consecutive Ones Submatrix problem has been studied in terms of parameterized complexity with respect to other parameters [Dom08, DGN10], it is unknown whether Consecutive Ones Submatrix is fixed-parameter tractable with respect to the parameter k. However, we can show that the following variant of Consecutive Ones Submatrix is fixed-parameter tractable:

Symmetric Consecutive Ones Submatrix
Input: A binary symmetric matrix A with ones on the main diagonal.
Parameter: A natural number k.
Question: Can A be transformed into a matrix having the consecutive ones property by k simultaneous row and column deletions?

Here, a simultaneous row and column deletion means that row i is deleted if and only if column i is deleted. To see that Symmetric Consecutive Ones Submatrix is fixed-parameter tractable, we exploit the fixed-parameter algorithm for Unit Interval Vertex Deletion shown in Chapter 5 and the fact that a graph is a unit interval graph if and only its augmented adjacency matrix has the consecutive ones property [Rob69, Rob78]; the augmented adjacency matrix of a graph $G = (\{v_1, v_2, \dots, v_n\}, E)$ is a matrix $A = (a_{ij})$ with $a_{ij} = 1$ if $\{v_i, v_j\} \in E$ or $i = j$, and $a_{ij} = 0$ otherwise. Observe that deleting a vertex v_i from G is equivalent to the simultaneous deletion of the i-th row and i-th column from its augmented adjacency matrix. Therefore, Symmetric Consecutive Ones Submatrix can be interpreted as taking as input the augmented adjacency matrix of an arbitrary graph and asking whether it can be transformed into the augmented adjacency matrix of a unit interval graph by deleting at most k vertices. It follows that Symmetric Consecutive Ones Submatrix is equivalent to Unit Interval Vertex Deletion. Chapter 5 presents a fixed-parameter algorithm for Unit Interval Vertex Deletion parameterized by the number of allowed vertex deletions. From this, it follows that Symmetric Consecutive Ones Submatrix is fixed-parameter tractable.

2 Prerequisites

This chapter gives the prerequisites necessary to understand the subsequent chapters of this work. The reader is expected to be familiar with the concept of NP-completeness. A detailed introduction into the concept is given by Garey and Johnson [GJ79].

2.1 Orders

An order $\sqsubseteq$ on a set X is a subset of the Cartesian product $X \times X$. For two objects $x, y \in X$, we write $x \sqsubseteq y$ if (x, y) is contained in $\sqsubseteq$. A pair of objects $x, y \in X$ is *incomparable* with respect to $\sqsubseteq$ if neither $x \sqsubseteq y$ nor $y \sqsubseteq x$ holds. An order $\sqsubseteq$ on X is *reflexive* if for every $x \in X$, it holds that $x \sqsubseteq x$. It is *transitive* if for every $x, y, z \in X$, it holds that $x \sqsubseteq y$ and $y \sqsubseteq z$ implies $x \sqsubseteq z$. It is *antisymmetric* if for every $x, y \in X$, $x \sqsubseteq y$ and $y \sqsubseteq x$ implies $x = y$.

A *partial order* is a reflexive, transitive, and antisymmetric order. A *total* order is a partial order for which no incomparable pairs exist. By convention, we write $\sqsubseteq$ for partial orders and $\preceq$ for total orders.

For a total order $\preceq$ on a set X, a *segment* of $\preceq$ is a set $[x, z] := \{y \mid x \preceq y \preceq z\}$ for two elements $x, z \in X$. If a total order $\preceq$ is a superset of a partial order $\sqsubseteq$, then $\preceq$ is a *linear extension* of $\sqsubseteq$. For a subset $S \subseteq X$, an object $x \in S$ is the *minimum object* in S with respect to $\preceq$ if all objects $y \in S$ satisfy $x \preceq y$. An object $z \in S$ is the *maximum object* in S with respect to $\preceq$ if all objects $y \in S$ satisfy $y \preceq z$.

Examples. On $\mathbb{N}$, the order $\leq$ is a total order. The segments of $\leq$ are the closed intervals $[a, b]$ for two numbers $a, b \in \mathbb{N}$. The minimum object in $[a, b]$ with respect to $\leq$ is a, the maximum object is b. The order "a divides b" with $a, b \in \mathbb{N}$ is a partial order. With respect to the latter, $(3, 5)$ is an incomparable pair because three does not divide five and five does not divide three.

2.2 Seriation

For a set X, a *similarity* is a function $d\colon X \times X \to \mathbb{R}$ satisfying $d(x,x) \geq d(x,y) = d(y,x) \geq 0$ for every $x, y \in X$. A total order $\preceq$ on X is *compatible* with the similarity d if $x \preceq y \preceq z$ implies $d(x,z) \leq d(x,y)$ and $d(x,z) \leq d(y,z)$. If such an order $\preceq$ exists, we call d Robinsonian. A *dissimilarity* is a function $d'\colon X \times X \to \mathbb{R}$ with $d'(x,y) = d'(y,x) \geq d'(x,x) = 0$ for all $x, y \in X$. An order $\preceq$ on X is *compatible* with the dissimilarity d' if $x \preceq y \preceq z$ implies $d'(x,z) \geq \max\{d'(x,y), d'(y,z)\}$. If a dissimilarity d allows for a compatible order, we call it *Anti-Robinsonian*.

For a (dis)similarity $d\colon X \times X \to \mathbb{R}$ and a set $S \subseteq X$, we let $d|_S\colon S \times S \to \mathbb{R}$ denote the (dis)similarity with $d|_S(x,y) = d(x,y)$ for all $x, y \in S$. For two (dis)similarities $d, d'\colon X \times X \to \mathbb{R}$ and a number $\lambda \in \mathbb{R}$, $d \pm d'$ is the (dis)similarity that maps (x,y) to $d(x,y) \pm d'(x,y)$ and $\lambda \pm d'$ is the (dis)similarity that maps (x,y) to $\lambda \pm d'(x,y)$. Note that $d - d'$ and $\lambda - d'$ do not necessarily satisfy $(d-d')(x,y) \geq 0$ or $(\lambda - d')(x,y) \geq 0$ for all $x, y \in X$, respectively.

For a (dis)similarity $d\colon X \times X \to \mathbb{R}$ not necessarily satisfying $d(x,y) \geq 0$ for all $x, y \in X$, we define the L_p-norms

$$\|d\|_p := \left(\frac{1}{2}\sum_{\substack{x \neq y\\ x,y \in X}} |d(x,y)|^p\right)^{\frac{1}{p}} \qquad \text{and} \qquad \|d\|_\infty := \max_{\substack{x \neq y\\ x,y \in X}} |d(x,y)|.$$

We only consider L_p norms for $p \geq 1$. These satisfy the triangle inequality $\|d + d'\|_p \leq \|d\|_p + \|d'\|_p$. We write $\|d\|_p^p$ instead of $(\|d\|_p)^p$.

2.3 Graph Theory

In this work, we only consider *undirected* simple graphs $G = (V, E)$, where V is the set of vertices and E is the set of edges. Throughout this work, we use $n := |V|$ and $m := |E|$. We call two vertices $v, w \in V$ *adjacent* or *neighbors* if $\{v, w\} \in E$. The *(open) neighborhood* $N(v)$ of a vertex $v \in V$ is the set of vertices that are adjacent to v. A *clique* is a graph in which every two distinct vertices are adjacent. A set of pairwise nonadjacent vertices is an *independent set*. The complement graph $\overline{G}$ of G is the graph on the same vertex set as G having an edge $\{v, w\}$ if and only if G does not have the edge $\{v, w\}$. Two graphs H and G are isomorphic if there is a bijective function f that maps vertices of H to vertices of G such that $\{v, w\}$ is an edge of H if and only if $\{f(v), f(w)\}$ is an edge of G.

Induced subgraphs. For a set of vertices $V' \subseteq V$, the *induced subgraph* $G[V']$ is the graph with the vertex set V' and the edge set $\{\{v, w\} \in E \mid v, w \in V'\}$. For $V' \subseteq V$, we use $G - V'$ as an abbreviation for $G[V \setminus V']$. A graph property Π is *hereditary* if every induced subgraph of a graph satisfying Π also satisfies Π. For a graph F, a graph G is *F-free* if it does not contain an induced subgraph that is isomorphic to F. We also say that G contains no induced F.

Connectivity. A *path* P from $v_1 \in V$ to $v_l \in V$ is a sequence $(v_1, v_2, \ldots, v_l) \in V^l$ with $\{v_i, v_{i+1}\} \in E$ for $i \in \{1, \ldots, l-1\}$. The path P *visits* the vertices $v_1, \ldots, v_l$. An edge $\{v_i, v_j\}$ for $|i - j| > 1$ is a *chord*. If $i \neq j$ implies $v_i \neq v_j$, then P is *simple*. If, additionally, P has no chords, then P is *induced*. An *(induced or chordless) cycle* is an (induced) path with an added edge $\{v_1, v_l\} \in E$. We use C_l to denote induced cycles of length l. A *hole* is an induced cycle of length greater than three. Two vertices $v, w \in V$ are *connected* in G if there is a path from v to w in G. A *vertex-cut* between v and w in G is a set $C \subseteq V$ such that v and w are not connected in $G - C$. A *connected component* of G is a maximal induced subgraph of G in which every pair of vertices is connected.

2.4 Parameterized Complexity

Parameterized complexity analysis aims at a multivariate complexity analysis of problems without giving up the demand for finding optimal solutions [DF99, FG06, Nie06]. For a finite alphabet Σ, a *parameterized problem* is a language $L \subseteq \Sigma^* \times \mathbb{N}$. The second component is called the *parameter* of the problem. A parameterized problem L is *fixed-parameter tractable* if it is decidable in $f(k)|x|^{O(1)}$ time whether $(x, k) \in L$, where f is a computable function only depending on k. The corresponding complexity class is *FPT*. An algorithm that decides $(x, k) \in L$ in $f(k)|x|^{O(1)}$ time is called *fixed-parameter algorithm*.

Parameter-Preserving Reductions. A *parameter-preserving reduction* from a parameterized problem L to a parameterized problem L' is a polynomial-time computable function $f : L \to L', (x, k) \mapsto (x', k)$ such that $(x, k) \in L$ if and only if $(x', k) \in L'$. Note that this definition is only a special case of general parameterized reductions found in the literature of parameterized complexity [DF99, FG06, Nie06] and of the polynomial-time many-one reductions from classical complexity theory [GJ79].

Bounded Search Tree Algorithms. Bounded search tree algorithms are one way to obtain fixed-parameter algorithms, which we explain by means of VERTEX COVER:

VERTEX COVER
Input: A graph G and a natural number k.
Question: Is there a set S with $|S| \leq k$ such that each edge of G has one of its endpoints in S?

What follows is a simple bounded search tree algorithm that solves VERTEX COVER in $O(2^k m)$ time, that is, the resulting algorithm is a fixed-parameter algorithm for VERTEX COVER parameterized by k. The bounded search tree algorithm for VERTEX COVER is a recursive algorithm that takes as input the graph G, the natural number k and a set S (initially $S = \emptyset$) and works as follows: if $|S| \leq k$, then search for an edge $\{u, v\}$ in G such that neither u nor v are in S. If no such edge exists, then (G, k) is a yes-instance and the algorithm can return S. Otherwise, as S shall contain a vertex of every edge in G, at least one of u or v must be put into S. Thus, we recursively apply the algorithm: (G, k) is a yes-instance if and only if one of the recursive calls with $(G, k, S \cup \{u\})$ and $(G, k, S \cup \{v\})$ returns that (G, k) is a yes-instance. We say that we recursively *branch* into the cases of putting u or v into S. If $|S| \geq k$, we stop making further recursive calls.

Each call of the algorithm makes two further recursive calls. Because $|S|$ increases with each nested recursive call and because a call does not make further recursive calls if $|S| \geq k$, the recursion depth is at most k. The corresponding recursive call tree, called *search tree*, has $O(2^k)$ *nodes*, each corresponding to a call of the algorithm. In each search tree node, an edge that has no endpoint in S is sought. This works in $O(m)$ time. Thus, each of the $O(2^k)$ nodes of the search tree can be processed in $O(m)$ time, which results in a total running time of $O(2^k m)$ for the search tree algorithm for VERTEX COVER.

3 Related Work and New Results

Section 3.1 summarizes related work. New results are summarized in Section 3.2. While this work focuses on Robinsonian similarities instead of Anti-Robinsonian dissimilarities because of their more obvious connections to unit interval graphs and the consecutive ones property, many results in literature concern Anti-Robinsonian dissimilarities. These are sometimes also called "Robinsonian". Our results one-to-one transfer to Anti-Robinsonian dissimilarities.

3.1 Related Work

Seriation. The seriation model used in this work has been established by Robinson [Rob51]. Anti-Robinsonian dissimilarities are recognizable in polynomial time [MR84, CF97]. Observing that a similarity $d\colon X \times X \to \mathbb{R}$ is Robinsonian if and only if $d' := \|d\|_\infty - d$ with $d'(x,x)$ for $x \in X$ set to zero is an Anti-Robinsonian dissimilarity, these algorithms are applicable to the recognition of Robinsonian similarities. Atkins et al. [ABH99] showed how to directly recognize Robinsonian similarities employing a spectral analysis of the corresponding similarity matrix.

Chepoi et al. [CFS09] have shown that the Anti-Robinson $(L_\infty, \mathbb{R})$-Fitting problem, which for a given dissimilarity $d\colon X \times X \to \mathbb{R}$ asks whether there is an Anti-Robinsonian dissimilarity $d'\colon X \times X \to \mathbb{R}$ with $\|d - d'\|_\infty \le \varepsilon$, is NP-hard. This is easily seen to imply the NP-hardness of Robinson $(L_\infty, \mathbb{R})$-Fitting by exploiting that a dissimilarity $d\colon X \times X \to \mathbb{R}$ is Anti-Robinsonian if and only if $\|d\|_\infty - d$ is a Robinsonian similarity. Chepoi and Seston [CS09] have shown a factor-16 approximation algorithm for Anti-Robinson $(L_\infty, \mathbb{R})$-Fitting. Barthélemy and Brucker [BB01] have shown NP-completeness of a problem related to Anti-Robinson $(L_p, \mathbb{N})$-Fitting; their problem description poses stricter requirements on the sought dissimilarity d'. As these stricter requirements are exploited in their NP-completeness proof, their NP-completeness result does not generalize to (Anti-)Robinson $(L_p, \mathbb{N})$-Fitting. Chepoi et al. [CFS09] have shown that, given a dissimilarity $d\colon X \times X \to \mathbb{R}$ and a total order $\preceq$, a dissimilarity $d'\colon X \times X \to \mathbb{R}$ that is compatible with $\preceq$ and minimizes $\|d - d'\|_\infty$ is computable in polynomial time. A straightforward transformation of their proof into an algorithm yields an algorithm that uses $O(|X|^4)$ real-number additions and comparisons.

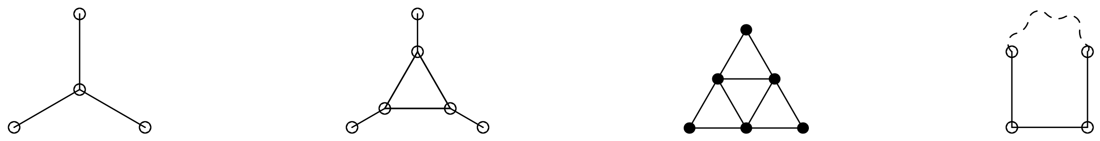

(a) Claw **(b)** Net **(c)** Tent **(d)** Holes

Figure 3.1: The (infinitely many) forbidden induced subgraphs for unit interval graphs. Holes are induced cycles of arbitrary length greater than three.

The consecutive ones property is recognizable in linear time [BL76, Hsu02]. The CONSECUTIVE ONES SUBMATRIX problem is NP-complete [HG02] and has been subject to parameterized complexity studies [Dom08, Dom09, DGN10].

Unit Interval Graphs. On unit interval graphs, many classical NP-hard problems [GJ79] are solvable in linear time. This includes the INDEPENDENT SET and CLIQUE problems [GLL82], the DOMINATING SET problem [Cha98], as well as the HAMILTON CYCLE and HAMILTON PATH problems [PD03]. Roberts has shown that unit interval graphs, defined as graphs whose vertices can be mapped to unit-length intervals on the real line such that two vertices are adjacent if and only if their corresponding intervals intersect, are equivalent to proper interval graphs and indifference graphs [Rob69, Rob78]. Wegner [Weg67] has shown that a graph is a unit interval graph if and only if it contains none of the infinitely many graphs shown in Figure 3.1 as induced subgraph [Rob78, BLS99]. Unit interval graphs can be recognized in linear time [CKN$^+$98, PD03, HH05].

UNIT INTERVAL EDITING has been shown to be NP-complete by Burzyn et al. [BBD06] even on cubic planar graphs. UNIT INTERVAL VERTEX DELETION is also NP-complete: the forbidden induced subgraph characterization of unit interval graphs implies that being a unit interval graph is a hereditary graph property. Because not all graphs are unit interval graphs (for examples, refer to Figure 3.1), and because unit interval graphs are recognizable in linear time [CKN$^+$98, PD03, HH05], this implies that the UNIT INTERVAL VERTEX DELETION problem is NP-complete by a general result due to Lewis and Yannakakis [LY80]. Marx [Mar09] has shown a fixed-parameter algorithm for the following related problem:

CHORDAL VERTEX DELETION
Input: A graph G.
Parameter: A natural number k.
Question: Is there a vertex set S with $|S| \leq k$ such that $G - S$ is hole-free?

3.2 New Results

Chapter 4, Seriation. Exploiting graph-theoretic results, we show that PARTIAL ROBINSONIAN SIMILARITY is NP-complete even if the input is restricted to $\{0,1\}$-similarities. In this case, we show that PARTIAL ROBINSONIAN $\{0,1\}$-SIMILARITY is solvable in $O((14k+14)^{k+1} \cdot k|X|^6)$ time by exploiting a fixed-parameter algorithm for the closely related UNIT INTERVAL VERTEX DELETION problem that is developed in Chapter 5. We show difficulties that arise when trying to generalize the used algorithm to the general PARTIAL ROBINSONIAN SIMILARITY problem.

Again exploiting graph-theoretic results, we show that ROBINSON $(L_p, \mathbb{N})$-FITTING and ROBINSON $(L_1, \{0,1\})$-FITTING are NP-complete. We give a formulation of ROBINSON $(L_1, \mathbb{R})$-FITTING as mixed integer program. The mixed integer program can easily be modified to solve ROBINSON (L_p, M)-FITTING for all combinations of $p \in \{1, \infty\}$ and $M \in \{\mathbb{N}, \mathbb{R}, \{0,1\}\}$. The mixed integer program formulation might be useful to solve these problem variants using sophisticated mixed integer solver software.

Finally, we consider the problem of, given a similarity $d\colon X \times X \to \mathbb{R}$ and a total order $\preceq$, finding a similarity $d'\colon X \times X \to \mathbb{R}$ that is compatible with $\preceq$ and minimizes $\|d - d'\|_p^p$ for $1 \leq p < \infty$. Algorithms for this problem are applicable to solve seriation problems where parts of the sought order are already known. We use convex programming techniques to show that the problem is solvable in polynomial time within arbitrary precision. For the special case that seeks to minimize $\|d - d'\|_1$ under the constraint that d' is a $\{0,1\}$-similarity, we present a dynamic programming algorithm running in $O(|X|^2)$ time, which is linear in the input size. Finally, we develop a simple combinatorial algorithm that minimizes $\|d - d'\|_\infty$ within precision $\bar{\varepsilon}$ using $O(|X|^2 \log \|d\|_\infty \bar{\varepsilon}^{-1})$ real-number additions and comparisons.

Chapter 5, Unit Interval Vertex Deletion. We present an easy-to-implement fixed-parameter algorithm that solves UNIT INTERVAL VERTEX DELETION in $O((14k+14)^{k+1} \cdot kn^6)$ time. The algorithm can be applied to solve PARTIAL ROBINSONIAN $\{0,1\}$-SIMILARITY as shown in Chapter 4. Moreover, we show that the problem is NP-hard even on {claw, net, tent}-free graphs. In this case, UNIT INTERVAL VERTEX DELETION is equivalent to CHORDAL VERTEX DELETION.

4 Seriation

This chapter studies approaches to transform similarities into Robinsonian similarities, on which the seriation problem is efficiently solvable. Herein, the focus is on the approaches described in Chapter 1: making similarities Robinsonian by removal of outliers or by minimally changing the similarities between objects. All of this chapters' results are summarized in a table on page 41. Section 4.1 first focuses on:

Partial Robinsonian Similarity
Input: A similarity $d\colon X \times X \to \mathbb{R}$ and a natural number k.
Question: Is $d|_{X\setminus S}$ Robinsonian for some subset $S \subseteq X$ with $|S| \leq k$?

Section 4.1 shows that Partial Robinsonian Similarity is NP-hard even if the input is restricted to $\{0,1\}$-similarities. We present a fixed-parameter algorithm to solve the problem restricted to $\{0,1\}$-similarities in $O((14k+14)^{k+1}k|X|^6)$ time, which as shown in Section 1.2 has applications in the measurement of indifference. Section 4.2 shows the NP-completeness of Robinson $(L_1,\{0,1\})$-Fitting and Robinson $(L_p,\mathbb{N})$-Fitting for $p<\infty$.

Robinson (L_p, M)-Fitting
Input: A similarity $d\colon X \times X \to M$ and a number $\varepsilon > 0$.
Question: Is there a Robinsonian similarity $d'\colon X \times X \to M$ with $\|d-d'\|_p^p \leq \varepsilon$ for $p<\infty$ and $\|d-d'\|_\infty \leq \varepsilon$ for $p=\infty$?

Additionally, we present a mixed integer program formulation of Robinson $(L_1,\mathbb{R})$-Fitting, which might help solving the problem using sophisticated mixed integer program solvers. Section 4.3 finally focuses on the following problem variant, which for $p=\infty$ and $M=\mathbb{R}$ has been considered by Chepoi et al. [CFS09]:

Restricted Robinson (L_p, M)-Fitting
Input: A similarity $d\colon X \times X \to M$, a total order $\preceq$ on X, and a number $\varepsilon > 0$.
Question: Is there a similarity $d'\colon X \times X \to M$ compatible with $\preceq$ and satisfying $\|d-d'\|_p^p \leq \varepsilon$ for $p<\infty$ and $\|d-d'\|_\infty \leq \varepsilon$ for $p=\infty$?

Algorithms solving Restricted Robinson (L_p, M)-Fitting are applicable if the object order sought by seriation is partly known. We show that for $M=\mathbb{R}$ the problem is, within arbitrary precision, solvable in polynomial time. Linear-time algorithms are presented for the case $p=1$ and $M=\{0,1\}$ and for the case $p=\infty$ and $M\in\{\mathbb{N},\mathbb{R}\}$.

4.1 Partial Robinsonian Similarities

Given a similarity $d\colon X \times X \to \mathbb{R}$, this section studies the problem of finding a small subset S of outliers in X such that d is Robinsonian on the remaining set $X \setminus S$. We show that Partial Robinsonian $\{0,1\}$-Similarity is NP-complete and fixed-parameter tractable parameterized by the number k of sought outliers. Both results are obtained by exploiting the close relation between Partial Robinsonian $\{0,1\}$-Similarity and the Unit Interval Vertex Deletion problem presented in Section 1.2.

Unit Interval Vertex Deletion
Input: A graph $G = (V, E)$ and a natural number k.
Question: Is $G - S$ a unit interval graph for some vertex set $S \subseteq V$ with $|S| \le k$?

According to Section 3.1, Unit Interval Vertex Deletion is NP-complete. Chapter 5 presents an $O((14k + 14)^{k+1} \cdot kn^6)$-time fixed-parameter algorithm for the problem. Combining these results with the following lemma, we show that Partial Robinsonian $\{0,1\}$-Similarity is NP-complete and fixed-parameter tractable.

Lemma 4.1. *There is an $O(|X|^2)$-time parameter-preserving reduction from* Partial Robinsonian $\{0,1\}$-Similarity *to* Unit Interval Vertex Deletion *and vice versa.*

Proof. Proposition 1.1 states that a similarity $d\colon X \times X \to \{0,1\}$ is Robinsonian if and only if the graph $\mathcal{G}(d) = (X, E)$ is a unit interval graph, where $E := \{\{x, y\} \mid x, y \in X, x \ne y, \text{ and } d(x, y) = 1\}$. Obviously, the graph $\mathcal{G}(d)$ is computable in $O(|X|^2)$ time.

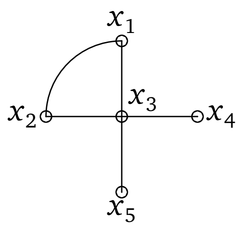
We first show the reduction from Partial Robinsonian $\{0,1\}$-Similarity to Unit Interval Vertex Deletion. For a similarity $d\colon X \times X \to \mathbb{R}$, (d, k) is a yes-instance of Partial Robinsonian $\{0,1\}$-Similarity if and only if there is a set $S \subseteq X$ such that $d|_{X\setminus S}$ is Robinsonian. This is the case precisely if $\mathcal{G}(d|_{X\setminus S}) = \mathcal{G}(d) - S$ is a unit interval graph, which is equivalent to $(\mathcal{G}(d), k)$ being a yes-instance of Unit Interval Vertex Deletion.

We now show the reduction from Unit Interval Vertex Deletion to Partial Robinsonian $\{0,1\}$-Similarity. Given a graph $G = (X, E)$, consider the function $d_G\colon X \times X \to \{0,1\}$ such that for each $x, y \in X$ it holds that $d_G(x, y) = 1$ if $\{x, y\} \in E$ or $x = y$, and $d_G(x, y) = 0$ otherwise. Then, for $x, y \in X$, one has $d_G(x, x) \ge d_G(x, y) = d_G(y, x) \ge 0$, implying that d_G is a similarity. Moreover, (G, k) is a yes-instance of Unit Interval Vertex Deletion if and only if there is a set $S \subseteq X$ with $|S| \le k$ such that $G - S$ is a unit interval graph. From $\mathcal{G}(d_G|_{X\setminus S}) = G - S$ it follows that this is precisely the case if the similarity $d_G|_{X\setminus S}$ is Robinsonian. This is equivalent to (d_G, k) being a yes-instance of Partial Robinsonian $\{0,1\}$-Similarity. □

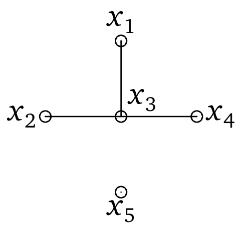
Chapter 5 shows an algorithm that solves Unit Interval Vertex Deletion on an n-vertex graph in $O((14k + 14)^{k+1} \cdot kn^6)$ time. Together with the NP-completeness of Unit Interval Vertex Deletion following from Section 3.1, Lemma 4.1 implies:

d	x_1	x_2	x_3	x_4	x_5
x_1	2	1	2	0	0
x_2	1	2	2	0	0
x_3	2	2	2	2	1
x_4	0	0	2	2	0
x_5	0	0	1	0	2

(a) The similarity d.



(b) The graph G_1.



(c) The graph G_2.

Figure 4.1: A similarity d and the corresponding graphs G_ε for $\varepsilon = 1$ and $\varepsilon = 2$.

Theorem 4.1. Partial Robinsonian $\{0,1\}$-Similarity *is NP-complete and can be solved in* $O((14k+14)^{k+1}k|X|^6)$ *time.*

Because, for a similarity $d\colon X \times X \to \{0,1\}$, (d,k) is a yes-instance of Partial Robinsonian $\{0,1\}$-Similarity if and only if it is a yes-instance of Partial Robinsonian Similarity, Theorem 4.1 also implies:

Corollary 4.1. Partial Robinsonian Similarity *is NP-hard.*

Exploiting a fixed-parameter algorithm for Unit Interval Vertex Deletion, we obtained a fixed-parameter algorithm for Partial Robinsonian $\{0,1\}$-Similarity. An obvious question to ask is whether this approach can generalized to obtain a fixed-parameter algorithm for Partial Robinsonian Similarity. This question has no straightforward answer, as the following example illustrates.

As stated by Roberts [Rob69, Rob78] and described in Section 1.2, a necessary condition for a similarity $d\colon X \times X \to \mathbb{R}$ to be Robinsonian is that all graphs G_ε are unit interval graphs, where G_ε is the graph with the vertex set X and $\{x,y\}$ being an edge if and only if $x \neq y$ and $d(x,y) \geq \varepsilon$. Figure 4.1 shows a similarity d and the corresponding graphs G_ε for $\varepsilon = 1$ and $\varepsilon = 2$, of which each contains an induced claw (shown in Figure 3.1). As claws are forbidden induced subgraphs for unit interval graphs, none of G_1 and G_2 is a unit interval graph.

Assume that, to transform G_1 and G_2 into unit interval graphs, we independently apply an algorithm for Unit Interval Vertex Deletion to G_1 and G_2. If the algorithm deletes x_5 from X, this makes G_1 a unit interval graph. However, deleting x_5 from X does not make G_2 a unit interval graph. Another vertex deletion (for example, x_2) is required to also transform G_2 into a unit interval graph. This transforms G_1 and G_2 into unit interval graphs by deleting a total number two vertices from X. We would obtain the same result if we started by deleting x_2 from X to first transform G_2 into a unit interval graph. However, if we remove the row and the column corresponding to x_3 in Figure 4.1a, then the entries in the shown table never increase when moving away from the main diagonal, that is, the order $x_1 \preceq x_2 \preceq x_4 \preceq x_5$ is compatible with $d|_{X\setminus\{x_3\}}$. It follows that d can be made Robinsonian by deleting a single object from X; Unit Interval Vertex Deletion algorithms that independently work on G_1 and G_2 might not find this optimal solution.

This problem could be circumvented if an algorithm enumerated *all* minimal vertex sets of size at most k whose deletion from a graph transforms it into a unit interval graph: simply enumerate all such sets S for G_1 and then try to delete as few as possible additional vertices from $G_2 - S$. However, because forbidden induced subgraphs for unit interval graphs may be arbitrarily large, there may be $O(n^k)$ such minimal vertex subsets of an n-vertex graph. Thus, we might not be able to enumerate them in $f(k) \cdot |G|^c$ time for a constant c. As a consequence, it remains an open question whether Partial Robinsonian Similarity is fixed-parameter tractable.

4.2 Robinson Fitting

This section studies the Robinson (L_p, M)-Fitting problem. First, Section 4.2.1 shows the NP-completeness of Robinson $(L_1, \{0,1\})$-Fitting. From this, the NP-completeness of Robinson $(L_p, \mathbb{N})$-Fitting with $p < \infty$ is derived. Then, Section 4.2.2 presents a mixed integer program formulation of Robinson $(L_1, \mathbb{R})$-Fitting. This mixed integer program is easily modifiable to additionally solve the problem variants Robinson (L_p, M)-Fitting for all combinations of $p \in \{1, \infty\}$ and $M \in \{\mathbb{R}, \mathbb{N}, \{0,1\}\}$. The mixed integer program formulation might help solving these problem variants using sophisticated mixed integer program solvers.

4.2.1 NP-Completeness

This subsection shows that Robinson $(L_p, \mathbb{N})$-Fitting is NP-complete. This complements the NP-completeness result for Robinson $(L_\infty, \mathbb{R})$-Fitting, which can be obtained by exploiting that a dissimilarity $d : X \times X \to \mathbb{R}$ is Anti-Robinsonian if and only if $\|d\|_\infty - d$ is a Robinsonian similarity and by exploiting the NP-completeness of Anti-Robinson $(L_p, \mathbb{R})$-Fitting [CFS09]:

Anti-Robinson (L_p, M)-Fitting
Input: A dissimilarity $d : X \times X \to M$ and a number $\varepsilon > 0$.
Question: Is there an Anti-Robinsonian dissimilarity $d' : X \times X \to M$ with $\|d - d'\|_p^p \leq \varepsilon$ for $p < \infty$ and $\|d - d'\|_\infty \leq \varepsilon$ for $p = \infty$?

Barthélemy and Brucker [BB01] have shown NP-completeness of a problem related to Anti-Robinson $(L_p, \mathbb{N})$-Fitting; however, they pose additional requirements on the sought dissimilarity d'. These requirements are exploited in their NP-completeness proof. Therefore, their NP-completeness result does not generalize to (ANTI-)ROBINSON $(L_p, \mathbb{N})$-FITTING. We now show that ROBINSON $(L_1, \{0,1\})$-FITTING is NP-complete and, from this, derive the NP-completeness of ROBINSON $(L_p, \mathbb{N})$-FITTING.

To show the NP-completeness of ROBINSON $(L_1, \{0,1\})$-FITTING, we exploit the characterization of Robinsonian similarities by unit interval graphs (see Section 1.2). We reduce UNIT INTERVAL EDITING to ROBINSON $(L_1, \{0,1\})$-FITTING.

UNIT INTERVAL EDITING
Input: A graph $G = (V, E)$ and a natural number k.
Question: Is there a unit interval graph $G' = (V, E')$ with $|E \Delta E'| \leq k$?

As shown by Burzyn et al. [BBD06], UNIT INTERVAL EDITING is NP-complete even on cubic planar graphs.

Lemma 4.2. ROBINSON $(L_1, \{0,1\})$-FITTING *is NP-complete.*

Proof. We first show that the problem is contained in NP: given a similarity $d' \colon X \times X \to \{0,1\}$ for an input similarity $d \colon X \times X \to \{0,1\}$, we can in polynomial-time check whether d is Robinsonian [MR84, CF97, ABH99]. To check $\|d - d'\|_1 \leq \varepsilon$, observe that the distance $\|d - d'\|_1$ always is an integer. Therefore, $\|d - d'\|_1 \leq \varepsilon$ is equivalent to $\|d - d'\|_1 \leq \lfloor \varepsilon \rfloor$. Thus, a Turing machine only has to read the integer part of the ε in an instance (d, ε); it can read the input in finite time.

To show that ROBINSON $(L_1, \{0,1\})$-FITTING is NP-hard, we now show a polynomial-time reduction from UNIT INTERVAL EDITING to ROBINSON $(L_1, \{0,1\})$-FITTING. For a graph $G = (V, E)$, let $d_G \colon V \times V \to \{0,1\}$ be a similarity such that $d_G(v, w) = 1$ if $v = w$ or $\{v, w\} \in E$, and $d_G(v, w) = 0$ otherwise. Obviously, d_G can be computed in polynomial time. To show that ROBINSON $(L_1, \{0,1\})$-FITTING is NP-hard, we show that (G, k) is a yes-instance of UNIT INTERVAL EDITING if and only if (d_G, k) is a yes-instance of ROBINSON $(L_1, \{0,1\})$-FITTING.

If $(G = (V, E), k)$ is a yes-instance of UNIT INTERVAL EDITING, then there is a unit interval graph $G' = (V, E')$ with $|E \Delta E'| \leq k$. By Proposition 1.1, it follows that the $\{0,1\}$-similarity $d_{G'}$ is Robinsonian. Moreover, because d_G and $d_{G'}$ are $\{0,1\}$-similarities, $\|d_G - d_{G'}\|_1$ counts the values that differ between d_G and $d_{G'}$, which is exactly the number of edges in $E \Delta E'$. Thus, one has $\|d_G - d_{G'}\|_1 \leq k$, implying that (d_G, k) is a yes-instance of ROBINSON $(L_1, \{0,1\})$-FITTING.

If (d_G, k) is a yes-instance of ROBINSON $(L_1, \{0,1\})$-FITTING, then there is a Robinsonian similarity $d' \colon V \times V \to \{0,1\}$ with $\|d_G - d'\|_1 \leq k$. By Proposition 1.1, the graph $G' = (V, E')$ with $E' := \{\{u, v\} \mid u, v \in V, u \neq v, d'(u, v) = 1\}$ is a unit interval graph. For the same reason as above, $\|d_G - d'\|_1 \leq k$ implies $|E' \Delta E| \leq k$. Thus, (G, k) is a yes-instance of UNIT INTERVAL EDITING. □

To generalize Lemma 4.2 to Robinson $(L_p, \mathbb{N})$-Fitting, we exploit the following observation. It implies that, without loss of generality, we may assume that the maximum value taken by a solution similarity d' to the Robinson $(L_p, \mathbb{R})$-Fitting problem is not larger than the maximum value taken by the input similarity.

Observation 4.1. *Let $d, d' \colon X \times X \to M$ be similarities such that d' is compatible with some total order $\preceq$ on X. Then, there is a similarity $d^* \colon X \times X \to M$ compatible with $\preceq$ and satisfying $\|d - d^*\|_p \le \|d - d'\|_p$ and $\|d^*\|_\infty \le \|d\|_\infty$.*

Proof. For each $x, y \in X$, choose $d^*(x,y) := \min\{\|d\|_\infty, d'(x,y)\}$. By this definition, $\|d^*\|_\infty \le \|d\|_\infty$ holds. From $d(x,y) \le \|d\|_\infty$ for all $x, y \in X$, we conclude

$$|d(x,y) - d^*(x,y)| = |d(x,y) - \min\{\|d\|_\infty, d'(x,y)\}| \le |d(x,y) - d'(x,y)|.$$

Thus, $\|d - d^*\|_p \le \|d - d'\|_p$. It remains to show that d^* is compatible with $\preceq$. For $x \preceq y \preceq z$, we show $d^*(x,z) \le d^*(x,y)$ and $d^*(x,z) \le d^*(y,z)$.

For the sake of contradiction, assume that $d^*(x,z) > d^*(x,y)$. By definition of d^*, this implies $\|d\|_\infty > d^*(x,y)$ and therefore $d^*(x,y) = d'(x,y)$. It follows that $d'(x,z) \ge d^*(x,z) > d^*(x,y) = d'(x,y)$, contradicting $\preceq$ being compatible with d'. Analogously, $d^*(x,z) \le d^*(y,z)$ can be shown. □

Theorem 4.2. Robinson $(L_p, \mathbb{N})$-Fitting *is NP-complete.*

Proof. We first show that the problem is contained in NP. Given a similarity $d' \colon X \times X \to \mathbb{N}$ for an input similarity $d \colon X \times X \to \mathbb{N}$, we can in polynomial-time check if d is Robinsonian [MR84, CF97, ABH99]. For the same reason as in the proof of Lemma 4.2, we can also check $\|d - d'\|_p^p \le \varepsilon$ in polynomial time.

To show that Robinson $(L_p, \mathbb{N})$-Fitting is NP-hard, we show that (d, ε) is a yes-instance of Robinson $(L_1, \{0,1\})$-Fitting, which is NP-complete by Lemma 4.2, if and only if (d, ε) is a yes-instance of Robinson $(L_p, \mathbb{N})$-Fitting.

If (d, ε) is a yes-instance of Robinson $(L_1, \{0,1\})$-Fitting, then there is a similarity $d' \colon X \times X \to \{0,1\}$ with $\|d' - d\|_1 \le \varepsilon$. Observe that for each $x, y \in X$, the difference $|d(x,y) - d'(x,y)| \in \{0,1\}$. We conclude that $\|d' - d\|_p^p = \|d' - d\|_1 \le \varepsilon$. It follows that (d, ε) is a yes-instance of Robinson $(L_p, \mathbb{N})$-Fitting.

If (d, ε) is a yes-instance of Robinson $(L_p, \mathbb{N})$-Fitting, then there is a similarity $d' \colon X \times X \to \mathbb{N}$ with $\|d - d'\|_p^p \le \varepsilon$. By Observation 4.1, we may assume that $\|d'\|_\infty \le \|d\|_\infty \le 1$, that is, we may assume d' to be a $\{0,1\}$-similarity. This again implies that $\|d - d'\|_1 = \|d - d'\|_p^p \le \varepsilon$. It follows that (d, ε) is a yes-instance of Robinson $(L_1, \{0,1\})$-Fitting. □

4.2.2 A Mixed Integer Program

This subsection presents a mixed integer program for, given a similarity $d\colon X \times X \to \mathbb{R}$, finding a Robinsonian similarity $d'\colon X \times X \to \mathbb{R}$ that minimizes $\|d - d'\|_1$. A mixed integer program consists of a linear objective function whose arguments are constrained by linear equations and inequalities. Subject to these constraints, the global minimum of the objective function shall be found. The arguments of the objective function are real numbers; some arguments may additionally be required to be integers. If no argument is required to be an integer, then the mixed integer program is a *linear program*, which can be solved in polynomial time [Kha79, Kar84]. In contrast, solving general mixed integer programs is NP-hard [GJ79].

The following mixed integer program formulation might be helpful to solve Robinson $(L_1, \mathbb{R})$-Fitting using sophisticated mixed integer solvers. The mixed integer program is easily modifiable to also solve Robinson (L_p, M)-Fitting for $p \in \{1, \infty\}$ and $M \in \{\mathbb{N}, \mathbb{R}\}$. However, it will not give insights into the complexity of Robinson $(L_p, \mathbb{R})$-Fitting, which, to our knowledge, is still open. This stands in contrast to Robinson $(L_p, \mathbb{N})$-Fitting, which Theorem 4.2 shows to be NP-complete.

Let $d\colon X \times X \to \mathbb{R}$ be a similarity. We present a mixed integer program in $O(|X|^2)$ variables and $O(|X|^3)$ constraints that enables us to find a Robinsonian similarity $d'\colon X \times X \to \mathbb{R}$ that minimizes $\|d - d'\|_1$. We seek to minimize

$$\|d - d'\|_1 = \frac{1}{2} \sum_{\substack{x \neq y \\ x,y \in X}} |d(x,y) - d'(x,y)|. \tag{4.1}$$

under the constraint that d' is Robinsonian. Here, for all $x, y \in X$, the value $d(x,y) \in \mathbb{R}$ is fixed and the similarity d' is to be determined by the optimum solution to the mixed integer program. That is, we consider $d'(x,y)$ as a variable. We transform (4.1) into the linear objective function $f(t) := \frac{1}{2} \sum_{x,y \in X, x \neq y} t_{xy}$, where the variables t_{xy} are determined by the linear constraints

$$\begin{aligned} &\forall x, y \in X, x \neq y : t_{xy} \geq d(x,y) - d'(x,y), \text{ and} \\ &\forall x, y \in X, x \neq y : t_{xy} \geq d'(x,y) - d(x,y). \end{aligned} \tag{4.2}$$

It remains to express the constraint of d' being Robinsonian in terms of linear equations and inequalities. As we do not only want to obtain the Robinsonian similarity d' that minimizes $\|d - d'\|_1$ but also the order with which it is compatible, we introduce an integer variable o_{xy} for each pair $x, y \in X$. Here $o_{xy} = 0$ corresponds to $x \preceq y$ and $o_{xy} = 1$ corresponds to $x \not\preceq y$. We have to make sure that the order induced by the variables o_{xy} has no incomparable pairs and that it is antisymmetric, reflexive, and transitive. The first three properties are ensured by the constraints

$$\begin{aligned}\forall x,y\in X, x\neq y: o_{xy}+o_{yx}&=1,\\ \forall x,y\in X, x\neq y: o_{xy}&\in\{0,1\},\\ \forall x\in X: o_{xx}&=0.\end{aligned}$$

It remains to ensure transitivity of the order induced by the variables o_{xy}: if $x\preceq y\preceq z$ holds, then $o_{xy}=o_{yz}=0$. As $x\preceq y\preceq z$ shall imply $x\preceq z$, we require

$$\forall x,y,z\in X: o_{xz}\le o_{xy}+o_{yz}.$$

Finally, d' shall be a similarity compatible with the order induced by the variables o_{xy}. The following constraints require d' to be symmetric and nonnegative:

$$\begin{aligned}&\forall x,y\in X: d'(x,y)=d'(y,x),\\ &\forall x,y\in X: d'(x,y)\ge 0.\end{aligned}$$

It remains to ensure that d' is compatible with $\preceq$. To this end, we exploit that Observation 4.1 guarantees the existence of an optimal solution d' satisfying $\|d'\|_\infty\le\|d\|_\infty$. Observe that since the input similarity d is fixed, $\|d\|_\infty$ is precomputable and also fixed. Therefore, the following constraints are indeed linear:

$$\forall x,y\in X: d'(x,y)\le\|d\|_\infty, \tag{4.3}$$

$$\begin{aligned}&\forall x,y,z\in X: \|d\|_\infty\cdot(o_{xy}+o_{yz})+d'(x,y)\ge d'(x,z),\\ &\forall x,y,z\in X: \|d\|_\infty\cdot(o_{xy}+o_{yz})+d'(y,z)\ge d'(x,z).\end{aligned} \tag{4.4}$$

If $o_{xy}=o_{yz}=0$, that is, if $x\preceq y\preceq z$ holds, then the constraints (4.4) ensure that $d'(x,y)$ and $d'(y,z)$ are at least $d'(x,z)$. If, on the contrary, $o_{xy}=1$ or $o_{yz}=1$, then the constraints (4.4) are trivially satisfied because of constraint (4.3).

Correctness. Given a similarity $d\colon X\times X\to\mathbb{R}$, let f^* be the optimum value of the objective function f and let d^* be a Robinsonian similarity that minimizes $\|d-d^*\|_1$. To show that the mixed integer program is correct, we show $\|d-d^*\|_1=f^*$.

We first show $\|d-d^*\|_1\le f^*$. If the mixed integer program yields an assignment to the variables $d'(x,y)$ under which the objective function f obtains the minimum value f^*, then d' is a Robinsonian similarity by constraints (4.3, 4.4). From constraint (4.2), it follows that $f^*\ge\|d-d'\|_1\ge\|d-d^*\|_1$.

We now show $f^*\le\|d-d^*\|_1$. By Observation 4.1, we may assume that $\|d^*\|_\infty\le\|d\|_\infty$. Moreover, there is an order $\preceq$ that is compatible with d^*. We now obtain an assignment to the variables of the mixed integer program as follows: for all $x,y\in X$, let $d'(x,y) = d^*(x,y)$, $t_{xy} = |d(x,y) - d'(x,y)|$, and $o_{xy} = 0$ if $x \preceq y$ and $o_{xy} = 1$ otherwise. The variables $d'(x,y)$, o_{xy}, and t_{xy} satisfy all constraints of the mixed integer program. Under this assignment, the objective function f obtains the value $\|d - d'\|_1$. That is, we have $\|d - d^*\|_1 = \|d - d'\|_1 = f \geq f^*$.

The presented mixed integer program formulation of Robinson $(L_1, \mathbb{R})$-Fitting can be used to solve seriation problems by transforming given similarity data into problem descriptions of mixed integer program solvers. It is straightforward to test the practical efficiency of the presented mixed integer program by generating problem descriptions for mixed integer program solvers like GLPK[1] and ILOG CPLEX[2]. If real-world similarity data is unavailable for experiments, then artificial similarity data can be obtained by perturbing a randomly generated Robinsonian similarity.

4.3 Restricted Robinson Fitting

This section focuses on a variant of Robinson (L_p, M)-Fitting that is motivated by seriation problems where parts of the sought order are already known. In this case, we can model the seriation problem as follows:

Partially Restricted Robinson (L_p, M)-Fitting
Input: A similarity $d\colon X \times X \to M$, a partial order $\sqsubseteq$ on X, and a number ε.
Question: Is there a similarity $d'\colon X \times X \to M$ compatible with a linear extension of $\sqsubseteq$ and $\|d - d'\|_p^p \leq \varepsilon$ for $p < \infty$ and $\|d - d'\|_\infty \leq \varepsilon$ for $p = \infty$?

Giving an empty order $\sqsubseteq$ as input yields the Robinson (L_p, M)-Fitting problem. Thus, following from Theorem 4.2 and Chepoi et al. [CFS09], respectively, Partially Restricted Robinson $(L_p, \mathbb{N})$-Fitting and Partially Restricted Robinson $(L_\infty, \mathbb{R})$-Fitting are NP-hard. In particular, it follows from Chepoi et al. [CFS09] that Partially Restricted Robinson $(L_\infty, \mathbb{R})$-Fitting is NP-hard even for $\varepsilon = 3$, implying that unless $\mathrm{P} = \mathrm{NP}$ the problem is not fixed-parameter tractable parameterized by ε.

We can solve Partially Restricted Robinson (L_p, M)-Fitting by first enumerating all linear extensions $\preceq$ of $\sqsubseteq$. If X contains Δ incomparable pairs with respect to $\preceq$, then there are at most 2^Δ linear extensions of $\sqsubseteq$, since for each of the Δ incomparable pairs $x, y \in X$, we can choose between $x \preceq y$ and $y \preceq x$. For each of the at most 2^Δ linear extensions $\preceq$ of $\sqsubseteq$, we then solve the following problem:

[1] http://www.gnu.org/software/glpk/
[2] http://www-01.ibm.com/software/integration/optimization/cplex/

Restricted Robinson (L_p, M)-Fitting
Input: A similarity $d\colon X \times X \to M$, a total order $\preceq$ on X, and a number $\varepsilon > 0$.
Question: Is there a similarity $d'\colon X \times X \to M$ compatible with $\preceq$ and satisfying $||d - d'||_p^p \leq \varepsilon$ for $p < \infty$ and $||d - d'||_\infty \leq \varepsilon$ for $p = \infty$?

This is congruent with the intuition that the seriation problem becomes easier the more of the sought total order is already known. It immediately follows that:

Proposition 4.1. *If* Restricted Robinson (L_p, M)-Fitting *is polynomial-time solvable, then* Partially Restricted Robinson (L_p, M)-Fitting *is fixed-parameter tractable with respect to the number of incomparable pairs in X as parameter.*

Chepoi et al. [CFS09] have shown that Restricted Robinson $(L_\infty, \mathbb{R})$-Fitting is solvable in polynomial time. As a consequence, Partially Restricted Robinson $(L_\infty, \mathbb{R})$-Fitting is fixed-parameter tractable parameterized by the number of incomparable pairs in X. We now focus on solving Restricted Robinson (L_p, M)-Fitting in polynomial time. Section 4.3.1 starts by showing that Robinson $(L_p, \mathbb{R})$-Fitting is polynomial-time solvable within arbitrary precision. Section 4.3.2 and Section 4.3.3 show linear-time algorithms for Restricted Robinson $(L_\infty, \mathbb{R})$-Fitting and Restricted Robinson $(L_1, \{0,1\})$-Fitting, respectively.

In this subsection, we use the term *simple arithmetic operation* to refer to an addition or comparison of two real numbers.

4.3.1 Polynomial-Time L_p-Fitting by Real-Valued Similarities

Given a similarity $d\colon X \times X \to \mathbb{R}$ and a total order $\preceq$, this section focuses on finding a similarity $d^*\colon X \times X \to \mathbb{R}$ compatible with $\preceq$ that minimizes $||d - d^*||_p^p$. While the similarity d^* that minimizes $||d - d^*||_\infty$ can be found in polynomial time [CFS09], we encounter a problem when minimizing $||d - d^*||_p^p$: in general, it might happen that we cannot output the binary numeral representation of the optimal solution d^* in polynomial time, as illustrated in Figure 4.2. We circumvent the problem by adopting the optimality concept used by Nesterov and Nemirovskii [NN87] in their book on convex optimization:

Definition 4.1. Given a similarity $d\colon X \times X \to \mathbb{R}$ and a total order $\preceq$ on X, let d^* be a similarity compatible with $\preceq$ and minimizing $||d - d^*||_p^p$. We say that Robinson (L_p, R)-Fitting is *within precision $\bar{\varepsilon}$ solvable in polynomial time* if we can compute, in time polynomial in the size of $(d, \preceq)$ and proportional to $\log \bar{\varepsilon}^{-1}$, a similarity d' with $||d - d'||_p^p \leq ||d - d^*||_p^p + \bar{\varepsilon}$ that satisfies $d'(x,z) \leq \min\{d'(x,y), d'(y,z)\} + \bar{\varepsilon}$ for all objects $x, y, z \in X$ with $x \preceq y \preceq z$.

d	x_1	x_2	x_3
x_1	2	1	2
x_2	1	2	1
x_3	2	1	2

d^*	x_1	x_2	x_3
x_1	2	$\sqrt{2}$	$\sqrt{2}$
x_2	$\sqrt{2}$	2	$\sqrt{2}$
x_3	$\sqrt{2}$	$\sqrt{2}$	2

Figure 4.2: The similarity d^* minimizes the distance $||d - d^*||_3^3$. Because d^* obtains irrational values, a Turing machine cannot output the binary numeral representation of d^* in finite time.

By Definition 4.1, if the required precision is increased by one bit, say for example, $\bar{\varepsilon}^{-1}$ is increased from 2^i to 2^{i+1}, then the factor $\log \bar{\varepsilon}^{-1}$ increases only by one. After a brief introduction into convex programming, we employ Definition 4.1 to prove the following theorem:

Theorem 4.3. Restricted Robinson $(L_p, \mathbb{R})$-Fitting *is, within arbitrary precision, solvable in polynomial time.*

The proof of Theorem 4.3 employs convex programming [NN87], which, like mixed integer programming as applied in Section 4.2.2, is a generalization of linear programming. A convex program consists of a convex function whose global minimum is to be found under given constraints on its arguments, where the set of all arguments satisfying the constraints, called *feasible region*, is convex.

The advantage of convex programs over linear programs is that the problem modeling capabilities of convex programs are not restricted to linear objective functions and constraints. Despite their increased modeling power, many convex programs are still solvable in polynomial time [NN87]. This is in contrast to optimization problems with non-convex objective functions or non-convex feasible regions. These problems are often NP-complete; a few examples shall illustrate this.

Linear programs are special forms of convex programs. They are solvable in polynomial time [Kha79, Kar84]. In contrast, solving mixed integer programs is generally NP-hard [GJ79]. Observe that the feasible region of mixed integer programs is generally not convex: for example, mixed integer programs allow for constraints that require a variable to be contained in the non-convex set $\{0, 1\}$. As another example, consider the objective function $f : \mathbb{R}^n \to \mathbb{R}, x \mapsto ||x - z||_p^p$ for fixed $z \in \mathbb{R}^n$ and $p \in [0, 1)$, whose global minimum shall be found under linear constraints. This problem is generally NP-hard [GJY10]. Here, the objective function f is non-convex. The feasible region, however, is convex because it is determined by linear constraints. As stated in Chapter 2, we only consider L_p-norms for $p \geq 1$, for which the minimization problem turns out to be polynomial-time solvable [NN87].

Proof of Theorem 4.3. For Robinson $(L_\infty, \mathbb{R})$-Fitting, Theorem 4.3 follows from Chepoi et al. [CFS09], who have shown that, for a given similarity $d\colon X \times X \to \mathbb{R}$ and a total order $\preceq$, a similarity $d^*\colon X \times X \to \mathbb{R}$ that is compatible with $\preceq$ and that minimizes $\|d - d^*\|_\infty$ is computable in polynomial time. It remains to show that Robinson $(L_p, \mathbb{R})$-Fitting is, within arbitrary precision, solvable in polynomial time for $p < \infty$. To this end, we employ a special-case convex program: assume that we are given linear functions $f_1, \ldots, f_m : \mathbb{R}^n \to \mathbb{R}$ called *constraints* and a vector $b \in \mathbb{R}^n$ of *constants*. The task is to find a vector $z \in \mathbb{R}^n$ of *variables* that satisfies the constraints $f_i(z) \leq 0$ for $i \in \{1, \ldots, m\}$ and that minimizes $\|z - b\|_p^p$. This is modeled by the following convex program, which is, within arbitrary precision, solvable in polynomial time [NN87, Section 6.3.2]:

$$\text{(CP)} \qquad \begin{aligned} &\text{minimize } f_0(z) = \sum_{j=1}^{n} |z_j - b_j|^p, \text{ where } z \in \mathbb{R}^n,\\ &\text{subject to } f_i(z) \leq 0 \text{ for } i \in \{1, \ldots, m\},\\ &\qquad\qquad \|z\|_2 \leq R, \text{ and } f_0(z) \leq V. \end{aligned}$$

Here, the constraints $\|z\|_2 \leq R$ and $f_0(z) \leq V$ are technical details that Nesterov and Nemirovskii [NN87] exploit to show that the convex program (CP) is, within arbitrary precision, solvable in polynomial time. The numbers R and V must be given as input to the convex program and must be fixed, that is, independent from the vector z of variables. The running time for solving the convex program (CP) is then independent from R but proportional to $\log V$ [NN87, Section 6.3.2].

Given a similarity $d\colon X \times X \to \mathbb{R}$ and a total order $\preceq$ on X, we model the task of finding a similarity $d'\colon X \times X \to \mathbb{R}$ compatible with $\preceq$ that minimizes $\|d - d'\|_p^p$ in terms of the convex program (CP). We seek to minimize the objective function

$$\|d - d'\|_p^p = \sum_{x \preceq y, x \neq y} |d(x,y) - d'(x,y)|^p$$

under the constraint that d' is compatible with $\preceq$. Here, the similarity d is fixed and given as input. The similarity d' is to be determined by the global minimum of the convex program. To this end, the convex program determines the value $d'(x,y)$ for each $x, y \in X$, that is, $d'(x,y)$ is a variable in the convex program. Intuitively, the similarity d corresponds to the vector b of constants in (CP) and the sought similarity d' corresponds to the vector z of variables in (CP). This convex program formulation uses $O(|X|^2)$ variables. It remains to ensure that the sought similarity d' is compatible with $\preceq$. To this end, we devise $O(|X|^3)$ linear constraints, which correspond to the constraints $f_i(z) \leq 0$ in (CP). First, we require that d' is nonnegative. For all $x \preceq y$, it shall hold that

$$-d'(x,y) \leq 0. \tag{4.5}$$

Moreover, the resulting similarity d' shall be compatible with $\preceq$, that is, for all objects $x_i \preceq x_j \preceq x_k$, the similarity d' shall satisfy

$$\begin{aligned} d'(x_i, x_k) - d'(x_j, x_k) &\leq 0 \\ d'(x_i, x_k) - d'(x_i, x_j) &\leq 0 \end{aligned} \tag{4.6}$$

For an optimal assignment to the variables $d'(x, y)$ of the convex program, we obtain a similarity d^* that is compatible with the given order $\preceq$ by choosing $d^*(x, y) = d^*(y, x) = d'(x, y)$ for all $x \preceq y, x \neq y$ and $d^*(x, x) = \|d'\|_\infty$. Moreover, any similarity d' compatible with $\preceq$ satisfies the constraints (4.5) and (4.6). Now, the correctness of the convex program follows in the same way as the correctness of the mixed integer program in Section 4.2.2.

To show that the convex program is solvable in polynomial time, it remains to find numbers R and V such that there is a similarity d' compatible with $\preceq$ that minimizes $\|d - d'\|_p^p$ and satisfies $\|d'\|_2 \leq R$ and $\|d - d'\|_p^p \leq V$. By Observation 4.1, there is a similarity $d' : X \times X \to \mathbb{R}$ compatible with $\preceq$ that minimizes $\|d - d'\|_p^p$ and satisfies $\|d'\|_\infty \leq \|d\|_\infty$. Because $\|d'\|_2^2$ is the sum of $1/2 \cdot |X| \cdot (|X| - 1)$ squares of values that do not exceed $\|d\|_\infty$, the similarity d' satisfies

$$\|d'\|_2 \leq R := \sqrt{|X|^2 \cdot \|d\|_\infty^2} = |X| \cdot \|d\|_\infty.$$

Exploiting the triangle inequality for L_p-norms, we also find the number V, since

$$\|d - d'\|_p \leq \|d\|_p + \|d'\|_p \leq 2\sqrt[p]{|X|^2 \cdot \|d\|_\infty^p}.$$

It follows that $\|d - d'\|_p^p \leq V := 2^p \cdot |X|^2 \cdot \|d\|_\infty^p$. Observe that $O(\log V) = O(\log |X| + \log \|d\|_\infty)$, since p is a problem-specific constant. Thus, $O(\log V)$ is polynomial in the input size. As the running time for solving (CP) is polynomial in the input size and proportional to $\log V$ [NN87, Section 6.3.2], Theorem 4.3 follows. Nesterov and Nemirovskii [NN87] also give a more detailed running time bound proportional to $n^2(m + n)^{3/2} \cdot \log(m + n)V\bar{\varepsilon}^{-1}$ for solving (CP), where in our case the number n of variables in $O(|X|^2)$ and the number m of constraints is $O(|X|^3)$. □

4.3.2 Linear-Time L_∞-Fitting

This subsection presents an algorithm that decides RESTRICTED ROBINSON $(L_\infty, \mathbb{R})$-FITTING using $O(|X|)^2$ simple arithmetic operations. We then show, given a similarity $d : X \times X \to \mathbb{R}$ and a total order $\preceq$ on X, how the algorithm can be employed to compute a similarity $d' : X \times X \to \mathbb{R}$ compatible with $\preceq$ and minimizing $\|d - d'\|_\infty$ *within a given precision* $\bar{\varepsilon} > 0$, that is, such that the minimum distance $\|d - d^*\|_\infty$ for any similarity $d^* : X \times X \to \mathbb{R}$ compatible with $\preceq$ satisfies $\|d - d'\|_\infty \leq \|d - d^*\|_\infty + \bar{\varepsilon}$.

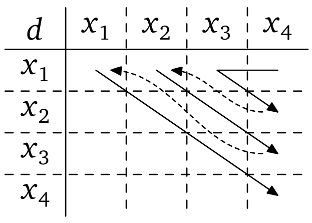


Figure 4.3: The secondary diagonals of d are processed starting at $d(x_1, x_4)$ and approaching the main diagonal. Each entry $d(x_i, x_k)$ is modified to satisfy (4.7), which expresses that entries do not decrease when moving away from the main diagonal starting at the entry $d(x_i, x_k)$.

We start with an intuitive description the algorithm. Given a similarity $d\colon X \times X \to \mathbb{R}$, a total order $x_1 \preceq \ldots \preceq x_n$ on X, and a number $\varepsilon > 0$, we search for a similarity $d'\colon X \times X \to \mathbb{R}$ that is compatible with $\preceq$ and that satisfies $\|d - d'\|_\infty \leq \varepsilon$. To this end, we try to make d compatible with $\preceq$ by increasing or decreasing the value $d(x_i, x_k)$ for each object pair $x_i, x_k \in X$ by at most ε. Obviously, to make d compatible with $\preceq$, for any two objects $x_i, x_k \in X$ with $x_i \preceq x_k$, the similarity d must be modified to satisfy

$$d(x_{i-1}, x_k) \leq d(x_i, x_k) \text{ and } d(x_i, x_{k+1}) \leq d(x_i, x_k), \tag{4.7}$$

if x_{i-1} or x_{k+1} exist, respectively. In the following, we say that $d(x_i, x_k)$ obtains its *smallest possible value* if for all similarities $d'\colon X \times X \to \mathbb{R}$ compatible with $\preceq$ and satisfying $\|d - d'\|_\infty \leq \varepsilon$, it holds that $d(x_i, x_k) \leq d'(x_i, x_k)$. Now, assume that we want to modify d to satisfy (4.7) for $d(x_i, x_k)$ and that $d(x_{i-1}, x_k)$ and $d(x_i, x_{k+1})$ already obtain their smallest possible values. If (4.7) cannot be satisfied by increasing $d(x_i, x_k)$ by at most ε, then we can show that no similarity d' compatible $\preceq$ and satisfying $\|d - d'\|_\infty \leq \varepsilon$ exists. In this case, $(d, \preceq, \varepsilon)$ is a no-instance. This is exploited by Algorithm 4.1, which intuitively works as follows: in the order shown in Figure 4.3, Algorithm 4.1 processes each entry $d(x_i, x_k)$ after $d(x_{i-1}, x_k)$ and $d(x_i, x_{k+1})$ (which are displayed on top and to the right of $d(x_i, x_k)$ in Figure 4.3, respectively) have already been set to their smallest possible values. If (4.7) is satisfiable for $d(x_i, x_k)$ by increasing $d(x_i, x_k)$ by at most ε, then $d(x_i, x_k)$ is set to its smallest possible value. Otherwise, "no" is returned. Processing starts at $d(x_1, x_n)$, because (4.7) is easy to satisfy for $d(x_1, x_n)$.

Lemma 4.3. *Algorithm 4.1 is correct and uses $O(|X|^2)$ simple arithmetic operations.*

Proof. The running time bound is easy to see. It remains to prove that Algorithm 4.1 is correct. We first show that if Algorithm 4.1 returns a similarity d', then d' is

Algorithm 4.1: Robinson $(L_p, \mathbb{R})$-Fitting

```
Input: A similarity $d\colon X \times X \to \mathbb{R}$, a total order $x_1 \preceq x_2 \preceq \ldots \preceq x_n$ on $X$, and a
       positive number $\varepsilon$.
Output: A similarity $d'\colon X \times X \to \mathbb{R}$ with $\|d - d'\|_\infty \leq \varepsilon$ compatible with $\preceq$, or
        "no" if no such similarity exists.
1  for $i := 1$ to $n$ do                          // setup sentinel elements $x_0$ and $x_{n+1}$.
2      $d(x_0, x_i) \leftarrow 0$;
3      $d(x_i, x_{n+1}) \leftarrow 0$
4  for $i := n$ to $1$ do
5      for $j := 1$ to $n - i + 1$ do              // choose $d(x_j, x_{i+j-1})$ as follows:
6          $\Delta \leftarrow \max\{d(x_{j-1}, x_{i+j-1}), d(x_j, x_{i+j})\} - d(x_j, x_{i+j-1})$;
7          if $\Delta > \varepsilon$ then return "no";
8          else $d(x_j, x_{i+j-1}) \leftarrow d(x_j, x_{i+j-1}) + \max\{\Delta, -\varepsilon\}$;
9  Make $d$ symmetric;
10 return $d$;
```

compatible with $\preceq$ and satisfies $\|d - d'\|_\infty \leq \varepsilon$. Then, we prove that if Algorithm 4.1 returns "no", then $(d, \preceq, \varepsilon)$ is a no-instance.

Assume that Algorithm 4.1 returns a similarity d'. Then, the similarity d' satisfies $d'(x_i, x_k) \geq \max\{d'(x_{i-1}, x_k), d'(x_i, x_{k+1})\}$ for each $x_i, x_k \in X$. That is, the values of d' never increase when moving away from the main diagonal. Hence, d' is compatible with $\preceq$. Moreover, the returned similarity d' satisfies $\|d - d'\|_\infty \leq \varepsilon$, because each value $d'(x_i, x_k)$ is obtained by increasing or decreasing $d(x_i, x_k)$ by at most ε. It follows that $(d, \preceq, \varepsilon)$ is a yes-instance.

To show that if Algorithm 4.1 returns "no", then $(d, \preceq, \varepsilon)$ is a no-instance, we employ the following definition: we say that $d(x_i, x_k)$ obtains its *smallest possible value* if for *all* similarities $d'\colon X \times X \to \mathbb{R}$ compatible with $\preceq$ and satisfying $\|d - d'\|_\infty \leq \varepsilon$, it holds that $d(x_i, x_k) \leq d'(x_i, x_k)$. We now reproduce the process of Algorithm 4.1 until it returns "no". Algorithm 4.1 processes d in the order shown in Figure 4.3, starting at $d(x_1, x_n)$. For the elements x_0 and x_n introduced in line 1 of Algorithm 4.1, we have $d(x_0, x_n) = d(x_1, x_{n+1}) = 0$. As a consequence, Algorithm 4.1 sets $d(x_1, x_n)$ to $\max\{0, d(x_1, x_n) - \varepsilon\}$ in line 8. This is the smallest possible value for $d(x_1, x_n)$. The remaining proof works inductively: assume that Algorithm 4.1 processes $d(x_i, x_k)$. In this case, $d(x_{i-1}, x_k)$ and $d(x_i, x_{k+1})$ have already been set to their smallest possible values. If Algorithm 4.1 does *not* return "no" in line 7, then $d(x_i, x_k)$ is set to its smallest possible value in line 8 of Algorithm 4.1, which is one of $d(x_i, x_k) - \varepsilon$, $d(x_{i-1}, x_k)$, or $d(x_i, x_{k+1})$.

Now, on the contrary, assume that Algorithm 4.1 returns “no” in line 7 and, for the sake of contradiction, assume that there is a similarity d' compatible with $\preceq$ that satisfies $\|d - d'\|_\infty \le \varepsilon$. As d' is compatible with $\preceq$, one has $d'(x_i, x_k) \ge d'(x_{i-1}, x_k)$ and $d'(x_i, x_k) \ge d'(x_i, x_{k+1})$. Because $d(x_{i-1}, x_k)$ and $d(x_i, x_{k+1})$ already obtain their smallest possible values, one has, in turn, $d'(x_{i-1}, x_k) \ge d(x_{i-1}, x_k)$ and $d'(x_i, x_{k+1}) \ge d(x_i, x_{k+1})$. Since Algorithm 4.1 returns “no” in line 7, it holds that $d(x_{i-1}, x_k) > d(x_i, x_k) + \varepsilon$ or $d(x_i, x_{k+1}) > d(x_i, x_k) + \varepsilon$. This, however shows

$$\begin{aligned} d'(x_i, x_k) &\ge \max\{d'(x_{i-1}, x_k), d'(x_i, x_{k+1})\} \\ &\ge \max\{d(x_{i-1}, x_k), d(x_i, x_{k+1})\} \\ &> d(x_i, x_k) + \varepsilon, \end{aligned}$$

which contradicts $\|d - d'\|_\infty \le \varepsilon$. Hence, there is no similarity d' compatible with $\preceq$ and satisfying $\|d - d'\|_\infty \le \varepsilon$. It follows that $(d, \preceq, \varepsilon)$ is a no-instance. □

Observe that, until now, Algorithm 4.1 merely solves the *decision problem* Robinson $(L_\infty, \mathbb{R})$-Fitting: for a yes-instance $(d, \preceq, \varepsilon)$, Algorithm 4.1 returns a similarity d' compatible with $\preceq$ that satisfies $\|d - d'\|_\infty \le \varepsilon$, but it does not necessarily return a similarity d' such that $\|d - d'\|_\infty$ is minimized. In the following, we show how Algorithm 4.1 can be exploited to compute a similarity d' compatible with $\preceq$ such that the minimum distance $\|d - d^*\|_\infty$ for any similarity $d^* \colon X \times X \to \mathbb{R}$ compatible with $\preceq$ satisfies $\|d - d'\|_\infty \le \|d - d^*\|_\infty + \bar{\varepsilon}$, where $\bar{\varepsilon} > 0$ is a given precision.

Theorem 4.4. *Given a similarity $d \colon X \times X \to \mathbb{R}$ and a total order $\preceq$ on X, a similarity $d' \colon X \times X \to \mathbb{R}$ that minimizes $\|d - d'\|_\infty$ within precision $\bar{\varepsilon}$ is computable using $O(|X|^2 \cdot \log \|d\|_\infty \bar{\varepsilon}^{-1})$ simple arithmetic operations.*

Proof. Given a similarity $d \colon X \times X \to \mathbb{R}$ and a total order $\preceq$ on X, let ε^* denote the minimum value for which $(d, \preceq, \varepsilon^*)$ is a yes-instance of Robinson $(L_\infty, \mathbb{R})$-Fitting. If $\varepsilon^* \ge \|d\|_\infty$, then $(d, \preceq, \varepsilon^*)$ trivially is a yes-instance since the Robinsonian similarity $d^* = 0$ satisfies $\|d^* - d\|_\infty \le \|d\|_\infty \le \varepsilon^*$. Thus, we have $0 \le \varepsilon^* < \|d\|_\infty$ and the minimum ε in the set $\{i\bar{\varepsilon} \mid 0 \le i \le \|d\|_\infty \bar{\varepsilon}^{-1}\}$ for which $(d, \preceq, \varepsilon)$ is a yes-instance satisfies $|\varepsilon - \varepsilon^*| \le \bar{\varepsilon}$. Since this set contains $O(\|d\|_\infty \bar{\varepsilon}^{-1})$ values, ε is computable using binary search by solving $O(\log \|d\|_\infty \bar{\varepsilon}^{-1})$ instances of Restricted Robinson $(L_\infty, \mathbb{R})$-Fitting. As Algorithm 4.1 solves a Restricted Robinson $(L_\infty, \mathbb{R})$-Fitting instance using $O(|X|^2)$ simple arithmetic operations and also outputs the corresponding Robinsonian similarity if it exists, Theorem 4.4 follows. □

Theorem 4.4 can be specialized for Robinson $(L_p, \mathbb{N})$-Fitting. To this end, observe that given a similarity $d \colon X \times X \to \mathbb{N}$ and an integer ε, Algorithm 4.1 returns a similarity that maps to integer values. Thus, by choosing precision $\bar{\varepsilon} = 1$, one obtains:

d'	x_{i-1}	x_i	x_j	x_k	x_n
x_{i-1}	1	1	1	0	0
x_i		1	1	1	0
x_j			1	1	0
x_k				1	1
x_n					1

Figure 4.4: As d' is compatible with $\preceq$, the values never increase when moving away from the main diagonal. The object x_k is the d'-boundary for x_i. The object x_j is the d'-boundary for x_{i-1} and satisfies $x_j \preceq x_k$.

Corollary 4.2. *Given a similarity $d\colon X \times X \to \mathbb{N}$ and a total order $\preceq$ on X, a similarity $d'\colon X \times X \to \mathbb{N}$ that is compatible with $\preceq$ and that minimizes $\|d - d'\|_\infty$ is computable in $O(|X|^2 \cdot \log \|d\|_\infty)$ time.*

Theorem 4.4 complements a result by Chepoi et al. [CFS09], who have shown that, for a given similarity $d\colon X \times X \to \mathbb{R}$ and a total order $\preceq$, a similarity $d^*\colon X \times X \to \mathbb{R}$ that is compatible with $\preceq$ and that minimizes $\|d - d^*\|_\infty$ is computable in polynomial time. A straightforward transformation of their proof into an algorithm yields an algorithm that uses $O(|X|^4)$ simple arithmetic operations. However, in contrast to our approach, which for a given precision $\bar{\varepsilon}$ only finds a similarity d' with $\|d - d'\|_\infty \leq \|d - d^*\|_\infty + \bar{\varepsilon}$, Chepoi et al. [CFS09] find the optimal solution d^*.

4.3.3 Linear-Time L_1-Fitting by $\{0,1\}$-Similarities

This subsection shows a dynamic programming algorithm that, given a similarity $d\colon X \times X \to \{0,1\}$ and a total order $x_1 \preceq x_2 \preceq \ldots \preceq x_n$ on X, finds a similarity $d'\colon X \times X \to \{0,1\}$ that is compatible with $\preceq$ and minimizes $\|d - d'\|_1$. We assume, without loss of generality, that all similarities $d'\colon X \times X \to \{0,1\}$ satisfy $d'(x,x) = 1$ for all $x \in X$. Observe that we can always modify a similarity to satisfy this assumption without increasing the distance $\|d - d'\|_1$.

The algorithm presented in this subsection is based on the following observation, which immediately follows from the fact that a similarity $d'\colon X \times X \to \{0,1\}$ is compatible with a total order $x_1 \preceq \ldots \preceq x_n$ on X if and only if its values never increase when moving away from the main diagonal in a table whose entry in the i-th row and k-th column displays $d'(x_i, x_k)$. In the following observation, which is illustrated in Figure 4.4, we call an object x_k the *d'-boundary* of an object x_i if x_k is the maximum object in $\preceq$ with $d'(x_i, x_k) = 1$.

Observation 4.2. *A similarity $d'\colon X \times X \to \{0,1\}$ is compatible with a total order $x_1 \preceq x_2 \preceq \ldots \preceq x_n$ on X if and only if for every object x_i and its d'-boundary x_k it holds that*

1. *$d'(x_i, x_j) = 1$ for all objects x_j with $x_i \preceq x_j \preceq x_k$,*
2. *$d'(x_i, x_l) = 0$ for all objects x_l with $x_{k+1} \preceq x_l$, and*
3. *the d'-boundary x_j of x_{i-1} satisfies $x_{i-1} \preceq x_j \preceq x_k$.*

Exploiting Observation 4.2, for two given similarities $d, d' : X \times X \to \{0, 1\}$, where d' is compatible with $\preceq$, we can recursively compute the distance $\|d - d'\|_1$ as follows: let $\delta_0 := 0$ and let $x_{k(i)}$ be the d'-boundary of x_i. Then, $\|d - d'\|_1 = \delta_n$, where

$$\delta_i := \delta_{i-1} + \sum_{j=i}^{k(i)} |d(x_i, x_j) - 1| + \sum_{l=k(i)+1}^{n} d(x_i, x_l). \tag{4.8}$$

In a similar way, we can recursively compute the minimum distance $\|d - d^*\|_1$ between a given similarity $d : X \times X \to \{0, 1\}$ and *any* similarity $d^* : X \times X \to \{0, 1\}$ compatible with $\preceq$. In contrast to (4.8), the similarity d^* is unknown here. Let $D_{0k} := 0$ for $0 \le k \le n$. Then, as explained below, the minimum distance $\|d - d^*\|_1$ for any similarity $d^* : X \times X \to \{0, 1\}$ compatible with $\preceq$ is D_{nn}, where for $1 \le i \le k \le n$,

$$D_{ik} := \underbrace{\sum_{j=i}^{k} |d(x_i, x_j) - 1|}_{=: A_{ik}} + \underbrace{\sum_{l=k+1}^{n} d(x_i, x_l)}_{=: B_{ik}} + \underbrace{\min_{i-1 \le j \le k} D_{i-1,j}}_{=: C_{ik}}. \tag{4.9}$$

Here, intuitively, the computation of D_{ik} takes place under the assumption that x_k is the d^*-boundary of x_i. The value A_{ik} is the cost for choosing $d^*(x_i, x_j) = 1$ for each x_j with $x_i \preceq x_j \preceq x_k$ and B_{ik} is the cost for choosing $d^*(x_i, x_l) = 0$ for each x_l with $x_{k+1} \preceq x_l$. Finally, C_{ik} recursively yields the minimum cost $D_{i-1,j}$ that results from choosing some object x_j with $x_{i-1} \preceq x_j \preceq x_k$ as d^*-boundary of x_{i-1}. Now, computing D_{nn} using dynamic programming enables us to prove:

Theorem 4.5. Restricted Robinson $(L_1, \{0, 1\})$-Fitting *is solvable in $O(|X|^2)$ time.*

Proof. We compute the value D_{nn} bottom-up. If the value $D_{i-1,j}$ is known for each j with $i - 1 \le j \le k$, then (4.9) can be used to compute D_{ik} in $O(|X|)$ time. This is up to further improvement: in $O(1)$ time, A_{ik} is computable from $A_{i,k-1}$, B_{ik} from $B_{i,k+1}$, and C_{ik} from $C_{i,k-1}$. This is exploited by Algorithm 4.2. Before Algorithm 4.2 computes D_{ik}, it computes $D_{i-1,j}$ for each j with $i - 1 \le j \le n$. Then, having A_{ik}, B_{ik}, and C_{ik} precomputed, Algorithm 4.2 computes D_{ik} in $O(1)$ time. As the computation of each D_{ik} for $1 \le i \le k \le n$ needs $O(1)$ time, it follows that the total running time of Algorithm 4.2 is $O(|X|^2)$.

By remembering the arguments for which C_{ik} obtains its minimum value, a similarity $d^* : X \times X \to \{0, 1\}$ compatible with the given order $\preceq$ and minimizing $\|d - d^*\|_1$ can be found within the same time bound. □

Algorithm 4.2: Restricted Robinson $(L_1, \{0,1\})$-Fitting

```
Input: A similarity $d\colon X \times X \to \mathbb{R}$, a total order $x_1 \preceq \ldots \preceq x_n$ on $X$, $\varepsilon > 0$.
Output: The minimum distance $\|d - d^*\|_1$ for any similarity $d^*\colon X \times X \to \{0,1\}$
        compatible with $\preceq$.
1 for $k := 0$ to $n$ do $D_{0k} \leftarrow A_{kk} \leftarrow B_{kn} \leftarrow 0$;
2 for $i := 1$ to $n$ do
3     $C_{ii} \leftarrow \min\{D_{i-1,i-1}, D_{i-1,i}\}$;
4     for $k := i+1$ to $n$ do
5         $A_{ik} \leftarrow A_{i,k-1} + |d(x_i, x_k) - 1|$;
6         $C_{ik} \leftarrow \min\{C_{i,k-1}, D_{i-1,k}\}$
7     for $k := n-1$ to $i$ do $B_{ik} \leftarrow B_{i,k+1} + |d(x_i, x_{k+1})|$;
8     for $k := i$ to $n$ do $D_{ik} \leftarrow A_{ik} + B_{ik} + C_{ik}$
9 return $D_{nn}$;
```

4.4 Conclusion

Having shown that Robinson $(L_p, \mathbb{N})$-Fitting is NP-complete, it remains open whether Robinson $(L_p, \mathbb{R})$-Fitting is NP-hard. The mixed integer program formulation of Robinson $(L_1, \mathbb{R})$-Fitting in Section 4.2.2 requires integer variables only to represent a compatible order. If these variables are fixed, the remaining mixed integer program becomes a polynomial-time solvable [Kha79, Kar84] linear program. Hence, finding the right order is the difficult part in Robinson $(L_1, \mathbb{R})$-Fitting. To model Robinson $(L_1, \mathbb{N})$-Fitting as mixed integer program, we also require the values of the sought similarity to be integer. This suggests that Robinson $(L_1, \mathbb{R})$-Fitting is, in some sense, easier than Robinson $(L_1, \mathbb{N})$-Fitting. However, we also find indications of Robinson $(L_p, \mathbb{R})$-Fitting being NP-hard: typical linear programs that minimize L_1-norms can easily be modified to also minimize L_∞-norms. That is, if we find a linear program solving Robinson $(L_1, \mathbb{R})$-Fitting in polynomial time, then it is likely to be modifiable to also solve Robinson $(L_\infty, \mathbb{R})$-Fitting in polynomial time, which was shown NP-hard by Chepoi et al. [CFS09]. Of course, Robinson $(L_1, \mathbb{R})$-Fitting could still be polynomial-time solvable without employing linear programming.

Exploiting convex programming, we have shown that Restricted Robinson $(L_p, \mathbb{R})$-Fitting is polynomial-time solvable within arbitrary precision. It would be interesting to find combinatorial algorithms that solve the problem faster than the shown convex programming approach. Moreover, it is open whether Restricted Robinson $(L_p, \mathbb{N})$-Fitting is polynomial-time solvable. The convex programming approach is not applicable here, as subsets of $\mathbb{N}^n$ are generally not convex. We have shown linear-time algorithms for the special cases Restricted Robinson $(L_1, \{0,1\})$-

Fitting and Restricted Robinson $(L_\infty, \mathbb{N})$-Fitting.

We introduced Partially Restricted Robinson (L_p, M)-Fitting to help solving seriation problems if parts of the sought order are already known. While we defined Partially Restricted Robinson (L_p, M)-Fitting to receive a partial order as input, it seems logical to study the complexity of Partially Restricted Robinson (L_p, M)-Fitting for different kinds of input orders. For example, one could consider strict weak orders (also called semi-orders) as input, which are total orders where "ties" are allowed. Also, the following variant of Partially Restricted Robinson (L_p, M)-Fitting seems interesting. It is interpretable as a seriation task where some objects have previously been put into a total order that shall now be completed by newly-found objects without rearranging previously ordered objects.

Incremental Constrained Robinson (L_p, M)-Fitting
Input: A similarity $d\colon X \times X \to M$, a number $\varepsilon > 0$, a set $Y \subseteq X$, and a total order $\preceq$ on Y compatible with $d|_Y$.
Question: Is there an linear extension of $\preceq$ on X that is compatible with a similarity $d'\colon X \times X \to M$ satisfying $\|d - d'\|_p^p \leq \varepsilon$?

This problem is, in particular, interesting in the context of parameterized complexity: if Partial Robinsonian Similarity is fixed-parameter tractable with respect to the number of sought outliers and if Incremental Constrained Robinson (L_p, M)-Fitting is fixed-parameter tractable with respect to the parameter $|X \setminus Y|$, then the general problem Robinson (L_p, M)-Fitting is fixed-parameter tractable with respect to the number of outliers in the input similarity—a structural parameter. Until now, neither the parameterized complexity of Partial Robinsonian Similarity nor of Incremental Constrained Robinson (L_p, M)-Fitting is known. We have made progress in this direction by showing that Partial Robinsonian $\{0,1\}$-Similarity is fixed-parameter tractable. The next logical step would be trying to show that Incremental Constrained Robinson $(L_1, \{0,1\})$-Fitting is fixed-parameter tractable with respect to the parameter $|X \setminus Y|$.

The hardness and tractability results presented in this chapter are summarized in Table 4.1 on the following page.

Problem	Complexity	Parameterized Complexity
Partial Robinsonian $\{0,1\}$-Similarity	**NP-complete**	$\boldsymbol{O((14k+14)^{k+1}k\|X\|^6)}$
Partial Robinsonian Similarity	**NP-complete**	open
Robinson $(L_1,\{0,1\})$-Fitting	**NP-complete**	open
Partially Restricted Robinson (L_p,M)-Fitting	**at least as hard as Robinson (L_p,M)-Fitting**	**$O(2^\Delta)$ instances of Restricted Robinson (L_p,M)-Fitting**[a]
Robinson $(L_p,\mathbb{N})$-Fitting	**NP-complete**	open
Robinson $(L_p,\mathbb{R})$-Fitting	open	open
Robinson $(L_\infty,\mathbb{R})$-Fitting	NP-complete even for $\varepsilon = 3$ [CFS09]	not FPT with parameter ε or $P = NP$
Restricted Robinson $(L_p,\mathbb{R})$-Fitting	**polynomial time**[b]	
Restricted Robinson $(L_\infty,\mathbb{R})$-Fitting	decision: **linear time**, optimization[c]: $\boldsymbol{O(\|X\|^2 \log \|d\|_\infty \bar{\varepsilon}^- 1)}$	
Restricted Robinson $(L_1,\{0,1\})$-Fitting	**linear time**	

[a] Here, Δ is the number of incomparable pairs in the given partial order.
[b] For optimization within arbitrary precision.
[c] Here, $\bar{\varepsilon}$ is the required precision. Previously shown to be polynomial-time solvable by Chepoi et al. [CFS09].

Table 4.1: Summary of complexity and tractability results. Results obtained in this work are shown in bold.

5 Unit Interval Vertex Deletion

This chapter presents fixed-parameter algorithms for the following problem:

Unit Interval Vertex Deletion
Input: A graph $G = (V, E)$.
Parameter: A natural number k.
Question: Is $G - S$ a unit interval graph for some vertex set $S \subseteq V$ with $|S| \leq k$?

In the following, we say that S is a unit interval vertex deletion set. We open with a brief summary of the main problems that emerge in the development of fixed-parameter algorithms for Unit Interval Vertex Deletion.

In Section 3.1, we have seen a characterization of unit interval graphs by infinitely many forbidden induced subgraphs [Weg67, BLS99], which are illustrated in Figure 5.1. If the number of vertices in the forbidden induced subgraphs for unit interval graphs was bounded, then we could apply a general result due to Cai [Cai96], which shows that a bounded search tree algorithm as discussed in Section 2.4 would immediately yield a simple fixed-parameter algorithm for Unit Interval Vertex Deletion. Unfortunately, as illustrated in Figure 5.1, holes of arbitrary lengths are forbidden induced subgraphs for unit interval graphs. Marx [Mar09] has shown a fixed-parameter algorithm for the related problem of making graphs hole-free:

Chordal Vertex Deletion
Input: A graph $G = (V, E)$.
Parameter: A natural number k.
Question: Is $G - S$ hole-free for some vertex set $S \subseteq V$ with $|S| \leq k$?

Section 5.1 shows that Marx' algorithm [Mar09] for Chordal Vertex Deletion can be easily extended to solve Unit Interval Vertex Deletion. However, Marx' algorithm relies on solving Chordal Vertex Deletion on a tree decomposition of width $\Omega(k^4)$ in the worst case, making his algorithm mainly a classification result. Circumventing Marx' algorithm [Mar09], whose total running time is not analyzed in Marx' work, this chapter shows:

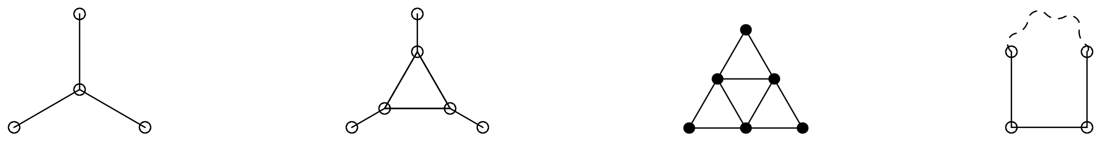

(a) Claw **(b)** Net **(c)** Tent **(d)** Holes

Figure 5.1: The (infinitely many) forbidden induced subgraphs for unit interval graphs. Holes are induced cycles of arbitrary length greater than three.

Theorem 5.1. Unit Interval Vertex Deletion *can be solved in* $O((14k+14)^{k+1} \cdot kn^6)$ *time.*

The corresponding fixed-parameter algorithm developed in this chapter exploits properties specific to unit interval graphs to gain an exponential speedup compared to Marx [Mar09]. It has applications in the social sciences, specifically in the measurement of indifference as discussed in Chapter 1 and by Roberts [Rob78].

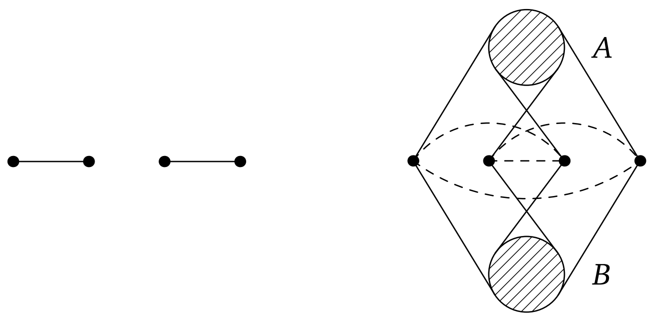
In Section 5.1, we show how Marx' algorithm for Chordal Vertex Deletion can be used to obtain a fixed-parameter algorithm for Unit Interval Vertex Deletion. Moreover, it is shown that Unit Interval Vertex Deletion is NP-hard even on {claw, net, tent}-free graphs, which, in this case, is equivalent to Chordal Vertex Deletion. The remainder of the chapter presents our fixed-parameter algorithm for Unit Interval Vertex Deletion. It is first outlined in Section 5.2. Then, Section 5.3 presents the details and thus proves Theorem 5.1.

5.1 Fixed-Parameter Tractability and a Stronger NP-Hardness Result

This section first shows that Unit Interval Vertex Deletion is fixed-parameter tractable. We exploit the fixed-parameter tractability of Chordal Vertex Deletion [Mar09]. Moreover, we prove that Unit Interval Vertex Deletion is NP-complete even on {claw, net, tent}-free graphs. In this case, Unit Interval Vertex Deletion is equivalent to Chordal Vertex Deletion.

Theorem 5.2. Unit Interval Vertex Deletion *is fixed-parameter tractable.*

Proof. A graph is a unit interval graph if and only if it is {claw, net, tent, hole}-free [Weg67, Rob78]. These forbidden induced subgraphs are shown in Figure 5.1. Marx [Mar09] has presented a fixed-parameter algorithm for Chordal Vertex Deletion. We turn this into a fixed-parameter algorithm for Unit Interval Vertex



(a) Input graph G **(b)** Output graph G'

Figure 5.2: Illustration of the construction made in the proof of Theorem 5.3. The hatched circles represent cliques that are adjacent to all explicitly shown vertices. The dashed edges in G' are the edges of G's complement graph. Observe that any two vertices that are incident to the same edge in G are part of the same hole in G'.

Deletion as follows: given a Unit Interval Vertex Deletion instance (G, k), first enumerate all minimal vertex sets S with $|S| \leq k$ such that $G - S$ is {claw, net, tent}-free. We can, at most k times, choose one of at most six vertices of a claw, net, or tent for inclusion in such a set S. It follows that there are at most 6^k such sets, since choosing a vertex that is not part of a claw, net, or tent would not result in a minimal set. Because we can find a claw, net, or tent in $O(n^6)$ time, we can enumerate these sets in $O(6^k n^6)$ time using a simple search tree algorithm as discussed in Section 2.4.

For each minimal vertex set S that makes $G - S$ {claw, net, tent}-free, apply the fixed-parameter algorithm for Chordal Vertex Deletion to the graph $G - S$ to test if it can be made hole-free by at most $k - |S|$ vertex deletions. It is not hard to verify that this is correct: because every unit interval vertex deletion set S for a graph G makes $G - S$ also {claw, net, tent}-free, we can partition S into a minimal set S' such that $G - S'$ is {claw, net, tent}-free and into a set S^* such that $G - (S' \cup S^*) = G - S$ is additionally hole-free and, therefore, a unit interval graph. □

We now prove that Unit Interval Vertex Deletion is NP-complete even on {claw, net, tent}-free graphs. Note that on these graphs, Chordal Vertex Deletion and Unit Interval Vertex Deletion are the same problem, because a {claw, net, tent}-free graph G is a unit interval graph if and only if G is hole-free [Weg67, Rob78] and because every induced subgraph of G is also {claw, net, tent}-free.

Theorem 5.3. Chordal Vertex Deletion *is NP-hard on {claw, net, tent}-free graphs.*

Proof. The problem is obviously contained in NP. To show NP-hardness, we employ a reduction from Vertex Cover, which is NP-complete even on C_3-free graphs [GJS76].

Vertex Cover
Input: A graph G and a natural number k.
Question: Is there a set S with $|S| \leq k$ such that $G - S$ has no edges?

First, we describe the reduction, then we prove its correctness. The construction made in the following is illustrated in Figure 5.2.

Let (G, k) be a Vertex Cover instance, where G is C_3-free. We construct an instance (G', k) for Chordal Vertex Deletion as follows: let $G' := \overline{G}$, where $\overline{G}$ is the complement graph of G. Add to G' two disjoint cliques A and B, each containing $k+1$ vertices, and make every vertex in A and every vertex in B adjacent to every vertex of $\overline{G}$. Because each of claw, net, and tent contains an independent set of size three, the constructed graph G' is {claw, net, tent}-free: since G is C_3-free, the complement $\overline{G}$ does not contain an independent set of size three. Moreover, there is no independent set of size three in G', since the vertices in the cliques A and B are adjacent to every vertex of $\overline{G}$. It remains to show that (G, k) is a yes-instance of Vertex Cover if and only if (G', k) is a yes-instance of Chordal Vertex Deletion.

If (G, k) is a yes-instance of Vertex Cover, then there is a set S of size k that makes $G - S$ edgeless. As a consequence, the complement of $G - S$ is a clique C. Thus, $G' - S$ contains three cliques A, B, and C, where every vertex of A and B is adjacent to every vertex in C by construction of G'. The graph $G' - S$ is obviously hole-free. Because $|S| \leq k$, (G', k) is a yes-instance of Chordal Vertex Deletion.

If (G', k) is a yes-instance of Chordal Vertex Deletion, then there is a set S with $|S| \leq k$ that makes $G' - S$ hole-free. Now, for the sake of contradiction, assume that there is an edge $\{u, v\}$ in $G - S$, and, therefore, no edge between u and v in G' by construction of G'. Because A and B each contain $k+1$ vertices and $|S| \leq k$, there are vertices $a \in A \setminus S$ and $b \in B \setminus S$; the tuple (a, u, v, b) forms a hole in $G' - S$. This contradicts $G - S$ being hole-free. Hence, S is a set of size at most k that makes $G - S$ edgeless and (G, k) is a yes-instance of Vertex Cover. □

5.2 Outline of a New Fixed-Parameter Algorithm

This section outlines a novel fixed-parameter algorithm for Unit Interval Vertex Deletion. We exploit the iterative compression technique by Reed et al. [RSV04, GMN09] to obtain a fixed-parameter algorithm for Unit Interval Vertex Deletion from the following, and as we will see, simpler problem:

Disjoint Unit Interval Vertex Deletion
Input: A graph G and a unit interval vertex deletion set X for G.
Parameter: $|X|$
Output: A unit interval vertex deletion set S with $|S| < |X|$ and $S \cap X = \emptyset$ or "no" if no such set exists.

Like Unit Interval Vertex Deletion, the Disjoint Unit Interval Vertex Deletion problem is NP-complete [FGMN09]. In contrast to a fixed-parameter algorithm that directly solves Unit Interval Vertex Deletion, a fixed-parameter algorithm for Disjoint Unit Interval Vertex Deletion can exploit the structural information that $G - X$ is a unit interval graph.

In Section 5.2.1, we show how a fixed-parameter algorithm for Disjoint Unit Interval Vertex Deletion can be extended to a fixed-parameter algorithm for Unit Interval Vertex Deletion exploiting the iterative compression technique [RSV04, GMN09]. This will show that Theorem 5.1 is implied by the following theorem:

Theorem 5.4. Disjoint Unit Interval Vertex Deletion *can be solved in* $O((14|X|-1)^{|X|-1} \cdot |X|n^5)$ *time.*

Then, it remains to prove Theorem 5.4. To this end, we develop a bounded search tree algorithm for Disjoint Unit Interval Vertex Deletion, which is outlined in Section 5.2.2. Its details are given in Section 5.3.

5.2.1 Iterative Compression

There are many problems where the destruction of cycles in graphs poses a significant subproblem. To name only two examples: in Feedback Vertex Set, one tries to transform a graph into a forest by a minimum number of vertex deletions; in Chordal Vertex Deletion, one tries to make a graph chordal, that is, hole-free. Fixed-parameter algorithms for such problems often employ the iterative compression technique [RSV04, GGH+06, DFL+07, GMN09]. Similarly, we successfully apply iterative compression to Unit Interval Vertex Deletion, which also involves the destruction of holes (see Figure 5.1).

This section shows how the iterative compression technique allows us to use a fixed-parameter algorithm for Disjoint Unit Interval Vertex Deletion to solve Unit Interval Vertex Deletion. More introductory examples on the technique are given in a survey by Guo et al. [GMN09].

The core of the iterative compression technique is a so-called *compression routine*: Given a graph G, a natural number k, and a unit interval vertex deletion set X with $|X| \leq k+1$ for G, the compression routine compress(G, X, k) outputs a unit interval vertex deletion set S for G with $|S| \leq k$ if such a set exists, or outputs "no" otherwise. We design a compression routine using a fixed-parameter algorithm for Disjoint Unit Interval Vertex Deletion, exploiting the fact that $G - X$ is a unit interval graph. Given such a compression routine, Algorithm 5.1 solves Unit Interval Vertex Deletion.

Algorithm 5.1: Iterative Compression

```
Input: A graph $G$ and its vertices $v_1, \dots, v_n$ in an arbitrary order, $k \in \mathbb{N}$.
Output: A unit interval vertex deletion set $S$ for $G$ with $|S| \leq k$ or “no”.
1 $S_0 \leftarrow \emptyset$;
2 for $i := 1$ to $n$ do
3     $S_i \leftarrow \text{compress}(G[\{v_1, \dots, v_i\}], S_{i-1} \cup \{v_i\}, k)$;
4     if $S_i =$ “no” then return “no”
5 return $S_n$
```

Correctness of Algorithm 5.1. We first show that if none of the S_i is “no”, then S_n is a unit interval vertex deletion set of size at most k for G. Observe that the compression routine never outputs unit interval vertex deletion sets larger than k. Thus, whenever line 3 of Algorithm 5.1 is reached, then S_{i-1} is a unit interval vertex deletion set for $G[\{v_1, \dots, v_{i-1}\}]$ of size at most k. It follows that the set $S_{i-1} \cup \{v_i\}$ is a unit interval vertex deletion set for the graph $G[\{v_1, \dots, v_i\}]$ of size at most $k+1$, which can then be compressed into a unit interval vertex deletion set S_i of size at most k. We conclude that if none of the S_i is “no”, then S_n is a unit interval vertex deletion set for G of size at most k.

It remains to show that if one of the S_i is “no”, then G does not allow for a unit interval vertex deletion set of size at most k. Observe that if G allows for a unit interval vertex deletion set of size at most k, then every induced subgraph of G also does. In contraposition, if $G[\{v_1, \dots, v_i\}]$ does not allow for a unit interval vertex deletion set of size at most k ($S_i =$ “no”) then neither does G.

Compression Routine. We now present a compression routine for Algorithm 5.1. In this compression routine, we exploit an algorithm for Disjoint Unit Interval Vertex Deletion. Recall that the task of the compression routine is, given a graph G, a natural number k, and a unit interval vertex deletion set X with $|X| \leq k+1$, to find a unit interval vertex deletion set S for G with $|S| \leq k$. If $|X| \leq k$, then the compression routine can return X unchanged. Thus, we assume that $|X| = k+1$ and arrive at the task of finding a Unit Interval Vertex Deletion S for G with $|S| < |X|$.

We partition the output unit interval vertex deletion set S into a set $S \backslash X$ of vertices that are not present in the input unit interval vertex deletion set X and into a set $S \cap X$ of vertices that are present in X. Using this partition, we reformulate the task of the compression routine: find a set S with $|S| < |X|$ such that $G - S = G - ((S \cap X) \cup (S \backslash X))$ is a unit interval graph. This, in turn, is equivalent to $S \backslash X$ being a unit interval vertex deletion set for $G - (S \cap X)$ with $|S \backslash X| < |X \backslash S|$. In the following, we exploit this equivalence in a compression routine and analyze its running time.

We find a unit interval vertex deletion set S with $|S| < |X|$ as follows: given X, try all possibilities to choose the set $X' := X \setminus S$ from X. Then, it remains to find a unit interval vertex deletion set that is disjoint from X' and smaller than $|X'|$ in the graph $G' := G - (S \cap X)$. That is, we solve the Disjoint Unit Interval Vertex Deletion instance (G', X'). Because $|X'| \leq |X|$ and because G' is an induced subgraph of G, Theorem 5.4 implies that this works in $O((14|X| - 1)^{|X'|} \cdot |X|n^5)$ time, where n is the number of vertices of G. We can compute G' in $O(n + m)$ time. For brevity, let $a := |X|n^5$ and $b := 14|X| - 1$. There are $\binom{|X|}{k'} = \binom{k+1}{k'}$ possibilities to choose X' with $|X'| = k'$ from X. Thus, the total running time of our compression routine is (up to a constant factor) upper-bounded by

$$a \cdot \sum_{k'=0}^{k+1} \binom{k+1}{k'} b^{k'} = a \cdot (1 + b)^{k+1} = (14|X|)^{k+1} \cdot |X|n^5.$$

Theorem 5.4 implies Theorem 5.1. It is now easily shown that Unit Interval Vertex Deletion is solvable in $O((14k + 14)^{k+1} \cdot kn^6)$ time. To this end, we analyze the running time of Algorithm 5.1: the for-loop initiated in line 2 of Algorithm 5.1 runs at most n times. In each run, compress(G, X) takes $O((14|X|)^{k+1} \cdot |X|n^5)$ time. In the call to compress in line 3, computing the induced subgraph of G takes $O(n + m)$ time and computing $S_{i-1} \cup \{v_i\}$ works in constant time. It follows that each of the n iterations costs $O((14|X|)^{k+1} \cdot |X|n^5)$ time. Because $|X| \leq k + 1$ holds in each iteration, this establishes Theorem 5.1.

In the remainder of this chapter, we focus on solving Disjoint Unit Interval Vertex Deletion in $O((14|X| - 1)^{|X|-1} \cdot |X|n^5)$ time, thus proving Theorem 5.4 and Theorem 5.1. The following subsection outlines an algorithm to solve Disjoint Unit Interval Vertex Deletion. In Section 5.3, we describe the outlined steps in detail.

5.2.2 Disjoint Unit Interval Vertex Deletion

This subsection outlines an algorithm for Disjoint Unit Interval Vertex Deletion. Given a graph G and a unit interval vertex deletion set X for G, we search for a vertex set S with $S \cap X = \emptyset$ and $|S| < |X|$ such that deleting the vertices in S from G destroys all claws, nets, tents, and holes (shown in Figure 5.1).

The algorithm roughly works as follows: first, similarly to the proof of Theorem 5.2, enumerate *all* minimal vertex sets of size at most $|X| - 1$ whose deletion transforms a graph into a {claw, net, tent, C_4, C_5, C_6}-free graph, henceforth called *almost unit interval graph*. We obtain a family of almost unit interval graphs. For each of these, it remains to find a minimum vertex set whose removal destroys all holes of length greater than six to transform it into a unit interval graph. If the size of this vertex

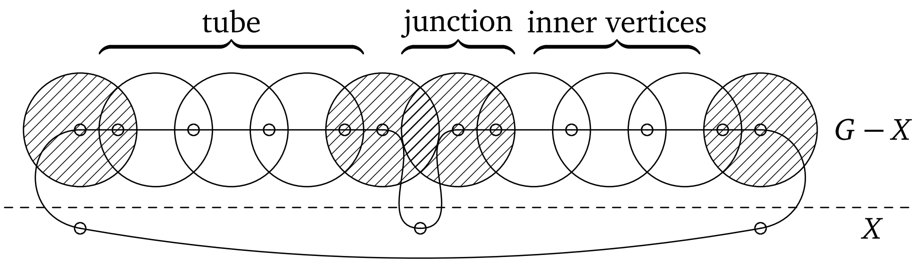


Figure 5.3: A hole that visits the maximal cliques of $G - X$, which are indicated by circles. Hatched circles show junctions. The vertices of $G - X$ are shown from left to right in a bicompatible elimination order. A maximal set of consecutive white cliques forms a tube. As illustrated, tubes may contain vertices of junctions. For the right tube, we indicate the inner vertices.

set together with the number of vertex deletions needed to transform a graph into an almost unit interval graph is at most $|X| - 1$, then we have found a solution.

To destroy all holes in an almost unit interval graph G, we show that each hole has a set of $14|X| - 1$ vertices of which at least one vertex can be assumed to be in a minimum unit interval vertex deletion set. This allows us to use a bounded search tree algorithm that, for each hole H in G, recursively branches into $14|X| - 1$ possibilities to destroy H. Because at most $|X| - 1$ vertices may be deleted from G to transform it into a unit interval graph, the recursion depth is bounded by $|X| - 1$. The corresponding search tree (also refer to Section 2.4) has $O((14|X|-1)^{|X|-1})$ nodes, which will result in the running time stated in Theorem 5.4.

To find this set of $14|X| - 1$ vertices for each hole in G, we exploit that $G - X$ is a unit interval graph. Unit interval graphs allow for a linear-time computable *bicompatible elimination order* of their vertices [PD03]:

Definition 5.1. Let $G = (V, E)$ be a graph. A total order $\preceq$ on V is a *bicompatible elimination order* for G if for each vertex $v \in V$, the sets $\{w \in N(v) \mid w \preceq v\}$ and $\{w \in N(v) \mid v \preceq w\}$ induce cliques in G.

Without loss of generality, we require that the vertices of a connected component of G form a segment of $\preceq$.

We will see that, with respect to a bicompatible elimination order, $G - X$ forms a sequence of maximal cliques such that the vertices of each maximal clique form a segment of the bicompatible elimination order. Figure 5.3 illustrates this together with the following classification of the maximal cliques of $G - X$:

1. A *junction* is a maximal clique in $G - X$ containing neighbors of vertices in X.

2. With respect to a bicompatible elimination order $\preceq$ for $G - X$, a *tube* is a maximal set of maximal cliques of $G - X$ that are not junctions and whose vertices form a segment of $\preceq$.

We say that a vertex is *contained in a tube* T if it is contained in a maximal clique of T. A hole *visits* a junction (or tube) if it contains a vertex of a junction (or tube). Vertices of a tube that are not in junctions are *inner vertices*.

Now, assume that there is a hole H in an almost unit interval graph G as illustrated in Figure 5.3. We show $14|X| - 1$ possibilities to destroy H, of which one is optimal. Each vertex of H in $G - X$ is contained in a junction or tube (or both). Section 5.3.1 shows that H contains at most $12|X|$ vertices in junctions of $G - X$. Then, Section 5.3.2 shows that H contains inner vertices of at most $2|X| - 1$ tubes. Additionally, Section 5.3.2 shows that there is a minimum unit interval vertex deletion set that contains vertices of H in junctions or a polynomial-time computable vertex of H in one of the $2|X| - 1$ tubes that it visits. Section 5.3.3 finally presents a bounded search tree algorithm that repeatedly searches for a hole H in G and branches into the following $14|X| - 1$ possibilities to destroy H:

1. delete one of the $12|X|$ vertices of H in junctions, or

2. delete an optimal, polynomial-time computable vertex of H in one of the $2|X| - 1$ tubes whose inner vertices are visited by H.

The next section will give the details of these steps; we show an algorithm for Disjoint Unit Interval Vertex Deletion running in $O((14|X| - 1)^{|X|-1} \cdot |X|n^5)$ time. This proves Theorem 5.1.

5.3 Algorithmic Details

This section presents the details of the Disjoint Unit Interval Vertex Deletion algorithm. Given a graph G and a unit interval vertex deletion set X for G, we search for a unit interval vertex deletion set S for G with $|S| < |X|$ and $S \cap X = \emptyset$. Following the outline in Section 5.2.2, we first transform the input graph G into a family of almost unit interval graphs. To this end, the following branching rule (starting with an empty set S) is employed:

Branching Rule 5.1. If $G - S$ contains a forbidden induced subgraph induced by a vertex set F with $|F| \leq 6$, then recursively branch into all possibilities of adding a vertex $v \in F \setminus X$ to S.

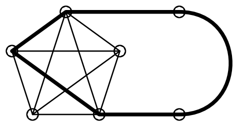


Figure 5.4: If a cycle contains three vertices of a clique, it cannot be a hole because it has a chord.

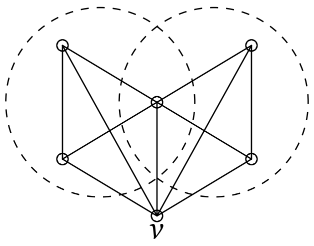


Figure 5.5: The neighborhood of v is covered by two cliques, shown as circles.

The algorithm stops branching if $|S| \geq |X| - 1$. If in this case $G - S$ still contains a forbidden induced subgraph, then S cannot be part of a unit interval vertex deletion set smaller than X and disjoint from X. Since each forbidden induced subgraph in G contains a vertex in X (otherwise, X would not be a unit interval vertex deletion set), Branching Rule 5.1 branches into at most five cases. Thus, we obtain $O(5^{|X|-1})$ sets S such that $G - S$ is an almost unit interval graph. In the following sections, it remains to transform these almost unit interval graphs into unit interval graphs.

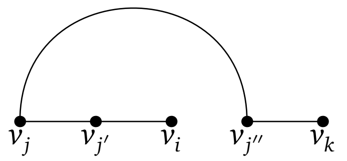
5.3.1 Bounding the Number of Vertices in Junctions

Following the outline in Section 5.2.2, this subsection shows that, for an almost unit interval graph G and a unit interval vertex deletion set X for G, a hole in G contains at most $12|X|$ vertices in junctions of $G - X$. Before proceeding in this respect, recall that a junction is a maximal clique in $G - X$ that contains neighbors of vertices in X (see Figure 5.3). The goal of this subsection is proving:

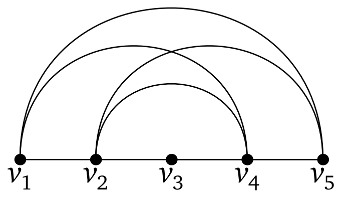
Lemma 5.1. *Let X be a unit interval vertex deletion set for an almost unit interval graph G. A hole in G contains at most $12|X|$ vertices in junctions of $G - X$.*

In the following, we provide the necessary observations to prove Lemma 5.1. First, observe that a cycle that contains more than two vertices of a clique has a chord. This is illustrated in Figure 5.4. As a result, no hole contains more than two vertices from a single clique.

In the following observation, we exploit that G is an almost unit interval graph. We say that a vertex subset of a graph G can be covered by two cliques if it is the union of two vertex sets which induce cliques in G. We assume that the cliques that cover a vertex subset of G only contain vertices of that vertex subset. The observation is illustrated in Figure 5.5.



(a) The vertices $v_j \preceq v_{j'} \preceq v_{j''}$ form the induced path $(v_{j'}, v_j, v_{j''})$, in wrong order.



(b) A maximal clique induced by the non-consecutive vertices v_1, v_2, v_4, and v_5.

Figure 5.6: The vertex orders from left to right are not bicompatible elimination orders, as they violate Observation 5.2.

Observation 5.1. *If a connected almost unit interval graph G contains a hole, then the neighborhood of each vertex in G can be covered by two cliques.*

Proof. Because the almost unit interval graph G contains no induced claw, net, tent, C_4, C_5, or C_6, it contains a hole with more than six vertices. Such a hole contains an independent set of size three. We now apply a result due to Fouquet [Fou93]:

> In a connected claw-free graph containing an independent set of size three, every vertex v satisfies exactly one of the following properties:
>
> 1. $N(v)$ can be covered by two cliques or
> 2. $N(v)$ contains an induced C_5.

Because G contains no induced C_5, the observation follows immediately. □

From this observation, one can conclude that if an almost unit interval graph G contains a hole, then the neighborhood of a unit interval vertex deletion set X in $G - X$ can be covered by $2|X|$ cliques. Note that these cliques are not necessarily maximal in $G - X$. Thus, Observation 5.1 does not yield an upper bound on the number of junctions of $G - X$.

In addition to Observation 5.1, we exploit $G-X$ being a unit interval graph to prove Lemma 5.1. Unit interval graphs are equivalent to proper interval graphs [Rob69, Rob78]. It follows that a graph is a unit interval graph if and only if it allows for a linear-time computable *bicompatible elimination order* [PD03], defined in Definition 5.1. The following observation says that maximal cliques of a unit interval graph form segments of any bicompatible elimination order and that vertices on induced paths occur in the same (or inverse) order as in any bicompatible elimination order. It immediately follows that the vertex orders shown in Figure 5.6 are not bicompatible elimination orders.

Observation 5.2. *Let $v_1 \preceq v_2 \preceq \ldots \preceq v_n$ be a bicompatible elimination order for a connected unit interval graph G.*

(1) If there is an induced path $P = (v_i, \ldots, v_k)$ with $v_i \preceq v_k$, then every vertex v_j on P satisfies $v_i \preceq v_j \preceq v_k$.

(2) If $v_i \preceq v_k$ and there is an edge between v_i and v_k, then the segment $[v_i, v_k]$ induces a clique in G. In particular, maximal cliques of G form segments.

Proof. We prove the two listed statements independently. To show (1), for the purpose of contradiction, assume that $P = (v_i, \ldots, v_j, \ldots, v_k)$ is an induced path with $v_j \preceq v_i \preceq v_k$ (the case $v_i \preceq v_k \preceq v_j$ can be proven analogously) such that v_j is the minimum vertex with respect to $\preceq$ that appears between v_i and v_k on P. Figure 5.6a illustrates this arrangement. Because there are induced subpaths of P from v_j to both v_i and v_k, the vertex v_j has two distinct neighbors $v_{j'}$ and $v_{j''}$ on P. Because v_j is the minimum vertex with respect to $\preceq$ that appears between v_i and v_k on P, it holds that $v_j \preceq v_{j'}$ and $v_j \preceq v_{j''}$. The vertices $v_{j'}$ and $v_{j''}$, which both are succeeding neighbors of v_j, are adjacent by Definition 5.1. This contradicts P being *induced*.

Prior to showing (2), we show that there is an edge between a vertex v_i and its direct successor v_{i+1} for $i \in \{1, \ldots, n-1\}$. By assumption, the graph G is connected. This implies the existence of a shortest (and hence, induced) path from v_i to v_{i+1}. By (1), this path can neither contain a predecessor of v_i nor a successor of v_{i+1}. Because v_{i+1} directly succeeds v_i in $\preceq$, v_i and v_{i+1} are the only vertices on the shortest path from v_i to v_{i+1}. Hence, they are adjacent.

We now show (2). Let $v_i \preceq v_k$ and assume that there is an edge between v_i and v_k. We have already shown that v_i is adjacent to its direct successor v_{i+1}. Because v_{i+1} and v_k are succeeding neighbors of v_i, the vertices v_i, v_{i+1}, and v_k form a clique by Definition 5.1. Inductively, it follows that all vertices v_j with $v_i \preceq v_j \preceq v_k$ are adjacent to v_k. By Definition 5.1, these vertices form a clique together with v_k because they are preceding neighbors of v_k in $\preceq$. Finally, a maximal clique in G contains an edge from its minimum vertex to its maximum vertex and, hence, forms a segment. □

To prove Lemma 5.1, we finally need the following definition, which is illustrated in Figure 5.7, and its subsequent observation.

Definition 5.2. Let G be a unit interval graph with a bicompatible elimination order $\preceq$ and let C be a clique of G. We define $\mathcal{S}(C)$ to be the set of vertices of all maximal cliques in G containing vertices of C.

Let $c_{\min}$ (and $c_{\max}$) be the minimum (or maximum, respectively) elements of $\mathcal{S}(C)$ with respect to $\preceq$. We define $\mathcal{S}_{\min}(C)$ (or $\mathcal{S}_{\max}(C)$) to be the vertex set of the (uniquely determined) maximal clique in G that contains $c_{\min}$ (or $c_{\max}$, respectively) and any vertex of C.

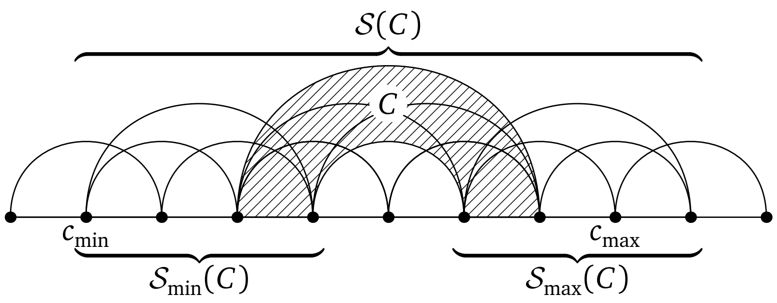


Figure 5.7: Illustration for Definition 5.2. The vertices are shown from left to right in a bicompatible elimination order. The hatched clique is C.

Observation 5.3. *Let G be a unit interval graph with a bicompatible elimination order $\preceq$. For every clique C of G and the vertex set C' of any maximal clique containing C, it holds that $\mathcal{S}(C) = C' \cup \mathcal{S}_{min}(C) \cup \mathcal{S}_{max}(C)$. Moreover, $\mathcal{S}(C)$ is a segment.*

Proof. By Definition 5.2, it holds that $C' \cup \mathcal{S}_{\min}(C) \cup \mathcal{S}_{\max}(C) \subseteq \mathcal{S}(C)$. Now, let $c_{\min}$ (and $c_{\max}$) be the minimum (or maximum, respectively) vertex of $\mathcal{S}(C)$ with respect to $\preceq$. Obviously, $\mathcal{S}(C) \subseteq [c_{\min}, c_{\max}]$. By Observation 5.2(2), the sets C', $\mathcal{S}_{\min}(C)$, and $\mathcal{S}_{\max}(C)$ form segments, because they induce maximal cliques. Since $\mathcal{S}_{\min}(C)$ and $\mathcal{S}_{\max}(C)$ contain vertices of C, both segments intersect with C'. It follows that $C' \cup \mathcal{S}_{\min}(C) \cup \mathcal{S}_{\max}(C)$ is the union of intersecting segments and, hence, a segment. By Definition 5.2, $C' \cup \mathcal{S}_{\min}(C) \cup \mathcal{S}_{\max}(C)$ contains $c_{\min}$ and $c_{\max}$. It follows that $\mathcal{S}(C) \subseteq [c_{\min}, c_{\max}] \subseteq C' \cup \mathcal{S}_{\min}(C) \cup \mathcal{S}_{\max}(C)$. □

We have now collected the necessary observations to prove Lemma 5.1, which for an almost unit interval graph G and a unit interval vertex deletion set X shows that a hole contains at most $12|X|$ vertices in junctions of $G - X$.

Proof of Lemma 5.1. By Observation 5.1, the neighborhood of X in $G - X$ can be covered by a set $\mathcal{C}$ of $2|X|$ cliques. By Definition 5.2, the vertices of all junctions containing vertices of a clique $C \in \mathcal{C}$ are in $\mathcal{S}(C)$. We show that a hole contains at most six vertices in $\mathcal{S}(C)$ and, thus, in all junctions containing vertices of C. A hole contains at most two vertices in the vertex set C' of any maximal clique containing C, at most two vertices of the clique induced by $\mathcal{S}_{\min}(C)$, and at most two vertices in the clique induced by $\mathcal{S}_{\max}(C)$. By Observation 5.3, one has $\mathcal{S}(C) = \mathcal{S}_{\min}(C) \cup \mathcal{S}_{\max}(C) \cup C'$. Hence, a hole contains at most six vertices in $\mathcal{S}(C)$. This worst-case scenario is illustrated in Figure 5.8.

A hole contains at most six vertices in all junctions containing vertices of C. From $|\mathcal{C}| \leq 2|X|$, it follows that a hole contains at most $12|X|$ vertices in junctions of $G - X$. □

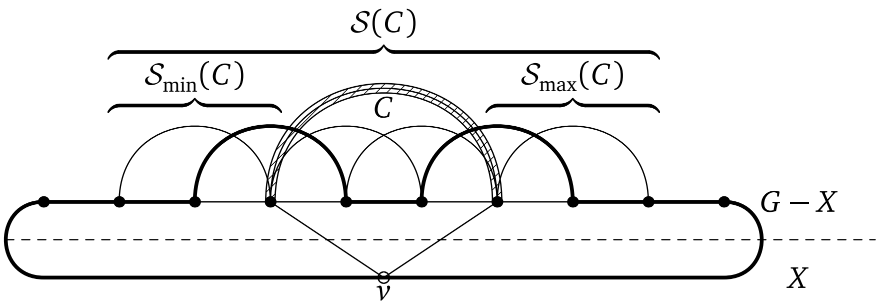


Figure 5.8: The neighborhood of v induces the two-vertex hatched clique C. The hole drawn in bold edges contains six vertices in junctions containing vertices of C.

5.3.2 Finding Optimal Solutions in Tubes

For an almost unit interval graph G and a unit interval vertex deletion set X, Lemma 5.1 showed that a hole H in G contains at most $12|X|$ vertices in junctions of $G-X$. Following the outline in Section 5.2.2, this subsection shows that H visits inner vertices of at most $2|X|-1$ tubes in $G-X$. Moreover, we show that there is a minimum unit interval vertex deletion set containing at least one vertex of H in junctions or a polynomial-time computable vertex of H in one of the $2|X|-1$ tubes whose inner vertices are visited by H. As discussed in the outline in Section 5.2.2, this allows us to solve Disjoint Unit Interval Vertex Deletion by trying at most $14|X|-1$ possibilities to destroy any hole H in G.

Recall that a tube with respect to a bicompatible elimination order $\preceq$ of $G-X$ is a maximal set of maximal cliques of $G-X$ that are not junctions and whose vertices form a segment of $\preceq$. Moreover, recall that inner vertices of a tube are those vertices that are not contained in junctions. Before stating the first main result of this subsection, we show the following observation, which implies that if two tubes contain a common vertex, then these tubes are equal.

Observation 5.4. *Let X be a unit interval vertex deletion set for a graph G and let $\preceq$ be a bicompatible elimination order for $G-X$. Let T_1 be a tube in $G-X$ and let T_2 be a set of maximal cliques of $G-X$ that are not junctions and whose vertices form a segment of $\preceq$. If T_1 and T_2 contain a common vertex, then $T_2 \subseteq T_1$.*

Proof. The set $T_1 \cup T_2$ only contains maximal cliques of $G-X$ that are not junctions. Moreover, the cliques in T_1 form a segment and the cliques in T_2 form a segment. Both segments contain a common vertex. Hence, the cliques in $T_1 \cup T_2$ form a segment. Because T_1 is a *maximal* set of maximal cliques that are not junctions and whose vertices form a segment, $T_1 = T_1 \cup T_2$, that is, $T_2 \subseteq T_1$. □

We now state the first of the two main results of this subsection.

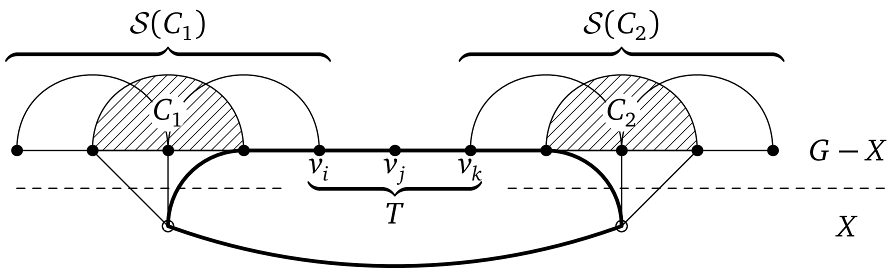


Figure 5.9: A hole (bold edges) visiting a tube T. The vertices of $G-X$ are shown from left to right in a bicompatible elimination order. The hatched cliques are neighborhoods of vertices in X. Observe that all vertices in the segments $\mathcal{S}(C_1)$ and $\mathcal{S}(C_2)$ are contained in junctions. The tube T contains the single inner vertex v_j.

Lemma 5.2. *Let X be a unit interval vertex deletion set for an almost unit interval graph G. A hole in G contains inner vertices of at most $2|X|-1$ tubes with respect to a bicompatible elimination order $\preceq$ for $G-X$.*

Proof. By Observation 5.1, the neighborhood of X can be covered by a set $\mathcal{C}$ of $2|X|$ cliques in $G-X$. Let $\mathcal{T}$ be the set of tubes that contain inner vertices visited by H. Recall from Definition 5.2 that $\mathcal{S}(C)$ is the set containing the vertices of all maximal cliques that contain vertices of C. Observe that for a clique $C \in \mathcal{C}$, all vertices in $\mathcal{S}(C)$ are contained in junctions, as Observation 5.3 shows that a vertex in $\mathcal{S}(C)$ is contained in the junction $\mathcal{S}_{\min}(C)$, the junction $\mathcal{S}_{\max}(C)$, or in a junction containing C. Moreover, Observation 5.3 shows that $\mathcal{S}(C)$ is a segment.

The proof now works as follows (see Figure 5.9 for illustrations): first, we show that each tube $T \in \mathcal{T}$ contains three vertices $v_i \preceq v_j \preceq v_k$ such that there are two cliques $C_1, C_2 \in \mathcal{C}$ with $v_i \in \mathcal{S}(C_1)$, $v_k \in \mathcal{S}(C_2)$ and $v_j \notin \mathcal{S}(C_1) \cup \mathcal{S}(C_2)$. Because $\mathcal{S}(C_1)$ and $\mathcal{S}(C_2)$ are segments, this implies that all vertices in $\mathcal{S}(C_1)$ precede all vertices of $\mathcal{S}(C_2)$ in $\preceq$. Then, we show that, for each clique $C \in \mathcal{C}$, there is at most one tube T containing vertices $v_i \preceq v_j$ with $v_i \in \mathcal{S}(C)$ and $v_j \notin \mathcal{S}(C)$. This already yields $|\mathcal{T}| \leq |\mathcal{C}|$. Now, observe that there is a clique $C_1 \in \mathcal{C}$ for which there is *no* clique $C_2 \in \mathcal{C}$ such that all vertices in $\mathcal{S}(C_1)$ precede the vertices in $\mathcal{S}(C_2)$ (the "last" clique). This yields $|\mathcal{T}| \leq |\mathcal{C}| - 1 \leq 2|X| - 1$.

Let $T \in \mathcal{T}$ and let v_i, with respect to $\preceq$, be the minimum vertex of H in T. By definition of $\mathcal{T}$, the hole H contains an inner vertex v_j of the tube T. By choice of v_i, it holds that $v_i \preceq v_j$. Moreover, we have $v_j \notin \mathcal{S}(C)$ for all cliques $C \in \mathcal{C}$ since all vertices in $\mathcal{S}(C)$ are in junctions and v_j is an inner vertex. All neighbors of v_i are in $G-X$, as v_i is a vertex of the tube T. By Observation 5.2(1), this implies that v_i has, with respect to $\preceq$, a preceding neighbor v_i' in H. Since v_i' and v_i are neighbors, there is a maximal clique containing v_i and v_i'. This maximal clique is

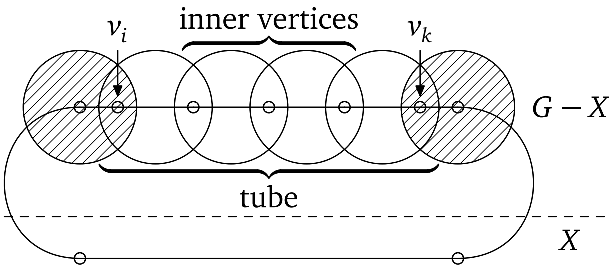


Figure 5.10: A hole illustrating Definition 5.3. Circles represent maximal cliques. The hatched cliques are junctions. The T-boundary for the shown hole is (v_i, v_k).

a junction, because v_i' is not in T by choice of v_i. This implies the existence of a clique $C \in \mathcal{C}$ with $v_i \in \mathcal{S}(C)$. Analogously, we find a clique $C \in \mathcal{C}$ for which $\mathcal{S}(C)$ contains, with respect to $\preceq$, the maximum vertex v_k of H in T. It follows that we find cliques $C_1, C_2 \in \mathcal{C}$ such that $v_i \in \mathcal{S}(C_1)$, $v_k \in \mathcal{S}(C_2)$ and $v_j \notin \mathcal{S}(C_1) \cup \mathcal{S}(C_2)$. By choice of v_i and v_k, it holds that $v_i \preceq v_j \preceq v_k$.

It remains to show that for a clique $C \in \mathcal{C}$, there is at most one tube $T \in \mathcal{T}$ containing vertices $v_i \preceq v_j$ such that $v_i \in \mathcal{S}(C)$ and $v_j \notin \mathcal{S}(C)$. Assume that there are two such tubes $T_1, T_2 \in \mathcal{T}$. We show that $T_1 = T_2$. Let $v_i \preceq v_k$ be vertices of T_1 and $v_i' \preceq v_k'$ be vertices of T_2 such that $v_i, v_i' \in \mathcal{S}(C)$ and $v_k, v_k' \notin \mathcal{S}(C)$. Because T_1 and T_2 are tubes, T_1 and T_2 contain the vertices in the segments $[v_i, v_k]$ and $[v_i', v_k']$, respectively. From $v_i, v_i' \in \mathcal{S}(C)$ and $v_k, v_k' \notin \mathcal{S}(C)$, we conclude that the segments $[v_i, v_k]$ and $[v_i', v_k']$ intersect. Therefore, the tubes T_1 and T_2 contain a common vertex. By Observation 5.4, this implies $T_1 = T_2$. □

Lemma 5.2 bounds the number of tubes whose inner vertices are visited by a hole. Following the outline in Section 5.2.2, it remains to show that there is a minimum unit interval vertex deletion set that contains vertices of a hole in junctions or a polynomial-time computable vertex in one of the $2|X| - 1$ tubes whose inner vertices are visited by the hole. To state the second main result of this subsection, we need the following concepts, which are illustrated in Figure 5.10.

Definition 5.3. Let X be a unit interval vertex deletion set for a graph G and let T be a tube in $G - X$ with respect to a bicompatible elimination order $\preceq$ such that T contains inner vertices visited by a hole H.

1. For two vertices v_i and v_k of H, we call (v_i, v_k) the *T-boundary* of H if, with respect to $\preceq$, v_i is the preceding neighbor in H of H's minimum inner vertex in T and if v_k is the succeeding neighbor in H of H's maximum inner vertex of T.

2. For the T-boundary (v_i, v_k) of a hole H, we call a (minimum) vertex-cut between v_i and v_k in $G - X$ a *(minimum) H-T-cut*.

Observation 5.5. *For the T-boundary (v_i, v_k) of a hole, the tube T contains v_i and v_k.*

Proof. We only show that v_i is a vertex of T. Analogously, it can be shown that v_k is a vertex of T. By Definition 5.3(1), v_i is the neighbor of an inner vertex $v_j \in [v_i, v_k]$ of T. There is a maximal clique C in $G - X$ that contains v_i and v_j, because v_i and v_j are neighbors. Because v_j is not part of a junction, all maximal cliques in $G - X$ containing v_j are not junctions. It follows that C is not a junction and that $\{C\}$ is a set containing a maximal clique that is not a junction and whose vertices form a segment of $\preceq$. Because the tube T and the set C contain the common vertex v_j, Observation 5.4 implies $C \in T$. In particular, T contains v_i. Analogously, it follows that T contains v_k. □

We can now state the second main result of this subsection. It shows that if for a hole H, we do not delete vertices in junctions, then we must delete a minimum H-T-cut for some tube T.

Lemma 5.3. *Let X be a unit interval vertex deletion set for a graph G. Let $\mathcal{T}$ be the set of tubes in $G - X$ with respect to a bicompatible elimination order $\preceq$ that contain inner vertices visited by a hole H.*

If a unit interval vertex deletion set S with $S \cap X = \emptyset$ does not contain vertices of H in junctions of $G - X$, then there is a unit interval vertex deletion set S' with $|S'| \leq |S|$ and a tube $T \in \mathcal{T}$ for which S' contains a minimum H-T-cut.

The remainder of this subsection is dedicated to the proof of Lemma 5.3. First, we present one further observation that is used to show Lemma 5.3.

Observation 5.6. *Let X be a unit interval vertex deletion set for a graph G and let T be a tube in $G - X$ with respect to a bicompatible elimination order $\preceq$. If a hole H visits T, then, for any two vertices $v_i \preceq v_k$ of H in T, the hole H has an induced subpath from v_i to v_k consisting of only vertices in the segment $[v_i, v_k]$.*

Proof. Because T is a tube, no vertex in $[v_i, v_k]$ has neighbors in X. From Observation 5.2(1), it follows that v_i has a succeeding neighbor in H and v_k has a preceding neighbor in H. As a consequence, H contains a vertex $v_j \in [v_i, v_k]$. Thus, H has an induced subpath P from v_i via v_j to v_k. Because no vertex in $[v_i, v_k]$ has neighbors in X, the induced path P contains only vertices $[v_i, v_k]$ by Observation 5.2(1). □

We now prove Lemma 5.3, which essentially shows that if for a hole H we do not delete vertices in junctions, then we must delete a minimum H-T-cut for some tube T. The proof is subdivided into two claims: we first show that a unit interval vertex deletion set S that is disjoint from a given unit interval vertex deletion set X contains an H-T-cut C for some tube $T \in \mathcal{T}$ if it does not contain vertices of H in

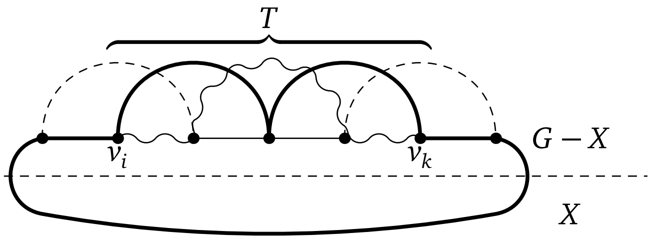


Figure 5.11: Difficulty in the proof of Claim 5.1. A hole H (bold edges) visits the tube T. The T-boundary of H is (v_i, v_k). If we replace the bold path from v_i to v_k through T by the waved path, we do not obtain an induced cycle because the dashed edges form chords.

junctions. Second, we show that a minimum H-T-cut C' contains a vertex of every hole that contains vertices within the T-boundary of H. Finally, we show that if C is not a *minimum* H-T-cut, then we can transform S into a unit interval vertex deletion set S' with $|S'| < |S|$ that contains C'.

Claim 5.1. Let X be a unit interval vertex deletion set for a graph G. Let the set $\mathcal{T}$ contain the tubes in G with respect to a bicompatible elimination order $\preceq$ whose inner vertices are visited by a hole H in G. If a unit interval vertex deletion set S with $S \cap X = \emptyset$ does not contain vertices of H in junctions, then S contains an H-T-cut for some tube $T \in \mathcal{T}$.

Proof. For the sake of contradiction, assume that S does not contain an H-T-cut for any tube $T \in \mathcal{T}$. We show that $G - S$ contains a hole. The proof is based on the following idea, which is simple but needs some care. For each tube $T \in \mathcal{T}$, consider the T-boundary (v_i, v_k) of H. By Observation 5.5, the tube T contains v_i and v_k, which by Observation 5.6 implies the existence of a subpath $P(T)$ of H from v_i to v_k in $G - X$ only containing vertices of T. As S does not contain an H-T-cut, there is a shortest (and therefore induced) path $Q(T)$ from v_i to v_k in $G - (S \cup X)$. Moreover, S does neither contain vertices of H in junctions nor in X. It follows that we obtain a cycle C in $G - S$ from H by replacing $P(T)$ by $Q(T)$ in H for each tube $T \in \mathcal{T}$. Unfortunately, C is not necessarily a hole, as Figure 5.11 illustrates. In the following, we construct a hole in $G - S$ from C, which contradicts the assumption that S is a unit interval vertex deletion set.

First, we show that C is a simple cycle, that is, it does not visit a vertex twice. By Observation 5.2(1), for each tube $T \in \mathcal{T}$ and its T-boundary (v_i, v_k), the induced paths $P(T)$ and $Q(T)$ only contain vertices in the segment $[v_i, v_k]$. In particular, they are vertex-disjoint from any induced subpath of H that does not contain vertices in $[v_i, v_k]$. Additionally, because two distinct tubes $T_1, T_2 \in \mathcal{T}$ do not contain common

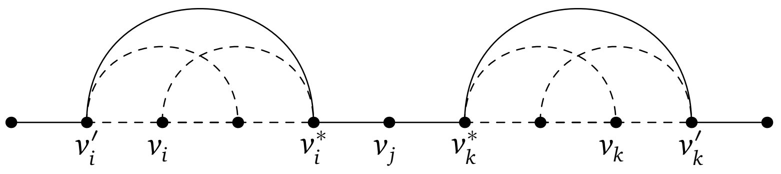


Figure 5.12: Illustration for the proof of Claim 5.2. The solid path is part of the hole H' that is assumed to exist. The dashed edges are implied by Observation 5.2(2) and connect v_i to v_k.

vertices (Observation 5.4), the paths $Q(T_1)$ and $Q(T_2)$ are vertex-disjoint. It follows that C is simple.

We now construct a hole in $G - S$ from the cycle C. The cycle C contains a vertex $v \in X$. For some tube $T \in \mathcal{T}$, consider a vertex v_j on the replacement path $Q(T)$. The vertex v_j is a vertex of C. The shortest path from v_j to v consists of at least three vertices, because v_j is contained in T and thus has no neighbors in X. From C being a simple cycle, it follows that two paths from v to v_j exist that, except for v and v_j, are vertex-disjoint. Because both paths consist of at least three vertices, we can conclude that there is a hole in $G - S$. This contradicts the assumption that S is a unit interval vertex deletion set. □

Claim 5.2. Let X be a unit interval vertex deletion set for a graph G. Let T be a tube in G with respect to a bicompatible elimination order $\preceq$ such that the inner vertices of T are visited by a hole H. If a vertex set S contains an H-T-cut, then no hole in $G - S$ contains vertices of the segment $[v_i, v_k]$, where (v_i, v_k) is the T-boundary of H.

Proof. For the sake of contradiction, assume that a hole H' in $G - S$ contains a vertex $v_j \in [v_i, v_k]$. The constructions made in this proof are illustrated in Figure 5.12. Let v_i^* be the minimum vertex of H' in T with $v_i \preceq v_i^*$ and let v_k^* be the maximum vertex of H' in T with $v_k^* \preceq v_k$. Because v_i^* and v_k^* are in T, they have no neighbors in X. Observation 5.2(1) implies that v_i^* has a preceding neighbor v_i' in H' and that v_k^* has a succeeding neighbor v_k' in H'. By choice of v_i^*, v_k^*, v_i', and v_k' and because v_i and v_k are in T by Observation 5.5, we have $v_i' \preceq v_i \preceq v_i^* \preceq v_j \preceq v_k^* \preceq v_k \preceq v_k'$ (they need not be pairwise distinct). Because v_i' and v_i^* are adjacent, $[v_i', v_i^*] \setminus S$ induces a clique in $G - (S \cup X)$ that contains v_i by Observation 5.2(2). Thus, if $v_i \neq v_i^*$, then v_i is adjacent to v_i^*. Analogously, v_k' is adjacent to v_k^* if $v_k' \neq v_k^*$. Since the vertices v_i^* and v_k^* are connected in $G - (S \cup X)$, there is a path from v_i to v_k in $G - (S \cup X)$, contradicting the assumption that S contains an H-T-cut. It follows that no hole in $G - S$ contains vertices of the segment $[v_i, v_k]$ if S contains an H-T-cut. □

We now prove Lemma 5.3. We show that if for a hole H, we do not delete vertices in junctions, then we must delete a minimum H-T-cut for some tube T.

Proof of Lemma 5.3. By assumption, the unit interval vertex deletion set S with $S \cap X = \emptyset$ does not contain vertices of the hole H in junctions of $G - X$. By Claim 5.1, S contains an H-T-cut C for some tube $T \in \mathcal{T}$. We show that there is a unit interval vertex deletion set S' with $|S'| \leq |S|$ that contains a minimum H-T-cut.

Let (v_i, v_k) be the T-boundary of H. We first show that $C' := C \cap [v_i, v_k]$ is an H-T-cut: if $G - X$ contains an induced path P from v_i to v_k, then P only contains vertices from the segment $[v_i, v_k]$ by Observation 5.2(1). As the H-T-cut C contains a vertex of P, which is in $[v_i, v_k]$, also $C' = C \cap [v_i, v_k]$ contains a vertex of P. It follows that C' is an H-T-cut.

If C' is a *minimum* H-T-cut, then the lemma is proven because $C' \subseteq S$. Otherwise, let C^* be a minimum H-T-cut. Because C^* is a minimum H-T-cut, one has $|C^*| \leq |C'|$ and $C^* \subseteq [v_i, v_k]$; otherwise, $C^* \cap [v_i, v_k]$ would be a smaller H-T-cut. We now choose $S' := (S \setminus C') \cup C^*$ and show that S' is a unit interval vertex deletion set for G.

Assume, for the sake of contradiction, that $G - S'$ contains a hole. Because $G - S$ is a unit interval graph, this hole contains a vertex that is in $G - S'$ but not in $G - S$, implying that the hole contains a vertex from $S \setminus S' = C' \setminus C^*$. However, C' only contains vertices in the segment $[v_i, v_k]$ and no hole in $G - (S' \cup X)$ contains vertices in this segment by Claim 5.2, because S' contains the H-T-cut C^*. It follows that S' is a unit interval vertex deletion set. Finally, $|C^*| \leq |C'|$ implies $|S'| \leq |S|$. □

5.3.3 Bounded Search Tree

Based on the results in the previous subsections, we now prove Theorem 5.4.

Proof of Theorem 5.4. Given an n-vertex graph G and a unit interval vertex deletion set X for G, we search for a unit interval vertex deletion set S for G with $|S| < |X|$ and $S \cap X = \emptyset$. We start with $S := \emptyset$ and apply Branching Rule 5.1 as long as $|S| < |X|$ to destroy forbidden induced subgraphs with at most six vertices. Because each such forbidden induced subgraph contains one vertex in the unit interval vertex deletion set X, we find such a forbidden induced subgraph in $O(|X|n^5)$ time and branch into at most five cases to add one of its vertices to S.

If $|S| \geq |X|$ and Branching Rule 5.1 is still applicable, return "no" because S cannot be contained in a unit interval vertex deletion set for G that is smaller than X. Otherwise, proceed as follows: compute a bicompatible elimination order $\preceq$ for $G - (S \cup X)$. This works in linear time [PD03] because $G - (S \cup X)$ is a unit interval graph. From this bicompatible elimination order, a set $\mathcal{C}$ of all maximal cliques of $G - (S \cup X)$ can easily be computed in $O(n^2)$ time by finding, for each vertex v in $G - (S \cup X)$, its last neighbor with respect to $\preceq$.

Now, sort the set $\mathcal{C}$ such that C_1 occurs before C_2 if, in $\preceq$, the minimum vertex of C_1 occurs before the minimum vertex of C_2. Because a unit interval graph has at most n maximal cliques [BLS99, p. 195], this is possible in $O(n \cdot n \log n)$ time, where $O(n)$ time can be used to find the minimum vertex of a clique. From the sorted set $\mathcal{C}$, compute a set $\mathcal{T}$ of all tubes in $G-(S \cup X)$ in $O(n)$ time as follows: repeatedly find the first clique C in $\mathcal{C}$ that is not a junction and not yet part of a tube and add C and all succeeding cliques in $\mathcal{C}$ to a new tube T until a junction is encountered.

Now, as long as $|S| < |X|$, repeatedly find a hole H in $G-S$ and add at least one vertex of H to S as follows: because $G-S$ is an almost unit interval graph and $G-(S \cup X)$ is a unit interval graph, recursively branch into at most $12|X|$ possibilities to choose a vertex of H from a junction for inclusion in S (Lemma 5.1) and into at most $2|X|-1$ possibilities to choose tube $T \in \mathcal{T}$ for which an H-T-cut shall be included in S (Lemma 5.2, Lemma 5.3).

In each search tree node, we branch into at most $14|X|-1$ cases (at most five cases for Branching Rule 5.1 and at most $14|X|-1$ for a hole in $G-S$). In each case, at least one vertex is chosen for inclusion in S. As a result, the corresponding search tree has at most $(14|X|-1)^{|X|-1}$ nodes.

To analyze the running time for processing each node, it remains to analyze the running time for finding holes and minimum H-T-cuts. A hole in $G-S$ can be found in $O(|X|(n+m))$ time by breadth-first search starting at each vertex in X. A minimum H-T-cut is computable in $O(\sqrt{n}m)$ time [Sch03, Theorem 9.8]. □

We have seen in Section 5.2.1 that Theorem 5.4 implies Theorem 5.1. It follows that Unit Interval Vertex Deletion can be solved in $O((14k+14)^{k+1} \cdot kn^6)$ time.

5.4 Conclusion

We presented an algorithm for Unit Interval Vertex Deletion running in $O((14k+14)^{k+1} \cdot kn^6)$ time. At this point, we revive the question of Marx [Mar09] whether Interval Vertex Deletion is fixed-parameter tractable. As our algorithm heavily relies on bicompatible elimination orders, which generally do not exist for interval graphs, it is not straightforward to extend the algorithm to Interval Vertex Deletion.

Very likely, an implementation of the algorithm would perform much better than the analyzed worst-case running time: as long as holes with fewer than $14|X|-1$ vertices exist in the graph, the algorithm never actually branches into $14|X|-1$ cases. A prospectively performance-increasing implementation trick is (besides searching for small forbidden induced subgraphs first) trying to destroy holes by deleting inner vertices of tubes: in this case, Lemma 5.3 has shown that a large set of vertices can be deleted. Since the presented algorithm is easy to implement, running experiments to reveal the actual running time behavior of the algorithm is desirable.

It would be interesting to see if Chordal Vertex Deletion or Disjoint Unit Interval Vertex Deletion on almost unit interval graphs is NP-hard. We have made progress in this direction by showing Chordal Vertex Deletion to be NP-complete on {claw, net, tent}-free graphs.

An obvious bottleneck for the running time of our algorithm is the time needed to find a claw, net, or tent in a graph. Developing faster methods to find these forbidden induced subgraphs would significantly improve the polynomial part in the running time of the presented algorithm. Employing data reduction, like the following reduction to a *problem kernel*, would allow us to execute the search for claws, nets, and tents in smaller input graphs.

For a Unit Interval Vertex Deletion instance (G, k), a polynomial-time computable Unit Interval Vertex Deletion instance (G', k') with $k' \leq k$ and $|G'| \leq f(k)$ is a *problem kernel* if (G, k) is a yes-instance if and only if (G', k') is a yes-instance, where f is a computable function only depending on k. A problem kernel for Disjoint Unit Interval Vertex Deletion would likewise be interesting. Besides searching for problem kernels for Unit Interval Vertex Deletion, other data reduction approaches are desirable: to reduce the number of possible ways of destroying a hole, data reduction rules to reduce the number of maximal cliques in a unit interval graph could be exploited. Ideally, one would reduce the number of maximal cliques to some constant value.

It is open to study the parameterized complexity of Unit Interval Editing, which asks whether a graph can be transformed into a unit interval graph by adding or removing at most k edges.

6 Conclusion

For future research, it seems natural to further exploit and find more connections between seriation on erroneous data and algorithmic graph theory. In this work, we have exploited graph-theoretic connections to show that Partial Robinsonian Similarity and Robinson $(L_p, \mathbb{N})$-Fitting are NP-complete. We also obtained a fixed-parameter algorithm for Partial Robinsonian $\{0,1\}$-Similarity parameterized by the number of sought outliers, which is applicable to measuring indifference. The corresponding fixed-parameter algorithm for Unit Interval Vertex Deletion exploits many tools and results specific to the involved graph classes, like unit interval graphs and the more general claw-free graphs. Probably, the large pool of graph-theoretic results yields more insights into seriation problems.

As discussed in Section 4.1, it is not obvious how the fixed-parameter algorithm for Unit Interval Vertex Deletion can be used for the general Partial Robinsonian Similarity problem. An extensive analysis of the class of graphs that represent Robinsonian similarities (graphs whose vertices are the given objects and whose edges are weighted by the similarities between objects) could lead the way to fixed-parameter algorithms for Robinson (L_p, M)-Fitting and Partial Robinsonian Similarity. Also, graph-theoretic results may be obtained from the field of seriation. For example, an interesting side result can be drawn from the algorithm for Restricted Robinson $(L_1, \{0,1\})$-Fitting presented in Section 4.3.3, which also solves the following graph editing problem:

Bicompatibility Editing
Input: A graph $G = (V, E)$ and a total order $\preceq$ on V.
Output: A graph $G' = (V, E')$ for which $\preceq$ is a bicompatible elimination order such that $|E \Delta E'|$ is minimized.

Exploiting the connection between Robinsonian similarities and unit interval graphs, Algorithm 4.2 from Section 4.3.3 solves this problem in $O(n^2)$ time.

It seems that implementing the algorithms presented in this work is feasible. Especially, it would be interesting to compare the performance of the mixed integer program in Section 4.2.2 with the performance of other algorithms solving seriation problems on inaccurate data [HHB08]. Additionally, experiments could reveal the actual running time behavior of the presented Unit Interval Vertex Deletion algorithm, compared to its analyzed worst-case running time.

Bibliography

[ABH99] Jonathan E. Atkins, Erik G. Boman, and Bruce Hendrickson. A spectral algorithm for seriation and the consecutive ones problem. *SIAM J. Comput.*, 28(1):297–310, 1999. Cited on pages 9, 18, 25, and 26.

[BB01] Jean-Pierre Barthélemy and François Brucker. NP-hard approximation problems in overlapping clustering. *Journal of Classification*, 18(2):159–183, 2001. Cited on pages 18 and 24.

[BBD06] Pablo Burzyn, Flavia Bonomo, and Guillermo Durán. NP-completeness results for edge modification problems. *Discrete Applied Mathematics*, 154(13):1824–1844, 2006. Cited on pages 19 and 25.

[BKMN10] René van Bevern, Christian Komusiewicz, Hannes Moser, and Rolf Niedermeier. Measuring indifference: Unit interval vertex deletion. In *Proceedings of the 36th International Workshop on Graph Theoretic Concepts in Computer Science*, Zarós, Crete, Greece, June 2010. Cited on page 2.

[BL76] Kellogg S. Booth and George S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *Journal of Computer and System Sciences*, 13:335–379, 1976. Cited on page 19.

[BLS99] Andreas Brandstädt, Van Bang Le, and Jeremy P. Spinrad. *Graph classes: a survey*. SIAM, Philadelphia, PA, USA, 1999. Cited on pages 19, 42, and 62.

[Cai96] Leizhen Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters*, 58(4):171–176, 1996. Cited on page 42.

[CF97] Victor Chepoi and Bernard Fichet. Recognition of Robinsonian dissimilarities. *Journal of Classification*, 14(2):311–325, 1997. Cited on pages 8, 9, 18, 25, and 26.

[CFS09] Victor Chepoi, Bernard Fichet, and Morgan Seston. Seriation in the presence of errors: NP-hardness of l_∞-fitting Robinson structures to dissimilarity matrices. *Journal of Classification*, 26(3):279–296, 2009. Cited on pages 18, 21, 24, 29, 30, 32, 37, 39, and 41.

[Cha98] Maw-Shang Chang. Efficient algorithms for the domination problems on interval and circular-arc graphs. *SIAM Journal on Computing*, 27(6):1671–1694, 1998. Cited on page 19.

[CKN$^+$98] Derek G. Corneil, Hiryoung Kim, Sridhar Natarajan, Stephan Olariu, and Alan P. Sprague. Simple linear time recognition of unit interval graphs. *Information Processing Letters*, 55(2):99–104, 1998. Cited on page 19.

[CS09] Victor Chepoi and Morgan Seston. Seriation in the presence of errors: A factor 16 approximation algorithm for l_∞-fitting Robinson structures to distances. *Algorithmica*, 2009. Available electronically. Cited on page 18.

[DF99] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Springer, 1999. Cited on page 16.

[DFL$^+$07] Frank K. H. A. Dehne, Michael R. Fellows, Michael A. Langston, Frances A. Rosamond, and Kim Stevens. An $O(2^{O(k)}n^3)$ FPT algorithm for the undirected feedback vertex set problem. *Theory of Computing Systems*, 41(3):479–492, 2007. Cited on page 46.

[DGN10] Michael Dom, Jiong Guo, and Rolf Niedermeier. Approximation and fixed-parameter algorithms for consecutive ones submatrix problems. *Journal of Computer and System Sciences*, 2010. To appear. Cited on pages 13 and 19.

[Dom08] Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property*. PhD thesis, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Cited on pages 12, 13, and 19.

[Dom09] Michael Dom. Algorithmic aspects of the consecutive-ones property. *Bulletin of the European Association for Theoretical Computer Science*, 98:27–59, 2009. Cited on pages 12 and 19.

[FG06] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer, 2006. Cited on page 16.

[FGMN09] Michael R. Fellows, Jiong Guo, Hannes Moser, and Rolf Niedermeier. A complexity dichotomy for finding disjoint solutions of vertex deletion problems. In *Proceedings of the 34th International Symposium on Mathematical Foundations of Computer Science (MFCS '09)*, volume 5734 of *LNCS*, pages 319–330. Springer, 2009. Cited on page 46.

[Fou93] Jean. L. Fouquet. A strengthening of Ben Rebea's lemma. *Journal of Combinatorial Theory Series B*, 59(1):35–40, 1993. Cited on page 52.

[Gau82] Hugh G. Gauch. *Multivariate Analysis in Community Ecology*, volume 1 of *Cambridge Studies in Ecology*. Cambridge University Press, 1982. Cited on page 8.

[GGH$^+$06] Jiong Guo, Jens Gramm, Falk Hüffner, Rolf Niedermeier, and Sebastian Wernicke. Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. *Journal of Computer and System Sciences*, 72(8):1386–1396, 2006. Cited on page 46.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979. Cited on pages 14, 16, 19, 27, and 31.

[GJS76] Michael R. Garey, David S. Johnson, and Larry J. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976. Cited on page 44.

[GJY10] Dongdong Ge, Xiaoye Jiang, and Yinyu Ye. A note on complexity of L_p minimization. Manuscript, 2010. Cited on page 31.

[GLL82] U. I. Gupta, D. T. Lee, and J. Y.-T. Leung. Efficient algorithms for interval graphs and circular-arc graphs. *Networks*, 12(4):459–467, 1982. Cited on page 19.

[GMN09] Jiong Guo, Hannes Moser, and Rolf Niedermeier. Iterative compression for exactly solving NP-hard minimization problems. In *Algorithmics of Large and Complex Networks*, volume 5515 of *Lecture Notes in Computer Science*, pages 65–80. Springer, 2009. Cited on pages 45 and 46.

[HG02] MohammadTaghi HajiAghayi and Yashar Ganjali. A note on the consecutive ones submatrix problem. *Information Processing Letters*, 83(3):163–166, 2002. Cited on pages 12 and 19.

[HH05] Pavol Hell and Jing Huang. Certifying LexBFS recognition algorithms for proper interval graphs and proper interval bigraphs. *SIAM Journal on Discrete Mathematics*, 18(3):554–570, 2005. Cited on page 19.

[HHB08] Michael Hahsler, Kurt Hornik, and Christian Buchta. Getting things in order: An introduction to the R package seriation. *Journal of Statistical Software*, 25(3):1–34, March 2008. Cited on pages 8 and 64.

[Hsu02] Wen-Lian Hsu. A simple test for the consecutive ones property. *Journal of Algorithms*, 43:1–16, 2002. Cited on page 19.

[Hub74] Lawrence J. Hubert. Problems of seriation using a subject by item response matrix. *Psychological Bulletin*, 81:976–983, 1974. Cited on page 8.

[Kar84] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing (STOC '84)*, pages 302–311, New York, NY, USA, 1984. ACM. Cited on pages 27, 31, and 39.

[Ken69] David G. Kendall. Incidence matrices, interval graphs and seriation in archaeology. *Pacific Journal of Mathematics*, 28(3):565–570, 1969. Cited on page 7.

[Kha79] Leonid G. Khachiyan. A polynomial algorithm in linear programming. *Doklady Akademii Nauk SSSR*, 244:1093–1096, 1979. Cited on pages 27, 31, and 39.

[Luc56] R. Duncan Luce. Semiorders and a theory of utility discrimination. *Econometrica*, 24:178–191, 1956. Cited on pages 7 and 11.

[LY80] John M. Lewis and Mihalis Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *Journal of Computer and System Sciences*, 20(2):219–230, 1980. Cited on page 19.

[Mar09] Dániel Marx. Chordal deletion is fixed-parameter tractable. *Algorithmica*, 2009. Available electronically. Cited on pages 19, 42, 43, and 62.

[MR84] Boris G. Mirkin and Sergei N. Rodin. *Graphs and Genes*. Springer, 1984. Cited on pages 9, 18, 25, and 26.

[Nie06] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006. Cited on page 16.

[NN87] Yurii Nesterov and Arkadii Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*. SIAM, 1987. Cited on pages 30, 31, 32, and 33.

[Nök91] Klaus Nökel. *Temporally Distributed Symptoms in Technical Diagnosis*. Springer, 1991. Cited on page 8.

[PD03] B. S. Panda and Sajal K. Das. A linear time recognition algorithm for proper interval graphs. *Information Processing Letters*, 87(3):153–161, 2003. Cited on pages 19, 49, 52, and 61.

[Pet99] William M. F. Petrie. Sequences in prehistoric remains. *Journal of the Anthropological Institute*, 29:295–301, 1899. Cited on page 7.

[Rob51] W. S. Robinson. A method for chronologically ordering archaeological deposits. *American Antiquity*, 16:293–301, 1951. Cited on pages 7, 8, and 18.

[Rob69] Fred S. Roberts. Indifference graphs. In *Proof Techniques in Graph Theory*, pages 139–146. Academic Press, New York, 1969. Cited on pages 10, 11, 12, 13, 19, 23, and 52.

[Rob78] Fred S. Roberts. *Graph Theory and Its Applications to Problems of Society*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1978. Cited on pages 7, 10, 11, 12, 13, 19, 23, 43, 44, and 52.

[Rob79] Fred S. Roberts. Indifference and seriation. *Annals of the New York Academy of Sciences*, 328:173–182, 1979. Cited on pages 9 and 11.

[RSV04] Bruce Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Operations Research Letters*, 32(4):299–301, 2004. Cited on pages 45 and 46.

[Sch03] Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, volume A. Springer, 2003. Cited on page 62.

[Weg67] Gerd Wegner. *Eigenschaften der Nerven homologisch-einfacher Familien im $\mathbb{R}^n$*. PhD thesis, Universität Göttingen, Germany, 1967. Cited on pages 19, 42, 43, and 44.