EPFL

Thèse n° 10 125

Results on Sparse Integer Programming and Geometric Independent Sets

Présentée le 3 mars 2023

Faculté des sciences de base
Chaire d'optimisation discrète
Programme doctoral en mathématiques

pour l'obtention du grade de Docteur ès Sciences

par

Jana Tabea CSLOVJECSEK

Acceptée sur proposition du jury

Prof. C. Hongler, président du jury
Prof. F. Eisenbrand, directeur de thèse
Prof. A. Sebo, rapporteur
Dr K.-M. Klein, rapporteur
Prof. M. Kapralov, rapporteur

École
polytechnique
fédérale
de Lausanne

2023

Planning is stupid and you should just
make it up as you go along.
— North of the Border

To my family and friends.

Abstract

An integer linear program is a problem of the form $\max\{c^\intercal x\colon Ax = b, x \geq 0, x \in \mathbb{Z}^n\}$, where $A \in \mathbb{Z}^{n\times m}$, $b \in \mathbb{Z}^m$, and $c \in \mathbb{Z}^n$. Solving an integer linear program is NP-hard in general, but there are several assumptions for which it becomes fixed parameter tractable. One example are block-structured integer programs, which exhibits a (recursive) block structure: The problem decomposes into independent and efficiently solvable sub-problems, if a small number of rows or columns are deleted from the constraint matrix. Prominent examples are 2-stage stochastic integer programs, n-fold integer programs and their generalizations. Previously known algorithms for these problems were based on the augmentation framework, a variant of local search tailored to integer programming. We propose a different approach. We provide new proximity bounds, independent of the number of sub-problems, for both n-fold and 2-stage stochastic integer programming. Further, we show that the relaxation can be solved efficiently via an adaptation of a parametric search framework.

We apply our techniques to n-fold and 2-stage integer programming, and integer programming with bounded primal or dual treedepth. For these cases, we obtain strongly polynomial algorithms, which are near-linear in the dimension of the problem. Moreover, unlike the augmentation algorithms, our approach is highly parallelizable.

For the second part of this thesis, the focus is shifted to a different NP-hard problem, the maximum (weight) independent set problem. We consider intersection graphs of axis-parallel rectangles or segments, both in the weighted and unweighted case. Given a set of weighted axis-parallel rectangles or segments, the task is to find a subset of pairwise non-intersecting objects with the maximum possible total weight. For weighted rectangles, the best-known polynomial-time approximation algorithm achieves an approximation factor of $\mathcal{O}(\log\log(n))$. In the unweighted setting, constant factor approximation algorithms are known. It remains open if there are also constant factor approximation algorithms for the weighted setting.

We give a parameterized approximation algorithm for finding a maximum weight independent set of axis-parallel rectangles. Given a parameter $k \in \mathbb{N}$ and $\varepsilon > 0$, the algorithm finds a set of non-overlapping rectangles with weight at least $(1-\varepsilon)\operatorname{opt}_k$ in $2^{\mathcal{O}(k\log(k/\varepsilon))}n^{\mathcal{O}(1/\varepsilon)}$ time, where opt_k is the maximum weight of a solution of cardinality at most k. Note that, our algorithm may return a solution consisting of more than k rectangles. To complement this, we also propose a parameterized approximation algorithm for the case of axis-parallel segments. Here the algorithm finds a solution with cardinality at most k and total weight at least $(1-\varepsilon)\operatorname{opt}_k$ in time $2^{\mathcal{O}(k^2\log(k/\varepsilon))}n^{\mathcal{O}(1)}$.

Lastly, we provide a nearly tight bound on the independence number of axis-parallel segments. More precisely, we prove that for any triangle-free intersection graph of n axis-parallel segments in the plane, the independence number of this graph is at least $n/4 + \Omega(\sqrt{n})$. We

complement this with a construction of a graph in this class, with an independence number at most $n/4 + c\sqrt{n}$ for an absolute constant c.

keywords: Integer Programming, n-fold IP, 2-stage stochastic IP, treedepth, Graver basis, fixed parameter tractable, Independence Number, Maximum Weight Independent Set of Rectangles.

Zusammenfassung

Ein ganzzahliges lineares Programm ist ein Problem der Form $\max\{c^\intercal x\colon Ax = b, x \geq 0, x \in \mathbb{Z}^n\}$, wobei $A \in \mathbb{Z}^{n\times m}$, $b \in \mathbb{Z}^m$, und $c \in \mathbb{Z}^n$. Das Lösen eines generellen, ganzzahligen linearen Programms ist NP-schwer, aber es gibt Annahmen unter denen es parametrisierbar ist. Ein Beispiel sind blockstrukturierte ganzzahlige Programme, mit einer (rekursiven) Blockstruktur: Das Problem zerfällt in unabhängige und effizient lösbare Teilprobleme, wenn eine kleine Anzahl von Zeilen oder Spalten aus der Matrize gelöscht werden. Beispiele sind 2-stufig stochastische und n-fache ganzzahlige Programme, sowie ihre Verallgemeinerungen. Bisher bekannte Algorithmen für diese Probleme basierten auf dem Augmentations-Framework, einer Variante der lokalen Suche, spezialisiert auf ganzzahlige Programmierung. Wir bieten neue Näherungsschranken, unabhängig von der Anzahl der Teilprobleme, sowohl für n-fache als auch für 2-stufig stochastische ganzzahlige Programmierung. Wir zeigen auch, dass die Relaxierung durch das Anapassen eines parametrischen Suchrahmens effizient gelöst werden kann.

Wir wenden unsere Techniken auf n-fache und 2-stufig ganzzahlige Programmierung sowie auf ganzzahlige Programmierung mit begrenzter primärer oder dualer Baumtiefe an. Für diese Fälle erhalten wir stark polynomielle Algorithmen, die nahezu linear in der Dimension sind. Im Gegensatz zu den Augmentierungsalgorithmen ist unser Ansatz ausserdem parallelisierbar.

Für den zweiten Teil dieser Arbeit wird der Fokus auf ein anderes NP-hartes Problem verlagert, nämlich das Problem der (gewichteten) maximalen unabhängigen Menge. Wir betrachten Schnittgraphen von achsenparallelen Rechtecken oder Segmenten, sowohl gewichtet als auch ungewichtet. Gegeben eine Menge von gewichteten achsenparallelen Rechtecken oder Segmenten, besteht die Aufgabe darin, eine Teilmenge von sich paarweise nicht überschneidenden Objekten mit dem maximal möglichen Gesamtgewicht zu finden. Für gewichtete Rechtecke hat der beste bekannte Polynomialzeit-Approximationsalgorithmus einen Approximationsfaktor von $\mathcal{O}(\log\log(n))$. Für ungewichtete Rechtecke sind Algorithmen zur Annäherung mit konstantem Faktor bekannt. Es bleibt offen, ob es auch Algorithmen zur Approximation mit konstantem Faktor für den gewichteten Fall gibt.

Wir geben einen parametrisierten Approximationsalgorithmus für die Suche nach einer unabhängigen Menge von achsenparallelen Rechtecken mit maxiamlem Gewicht. Für Parameter $k \in \mathbb{N}$ und $\varepsilon > 0$, findet der Algorithmus eine Menge nicht überlappender Rechtecke mit einem Gewicht von mindestens $(1-\varepsilon)\,\mathrm{opt}_k$ in $2^{\mathcal{O}(k\log(k/\varepsilon))}n^{\mathcal{O}(1/\varepsilon)}$ Zeit, wobei opt_k das maximale Gewicht einer Lösung von höchstens k Rechtecken ist. Unser Algorithmus kann aber auch eine Lösung liefern, die aus mehr als k Rechtecken besteht. Wir präsentieren auch einen parametrisierten Approximationsalgorithmus für den Fall achsenparalleler Segmente. Hier findet der Algorithmus eine Lösung von höchstens k Segmenten und einem Gesamtgewicht

von mindestens $(1-\varepsilon)\,\mathrm{opt}_k$ in $2^{\mathcal{O}(k^2\log(k/\varepsilon))}n^{\mathcal{O}(1)}$ Zeit.
Schliesslich liefern wir eine nahezu enge Schranke für die Unabhängikeitszahl achsparalleler Segmente. Genauer gesagt beweisen wir, dass für jeden dreiecksfreien Schnittgraphen von n achsenparallelen Segmenten die Unabhängigkeitszahl mindestens $n/4+\Omega(\sqrt{n})$ beträgt. Wir ergänzen dies durch eine Konstruktion eines Graphen dieser Klasse mit einer Unabhängigkeitszahl von höchstens $n/4+c\sqrt{n}$, für eine absolute Konstante c.

Schlüsselwörter: ganzzahlige Programmierung, n-faches IP, 2-stufig stochastisches IP, Baumtiefe, Graverbasis, parametrisierter Algorithmus, Unabhängigkeitszahl, maximal gewichtete unabhängige Menge von Rechtecken.

Contents

Introduction

Discrete optimization considers problems in which discrete solutions are desirable. Some examples are the traveling salesman problem, scheduling, and many more. Most of these problems are considered *hard* to solve. Meaning, that there is no efficient algorithm solving the problem. In our context, efficiency is measured by the time needed to solve the problem as a function on the input size. An algorithm is considered to be efficient if it runs in polynomial time on the input size. On the other hand, a problem is considered NP-hard if it is unlikely that there exists such a polynomial time algorithm. Unlikely means that if there is a polynomial time algorithm for one NP-hard problem, then all of them can be solved in polynomial time, which is excluded by a broadly believed conjecture. This notion was introduced in 1972 by Karp [Kar72] where he presented 21 NP-hard problems.

There are two main approaches to NP-hard problems. In the field of parameterized complexity, the idea is to assume that certain *parameters* of the problem are small. This can lead to algorithms running in polynomial time on the input size, as long as the parameter is fixed. Ideally, for some parameter k of the problem, one seeks an algorithm with a running time of the form $f(k)\mathsf{poly}(n)$, where n is the instance size and f a function only dependent on the parameter k. Such an algorithm is also called *fixed parameter tractable*, parameterized by k. In the field of *approximation algorithms*, NP-hard optimization problems are attacked from a different angle. Here the goal is to design an efficient algorithm computing a solution which is provably only slightly worse than the optimal solution. More precisely, given a parameter c, an approximation algorithm returns a solution of cost at most c times the cost of the optimum solution, or, when the objective is to maximize a profit, of gain at least $1/c$ times the gain of the optimum solution. The parameter c is the *approximation factor* of the algorithm, a constant, which is possibly dependent on the instance size. Naturally, these two concepts can also be combined to try and design approximation algorithms running in parameterized time.

One of Karp's classical NP-hard problems is the *integer linear programming problem*, an instance of which is called an *integer linear program*. While there are several equivalent descriptions of an integer linear program, in this thesis they are mainly considered to be given in the form

$$\max\{c^\mathsf{T} x : \ Ax = b,\ x \geq 0,\ x \in \mathbb{Z}^m\}, \tag{♠}$$

where $A \in \mathbb{Z}^{n\times m}$ is the *constraint matrix*, $b \in \mathbb{Z}^n$ the *target vector*, and $c \in \mathbb{Z}^m$ the *objective function*.

While solving an integer linear program is NP-hard in general, there are some natural as-

sumptions on the constraint matrix A for which (♠) is solvable in polynomial time. Famous examples include *totally unimodular* and *bimodular integer programming* [HK10, AWZ17] or integer programs with a *constant number of variables* [LJ83, Kan87]. Another example are *block-structured* integer programs in which the constraint matrix exhibits a (recursive) block structure.

Intuitively, a matrix is *block structured* if, after removing a small number of rows or columns, it decomposes into many small independent blocks. Two famous examples with polynomial time algorithms are *2-stage stochastic* integer programming [HS03] and its transposed version, *n-fold* integer programming [LHOW08].

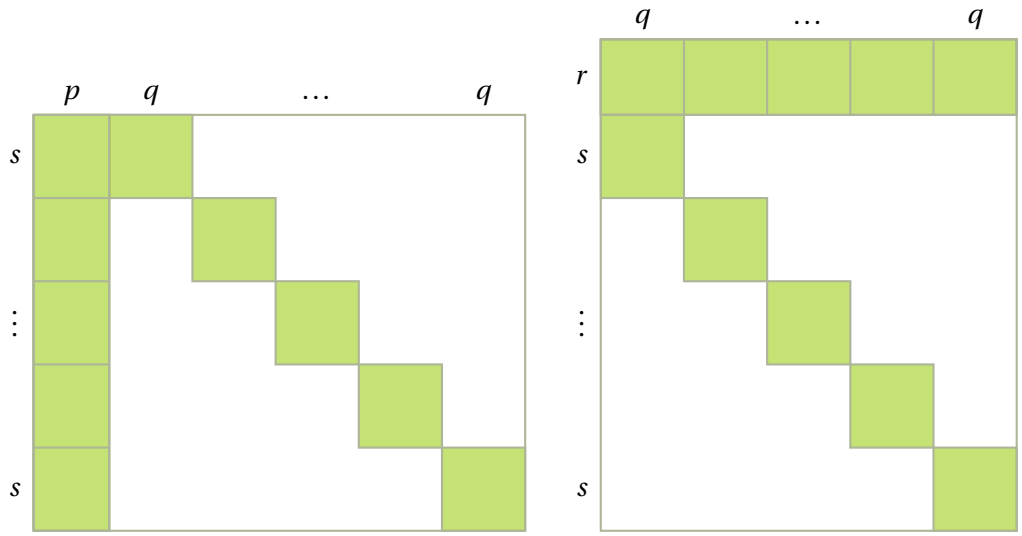

Figure 1: A schematic view of a 2-stage stochastic matrix (left panel) and an n-fold matrix (right panel). All non-zero entries are contained in the blocks depicted in green.

A *2-stage stochastic* matrix decomposes into blocks with at most q columns each, after deleting the first p columns (Figure 1, left panel). The terminology is borrowed from the field of stochastic integer optimization, a model for discrete optimization under uncertainty. Here, the first p variables, also called *global variables*, correspond to a decision made in the first stage, whereas the n blocks involving q variables each represent a usually large number of different scenarios arising in the second stage of stochastic optimization.

Similarly, an *n-fold* matrix decomposes into n blocks with at most s rows each, after deleting the first r rows. Looking at the corresponding integer linear program, the n blocks can be seen as "small" integer linear programs, each spanning over an independent subset of variables. The only constraints spanning over all variables and thus linking the small programs, are given by the first r rows, called *linking constraints*.

The two concepts can be generalized by allowing further recursive levels in the block structure, leading to a *multistage stochastic* matrix (Figure 2, left panel), respectively, a *tree-fold* matrix (Figure 2, right panel). In this generalization, the diagonal blocks are not necessary of small

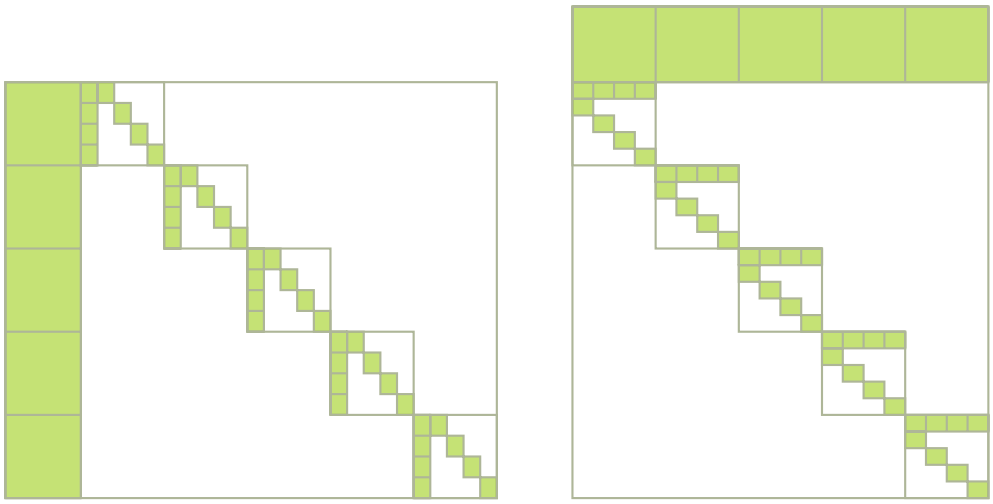

Figure 2: A schematic view of multistage stochastic matrix (left panel) and a tree-fold matrix (right panel). All non-zero entries are contained in the blocks depicted in green.

dimension, but we assume that they admit the same block structure again, this time with one recursion level fewer. One common way to explain this recursive structure is through the notion of the *primal*, respectively, *dual treedepth* of a matrix.

In the first part of this thesis, we focus on such block structured integer linear programs, providing some of the currently fastest algorithms for different types of block structures. Traditionally, algorithms for block structured integer linear programming were based on an augmentation framework. For this, the algorithm iteratively improves a solution, ultimately converging to the optimum solution. Another important technique used, also in the augmentation framework, are *proximity bounds*. This is a result stating that, for an optimal solution $x^\star$ of the relaxation, there is a optimal integral solution close by. Until recently, these proximity bounds were dependent on the number of blocks n. We improve on this, by providing a proximity bound independent of the number of blocks for both n-fold and 2-stage stochastic integer programming. This renders the augmentation framework obsolete and provides a different way to deal with block structured integer programming, leading to the currently fastest algorithms. Another benefit of this new approach is that it yields parallel algorithms, which was not possible with the augmentation framework since it is inherently sequential.

In the case of n-fold integer programming, the standard linear relaxation can not be used to obtain a proximity bound independent on n. In this case, we introduce a stronger relaxation by restricting the solution space of the n small linear programs to their integral hull. This relaxation proves strong enough for a proximity bound independent of n, while not removing any integral solution of the original linear program. Additionally to a proximity bound, we also provide an algorithm solving this stronger relaxation in near linear time. Our main results on block structured integer programming are:

(i) We provide a new proximity bound for n-fold integer programming. For an optimal vertex solution $x^\star$ of our strengthened relaxation, there exists an optimal integral solution $x^\diamond$ such that the distance $\|x^\star - x^\diamond\|_1$ is bounded by a function independent of the number of variables (and thus the number of blocks n). More precisely, the bound depends only on the maximum absolute value $\|A\|_\infty$ in the constraint matrix A, the number of linking constraints r and the structure of the n small blocks.

(ii) We show how to efficiently solve an n-fold linear program, even when considering the strengthened relaxation introduced above. Let T be the running time needed to solve any of the n small linear programs. We provide an algorithm solving the n-fold linear program in parallel on n processors, where each processor carries out $2^{\mathcal{O}(r^2)}(T\log(n))^{r+1}$ operations. This result can be further refined if the individual block problems can be efficiently solved in parallel as well.

(iii) Combining the above results, we obtain an algorithm for n-fold integer programming with a running time of

$$2^{\mathcal{O}(rs^2)}(rs\|A\|_\infty)^{\mathcal{O}(r^2s+s^2)}(nq)^{1+o(1)}.$$

This is the first parallel algorithm for n-fold integer programming.

(iv) In the case of 2-stage stochastic and multistage stochastic integer programming, we also provide a new proximity bound, this time for the standard linear relaxation. For each optimal solution $x^\star$ to the linear relaxation, there exists an optimal integral solution $x^\diamond$ such that $\|x^\star - x^\diamond\|_\infty$ is bounded by a function dependent on the maximum absolute value $\|A\|_\infty$ in the constraint matrix A, and the primal treedepth of A.

(v) Using this proximity bound, we give an algorithm for 2-stochastic integer programming with a running time of

$$2^{((p+q)\|A\|_\infty)^{\mathcal{O}(p(p+q))}} \cdot t^{1+o(1)},$$

where t is the number of variables.

(vi) Using a recursive application of our results, we obtain fixed parameter tractable algorithms for integer linear programming, parameterized by the primal treedepth, or respectively, parameterized by the dual treedepth. Both these algorithms are near linear in the number of variables and run in parallel.

While most of our results are still state of the art, Klein and Reuter further investigated this new approach for primal treedepth. They since found an algorithm for integer linear programming parameterized by the primal treedepth, with an improved bound on the parameter complexity [KR22].

For the second part of this thesis, the focus is shifted to a different famous NP-hard problem, namely the *Maximum Independent Set Problem*. Most commonly, this problem is defined

on graphs and given as follows. For a graph $G = (V, E)$ with vertex set V and edge set E, an *independent set* is a subset of its vertices which are pairwise non-adjacent. Since a single vertex is always an independent set, finding any independent set becomes trivial. However, the goal is to find a large independent set. That is, a maximum size independent set, or alternatively an independent set containing at least k vertices for some given constant k.

This problem is well studied over different classes of graphs. While it becomes polynomial time solvable for certain classes like interval graphs or bipartite graphs, it remains NP-hard for many other classes like planar graphs or intersection graphs. The *Maximum Weight Independent Set Problem* is an important generalization, in which a weight function $\omega\colon V \to \mathbb{R}$ on the vertices is provided, and the goal is to find an independent set of maximum weight. The classical unweighted version corresponds to assigning weight 1 to all vertices.

We focus on the class of geometric intersection graphs, in which the vertices are geometric objects in the plane and two vertices are adjacent if they intersect. In this setting, an independent set is a set of disjoint objects. More precisely, we study this class restricted to axis-parallel rectangles, or respectively, axis-parallel segments, both in the weighted and unweighted setting.

Finding a maximum independent set is NP-hard both for axis-parallel rectangles [FPT82] and axis-parallel segments [KN90]. From the parameterized perspective, finding a maximum independent set of axis-parallel rectangles is W[1]-hard when parameterized by k, the number of rectangles in the solution. This still holds in the unweighted setting and when all rectangles are squares [Mar05]. Therefore, it is unlikely that there is an exact algorithm with a running time of the form $f(k)n^{\mathcal{O}(1)}$, even in this restricted setting. However, the setting allows for efficient approximation algorithms. For finding a maximum weight independent set of axis-parallel rectangles, there exist a QPTAS [AW13], and the currently best approximation factor with a polynomial time algorithm is $\mathcal{O}(\log\log(n))$ [CW21]. It remains an important open question if there is a polynomial time algorithm with a constant approximation factor. In the unweighted case however, there exist constant factor approximation algorithms for finding a maximum independent set of axis-parallel rectangles [Mit21, GKM$^+$22].

In the following, we outline one useful technique to find approximation algorithms for independent sets of axis-parallel rectangles, which is also adapted in this thesis. The approach is based on finding an appropriate grid, which is then used to locally restrict the possible intersections. More precisely, the aim is a fixed size grid (parameterized by the solution size) such that each rectangle of the optimum solution contains a grid point. In case this is not necessarily possible, an approximation approach to this might be chosen. Then, each rectangle in the optimal solution can be characterized by the grid points in its interior. Using the bounded grid size, this allows to guess the positioning of the optimal solution, but without identifying the exact rectangles. Since the rectangles are axis-parallel, knowing their position on the grid is heavily restricting the possible intersections. This often results in a more manageable problem, which can either be solved or at least approximated.

We apply this technique to the maximum weight independent set problem of both axis-parallel rectangles and axis-parallel segments. In doing so, we obtain the following results.

(vii) Given a set of n weighted, axis-parallel rectangles, a parameter k, and $\varepsilon > 0$. Let opt_k be the maximum weight of an independent set of size k. We provide an algorithm computing an independent set with weight at least $(1-\varepsilon)\,\mathrm{opt}_k$ in time

$$2^{\mathcal{O}(k\log(k/\varepsilon))}n^{\mathcal{O}(1/\varepsilon)}.$$

(viii) In the setting of axis-parallel segments, we give an approximation algorithm, which also respects the cardinality of the optimal solution. Given a set of n weighted, axis-parallel segments, a parameter k, $\varepsilon > 0$ and opt_k the maximum weight of an independent set of size at most k. We provide an algorithm computing an independent set of cardinality at most k and with weight at least $(1-\varepsilon)\,\mathrm{opt}_k$ in time

$$2^{\mathcal{O}(k^2\log(k/\varepsilon))}n^{\mathcal{O}(1)}.$$

Another approach to independent sets is to study structural results. Given a class of graphs, we are interested in bounding the independence number, which is the maximum size of an independent set, for any graph in the class. The size of an independent set is heavily influenced by the instance size and by cliques, a set of pairwise adjacent vertices. For this reason, the independence number is often bounded by a function of the instance size and the clique covering number. The clique covering number is the minimal amount of cliques needed to cover all vertices of the graph. Alternatively, the class of graphs can be restricted in order to avoid handling the clique covering number. An example for such a restriction are triangle-free graphs, for which the maximum clique size is bounded by 2.

In this line of thinking, we provide a nearly tight bound on the independence number of axis-parallel segments, relying on a classical result of Erdős Szekeres [ES35].

(ix) Let G be a triangle-free intersection graph of n axis-parallel segments. Then the independence number of G is at least $\frac{n}{4} + c_1\sqrt{n}$, for some absolute constant c_1.

(x) On the other hand, for any $n \in \mathbb{N}$, we give a set of n axis-parallel segments with a triangle-free intersection graph G, such that the independence number of G is at most $\frac{n}{4} + c_2\sqrt{n}$, for some absolute constant c_2.

Summarizing Overview and Sources

Basic notations, results and concepts used throughout this thesis are presented in Chapter 1. This is not meant to be an introduction to the area, and a certain background on the topics is assumed.

In Part I, we focus on block structured integer linear programs, deriving algorithms for different types of block structures. We propose a different approach to previously known algorithms, which were based on the augmentation framework.

In Chapter 2, we consider n-fold and tree-fold integer programming. We use a strengthened relaxation and derive an algorithm, which relies on parametric search and a new proximity bound. This chapter is based on the results in the article [CEH+21].

In Chapter 3, we then consider 2-stage stochastic and multistage stochastic integer programming. We derive a new proximity bound and use it to derive a new algorithm. This chapter is based on the results in the article [CEP+21].

In Part II, we focus on independent sets of geometric intersection graphs.

In Chapter 4, we derive approximation algorithms for the maximum weight independent set problem for intersection graphs of axis-parallel rectangles, or respectively, axis-parallel segments. This chapter is based on the results in the arXiv preprint [CPW22].

In Chapter 5, we study the independence number of intersection graphs of axis-parallel segments. We derive a nearly tight bound on the independence number. This chapter is based on the results in the arXiv preprint [CCPW22].

1 Preliminaries

This chapter familiarizes the reader with basic notations, results and concepts used throughout this thesis.

1.1 Running Times, Parameters and Algorithms

A real RAM model of computation is used to compute the running time of all algorithms in this thesis. Meaning that each memory cell stores a real number of arbitrary bitlength and precision; and arithmetic operations (including rounding) are assumed to be of unit cost. Here, the *encoding length* of an instance includes the bitlength of each number stored. The *size* of an instance on the other hand is the number of memory cells used to store the instance. For the running time analysis of the parallel algorithms in this thesis, a PRAM model is used.

If an algorithm is independent of the encoding length and polynomial in the size of an instance, we say that the algorithm is *strongly polynomial*. Note that this notion only makes sense in a model where input numbers occupy single memory cells on which unit-cost arithmetic operations are allowed. This also implies that the running time of a strongly polynomial algorithm is not allowed to depend on the bitlength of the input numbers.

1.1.1 Fixed Parameter Tractability

Often, there are structural or other parameters of a problem which can be useful in the design of efficient algorithms. In other words, assuming certain parameters to be bounded, the problem admits a polynomial time algorithm. Such algorithms are called *fixed parameter tractable* and can be understood in two ways.

Weak fixed parameter tractable algorithms have running time of the form

$$f(p_1,\dots,p_k)\cdot|I|^{\mathcal{O}(1)},$$

where f is a computable function, $p_1,\dots,p_k$ the parameters and $|I|$ the total encoding length of the input I. For *strong fixed parameter tractable algorithms* we require a time complexity of

the form

$$f(p_1,\dots,p_k)\cdot n^{\mathcal{O}(1)},$$

where f is a computable function, $p_1,\dots,p_k$ the parameters, and n instance size.

1.1.2 Approximation Algorithms

An approximation algorithm returns a solution which is only slightly worse than the optimal solution. Given a parameter $\varepsilon > 0$, an approximation algorithm returns a solution of cost at most $(1+\varepsilon)$ times the cost of the optimum solution, or, when the objective is to maximize a profit, of gain at least $(1-\varepsilon)$ times the gain of the optimum solution. If the running time is of the order $n^{g(\varepsilon)}$, where n is the instance size and $g(\varepsilon)$ a function only depending on ε, such an algorithm is called a *Polynomial Time Approximation Scheme* (PTAS). If the algorithm runs in time $n^{g(\varepsilon)\mathsf{poly}(\log(n))}$ it is also called a *Quasi Polynomial Time Approximation Scheme* (QPTAS).

1.1.3 Linear Algorithms

The concept of accessing numbers in the input of an algorithm by *linear queries* only is a common feature of many algorithms and crucial in Sections 2.2 and 3.5. Let $\lambda_1,\dots,\lambda_k$ be numbers in the input of an algorithm. The algorithm is *linear* in this part of the input if it does not query the value of these numbers, but queries linear comparisons instead. This means that the algorithm generates numbers $a_1,\dots,a_k \in \mathbb{R}$ and $\beta \in \mathbb{R}$ and queries whether

$$a_1\lambda_1 + \cdots + a_k\lambda_k \le \beta \tag{1.1}$$

holds. Such a query is to be understood as a call to an oracle not implemented in the algorithm itself.

Many sorting algorithms such as *quicksort* or *merge sort* are linear in the input numbers $\lambda_1,\dots,\lambda_k$ to be sorted. Another example is the *simplex algorithm* for a linear program of the form $\max\{c^\intercal x\colon Ax \le b\}$ which is linear in b and c. In fact, the only point where b and c are used is to test the feasibility and optimality of a basis $B \subseteq \{1,\dots,m\}$. To see that testing this only needs linear queries on b and c, assume that $A \in \mathbb{R}^{n\times m}$ is of full column rank and recall that a basis is a set of m indices corresponding to linearly independent rows of A. The basis is feasible if $A(A_B^{-1}b_B) \le b$ holds. These are n linear queries involving components of b. Similarly, B is an optimal basis if $c^T A_B^{-1} \ge 0$ holds. This can be checked with m linear queries involving components of c.

We use the following notation. If an algorithm is linear in some part of the numbers in its input λ, this part is denoted with a bar, i.e. by $\overline{\lambda}$. In this case we say that $\overline{\lambda}$ is given *symbolically*, implying that it can only be used through queries. In the case of the simplex algorithm, we can write that it solves a linear program of the form $\max\{\overline{c}^\intercal x\colon Ax \le \overline{b}\}$ to indicate which input numbers are given symbolically, i.e. only accessible via linear queries.

1.2 (Integer) Linear Programming

Consider an instance of the linear programming problem given as follows:

$$\begin{aligned} \max\ & c^\intercal x \\ & Ax = b \\ & x \geq 0 \end{aligned} \tag{♠}$$

where $A \in \mathbb{Z}^{n\times m}$ is the *constraint matrix*, $b \in \mathbb{Z}^n$ the *target vector* and $c \in \mathbb{Z}^m$ the *objective function*. Such an instance of the linear programming problem is called a *linear program* (LP). Formally, define a linear program in the form (♠) as the 4-tuple $P = (x, A, b, c)$. In some cases additionally an *upper bound* $u \in \mathbb{Z}_{\geq 0} \cup \{\infty\}$ is used to describe the linear program. This allows a more concise notion using less variables and constraints for the same program. In the case where (♠) models an instance of the integer linear programming problem, an integrality constraint $x \in \mathbb{Z}^m$ is added. Such an instance is called an *integer linear program*.

Note that there are several equivalent forms to describe an (integer) linear program. While the above described is the main form considered during this thesis, other forms be used to simplify certain arguments. The notations presented below are extended to these forms in the obvious ways.

For a linear program $P = (x, A, b, c)$ in the form (♠), denote by $\mathrm{Sol}^{\mathbb{R}}(P)$ and $\mathrm{Sol}^{\mathbb{Z}}(P)$ the sets of fractional and integral solutions to P, respectively. That is, $\mathrm{Sol}^{\mathbb{R}}(P)$ is the polytope consisting of $x \in \mathbb{R}^n_{\geq 0}$ satisfying $Ax = b$, while $\mathrm{Sol}^{\mathbb{Z}}(P)$ comprises all integer points in $\mathrm{Sol}^{\mathbb{R}}(P)$. Further, define $\mathrm{opt}^{\mathbb{R}}(P)$ and $\mathrm{opt}^{\mathbb{Z}}(P)$ as the optimum values of any fractional or, respectively, any integral solutions to P. That is,

$$\mathrm{opt}^{\mathbb{R}}(P) = \inf\{\, c^\intercal x :\ x \in \mathrm{Sol}^{\mathbb{R}}(P)\,\} \qquad \text{and} \qquad \mathrm{opt}^{\mathbb{Z}}(P) = \inf\{\, c^\intercal x :\ x \in \mathrm{Sol}^{\mathbb{Z}}(P)\,\}.$$

A vector $x \in \mathrm{Sol}^{\mathbb{R}}(P)$ is an *optimal fractional solution* to P if $c^\intercal x = \mathrm{opt}^{\mathbb{R}}(P)$. *Optimal integral solutions* are defined analogously.

Given two (integral) solutions $x, y \in \mathrm{Sol}^{\mathbb{R}}(P)$ respectively $\mathrm{Sol}^{\mathbb{Z}}(P)$, their difference $x - y$ is an (integral) solution to $P' = (x, A, 0, c)$, the linear program obtained by setting the target vector to 0. This solution set is called the *kernel* of A and denoted by $\ker^{\mathbb{R}}(A)$. The *integer kernel* of A, denoted $\ker^{\mathbb{Z}}(A)$, consists of all integer vectors in $\ker^{\mathbb{R}}(A)$. Note that the kernel is only dependent on the matrix and consequently also used outside of linear programs.

1.2.1 Block Structured Matrices

A matrix A is *block decomposable*, if it can presented as

$$A = \begin{pmatrix} A_1 & \\ & A_2 \end{pmatrix},$$

where all non-zero entries are contained in the blocks A_1 and A_2. The *block decomposition* of a matrix A is its unique presentation as

$$A = \begin{pmatrix} D_1 & & & \\ & D_2 & & \\ & & \ddots & \\ & & & D_n \end{pmatrix},$$

where all non-zero entries are contained in the blocks $D_1, \ldots, D_n$ which are not block decomposable themselves.

For non-negative integers p and q, a matrix A is 2-stage stochastic or (p,q)-*stochastic* if, after deleting the first p columns, the matrix admits a block decomposition such that each block has at most q columns. In other words, a (p,q)-stochastic matrix can be written as

$$A = \begin{pmatrix} C_1 & D_1 & & & \\ C_2 & & D_2 & & \\ \vdots & & & \ddots & \\ C_n & & & & D_n \end{pmatrix}, \qquad (\diamondsuit)$$

where the blocks $C_1, \ldots, C_n$ have p columns and the blocks $D_1, \ldots, D_n$ have at most q columns each. In general, a presentation of matrix A as in $(\diamondsuit)$ shall be called a *stochastic decomposition* of A.

Similarly, for two non-negative integers r and s, a matrix A is an n-fold if $A^\intercal$ is (r,s)-stochastic. Meaning that an n-fold matrix can be written as

$$A = \begin{pmatrix} B_1 & B_2 & \ldots & B_n \\ D_1 & & & \\ & D_2 & & \\ & & \ddots & \\ & & & D_n \end{pmatrix}, \qquad (1.2)$$

where the blocks $B_1, \ldots, B_n$ have r rows and the blocks $D_1, \ldots, D_n$ have at most s rows each. Note that for an n-fold matrix A, the linear program $P = (x, A, b, c)$ decomposes into n independent linear programs P_i linked only by the first r constraints. This viewpoint also provides a natural decomposition of the vectors x, b and c into small bricks. More precisely, partition x into the bricks $x_1, \ldots, x_n$, so that x_i corresponds to the columns of the matrix D_i, for each $i \in \{1, \ldots, n\}$. Partition c into the bricks $c_0, c_1, \ldots, c_n$ in the same fashion, and partition b into the bricks $b_0, b_1, \ldots, b_n$ so that b_0 corresponds to the rows of matrices B_i, while b_i corresponds to the rows of the matrix D_i, for $i = 1, \ldots, n$ Then, the linear program P_i is given by $P_i = (x_i, D_i, b_i, c_i)$. Observe that, in a solution x of P, each brick x_i is in the polyhedron

$\mathrm{Sol}^{\mathbb{R}}(P_i)$, given by the solution set of P_i. This description can be generalized to

$$\begin{aligned} \max\ & c_1^\intercal x_1 + \cdots + c_n^\intercal x_n \\ & B_1 x_1 + \cdots + B_n x_n = b_0 \\ & x_i \in Q_i \quad i = 1, \ldots, n, \end{aligned} \tag{★}$$

where the $Q_i \subseteq \mathbb{R}^{q_i}_{\geq 0}$ are polyhedra. For a n-fold matrix, the Q_i correspond to the polyhedra $\mathrm{Sol}^{\mathbb{R}}(P_i)$. In the general structure however, they could also be described differently. A linear program P described as in (★) is referred to as linear program with an *n-fold structure*.

These two types of block structured matrices are often generalized by allowing further recursive levels in the block structure leading to *multistage stochastic* and *tree-fold* matrices. This recursive structure can be explained through the *primal treedepth*, or respectively, the *dual treedepth* of a matrix. As the name already suggests, the two notions are in a certain sense dual to each other.

There are multiple equivalent ways do define the primal and dual treedepth of a matrix. For us, it will be most convenient to rely on a recursive approach. First, we recursively define the *primal depth* of A, denoted $\mathsf{depth}_{\mathsf{P}}(A)$:

- if A has no columns, then its primal depth is 0;
- if A is block-decomposable, then its primal depth is equal to the maximum among the primal depths of the blocks in its block decomposition;
- if A has at least one column and can not be decomposed into blocks, then the primal depth of A is one larger than the depth of the matrix obtained from A by removing its first column.

Observe that, by a straightforward induction, in a matrix of primal depth d, every row contains at most d non-zero entries. The *primal treedepth* of A, denoted $\mathsf{td}_{\mathsf{P}}(A)$, is the smallest integer d such that the rows and columns of A can be permuted into a matrix with primal depth d.

Define the *dual depth* of A as $\mathsf{depth}_{\mathsf{D}}(A) = \mathsf{depth}_{\mathsf{P}}(A^\intercal)$, and define the *dual treedepth* of A as $\mathsf{td}_{\mathsf{D}}(A) = \mathsf{td}_{\mathsf{P}}(A^\intercal)$. Note that, exchanging rows and columns in the definition of the primal depth of A, gives a characterization of the dual depth of A. In particular, the dual treedepth of A is the smallest integer d such that the rows and columns of A can be permuted into a matrix with dual depth d.

For the sake of future application, we now observe that the block decomposition can be computed efficiently. More precisely, we have the following lemma.

Lemma 1.1. *For a matrix A with t rows and at most δ non-zero entries in each row the block decomposition of A can be computed in time $\mathcal{O}(\delta \log(\delta t))$ on t processors.*

To see this, assume that A is given as a list of its non-zero entries specifying the row and column index as well as the coefficient. Since A has at most δ non-zero entries in each row,

this list has a length of at most δt. After sorting the list, it can be split into blocks by comparing consecutive entries in parallel.

Observe that using this result, checking whether the input matrix or its transposed is (p,q)-stochastic can be done in time $\mathcal{O}((p+q)\log((p+q)t))$ on t processors. Similarly, a recursive application on the blocks can be used to verify whether the matrix has primal or dual depth at most d in time $\mathcal{O}(d^2\log(dt))$ on t processors. Note that in the definition of primal and dual treedepth, we allow the matrix to take the specified form after applying a permutation of the rows and columns.

This permutation can be computed in linear fixed parameter tractable time without further assumptions. We discuss this for the primal treedepth and (r,s)-stochastic matrices, since working with the transposed matrix gives the same results for the dual treedepth and n-fold matrices. The following definition will be useful for the computation: the *primal graph* of a matrix A is the graph whose vertex set consists of the columns of A and two columns are considered adjacent if they contain non-zero entries in the same row. As observed in previous works (see e.g. [KLO18, EHK+19]), the primal treedepth of A coincides with the (graph-theoretic) treedepth of its primal graph of A. For this graph problem and some fixed parameter d, we can use an algorithm due to Reidl et al. [RRVS14] which either finds an *elimination forest* of depth at most d, or asserts that no such permutation exists. From the given elimination forest, one can then recover a column permutation of A with depth at most d. This algorithm takes time $2^{\mathcal{O}(d^2)}\cdot(dt)$, where dt is an upper bound on the number of columns of A.

In the case of looking for a column permutation of A such that the resulting matrix is (p,q)-stochastic, the problem boils down to the following: Given the primal graph of A, verify whether there exists a set of r vertices such that after removing them, every connected component of the resulting graph has at most s vertices. This problem has been studied under the name *Component Order Connectivity* by Drange et al. [DDvtH16], who gave an algorithm with running time $2^{\mathcal{O}(p\log q)}\cdot dt$.

Note that both of these algorithms are sequential. Therefore, all algorithms given in this thesis assume that the matrix is already suitably organized, that is in one of the forms specified above.

1.2.2 Graver Basis

The *conformal (partial) order* $\sqsubseteq$ on $\mathbb{R}^n$ is the relation defined such that, for two vectors $x, y \in \mathbb{R}^n$, the relation $x \sqsubseteq y$ holds if the following conditions are satisfied for each $i \in \{1,\dots,n\}$:

- $|x_i| \leq |y_i|$
- $x_i y_i \geq 0$

where x_i and y_i are the ith entries of x and y, respectively. The first condition implies that

x is component wise smaller than y. The second condition ensures that x and y are sign-compatible, meaning that they lie in the same orthant.

The *Graver Basis* of a matrix A, denoted by $\mathcal{G}(A)$, is the set of $\sqsubseteq$-minimal vectors in the integer kernel $\ker^{\mathbb{Z}}(A)$ of A. These minimal elements are also called *indecomposable* in $\ker^{\mathbb{Z}}(A)$, a notion coming from the fact that they can not be written as the sum of two sign-compatible non-zero elements of $\ker^{\mathbb{Z}}(A)$. On one hand, a $\sqsubseteq$-minimal vector in $\ker^{\mathbb{Z}}(A)$ is clearly indecomposable. For the other direction, note that for $x \sqsubseteq y \in \ker^{\mathbb{Z}}(A)$ it holds that $y - x \sqsubseteq y$ which implies that y is decomposable.

It follows from Dickson's Lemma, see [Dic13, HS03], that $\mathcal{G}(A)$ is always finite. However we are interested in more precise bounds on the lengths of vectors in $\mathcal{G}(A)$ depending on different norms. For $p \in [1,\infty]$, define the ℓ_p *Graver measure* of A as

$$g_p(A) := \max_{v \in \mathcal{G}(A)} \|v\|_p.$$

In the case of block structured matrices, bounds on the ℓ_p Graver measure are usually obtained by using the Steinitz Lemma, which holds for arbitrary norms.

Theorem 1.2 (Steinitz [Ste13], Grinberg and Sevast'yanov [GS80])**.**
Let $x_1, \dots, x_n \in \mathbb{R}^r$ such that

$$\sum_{i=1}^{n} x_i = 0 \quad \text{and} \quad \|x_i\| \leq 1 \text{ for each } i.$$

There exists a permutation $\pi \in S_n$ such that all partial sums satisfy

$$\|\sum_{j=1}^{k} x_{\pi(j)}\| \leq r \text{ for all } k = 1, \dots, n.$$

The Steinitz Lemma proved itself not only useful in bounding the ℓ_p Graver measure of block-structured matrices, but proving proximity results, more direct applications also turned out useful.

Next, some important results bounding the ℓ_p Graver measure of different types of integer matrices are presented.

Theorem 1.3 (Eisenbrand et al. [EHK18])**.** *For every integer matrix A with n rows,*

$$g_\infty(A) \leq g_1(A) \leq (2n\|A\|_\infty + 1)^n.$$

Note that the Graver basis of a matrix A is completely described by $\ker(A)$, so without changing $\mathcal{G}(A)$ the rows of A can be restricted to a maximal linearly independent subset. Then the number of columns m is not smaller than the number of rows n. Hence, the following bound

independent of the number of rows in A can be derived.

Corollary 1.4. *For every integer matrix A with m columns,*

$$g_\infty(A) \leq g_1(A) \leq (2m\|A\|_\infty + 1)^m.$$

In the case of matrices with bounded primal treedepth, the following result was known at the time the results in Chapter 3 were derived.

Theorem 1.5 (Eisenbrand et al. [EHK+19])**.** *There is a computable function $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that for every integer matrix A,*

$$g_\infty(A) \leq f(\mathsf{td}_\mathsf{P}(A), \|A\|_\infty).$$

The proof of Theorem 1.5 given by Eisenbrand et al. [EHK+19] shows that, roughly speaking, $f(\mathsf{td}_\mathsf{P}(A), \|A\|_\infty)$ is a $\mathsf{td}_\mathsf{P}(A)$-fold exponential function of $\|A\|_\infty$. Recently, this result was substantially improved by Klein and Reuter to a triple exponential bound in $\mathsf{td}_\mathsf{P}(A)$ [KR22]. More precisely, for $d = \mathsf{td}_\mathsf{P}(A)$, they show that

$$g_\infty(A) \leq 2^{(d\|A\|_\infty)^{\mathcal{O}(d^{3d+1})}}.$$

On the other hand, bounding the Graver measure as a function of the dual treedepth yields a better parameter dependence.

Theorem 1.6 (Knop [KPW20])**.** *For any integer matrix A, one has*

$$g_1(A) \leq (2\|A\|_\infty + 1)^{2^{\mathsf{td}_\mathsf{D}(A)}-1}.$$

1.3 Results in Graph Theory

A graph $G = (V, E)$ is given by its *vertex set* V and its *edge set* $E \subseteq V \times V$. A *tree decomposition* of a graph G is a tree T together with a function bag: $V(T) \to 2^{V(G)}$ mapping the nodes of T to subsets of vertices of G, called *bags*. Further, the following conditions must be satisfied:

- for every vertex u of G, the nodes of T whose bags contain u must form a connected, nonempty subtree of T, and
- for every edge uv of G, there must exist a node of T whose bag contains both u and v.

The *width* of a tree decomposition (T, bag) is defined as

$$\max_{x \in V(T)} |\text{bag}(x)| - 1.$$

The *treewidth* of G is the minimum width of a tree decomposition of G.

1.3.1 Graph Classes

An important class of graphs are *geometric intersection graphs*. Given a family of geometric objects $\mathcal{S}$ in the plane, the *intersection graph* of $\mathcal{S}$, denoted $G(\mathcal{S})$, is the graph with vertex set $\mathcal{S}$ and where two objects are adjacent if and only if they intersect. The class of geometric intersection graphs can be further specified by restricting to a certain type of geometric objects like discs, rectangles or even axis-parallel rectangles. This thesis focuses on axis-parallel rectangles and axis-parallel segments.

Another class of graphs we consider are graphs induced by *Constraint Satisfaction Problems*. In the *Constraint Satisfaction Problem* (CSP) one seeks to assign labels to a set of variables such that they fulfill given constraints. For an *Arity*-2 *Valued Constraint Satisfaction Problem* (2-VCSP) an additional cost function is given for each of the constraints and each constraint is restricted to 2 variables. The goal is to assign the labels in order to maximize the cost function. More precisely, an instance of the 2-VCSP is given by

- a finite set of *variables* X;
- a finite domain D_x for each variable $x \in X$. This is the set of possible labels for x;
- a finite set of constraints C. Each constraint $c \in C$ consists of an ordered pair of variables $x_c = (x, y) \in X \times X$ and a cost function $f_c \colon D_x \times D_y \to \mathbb{R}$. We assume that f_c is given as the set of all possible triples $(d_x, d_y, f_c(x, y)) \in D_x \times D_y \times \mathbb{R}$.

The goal is to compute the maximum value of the function

$$f(u) \coloneqq \sum_{c \in C} f_c(u|_{x_c}),$$

over all valid label assignments $u \in \prod_{x \in X} D_x$ to the variables of X. The value $f(u)$ is called the *revenue* of the label assignment u.

Each instance of 2-VCSP induces an undirected graph, called the *Gaifman graph*: the vertex set is the set of variables X, and for every pair of distinct variables $x, y \in X$, there is an edge xy if and only if there is a constraint $c \in C$ such that $x_c = (x, y)$. Given a class $\mathcal{H}$ of graphs, a restriction of 2-VCSP to $\mathcal{H}$ can be defined by focusing only on instances whose Gaifman graph is in $\mathcal{H}$. In this thesis, we focus only on instances of 2-VCSP where the Gaifman graph has bounded treewidth. In this setting, it is well-known that standard dynamic programming solves 2-VCSP efficiently [Fre90].

Theorem 1.7 (Freuder [Fre90])**.** *2-VCSP can be solved in time $\Delta^{\mathcal{O}(t)} \cdot |X|^{\mathcal{O}(1)}$ when the Gaifman graph has treewidth at most t and all domains are of size at most Δ.*

Freuder [Fre90] actually only considered the unweighted 2-CSP, however, as pointed out in e.g., [CRZ20, RWZ21], this dynamic-programming approach can be adapted to the weighted setting. Also, Freuder assumes that a suitable tree decomposition is given on input. Such a tree decomposition can be provided within the stated time complexity by, for instance, the 4-approximation algorithm of Robertson and Seymour [RS95].

In Section 4.3 we also use standard 2-CSPs. These can be modeled by 2-VCSPs where all the constraints are hard: revenue functions f_c only assign value 0 if the constraint is satisfied, or $-\infty$ if the constraint is not satisfied. The task is to find a variable assignment that satisfies all constraints, that is, yields revenue 0.

1.3.2 Grids

An important property of axis-parallel rectangles is, that they can be described using only horizontal and vertical lines. A useful tool to leverage this fact is provided by *grids*. A grid is a finite set of horizontal and vertical lines in the plane, called *grid lines*. The *size* $|G|$ of a grid G is the total number of lines it contains. The intersection of a horizontal and a vertical line of a grid G, is called a *grid point* of G. The set of grid points of G is denoted by points(G). The lines of a grid G divide the plane into *grid cells* of G. Each such grid cell is a rectangle, possibly with one or two sides extending to infinity, and has at most four *corners*: the grid points lying on its boundary.

Part I

Block Structured Integer Programming

2 Treefold Integer and Linear Programming in Strongly Polynomial and Near Linear Time

This chapter contains an overworked version of [CEH+21] which is joint work with Friedrich Eisenbrand, Christoph Hunkenschröder, Lars Rohwedder and Robert Weismantel.

2.1 Introduction

We consider *n-fold* and *treefold* integer and linear programming problems (see Section 1.2.1). In particular, we even consider integer linear programs in any n-fold structure (★). Recall that such a linear program is given by

$$\begin{aligned} \max\ & \bar{c}_1^\intercal x_1 + \cdots + \bar{c}_n^\intercal x_n \\ & B_1 x_1 + \cdots + B_n x_n = b_0 \qquad (\bigstar) \\ & x_i \in Q_i \quad i = 1, \ldots, n, \end{aligned}$$

Here, the $Q_i \subseteq \mathbb{R}^{q_i}_{\geq 0}$, $i = 1, \ldots, n$ are polyhedra, the $B_i \in \mathbb{Z}^{r \times q_i}$, $i = 1, \ldots, n$ integer matrices and $b_0 \in \mathbb{Z}^r$ is an integer vector. In the case where (★) is to model an integer linear program, we also have the integrality constraint $x_i \in \mathbb{Z}^{q_i}$, $i = 1, \ldots, n$. The r given by the B_i are the *linking constraints* of (★) and, if they are removed, the problem decomposes into n independent (integer) linear programs. This means that Q_i is described by one of the following polyhedra

$$\begin{gathered} \{x_i \colon D_i x_i = b_i, x_i \geq 0\}, \text{ or} \\ \operatorname{conv}\{x_i \colon D_i x_i = b_i, x_i \geq 0, x \in \mathbb{Z}^{q_i}\}, \end{gathered}$$

where $D_i \in \mathbb{Z}^{s \times q_i}$ is an integral matrix, $b_i \in \mathbb{Z}^s$ an integral vector, and conv the convex hull of a set of points. Most results and running times to solve a linear program in n-fold structure (★) depend on parameters of the matrices D_i. Recall that the notation $\bar{c}_i$ is used to denote that the vector c_i is given symbolically, that is, it can only be accessed through linear queries. This

notion is used to describe that we seek an algorithm linear in $c_1, \dots, c_n$, see Section 1.1.3 for a definition. Throughout this chapter, we use

$$\gamma := \max_{1 \le i \le n} g_1(D_i)$$

as a bound on the ℓ_1 Graver measure of all D_i. Further,

$$\Delta := \max_{1 \le i \le n} \{\|B_i\|_\infty, \|D_i\|_\infty\}$$

is an upper bound on each entry in the matrices B_i and D_i for $i = 1, \dots, n$.

Our contribution

We present two main results regarding linear programs in n-fold structure.

(i) We show how to efficiently solve a linear program with n-fold structure (★) by adapting the framework of Norton et al. [NPT92] to the setting of linear programs with n-fold structure. We leverage the inherent parallelism by using the multidimensional search technique of Megiddo [Meg84, Dye86, Cla86] and obtain the following result (Theorem 2.1): Let $T_{\max}$ be an upper bound on the running time for solving a linear programming problem $\max\{\bar{c}_i^\intercal x_i : x_i \in Q_i\}$. Then (★) can be solved in parallel on n processors, where each processor carries out $2^{\mathcal{O}(r^2)}(T_{\max}\log(n))^{r+1}$ operations. For technical reasons, we make the mild assumption that the algorithm is linear in the objective function c_i, see Section 1.1.3 for a definition. This result can be further refined if the individual block problems $\max\{\bar{c}_i^\intercal x_i, x_i \in Q_i\}$ can be efficiently solved in *parallel* as well.

(ii) We furthermore provide a new proximity bound for the integer programming variant of (★) (Theorem 2.5). If all polyhedra Q_i are integral and $x^\star$ a vertex solution of the corresponding linear program (★), then there exists an optimal solution $x^\diamond$ of the integer programming problem such that

$$\|x^\star - x^\diamond\|_1 \le (2r\Delta\gamma + 1)^{r+4}. \tag{2.1}$$

Here, Δ is an upper bound on the absolute value of each entry in matrices B_i and D_i, and γ a bound on the ℓ_1 Graver measure for all D_i. Where for all $i = 1, \dots, n$, we assume D_i to be the matrix describing Q_i. Our new contribution is that this bound is independent of *n and q*.

Using (i) and (ii) we also derive an algorithm for the integer programming problem (★). We first solve the continuous relaxation of (★) using (i). Here we assume that for $i = 1, \dots, n$, each Q_i is integral or otherwise we replace it by the convex hull of its integer solutions. Then we find an optimal integer solution via dynamic programming using (ii) to restrict the search space. All techniques above can also be applied recursively, e.g., to solve linear programs with

n-fold structure, where the polytopes Q_i, $i = 1,\dots,n$, have a a n-fold structure as well.

We want to emphasize three main applications of the techniques described above. First, we use them to derive a fixed parameter tractable algorithm for n-fold integer and linear programming. Here, the polyhedra Q_i are given by systems $D_i x_i = b_i$, $0 \le x_i \le u$, with $D_i \in \mathbb{Z}^{s\times q}$ all of the same dimension. Algorithms for n-fold integer programming have been used, for example in [KK18, CMYZ17, JKMR21] to derive novel fixed parameter tractable results in scheduling. Moreover, they have been successfully applied to derive fixed parameter tractable results for string and social choice problems [KKM20b, KKM20a].

On the other hand, we use a recursive application of the techniques to get a fixed parameter tractable algorithm for integer linear programming (♠) parameterized by the *dual treedepth* of the constraint matrix.

(iii) We obtain a strongly polynomial and nearly linear time algorithm for n-fold integer programming. Our algorithm requires

$$2^{\mathcal{O}(rs^2)}(rs\Delta)^{\mathcal{O}(r^2s+s^2)}(nq)^{1+o(1)}$$

arithmetic operations, or alternatively $2^{\mathcal{O}(r^2+rs^2)}\log^{\mathcal{O}(rs)}(nq\Delta)$ parallel operations on $(rs\Delta)^{\mathcal{O}(r^2s+s^2)}nq$ processors. Previous algorithms in the literature either had at least a quadratic (and probably higher) dependence on nq or an additional factor of φ, which is the encoding size of the largest integer in the input. Moreover, this is the first parallel algorithm.

(iv) We present an algorithm for n-fold linear programming which requires

$$2^{\mathcal{O}(r^2+rs^2)}(nq)^{1+o(1)}$$

arithmetic operations, or alternatively $2^{\mathcal{O}(r^2+rs^2)}\log^{\mathcal{O}(rs)}(nq)$ parallel operations on nq processors. This extends the class of linear programs known to be solvable in strongly polynomial time and those known to be parallelizable.

(v) In terms of dual treedepth, we also obtain a strongly polynomial algorithm. More precisely, our algorithm requires

$$d^{\mathcal{O}(d^2 2^d)}\|A\|_\infty^{\mathcal{O}(2^d)}t^{1+o(1)}$$

arithmetic operations, or alternatively $\log^{\mathcal{O}(d2^d)}(d\|A\|_\infty t)$ parallel arithmetic operations on $d^{\mathcal{O}(d^3)}\|A\|_\infty^{\mathcal{O}(2^d)}t$ processors. Here t is the number of variables. Furthermore, we present a running time analysis for the corresponding linear programming case.

Further related work

Closely related are dynamic programming approaches to integer linear programming [Pap81]. In [EW20a] it was shown that an integer linear program $\max\{c^\intercal x\colon Ax = b, 0 \le x \le u, x \in \mathbb{Z}^n\}$ with $A \in \mathbb{Z}^{s\times t}$ can be solved in time $(s\|A\|_\infty)^{\mathcal{O}(s^2)}t$ and in time $(s\|A\|_\infty)^{\mathcal{O}(s)}$ if there are no upper bounds on the variables. Jansen and Rohwedder [JR19] obtained better constants in the exponent of the running time of integer programs without upper bounds. Assuming the Exponential Time Hypothesis, a tight lower bound was presented by Knop et al. [KPW20].

The first fixed parameter tractable algorithm for n-fold integer programming is due to Hemmecke et al. [HOR13] and is with respect to parameters $\|A\|_\infty, r, s$ and q. Their running time is $\mathcal{O}(n^3\varphi)\|A\|_\infty^{\mathcal{O}(q(rs+sq))}$ where φ is the encoding size of the largest value of a component in the input.

The exponential dependence on q was removed by Eisenbrand et al. [EHK18] and Koutecký et al. [KLO18]. The first strongly polynomial algorithm for n-fold integer programs was given by Koutecký et al. [KLO18]. The previously fastest algorithms for n-fold integer programming were provided by Jansen, Lassota and Rohwedder [JLR20] and Eisenbrand et al. [EHK+19]. While the first work has a slightly better parameter dependency, the second work achieves a better dependency on the number of variables and is strongly polynomial. The results above are based on an augmentation framework, which differs significantly from our methods. In this framework an algorithm iteratively augments an integral solution, ultimately converging to the optimal integral solution. This requires $\Omega(n)$ sequential iterations, making parallelization hopeless.

Other variants of recursively defined block structured integer programming problems were also considered in the literature. Notable cases include the tree-fold integer programming problem introduced by Chen and Marx [CM18]. This case is closely related to dual treedepth and can be analyzed in a similar way using our theorems. The currently best algorithms parameterized by dual treedepth obtain a running time of $\|A\|_\infty^{\mathcal{O}(\mathsf{td}_\mathsf{D}(A)2^{\mathsf{td}_\mathsf{D}(A)})}\varphi^2 n^{1+o(1)}$, where φ is the encoding size of the largest value of the input [EHK+19].

A crucial ingredient of our algorithms is the use of a relaxation of (★) where Q_i is integral for $i = 1,\dots,n$. For many settings (e.g., for the n-fold case) this is stronger than using a naive relaxation of (★) obtained by simply dropping the integrality constraints. This idea of using a stronger relaxation was also used by Koutecký et al. in [KKL+19], where they consider a high-multiplicity setting. Roughly speaking, they consider linear programs in n-fold structure with only a few types of polyhedra Q_i but possibly repeated several times. Using the stronger relaxation, the authors also obtain a proximity result independent of the dimension. However, their result still depends on the number of variables per block and the number of different polyhedra types.

Structure of the chapter

Section 2.2 is dedicated to solving the linear relaxation, that is item (i). We give an adaptation of the parametric search framework by Norton, Plotkin, and Tardos (Section 2.2.1) and combine it with Megiddo's multidimensional search technique in Section 2.2.2. In Section 2.3 we provide a proximity result for the stronger relaxation, that is item (ii). Which is then used in Section 2.4 to establish a dynamic program. Applications of our techniques are discussed in Section 2.5.

2.2 Solving the LP by Parametric Search and Parallelization

We now describe how to solve a linear program given in n-fold structure (★) efficiently and in parallel. The method that we lay out is based on a technique of Norton, Plotkin and Tardos [NPT92]. The authors of this paper show the following. Suppose there is an algorithm that solves a linear programming problem in time $T(n)$, where n is some measure of the length of the input and let us suppose that we change this linear program by adding r additional constraints. Norton et al. show that this augmented linear program can be solved in time $(T(n))^{r+1}$. A straightforward application of this technique to our setting would yield the following. Consider the starting linear program in n-fold structure (★). Removing the r linking constraints, the n individual linear programs $\max\{\bar{c}_i^\intercal x\colon x \in Q_i\}$ can be solved a total running time of $\Omega(n)$. Using the result [NPT92] out of the box would then yield a running time bound of $\Omega(n^{r+1})$. The main result of this section is to improve this to:

Theorem 2.1. *Suppose there are algorithms that solve* $\max\{\bar{c}_i^\intercal x\colon x \in Q_i\}$ *on* R_i *processors using at most* $T_{\max}$ *operations on each processor and suppose that these algorithms are linear in* $\bar{c}_i$*. Then there is an algorithm solving a linear program in* n*-fold structure* (★) *on* $R = \sum_i R_i$ *processors, requiring* $2^{\mathcal{O}(r^2)}(T_{\max}\log(R))^{r+1}$ *operations on each processor. This algorithm is linear in* $\bar{c}$*.*

Remark 2.2. The problems $\max\{\bar{c}_i^\intercal x\colon x \in Q_i\}$ can be solved independently in parallel. If $T_{\max}$ is an upper bound on the running times of these algorithms, then Theorem 2.1 provides a sequential running time of

$$2^{\mathcal{O}(r^2)} n \cdot (T_{\max}\log(n))^{r+1} = 2^{\mathcal{O}(r^2)} n^{1+o(1)} \cdot T_{\max}^{r+1}$$

to solve the linear program in n-fold structure (★). Theorem 2.1 is stated in greater generality in order to use the potential of parallel algorithms that solve the problems $\max\{\bar{c}_i^\intercal x\colon x \in Q_i\}$ themselves. This makes way for a refined analysis of linear programming problems that are in recursive block structure as demonstrated in our applications.

The novel elements of this chapter are the following. We leverage the parallelism exhibited in the solution of the Lagrange dual in the framework of Norton et al. [NPT92]. Furthermore, we present an analysis of the multidimensional search technique of Megiddo [Meg84] in the framework of linear algorithms to clarify how this technique can be used in our setting.

Roughly speaking, multidimensional search deals with the following problem. Given m hyperplanes $a_i^\intercal \lambda = \overline{f}_i, i = 1,\dots,m$ and $\overline{\lambda} \in \mathbb{R}^r$, the task is to understand the orientation of $\overline{\lambda}$ with respect to each hyperplane using only few linear queries on $\overline{\lambda}$. Megiddo shows how to do this in time $2^{\mathcal{O}(r)} \cdot m \log^2(m)$ while the total number of linear queries involving $\overline{\lambda}$ is bounded by $2^{\mathcal{O}(r)} \log(m)$. This is crucial for us and implicit in his analysis, we make it explicit here. Finally, the algorithm itself is linear in $\overline{c}$. In [NPT92] it is not immediately obvious that linearity can be preserved. However, this is important for linear programs with a recursive block structure.

2.2.1 The Technique of Norton et al.

We now explain the algorithm to solve a linear program in n-fold structure (★). In the following we write

$$\begin{aligned} B &= (B_1 \ \dots \ B_n) \in \mathbb{R}^{r \times (q_1 + \dots + q_n)}, \\ x^\intercal &= (x_1^\intercal \ \dots \ x_n^\intercal) \in \mathbb{R}^{q_1 + \dots + q_n}, \quad \text{and} \\ \overline{c}^\intercal &= (\overline{c}_1^\intercal \ \dots \ \overline{c}_n^\intercal) \ \in \mathbb{Z}^{q_1 + \dots + q_n}. \end{aligned}$$

It is well known that a linear program can be solved via *Lagrangian relaxation*. We refer to standard textbooks in optimization for further details, see, e.g. [BV14, Sch98]. By dualizing the linking constraints of (★) the Lagrangian $L(\lambda)$ with weight $\lambda \in \mathbb{R}^r$ is the linear programming problem

$$\begin{aligned} L(\lambda) = \max \ & \overline{c}^\intercal x - \lambda(Bx - b_0) \\ & x_i \in Q_i \qquad i = 1,\dots,n. \end{aligned} \tag{2.2}$$

The Lagrangian dual is the task to solve the convex optimization problem

$$\min_{\overline{\lambda} \in \mathbb{R}^r} L(\overline{\lambda}). \tag{2.3}$$

For a given symbolic vector $\overline{\lambda}$, the value of $L(\overline{\lambda})$ can be found by solving the independent optimization problems

$$P_i = \max\{(\overline{c}_i^\intercal - \overline{\lambda}^\intercal B_i) x_i : \ x_i \in Q_i\} \tag{2.4}$$

with the corresponding algorithms which are linear in the objective function vector. The objective function vector of (2.4) is $\overline{c}_i - B_i^\intercal \overline{\lambda}$. A query on this objective function vector posed by the algorithm solving (2.4) can be put in the form

$$a^\intercal \overline{\lambda} \leq \overline{f}. \tag{2.5}$$

Here $\overline{f}$ is an affine function in the components of $\overline{c}_i$ defined by the query.

We are now ready to describe the main idea of the framework by Norton et al. Assume that we have an algorithm $\mathcal{A}_k$ solving the following restricted Lagrangian

$$\min_{\overline{\lambda} \in S} L(\overline{\lambda}), \tag{2.6}$$

where S is any k-dimensional affine subspace of $\mathbb{R}^r$ defined by $S = \{\lambda \in \mathbb{R}^r : D\lambda = \overline{d}\}$ for a matrix $D \in \mathbb{R}^{(r-k)\times r}$ of $r-k$ linear independent rows and a symbolic vector $\overline{d} \in \mathbb{R}^{r-k}$. To be precise, $\mathcal{A}_k$ is delivering the optimal solution $\overline{\lambda}_S$ of (2.6) (as an affine function in $\overline{c}$ and $\overline{d}$) as well as an optimal solution $x^\star \in Q_1 \times \cdots \times Q_n$ of the linear program $L(\overline{\lambda}_S)$. Assume the algorithm $\mathcal{A}_k$ to be linear in $\overline{c}$ and $\overline{d}$. Further, assume that in the symbolic vector $\overline{d}$ each component is a linear function in $\overline{c}$. It follows that the restricted Lagrangian (2.6) stems from constraining the vectors λ to satisfy $n-k$ linearly independent queries of the form (2.5) with equality.

The algorithm $\mathcal{A}_0$ solves the restricted Lagrangian dual for a subspace consisting of a single symbolic point $S = \{\overline{\lambda}_0\}$. In other words, the algorithm $\mathcal{A}_0$ simply returns $\overline{\lambda}_0 = D^{-1}\overline{d}$ which is an affine function in $\overline{c}$, together with a corresponding optimal solution to (2.2) for $\overline{\lambda}_0$. Note that, the algorithm $\mathcal{A}_r$ solves the unrestricted version of the Lagrangian dual (2.3).

Now, we describe how to construct the algorithm $\mathcal{A}_{k+1}$ with the algorithm $\mathcal{A}_k$ at hand. To this end, let $S = \{\lambda \in \mathbb{R}^r : D\lambda = \overline{d}\}$ be of dimension $k+1$, i.e., we assume that D consists of $r-k-1$ linearly independent rows. Furthermore, let $\overline{\lambda}_S \in S$ be an optimal solution of the restricted Lagrangian dual (2.6) which is unknown to us.

Let $\mathcal{E}$ by the algorithm evaluating the Lagrangian $L(\overline{\lambda})$. The idea is to run $\mathcal{E}$ on $L(\overline{\lambda}_S)$. Even though $\overline{\lambda}_S$ is unknown, this is possible if each linear query (2.5) occurring in $\mathcal{E}$ is answered as if it was queried for $\overline{\lambda}_S \in S$. The symbolic point $\overline{\lambda}_S$ is then any point satisfying all queries.

Let $a^\intercal\overline{\lambda} \le \overline{f}$ be a query (2.5) given by $\mathcal{E}$. If a is in the span of rows of D, then the query can be answered by a linear query on the right hand sides $\overline{d}$. Since the components of $\overline{d}$ are linear functions in $\overline{c}$, the query is also linear in $\overline{c}$. Thus, we may assume a is not in the span of the rows of D. We next show that, by three calls on the algorithm $\mathcal{A}_k$, we can decide whether

(i) $a^\intercal\lambda^\star = \overline{f}$ for some optimal solution $\lambda^\star \in S$,
(ii) $a^\intercal\lambda^\star > \overline{f}$ for each optimal solution $\lambda^\star \in S$, or
(iii) $a^\intercal\lambda^\star < \overline{f}$ for each optimal solution $\lambda^\star \in S$

which means that we can answer the query as if it was asked for $\overline{\lambda}_S$. Let $\varepsilon > 0$ be an unspecified small value. We use $\mathcal{A}_k$ to find optimal solutions $\overline{\lambda}_L, \overline{\lambda}_R$ and $\overline{\lambda}_0$ of the Lagrangian restricted to

$$\begin{aligned} S_L &= S \cap \{\lambda : a^\intercal\lambda = \overline{f} - \varepsilon\} \\ S_0 &= S \cap \{\lambda : a^\intercal\lambda = \overline{f}\} \\ S_R &= S \cap \{\lambda : a^\intercal\lambda = \overline{f} + \varepsilon\} \end{aligned}$$

respectively. Since $\overline{\lambda}_L, \overline{\lambda}_R$ and $\overline{\lambda}_0$ are affine functions in $\bar{c}$, we can compare their corresponding values and since $L(\lambda)$ is convex, these comparisons allow to decide whether (i), (ii) or (iii) holds.

The value ε does not have to be provided explicitly but can be treated symbolically. For instance, in the recursion on S_L, the algorithm $\mathcal{A}_k$ produces a series of linear queries of the form $u^\intercal \bar{c} + y\varepsilon \leq z$. If y is positive, this is the query $u^\intercal \bar{c} < z$ which can be answered by querying $u^\intercal \bar{c} \leq z$ and $u^\intercal \bar{c} \geq z$.

We have shown how to simulate an algorithm that computes $L(\lambda)$ as if it was on input $\overline{\lambda}_S$. It remains to describe how to retrieve $\overline{\lambda}_S$ as a linear function in $\bar{c}$. This is done as follows. Let $x^\star$ be an optimal solution of $L(\overline{\lambda}_S)$ found by the above simulation. Let $U\lambda = \overline{u}$ be the system of equations given by all those linear queries of $\mathcal{A}_k$ which were answered with (i). The value of $L(\overline{\lambda}_S)$ is equal to

$$\max\left\{\bar{c}^\intercal x^\star - \lambda^\intercal(Ax^\star - b)\colon \lambda \in \mathbb{R}^r, \begin{pmatrix} D \\ U \end{pmatrix}\lambda = \begin{pmatrix} \overline{d} \\ \overline{u} \end{pmatrix}\right\}.$$

If $(Ax^\star - b)^\intercal$ can be expressed as a linear combination of the rows of D and U, then any point in the subspace given is optimal and one can be found with Gaussian elimination. Otherwise, the Lagrange dual restricted to S is unbounded.

It remains to analyze the running time of the algorithm $\mathcal{A}_{k+1}$. Let T be the running time of the algorithm $\mathcal{E}$ evaluating $L(\lambda)$ if each linear query (2.5) counts as a single operation. Then, the running time of A_{k+1} is $3 \cdot T$ times the running time of the algorithm A_k. This shows that the running time of $\mathcal{A}_r$ is bounded by $(3 \cdot T)^{r+1}$.

2.2.2 Acceleration by Parallelization and Multidimensional Search

A linear program in n-fold structure (★) has the important feature that, for a given $\overline{\lambda} \in \mathbb{R}^r$, the value of $L(\overline{\lambda})$ can be computed by solving the n linear programming problems $P_1, \ldots, P_n$ (2.4) in parallel. We now explain how to exploit this and, in consequence, prove Theorem 2.1.

For the sake of a more accessible treatment, let us assume for now that each of these problems can be solved in time $T_{\max}$ on an individual processor. This means that the evaluation of $L(\lambda)$ can be carried out with algorithm $\mathcal{E}$ on n processors by algorithms which are linear in their respective objective function vectors. We are now looking again at the construction of the algorithm $\mathcal{A}_{k+1}$ with the algorithms $\mathcal{A}_k$ and $\mathcal{E}$ at hand.

In a single step of the parallel algorithm $\mathcal{E}$, there are at most n queries of the form (2.5) coming from the individual sub-problems P_i, $i = 1, \ldots, n$. In the previous subsection, these queries were answered one-by-one according to $\overline{\lambda}_S$ by calling $\mathcal{A}_k$ three times. This can be reduces massively by using Megiddo's multidimensional search technique and Clarkson and Dyer's improvement [Dye86, Cla86].

Theorem 2.3 (Megiddo)**.** *Let $\overline{\lambda} \in \mathbb{R}^r$ be a symbolic vector and consider a set of m hyperplanes*

$$H_i = \{\lambda \in \mathbb{R}^r : a_i^\intercal \lambda = \overline{f_i}\}, \quad i = 1, \dots, m.$$

There is an algorithm that determines for each hyperplane whether

(i) $a_i^\intercal \overline{\lambda} = \overline{f_i}$,
(ii) $a_i^\intercal \overline{\lambda} < \overline{f_i}$, *or*
(iii) $a_i^\intercal \overline{\lambda} > \overline{f_i}$

in $2^{\mathcal{O}(r)} \log^2(m)$ *operations on* $\mathcal{O}(m)$ *processors. Moreover, the total (sequential) number of comparisons dependent on* $\overline{\lambda}$ *is at most* $2^{\mathcal{O}(r)} \log(m)$.

To get an intuition on why the number of queries involving $\overline{\lambda}$ is this low, we consider the base-case $r = 1$. Assume that $a_i \neq 0$ for all i and compute the median $\overline{M}$ of the numbers $\overline{f_i}/a_i$. Then, for each hyperplane H_i one checks whether $\overline{f}_i/a_i \leq \overline{M}$ and $\overline{f_i}/a_i \geq \overline{M}$ holds. The standard median computation [BFP+72] requires $\mathcal{O}(\log(m))$ operations on $\mathcal{O}(m)$ processors. Now, compare $\overline{\lambda} \leq \overline{M}$ and $\overline{\lambda} \geq \overline{M}$. From the result, we can derive an answer for $m/2$ of the hyperplanes. Thus, the total number of linear queries involving $\overline{\lambda}$ is bounded by $\mathcal{O}(\log(m))$.

Note that we stated Theorem 2.3 in the framework of algorithms linear in $\overline{\lambda}$ and $\overline{f_i}$, $i \in \{1, \dots, m\}$. This version is implicitly proven in the papers [Meg84, Dye86, Cla86], nevertheless we provide a proof after discussing its consequences.

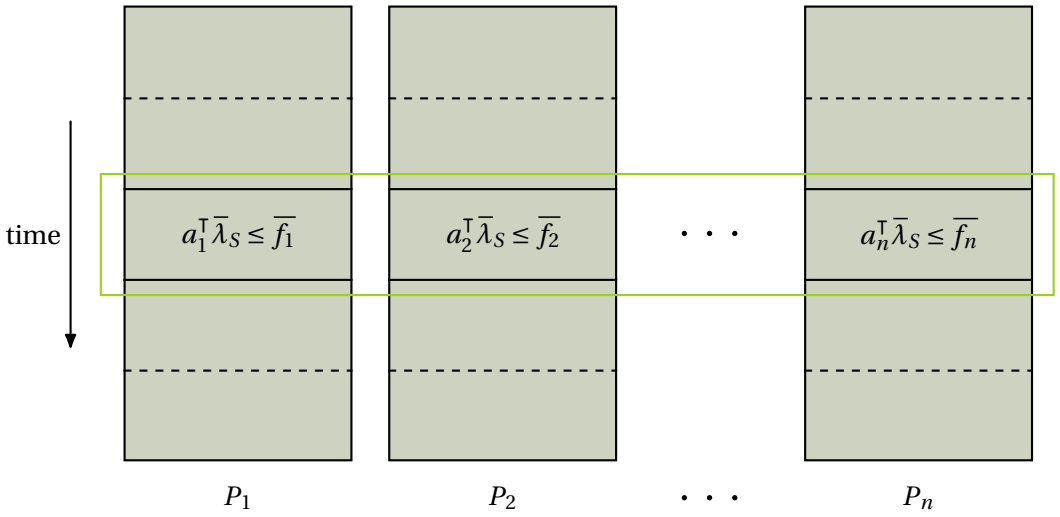

Figure 2.1: The algorithm $\mathcal{E}$ running the optimization problems P_i in parallel.

To evaluate the Lagrangian at $\overline{\lambda}_S$, the algorithm $\mathcal{E}$ runs the optimization problems $P_1, \dots, P_n$ in parallel. At each given step in time, these n algorithms can all make a single query of the form (2.5)

$$a_1^\intercal \overline{\lambda}_S \leq \overline{f_1}\ , \ \dots\ , \ a_n^\intercal \overline{\lambda}_S \leq \overline{f_n},$$

see Figure 2.1. Recall that each $\overline{f_i}$ is an affine function of $\bar{c}$. These n queries can now be answered with Megiddo's algorithm in time $2^{\mathcal{O}(r)}\log^2(n)$ operations on n processors and a total number of $2^{\mathcal{O}(r)}\log(n)$ linear comparisons on $\overline{\lambda}_S$. As explained in the previous section, these $2^{\mathcal{O}(r)}\log(n)$ linear queries can be answered by 3 calls to the algorithm $\mathcal{A}_k$. Say that the parallel algorithms in $\mathcal{E}$ solving the linear programs P_i have a maximum running time of $T_{\max}$. Then using n processors, the algorithm $\mathcal{A}_{k+1}$ has a total running time of $2^{\mathcal{O}(r)}\log^2(n)\,T_{\max}$ plus $2^{\mathcal{O}(r)}\log(n)\,T_{\max}$ times the running time of $\mathcal{A}_k$. This shows that the running time of $\mathcal{A}_r$ is bounded by $2^{\mathcal{O}(r^2)}(T_{\max}\log(n))^{r+1}$. For a complete proof of Theorem 2.1 it remains to show how to leverage parallel algorithms for the P_i, and how to find an optimal solution $x^\star$ of (★).

Proof of Theorem 2.1. We follow the lines of the argument above, assuming now that the linear optimization problems $P_i = \max\{(\bar{c}_i^\intercal - \overline{\lambda}^\intercal B_i)x_i \colon x_i \in Q_i\}$ are solved on R_i processors using at most $T_{\max}$ operations on each processor. In this case, we obtain that the algorithm $\mathcal{A}_r$ requires $2^{\mathcal{O}(r^2)}(T_{\max}\log(R))^{r+1}$ operations on $R = \sum_{i=1}^n R_i$ processors. The optimal solution $\overline{\lambda}_{OPT}$ of the Lagrangian dual (2.3) can thus be found in this time bound. The remainder of the proof is dedicated to show how to find an optimal solution $x^\star$ of the linear program in n-fold structure (★).

The algorithm $\mathcal{A}_r$ proceeds by evaluating $L(\overline{\lambda}_{OPT})$ at the unknown point $\overline{\lambda}_{OPT}$. To do so, it recursively makes various calls to the algorithm $\mathcal{E}$ to find optimal solutions to different restricted Lagrangian duals (2.6). These optimal solutions are vertices $v_1,\dots,v_\ell$ of the polytope $Q_1 \times \cdots \times Q_n$. Note that replacing the condition of $x \in Q_1 \times \cdots \times Q_n$ by $x \in \operatorname{conv}\{v_1,\dots,v_n\}$ in the Lagrangian (2.2) does not change the optimal solution of the Lagrangian dual (2.3). Therefore, the optimal value of the primal linear program is also not affected by restricting x to $\operatorname{conv}\{v_1,\dots,v_n\}$. Thus, to find an optimal solution to a linear program in n-fold structure (★), it is sufficient to solve the restricted linear program

$$\begin{aligned} \max \sum_{i=1}^{\ell} (\bar{c}^\intercal v_i)\,\mu_i & \\ B\left(\sum_{i=1}^{\ell} \mu_i v_i\right) &= b_0 \\ \sum_{i=1}^{\ell} \mu_i &= 1 \\ \mu &\geq 0. \end{aligned}$$

The dual of this linear program is a linear program in $r+1$ dimensions and with ℓ constraints. This can be solved in time $2^{\mathcal{O}(r^2)} \cdot \ell$ with Megiddo's algorithm [Meg84].

We now argue that the number ℓ of vertices is bounded by $2^{\mathcal{O}(r^2)}(T_{\max}\log R)^r$ which, in turn, implies that this additional work can be done in the claimed running time bound on one processor. The algorithm $\mathcal{A}_{k+1}$ runs $\mathcal{E}$ and makes $2^{\mathcal{O}(r)}\,T_{\max}\log(R)$ calls to the algorithm $\mathcal{A}_k$. This shows that the total number of calls incurred by the algorithm $\mathcal{A}_r$ is bounded by

$2^{\mathcal{O}(r^2)}(T_{\max}\log R)^r$. This in turn bounds the number of vertices as claimed and finishes the proof of the main result of this section. □

Let us now turn to the proof of Theorem 2.3, adapted to the formulation using algorithms linear in $\overline{\lambda}$ and the $\overline{f_i}$. Note again that we closely follow the proofs provided in [Meg84, Cla86, Dye86], with the difference that we pay attention to the fact that the algorithm is linear in $\overline{\lambda}$ and the $\overline{f_i}$.

Proof of Theorem 2.3. We show that in $2^{\mathcal{O}(r)}\log(m)$ operations on $\mathcal{O}(m)$ processors with $2^{\mathcal{O}(r)}$ comparisons dependent on $\overline{\lambda}$, we can determine the location of $\overline{\lambda}$ with respect to half of the hyperplanes. The claimed result then follows with $\mathcal{O}(\log(m))$ repetitions. The algorithm solves the problem by recursively solving the same problem in smaller dimensions.

Consider the base case in dimension $r = 1$. We start by dealing with all hyperplanes $H_i = \{\lambda \in \mathbb{R}^r : a_i^\mathsf{T}\lambda = \overline{f_i}\}$ where $a_i = 0$. For this, it is sufficient to check whether $\overline{f_i} \le 0$ and $\overline{f_i} \ge 0$, which can be done in parallel. Hence, assume now $a_i \neq 0$ for all hyperplanes H_i. Next, we compute the median $\overline{M}$ of the numbers $\overline{f_1}/a_1, \ldots, \overline{f_m}/a_m$ and for each hyperplane H_i whether $\overline{f_i}/a_i \le \overline{M}$ and $\overline{f_i}/a_i \ge \overline{M}$ hold. The standard median computation [BFP+72] requires $\mathcal{O}(\log(m))$ operations on $\mathcal{O}(m)$ processors. Now compare $\overline{\lambda} \le \overline{M}$ and $\overline{\lambda} \ge \overline{M}$. From the result we can derive an answer for $m/2$ of the hyperplanes. The total number of linear queries involving $\overline{\lambda}$ is bounded by $\mathcal{O}(\log(m))$.

Now consider the dimension $r > 1$. In this case, the following observation is crucial in reducing the problem to a smaller dimension. Let H_i and H_j be two hyperplanes with $(a_i)_1/(a_i)_2 \le 0$ and $(a_j)_1/(a_j)_2 > 0$. We can define two new hyperplanes, one parallel to the λ_1-axis and one parallel to the λ_2-axis such that the location of $\overline{\lambda}$ with respect to these new hyperplanes implies its location with respect to either H_i or H_j.With this in mind, we start by transforming the coordinate system such that many pairs with this property exist.

Similar to the base case, we first consider all hyperplanes H_i with $(a_i)_2 = 0$ and solve half of them recursively in dimension $r-1$. For the remainder we assume $(a_i)_2 > 0$ for all H_i by simply swapping the signs on all other hyperplanes. Now compute the median M of $(a_i)_1/(a_i)_2$ and, for each hyperplane H_i, determine whether $(a_i)_1/(a_i)_2 \le M$ and $(a_i)_1/(a_i)_2 \ge M$ holds. This takes $\mathcal{O}(\log(m))$ operations on $\mathcal{O}(m)$ processors.

Next, transform the coordinate system by using the automorphism given by $F \in \mathbb{R}^{r\times r}$ such that

$$(\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_r)^\mathsf{T} F = (\lambda_1 - M\lambda_2, \lambda_2, \lambda_3, \ldots, \lambda_r)^\mathsf{T}.$$

For each H_i, define a new hyperplane $H_i' = \{\lambda' \in \mathbb{R}^r : a_i'^\mathsf{T}\lambda' = \overline{f_i}\}$ with $a' = aF$. Then, locating $\overline{\lambda'} = (F^\mathsf{T})^{-1}\overline{\lambda}$ with respect to the hyperplanes H_i' is equivalent to locating $\overline{\lambda}$ with respect to the hyperplanes H_i. Hence, it suffices to construct an algorithm for the new hyperplanes H_i'. Then, running this algorithm with each comparison $u^\mathsf{T}\overline{\lambda'} + v^\mathsf{T}\overline{f} \le w$ transformed to $(u^\mathsf{T}(F^\mathsf{T})^{-1})\overline{\lambda'} + v^\mathsf{T}\overline{f} \le w$ yields an algorithm for the original hyperplanes H_i.

We have established that for one half of the new hyperplanes $(a'_i)_1 \leq 0$ holds and for the other half, $(a'_i)_1 \geq 0$ holds. Notice also that $(a'_i)_2 = (a_i)_2 > 0$. Now, form a maximum number of pairs of hyperplanes such that each pair (H'_i, H'_j) satisfies $(a'_i)_1 \leq 0$ and $(a'_j)_1 > 0$. For all remaining hyperplanes $(a'_i)_1 = 0$ holds and we solve half of them recursively in dimension $r-1$. Hence, we focus on the pairs and alter the hyperplanes once more. For each pair (H'_i, H'_j) define the two hyperplanes

$$H''_i = \left\{ \lambda' \in \mathbb{R}^r : \underbrace{\left((a'_j)_1 a'_i - (a'_i)_1 a'_j\right)}_{=:a''_i}{}^{\mathsf{T}} \lambda' = \underbrace{(a'_j)_1 \overline{f_i} - (a'_i)_1 \overline{f_j}}_{=:\overline{f''_i}} \right\},$$

$$H''_j = \left\{ \lambda' \in \mathbb{R}^r : \underbrace{\left((a'_j)_2 a'_i - (a'_i)_2 a'_j\right)}_{=:a''_i}{}^{\mathsf{T}} \lambda' = \underbrace{(a'_j)_2 \overline{f_i} - (a'_j)_1 \overline{f_j}}_{=:\overline{f''_j}} \right\}.$$

These hyperplanes are a combination of H'_i and H'_j which eliminates either the first or second coordinate. Hence, locating $\overline{\lambda'}$ with respect to the hyperplanes of the first kind (those where λ_1 is eliminated) is a problem in dimension $r-1$. We now recursively solve the problem for half of these hyperplanes. For those pairs where the first kind of hyperplanes is solved, we recurse again on the hyperplanes of the second kind. Thus, we located $\overline{\lambda'}$ with respect to both H''_i and H''_j for at least $1/4$ of the pairs (H'_i, H'_j).

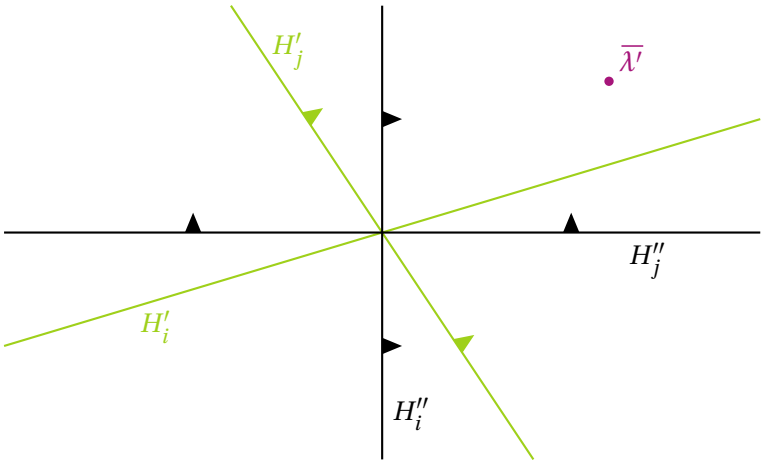

Figure 2.2: Megiddo's multidimensional search algorithm illustrated in $r = 2$ dimensions.

It remains to derive the location with respect to one of H'_i and H'_j. In two dimensions this is illustrated in Figure 2.2. Intuitively, if for example, $\overline{\lambda'}$ is to the right of H''_i and above H''_j, then

it is to the top-right of H'_j. Notice that we can write

$$\underbrace{\left((a'_i)_2(a'_j)_1 - (a'_i)_1(a'_j)_2\right)}_{=:\mu_i>0} a'_i = \underbrace{(a'_i)_2}_{\geq 0} a''_i - \underbrace{(a'_i)_1}_{\leq 0} a''_j, \quad (a'_i)_2\overline{f''_1} - (a'_i)_1\overline{f''_j} = \mu_i\overline{f_i}$$
$$\underbrace{\left((a'_j)_1(a'_i)_2 - (a'_j)_2(a'_i)_1\right)}_{=:\mu_j<0} a'_j = \underbrace{(a'_j)_2}_{\geq 0} a''_i - \underbrace{(a'_j)_1}_{>0} a''_j, \quad (a'_j)_2\overline{f''_i} - (a'_j)_1\overline{f''_j} = \mu_j\overline{f_j}.$$

From the coefficients above it follows that if the signs of the comparisons between $a''^{\intercal}_i\overline{\lambda'}, \overline{f''_i}$ and $a''^{\intercal}_j\overline{\lambda'}, \overline{f''_j}$ are equal, this implies an answer for H'_j, and if they differ, it implies an answer for H'_i.

This means that we know the location of $\overline{\lambda}$ with respect to at least $1/8$ of the paired up hyperplanes. After repeating the same procedure a constant number of times, we know the location of $\overline{\lambda}$ with respect to at least half of the hyperplanes. This procedure requires $\mathcal{O}(\log(m))$ operations on $\mathcal{O}(m)$ processors and no comparisons, except for those made by recursions. The number of recursive calls to dimension $r-1$ is also constant. Leading to a total running time (including recursions) of $2^{\mathcal{O}(r)}\log(m)$ on $\mathcal{O}(m)$ processors and $2^{\mathcal{O}(r)}$ comparisons on $\overline{\lambda}$. We note that the arithmetic operations on $\overline{f_i}$ increase in the recursions, since each operation is made on a linear function in $\overline{f_i}$. However, the cardinality of the support of these functions is always bounded by $2^{\mathcal{O}(r)}$. Hence, this overhead is negligible. □

2.3 Proximity

Our goal is to describe a relaxation of the integer program $P = (x, A, b, c)$ in n-fold structure (★), which can be solved efficiently with the techniques of Section 2.2 and whose ℓ_1 distance between its optimal solution and an optimal integral solution is independent of n. The following lemma shows that the standard relaxation, obtained by relaxing the integrality constraints $x_i \in \mathbb{Z}^{q_i}$ to $x_i \in \mathbb{R}^{q_i}$ for $i = 1, \ldots, n$, is not sufficient.

Lemma 2.4. *There exists a family of block-structured integer programming problems such that the ℓ_∞-distance of an optimal solution $x^\star$ of the standard LP-relaxation to each integer optimal solution $x^\diamond$ is bounded from below by*

$$\|x^\star - x^\diamond\|_\infty = \Omega(n).$$

Proof. We construct a family of problems in which the number n of blocks is odd and each block has two variables. Denote the variables by $(x_i)_1$ and $(x_i)_2$ for $1 \leq i \leq n$. Consider the linear program given by maximizing following objective function

$$\sum_{i=1}^{n} (2+\varepsilon)(x_i)_1 + (3-\varepsilon)(x_i)_2$$

under the constraints given by the polyhedra

$$Q_i = \{x_i \colon 2(x_i)_1 + 3(x_i)_2 = 3, x_i \geq 0\}, \quad i = 1, \ldots, n-1,$$
$$Q_n = \{x_n \colon 2(x_n)_1 + 3(x_n)_2 = 6n\}.$$

Notice that for these constraints, the unique optimal fractional solution is given by $(x_i^\star)_1 = 3/2$, $(x_i^\star)_2 = 0$ for $i = 1, \ldots, n-1$ and $(x_n^\star)_1 = 3n$, $(x_n^\star)_2 = 0$. We now add the linking constraint

$$\sum_{i=1}^{n-1} \big((x_i)_1 + (x_i)_2\big) - \big((x_n)_1 + (x_n)_2\big) = \frac{3(n-1)}{2} - 3n$$

which is satisfied by the optimal fractional solution $x^\star$. It may be checked that the optimal integral solution $x^\diamond$ is defined by setting $(x_i^\diamond)_1 = 0$, $(x_2^\diamond)_1 = 1$ for $i = 1, \ldots, n-1$, and setting $(x_n^\diamond)_1 = 3n - 3(n-1)/2$, $(x_n^\diamond)_2 = n-1$. We obtain that $(x_n^\diamond)_2 - (x_n^\star)_2 = n-1$ giving the claimed lower bound. □

We are interested in a relaxation whose solution is closer to the optimal integer solution. Using the n-fold structure (★) of the problem, we replace the polyhedra Q_i by their integer hull $(Q_i)_I$, removing only fractional solutions but no integer solutions of the standard relaxation. Note that this strengthened relaxation is still a linear program in n-fold structure as described in (★). We now show that for a linear program in n-fold structure (★) with integral polyhedra Q_i, an optimal vertex solution is close to an optimal integer solution.

Before we state and prove the main result of this section, we will need some additional notation. Suppose that P is a linear program in n-fold structure (★) with integral polyhedra Q_i. Let $x_1, \ldots, x_n$ be the partition of the vector of variables x so that x_i corresponds to the columns of B_i and variables of Q_i, for each $i \in \{1, \ldots, n\}$. Partition c into $c_1, \ldots, c_n$ in the same fashion. For each $i \in \{1, \ldots, n\}$, suppose Q_i is the integer hull of the polyhedron $\{x_i \colon D_i x_i = b_i, x_i \geq 0\}$ in dimension q_i. Further, let

$$\gamma := \max_{1 \leq i \leq n} g_1(D_i)$$

be a bound on the ℓ_1 Graver measure of the D_i and

$$\Delta := \max_{1 \leq i \leq n} \{\|B_i\|_\infty, \|D_i\|_\infty\}$$

an upper bound on each entry in the matrices B_i and D_i for $i = 1, \ldots, n$.

Now we are ready to state our main result.

Theorem 2.5. *Given a linear program P in n-fold structure (★) with integral polyhedra Q_i and an optimal vertex solution $x^\star$ of P. Then there exists an optimal integer solution $x^\diamond$ of P with*

$$\|x^\star - x^\diamond\|_1 \leq (r\Delta\gamma)^{\mathcal{O}(r)}.$$

Theorem 2.5 shows that the proximity bound for $\|x^\star - x^\diamond\|_1$ does neither depend on the number of blocks n nor on the ambient dimension of the block polyhedra Q_i, using known bounds on the Graver basis, see e.g. Theorem 1.3.

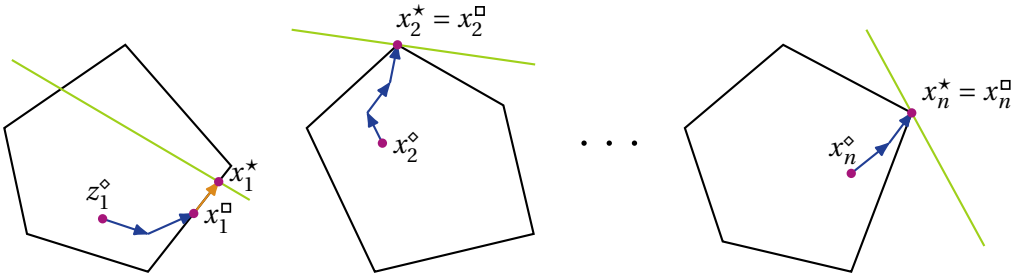

Figure 2.3: Overview of the proof of Theorem 2.5 with $r = 2$ linking constraints.

Next we present an overview of the proof, see Figure 2.3.

- Let $x_i^\square$ be a nearest integer point to $x_i^\star$ with respect to the ℓ_1-norm that lies on the minimal face of Q_i containing $x_i^\star$. In Lemma 2.7 we show that $\|x_i^\star - x_i^\square\|_1 \le r^2\gamma$.
- Let $x^\diamond$ be an optimal integer solution such that $\|x^\square - x^\diamond\|_1$ is minimal. In Theorem 2.10 we show that $\|x^\square - x^\diamond\|_1 \le (r\Delta\gamma)^{\mathcal{O}(r)}$.
- In Lemma 2.6 we show that at most r of the $x_i^\star$ are non-integral and in particular not equal to $x_i^\square$. Theorem 2.5 then follows by applying the above bounds:

$$\begin{aligned}\|x^\star - x^\diamond\|_1 &\le \|x^\star - x^\square\|_1 + \|x^\square - x^\diamond\|_1\\ &\le r^3\gamma + (r\Delta\gamma)^{\mathcal{O}(r)}\\ &\le (r\Delta\gamma)^{\mathcal{O}(r)}.\end{aligned}$$

Lemma 2.6. *Let $x^\star$ be a vertex solution of a linear program P in n-fold structure (★). All but r of the $x_i^\star$ are vertices of the respective Q_i. Thus if the Q_i are integral polyhedra, all but r of the $x_i^\star$ are integral.*

Proof. For the sake of contradiction, suppose that $x_1^\star, \dots, x_{r+1}^\star$ are not vertices of the respective Q_i. Then, for each $i = 1, \dots, r+1$ there exists a non-zero vector $d_i \in \mathbb{R}^{q_i}$ such that $x_i^\star \pm d_i \in Q_i$. Consider the $r+1$ vectors of the form $B_i d_i \in \mathbb{R}^r$. They have to be linearly dependent and thus, there exist $\lambda_1, \dots, \lambda_{r+1} \in \mathbb{R}$ not all zero such that

$$\sum_{i=1}^{r+1} \lambda_i B_i d_i = 0$$

By scaling the λ_i we can suppose that $x_i^\star \pm \lambda_i d_i \in Q_i$. Consider

$$d = (\lambda_1 d_1, \dots, \lambda_{r+1} d_{r+1}, 0, \dots, 0) \in \mathbb{R}^{q_1+\dots+q_n} \setminus \{0\}$$

Then $x^\star + d$ and $x^\star - d$ are both feasible solutions of P and $x^\star = \frac{1}{2}(x^\star + d) + \frac{1}{2}(x^\star - d)$. So $x^\star$ is a convex combination of two feasible points of P and thus not a vertex. □

Lemma 2.7. *Let Q_i be an integer polyhedron of a linear program P in n-fold structure (★) and $x^\star$ an optimal solution to P. Further, let F_i be the minimal face of Q_i containing $x_i^\star$. Then there exists an integer point $x_i^\square \in F_i \cap \mathbb{Z}^{q_i}$ with*

$$\|x_i^\star - x_i^\square\|_1 \leq r^2\gamma.$$

The proof of this result uses standard arguments of polyhedral theory, see, e.g. [Sch98].

Proof. We show that for a k dimensional face F_i, the bound $k^2\gamma$ holds and prove the result by induction. Because of Lemma 2.6 we can then conclude that the dimension k of F_i is bounded by $k \leq r$, leading to the wanted bound. If F_i is 0 dimensional, the assumption clearly holds, henceforth we assume F_i to be $k \geq 1$ dimensional. Let $w \in F_i$ be an arbitrary vertex of F_i and let C_i be the cone

$$C_i = \{\lambda(f - w) \colon f \in F_i, \lambda \geq 0\}.$$

The extreme rays of this cone are elements of the Graver basis D_i representing Q_i, see e.g. [Onn10a, Lemma 3.15]. By Carathéodory's Theorem [Sch98, p. 94] there exist k Graver basis elements $g_1, \ldots, g_k$ such that

$$x_i^\star = w + \sum_{j=1}^{k} \lambda_j g_j.$$

Consider the integer point $z = w + \sum_{j=1}^{k} \lfloor \lambda_j \rfloor g_j \in \mathbb{Z}^{q_i}$. There are two cases. If $z \in F_i$, then, by the triangle inequality, the distance in ℓ_1-norm of $x_i^\star$ to the nearest integer point in F_i is bounded by $k\gamma$. Otherwise, the line segment between $x_i^\star$ and z exits F_i in a lower dimensional face of Q_i contained in F_i. Call $\widetilde{x}_i$ the intersection point. We apply an inductive argument now: there is an integer point on this lower dimensional face that has ℓ_1-distance at most $(k-1)^2\gamma$ from $\widetilde{x}_i$. The bound $k^2\gamma$ follows by applying the triangle inequality and the recursion. □

In the following, we keep the notation of Lemma 2.7 and denote $(x_1^\square, \ldots, x_n^\square) \in \mathbb{Z}^{q_1+\cdots+q_n}$ by $x^\square$. If $x^\diamond \in \mathbb{Z}^{q_1+\cdots+q_n}$ is a feasible integer solution of P, the linear program in n-fold structure (★) with integral Q_i, then each $x_i^\diamond$ is in Q_i. In particular $x_i^\square - x_i^\diamond$ is in the integer kernel of D_i, the matrix representing Q_i. Therefore, there exists a multiset L_i of Graver basis elements of this matrix that are sign-compatible with $x_i^\square - x_i^\diamond$ such that

$$x_i^\square - x_i^\diamond = \sum_{g \in L_i} g.$$

Lemma 2.8. *For each $i = 1, \dots, n$ and each submultiset $H_i \subseteq L_i$ one has*

(i)

$$x_i^{\diamond} + \sum_{h \in H_i} h \in Q_i$$
$$x_i^{\square} - \sum_{h \in H_i} h \in Q_i.$$

(ii) There exists an $\varepsilon > 0$ such that

$$x^{\star} - \varepsilon \sum_{h \in H_i} h \in Q_i.$$

Proof. Assertion (i) follows from standard arguments (see e.g. [Onn10a]) as follows. Let Q_i be the integer hull of $\{x_i : D_i x_i = b_i, x_i \geq 0\}$. Then, one has

$$D_i \left(z_i^{\diamond} + \sum_{h \in H_i} h \right) = b_i,$$
$$D_i \left(x_i^{\square} - \sum_{h \in H_i} h \right) = b_i$$

and both $x^{\diamond} + \sum_{h \in H_i} h$ and $x_i^{\square} - \sum_{h \in H_i} h$ are integral. Since the Graver basis elements of H_i are sign-compatible with $x_i^{\square} - x_i^{\diamond}$ the non-negativity is satisfied by both points as well. Thus both points are feasible integer points of the system $D_i x_i = b_i$, $x_i \geq 0$ which implies that they lie in Q_i.

For the proof of Assertion (ii), let the polyhedron Q_i be described by the inequalities

$$Q_i = \{x \in \mathbb{R}^{q_i} : E_i x_i \leq p_i\}$$

for some integer matrix $E_i \in \mathbb{Z}^{t_i \times q_i}$ and integer vector $p_i \in \mathbb{Z}^{t_i}$. Let $I_i \subseteq \{1, \dots, t_i\}$ be the index set corresponding to the inequalities of $E_i x_i \leq p_i$ satisfied by $x_i^{\star}$ with equality. The inequality description of F_i is obtained from $E_i x_i \leq p_i$ by setting the inequalities indexed by I_i to equality.

Since $x_i^{\square} \in F_i$, all the inequalities indexed by I_i and possibly more are also tight at $x_i^{\square}$. However, by subtracting $\sum_{h \in H_i} h$ from $x_i^{\square}$ one obtains a point of Q_i. Therefore, starting at $x_i^{\star}$ we can move in the direction of $-\sum_{h \in H_i} h$ for some positive amount without leaving Q_i. This means that Assertion (ii) holds. □

Lemma 2.9. *Given a linear program P in n-fold structure (★) with integral polyhedra Q_i, and a feasible integer solution $x^{\square}$ to P as provided by Theorem 2.10. Suppose that $x^{\diamond}$ is an optimal integer solution to P minimizing $\|x^{\square} - x^{\diamond}\|_1$. For each $i = 1, \dots, n$, let L_i be a multiset of Graver*

basis elements of D_i, the matrix representing Q_i such that $h \sqsubseteq x_i^{\square} - x_i^{\diamond}$ for each $h \in L_i$ and

$$x_i^{\square} - x_i^{\diamond} = \sum_{h \in L_i} h$$

Then, each selection of sub-multisets $H_1 \subseteq L_1, \ldots, H_n \subseteq L_n$ satisfies

$$\sum_{i=1}^{n} \sum_{h \in H_i} B_i h \neq 0.$$

Proof. For the sake of contradiction, let $H_1 \subseteq L_1, \ldots, H_n \subseteq L_n$ be a selection of sub-multisets such that

$$\sum_{i=1}^{n} \sum_{h \in H_i} B_i h = 0 \tag{2.7}$$

holds. By Lemma 2.8 (i) one has $x_i^{\diamond} + \sum_{h \in H_i} h \in Q_i$ for each $i = 1, \ldots, n$. Together, this implies that

$$x^{\diamond} + \left(\sum_{h \in H_1} h, \ \ldots, \ \sum_{h \in H_n} h \right) \in \mathrm{Sol}^{\mathbb{Z}}(P) \tag{2.8}$$

is an integer solution of P. Similarly, Lemma 2.8 (ii) implies that there exists an $\varepsilon > 0$ such that

$$x^{\star} - \varepsilon \left(\sum_{h \in H_1} h, \ \ldots, \ \sum_{h \in H_n} h \right) \in \mathrm{Sol}^{\mathbb{R}}(P).$$

is a feasible solution of P. Since $x^{\diamond}$ and $x^{\star}$ were optimal integral and fractional solutions of P respectively, this implies that

$$\sum_{i=1}^{n} c_i^{\intercal} \sum_{h \in H_i} h = 0,$$

and thus the objective values of $z^{\diamond}$ and (2.8) are the same. This however is a contradiction to the minimality of $\|x^{\square} - x^{\diamond}\|_1$, as the optimal integer solution (2.8) is closer to $x^{\square}$. □

As in the proximity result presented in [EW20a] we make use of the Steinitz lemma (Theorem 1.2) to bound, in this case $\|x^{\square} - x^{\diamond}\|_1$. The proof follows closely the ideas in [EHK18].

Theorem 2.10. *Given a linear program P in liked block form (★) with integer polyhedra Q_i and a feasible integer solution $x^{\square}$ to P as provided by Theorem 2.10. Suppose that $x^{\diamond}$ is an optimal integer solution to P minimizing $\|x^{\square} - x^{\diamond}\|_1$. Then it holds that*

$$\|x^{\square} - x^{\diamond}\|_1 \leq (2r\Delta\gamma + 1)^{r+4}.$$

Proof. We use the notation of the statement of Lemma 2.9. Denote the matrix $(B_1, \ldots, B_n) \in$

$\mathbb{Z}^{r\times(q_1+\cdots+q_n)}$ by B. For an optimal fractional solution $x^\star$ of P, we have

$$\begin{aligned} 0 = B(x^\star - x^\diamond) &= B(x^\star - x^\square) + B(x^\square - x^\diamond) \\ &= B(x^\star - x^\square) + \sum_{i=1}^{n} \sum_{h\in L_i} B_i h. \end{aligned}$$

Since the ℓ_1-norm of each Graver basis element $h \in L_i$ is assumed to be bounded by γ, the ℓ_∞-norm of each $B_i h$ is bounded by $\Delta\gamma$. The Steinitz Lemma (Theorem 1.2) implies that the set

$$\{B_i h\colon h \in L_i,\, i = 1,\ldots,n\} \subseteq \mathbb{Z}^r$$

can be permuted in such a way such that the distance in the ℓ_∞-norm of each prefix sum to the line-segment spanned by 0 and $B(x^\star - x^\square)$ is bounded by $\Delta\gamma$ times the dimension r, i.e., by $R := r\Delta\gamma$. Note that each such prefix sum is an integer point. The number of integer points within ℓ_∞-distance R to the line segment spanned by 0 and $B(x^\star - x^\square)$ is at most

$$(\|B(x^\star - x^\square)\|_1 + 1)\cdot(2R+1)^r.$$

Where

$$\begin{aligned} \|B(x^\star - y^\square)\|_1 &\le \sum_{i=1}^{n} \|B_i(x_i^\star - x_i^\square)\|_1 \\ &\le r\cdot\Delta r^3\gamma. \end{aligned}$$

The last inequality follows from Lemmas 2.6 and 2.7. Thus number of integer points is bounded by

$$(r^4\Delta\gamma + 1)\cdot(2r\Delta\gamma+1)^r$$

If $\sum_{i=1}^{n}|L_i|$ is larger than this bound, then there exist two equal prefix sums in the Steinitz rearrangement of the vectors. This yields sub-multisets $H_1 \subseteq L_1,\ldots,H_n \subseteq L_n$ consisting of the vectors in between these equal prefix sums, for which one has

$$\sum_{i=1}^{n}\sum_{h\in H_i} B_i h = 0.$$

By Lemma 2.9 this is not possible, which implies that

$$\|x^\square - z^\diamond\|_1 \le (r^4\Delta\gamma+1)(2r\Delta\gamma+1)^r = (r\Delta\gamma)^{\mathcal{O}(r)}. \qquad \square$$

Applying the triangle inequality $\|x^\star - x^\diamond\|_1 \le \|x^\star - x^\square\|_1 + \|x^\square - x^\diamond\|_1$ then proves Theorem 2.5.

2.4 A Dynamic Program

Let $x^\star$ be an optimal vertex solution of a linear program P in n-fold structure ($\bigstar$) with integral polyhedra Q_i. We now describe a dynamic programming approach which computes an optimal integer solution of P. Note that this approach closely resembles a method from [EHK+19].

First we observe that by Theorem 2.5 it is sufficient to search for an optimal integral solution $x^\diamond$ of P which satisfies $\|x^\star - x^\diamond\|_1 \leq (r\Delta\gamma)^{\mathcal{O}(r)}$. For simplicity of presentation, assume n to be a power of 2. We now construct a binary tree with n leaves. Let $v_{j,k}$ be the $(j+1)$-th node on the $(k-1)$-th layer from the bottom. Each node $v_{j,k}$ corresponds to the interval of the form $[j2^k+1,(j+1)2^k]$. In particular, the root node $v_{0,\log(n)}$ corresponds to the interval $[1,n]$. Starting from the leaves, we compute solutions $z_{j2^k+1},\ldots,z_{(j+1)2^k}$ for each $v_{j,k}$ using the solutions its two children.

An integer vector z with $\|x^\star - z\|_1 \leq (r\Delta\gamma)^{\mathcal{O}(r)}$ also satisfies

$$\left\| \sum_{i=j2^k+1}^{(j+1)2^k} B_i \left(x_i^\star - z_i\right) \right\|_\infty \leq (r\Delta\gamma)^{\mathcal{O}(r)}$$

for each $v_{j,k}$. Which implies the following bounds

$$\sum_{i=j2^k+1}^{(j+1)2^k} B_i x_i^\star - (r\Delta\gamma)^{\mathcal{O}(r)}\mathbf{1} \leq \sum_{i=j2^k+1}^{(j+1)2^k} B_i z_i \leq \sum_{i=j2^k+1}^{(j+1)2^k} B_i x_i^\star + (r\Delta\gamma)^{\mathcal{O}(r)}\mathbf{1},$$

where $\mathbf{1}$ is the all-ones vector. Let $S_{j,k} \subseteq \mathbb{Z}^r$ be the set of integer vectors $d \in \mathbb{Z}^r$ which satisfy the above bounds, namely

$$\sum_{i=j2^k+1}^{(j+1)2^k} B_i x_i^\star - (r\Delta\gamma)^{\mathcal{O}(r)}\mathbf{1} \leq d \leq \sum_{i=j2^k+1}^{(j+1)2^k} B_i x_i^\star + (r\Delta\gamma)^{\mathcal{O}(r)}\mathbf{1},$$

Clearly, the cardinality of each $S_{j,k}$ satisfies

$$|S_{j,k}| \leq (r\Delta\gamma)^{\mathcal{O}(r^2)}.$$

Now, we generate all these sets $S_{j,k}$ and compute for each node $v_{j,k}$ and each $d \in S_{j,k}$ a partial solution $z_{j2^k+1},\ldots,z_{(j+1)2^k}$ bottom-up. The optimal integral solution $x^\diamond$ will satisfy the following condition: If d is the the correct guess, that is to say,

$$\sum_{i=j2^k+1}^{(j+1)2^k} B_i x_i^\diamond = d,$$

then our computed partial solution z satisfies

$$\sum_{i=j2^k+1}^{(j+1)2^k} c_i^\intercal z_i \geq \sum_{i=j2^k+1}^{(j+1)2^k} c_i^\intercal x_i^\diamond.$$

For the leaves, we solve the individual block using a presumed algorithm we have for them, that is, we compute the optimal solution to

$$\max\{c_i^\intercal z_i \colon B_i z_i = d,\, z_i \in Q_i,\, z_i \in \mathbb{Z}^{q_i}_{\geq 0}\}. \tag{2.9}$$

For an inner node $v_{j,k}$ with children $v_{2j,k-1}$ and $v_{2j+1,k-1}$ we consider all $d' \in S_{2j,k-1}$ and $d'' \in S_{2j+1,k-1}$ with $d = d' + d''$ and take the best solution among all combinations. Indeed, if d is the correct guess, then the correct guesses d' and d'' are among these candidates.

After computing all solutions for the root, we obtain an optimal solution by taking the solution for $b_0 \in S_{0,\log(n)}$.

This algorithm leads to the following theorem.

Theorem 2.11. *Given a linear program P in n-fold structure (★) with integral polyhedra Q_i and an optimal vertex solution $x^\star$ to P. There is a linear algorithm rounding $x^\star$ to an optimal integral solution of P in*

$$\mathcal{O}\big((r^2\log(r\Delta\gamma) + T_{\max})\log(n)\big)$$

operations on each of $(r\Delta\gamma)^{\mathcal{O}(r^2)}R$ processors. Here $R = \sum_i R_i$ is the sum of the processor requirements of the algorithms for the block problems (2.9) and $T_{\max}$ is the maximum number of operations used to solve any of them.

2.5 Applications

In this section, we derive running time bounds for concrete cases of integer linear programs in n-fold structure (★) using the theorems from the earlier sections. All algorithms in this section are linear algorithms in the sense of the definition Section 1.1.3.

Our base case is the trivial integer linear program

$$\max\{cx \colon Ax = b, 0 \leq x \leq u, x \in \mathbb{Z}\}, \tag{2.10}$$

where $c \in \mathbb{R}, A \in \mathbb{Z}^{s\times 1}$, $b \in \mathbb{Z}^s$, and $u \in \mathbb{Z} \cup \{\infty\}$. Clearly this problem can be solved in $\mathcal{O}(s)$ operations.

Corollary 2.12. *Let $c \in \mathbb{R}^q, A \in \mathbb{Z}^{s\times q}$, $b \in \mathbb{Z}^s$. Consider the integer linear program $P = (x, A, b, c)$ in the form (♠) with upper bounds $u \in \mathbb{Z}_{\geq 0} \cup \{\infty\}$. An optimal integral solution to P can be found in*

$$(s\|A\|_\infty)^{\mathcal{O}(s^2)} q^{1+o(1)}$$

arithmetic operations, or alternatively in $2^{\mathcal{O}(s^2)}\log(\|A\|_\infty)\log^{s+1}(q)$ *arithmetic operations on* $(s\|A\|_\infty)^{\mathcal{O}(s^2)}q$ *processors.*

Proof. Using Theorem 2.1 with $Q_i = \mathbb{R}$ for all i, we can solve the relaxation in $2^{\mathcal{O}(s^2)}\log^{s+1}(q)$ operations on q processors. Then, with an upper bound of $\mathcal{O}(\|A\|_\infty)$ on the ℓ_1-Graver measure of (2.10), we can use Theorem 2.11 to derive an optimal integral solution to P in $\mathcal{O}\big((s^2\log(s\|A\|_\infty))\log(q)\big)$ operations on $(s\|A\|_\infty)^{\mathcal{O}(s^2)}q$ processors. The sequential running time follows with the fact that $\log^k(n) \le k^{2k}n^{o(1)}$. □

Corollary 2.13. *Consider the integer linear program* $P = (x, A, b, c)$ *in the form* (♠) *with upper bounds* $u \in \mathbb{Z}_{\ge 0}\cup\{\infty\}$*, where* A *is an* n*-fold matrix as in* (1.2)*. An optimal integral solution to* P *can be found in*

$$2^{\mathcal{O}(rs^2)}(rs\|A\|_\infty)^{\mathcal{O}(r^2s+s^2)}(nq)^{1+o(1)}$$

arithmetic operations, or alternatively in $2^{\mathcal{O}(r^2+rs^2)}\log^{\mathcal{O}(rs)}(nq\|A\|_\infty)$ *arithmetic operations on* $(rs\|A\|_\infty)^{\mathcal{O}(r^2s+s^2)}nq$ *processors.*

Proof. We start by solving the relaxation of P given by the linear program in n-fold structure (★) where each block constraint $x_i \in \{x_i\colon D_ix_i = b_i\}$ is replaced by its integral hull. Using Theorem 2.1 we can solve this relaxation in $2^{\mathcal{O}(r^2+rs^2)}\log^{r+1}(\|A\|_\infty)\log^{\mathcal{O}(rs)}(q)\log^{r+1}(nq)$ operations on $(s\|A\|_\infty)^{\mathcal{O}(s^2)}nq$ processors. From Theorem 1.3 we derive that the Graver measure of D_i is at most $(2s\|A\|_\infty+1)^s$. Then using Theorem 2.11, we derive an optimal integer solution to P in $2^{\mathcal{O}(r+s)^2}\log^{r+s+1}(q)\log(rs\|A\|_\infty)\log(n)$ operations on $(rs\|A\|_\infty)^{\mathcal{O}(r^2s+s^2)}nq$ processors. □

Another parameter under consideration for integer programming is the dual treedepth of the constraint matrix A. Though sometimes the running time is analyzed in a more fine-grained way by introducing additional parameters, the best running time in literature purely dependent $d = \mathsf{td}_{\mathsf{D}}(A)$ and $\|A\|_\infty$ have a parameter dependency of $\|A\|_\infty^{\mathcal{O}(d2^d)}$ [KPW20].

Corollary 2.14. *Let* $P = (x, A, b, c)$ *be a linear program in the form* (♠) *on* t *variables. Suppose* A *has dual depth* $\mathsf{depth}_{\mathsf{D}}(A) = d$*. Then, an optimal integral solution to* P *can be found in*

$$d^{\mathcal{O}(d^22^d)}\|A\|_\infty^{\mathcal{O}(2^d)}t^{1+o(1)}$$

arithmetic operations, or alternatively in $\log^{\mathcal{O}(d2^d)}(d\|A\|_\infty t)$ *parallel arithmetic operations on* $d^{\mathcal{O}(d^3)}\|A\|_\infty^{\mathcal{O}(2^d)}t$ *processors.*

Proof. We show the result by induction on the dual depth for a slightly more general problem. Split the constraint matrix into $A = \binom{B}{D}$ such that $B \in \mathbb{Z}^{r\times t}$ for some $r \ge 2$ and $D \in \mathbb{Z}^{h\times t}$. The requirement $r \ge 2$ is to avoid some corner cases in the proof. Denote the dual depth of D by $d = \mathsf{depth}_{\mathsf{D}}(D)$. The induction hypothesis is that in this setting, an optimal integral solution to

P can be found in

$$T(d,r,t) := \log^{(r+1)2^d \sum_{i=0}^{d}(3Mr+i)/2^i}(r\|A\|_\infty t) = \log^{\mathcal{O}(r^2 2^d)}(\|A\|_\infty t)$$

arithmetic operations on

$$R(d,r,t) := \|A\|_\infty^{M2^d \cdot \sum_{i=0}^{\infty}(r+i)^3/2^i} t = \|A\|_\infty^{\mathcal{O}(r^3 2^d)} t$$

processors. Here, M is a sufficiently large constant. In particular, we suppose that M is larger than any hidden constant appearing in the $\mathcal{O}$-notation of Theorems 2.1 and 2.11. To get the wanted result for $r = 0$ we can simply solve the problem with $r = 2$ and B the trivial zero matrix.

As a base case, we consider $\binom{B}{D}$ with $\mathsf{depth}_{\mathsf{D}}(D) = 0$. By Corollary 2.12 the induction hypothesis holds for any $r \geq 1$.

Now suppose that A is split into $\binom{B}{D}$ with $d = \mathsf{depth}_{\mathsf{D}}(D) \geq 1$. Compute the block decomposition of D into the $n \geq 1$ blocks $D_1, \ldots, D_n$ and split B into according blocks $B_1, \ldots, B_n$. For $i = 1 \ldots, n$, say that q_i is the number of columns in D_i and B_i. Consider the linear program P in n-fold structure (★) where the linking constraints are given by $Bx = b_0$ and the block constraints $x_i \in Q_i$ by setting Q_i to be the integer hull of $\{x_i : D_i x_i = b_i\}$. To find an optimal integral solution of P we first use Theorem 2.1 to find an optimal fractional solution of P and then derive an optimal integral solution with Theorem 2.11.

Note that for $i = 1, \ldots, n$ the matrix $\binom{B_i}{D_i}$ can also be seen as $\binom{B'_i}{D'_i}$ with B'_i having $r+1$ rows and $\mathsf{depth}_{\mathsf{D}}(D'_i) \leq d-1$. Similarly, D_i can be seen as D'_i with an additional row on the top. First we analyze the processor usage of these two steps. By Theorem 2.1, solving the relaxation requires

$$\sum_{i=1}^{n} R(d-1,2,q_i) \leq \|A\|_\infty^{M2^{d-1}\sum_{i=0}^{\infty}(2+i)^3/2^i}(q_1+\cdots+q_n) \leq R(d,2,h) \leq R(d,r,h)$$

processors. From Theorem 1.6 we have $g_1(D_i) \leq (2\|A\|_\infty + 1)^{2^d-1}$ for each $i = 1, \ldots, n$. This implies that $r\|A\|_\infty \max\{g_1(D_i) : 1 \leq i \leq n\}$ is bounded by $\|A\|_\infty^{r2^d}$. Then, by Theorem 2.11 for deriving an optimal integral solution we require at most

$$\begin{aligned}
&\Big(r\|A\|_\infty \max_{1\leq i\leq n}\{g_1(D_i)\}\Big)^{Mr^2} \sum_{i=1}^{n} R(d-1,r+1,q_i) \\
&\qquad \leq \|A\|_\infty^{M2^d r^3} \cdot \|A\|_\infty^{M2^{d-1}\sum_{i=0}^{\infty}(r+1+i)^3/2^i}(q_1+\cdots+q_n) \\
&\qquad = \|A\|_\infty^{M2^d r^3} \cdot \|A\|_\infty^{M2^d \sum_{i=1}^{\infty}(r+i)^3/2^i} t \\
&\qquad = \|A\|_\infty^{M2^d \sum_{i=0}^{\infty}(r+i)^2/2^i} t = R(d,r,t).
\end{aligned}$$

processors. This proves the closed formula for the number of processors.

We now proceed to prove the number of arithmetic operations required. Since this depends mildly on the processor usage, we will first upper bound the number of processors by

$$R(d,2,t) = \|A\|_\infty^{M2^d \sum_{i=0}^{\infty}(2+i)^3/2^i} t \leq \|A\|_\infty^{M^2 2^d} t,$$

which holds for sufficiently large M. By Theorem 2.1, the number of arithmetic operations required to solve the relaxation is at most

$$\begin{aligned}
&2^{Mr^2}\left(\max_{1\leq i\leq n}\left\{T(d-1,2,q_i)\right\}\cdot\log\left(R(d-1,1,t)\right)\right)^{r+1}\\
&\qquad\leq \frac{1}{2}\left(2^{Mr}\cdot T(d-1,r,t)\cdot\log\left(R(d-1,2,t)\right)\right)^{r+1}\\
&\qquad\leq \frac{1}{2}\left(2^{Mr}\log^{2^d\sum_{i=0}^{d-1}(3Mr+i)/2^i}(\|A\|_\infty t)\cdot M^2 2^d\log(\|A\|_\infty t)\right)^{r+1}\\
&\qquad\leq \frac{1}{2}\left(\log^{2^d\sum_{i=0}^{d-1}(3Mr+i)/2^i}(\|A\|_\infty t)\cdot\log^{3Mr+d}(\|A\|_\infty t)\right)^{r+1}\\
&\qquad\leq \frac{1}{2}\left(\log^{2^d\sum_{i=0}^{d}(3Mr+i)/2^i}(\|A\|_\infty h)\right)^{r+1} \leq \frac{1}{2}T(d,r,t).
\end{aligned}$$

By Theorem 2.11 the number of arithmetic operations to deduce an integer solution is at most

$$\begin{aligned}
&M\big(r^2\log(r\|A\|_\infty\max_{1\leq i\leq n}\{g_1(D_i)\})+\max_{1\leq 1\leq n}\{T(d-1,r+1,q_i)\}\big)\log(n)\\
&\qquad\leq M\big(r^3 2^d\log(\|A\|_\infty)+\log^{(r+2)2^{d-1}\sum_{i=0}^{d-1}(3M(r+1)+i)/2^i}(\|A\|_\infty t)\big)\log(n)\\
&\qquad\leq \frac{1}{2}\big(\log^{2Mr+d}(\|A\|_\infty t)+\log^{M}(\|A\|_\infty t)\cdot\log^{(r+1)2^d\sum_{i=0}^{d-1}(3Mr+i)/2^i}(\|A\|_\infty t)\big)\\
&\qquad\leq \frac{1}{2}\log^{(r+1)2^d\sum_{i=0}^{d}(3Mr+i)/2^i}(\|A\|_\infty t)=\frac{1}{2}T(d,r,t),
\end{aligned}$$

where in the second inequality we use that $(r+2)(r+1)\leq 2(r+1)r$ since $r\geq 2$. □

Continuous variables

We now consider the continuous cases of linear programs in n-fold structure. Although linear programming has polynomial algorithms, no strongly polynomial algorithm is known for the general case and there is no PRAM algorithm running in polylogarithmic time on a polynomial number of processors unless NC = P [GHR+95]. For the following corollaries, we only use Theorem 2.1.

The continuous variant of the base case (2.10) is easily solvable in $O(s)$ operations.

Corollary 2.15. *Let $c \in \mathbb{R}^q$, $A \in \mathbb{Z}^{s \times q}$, $b \in \mathbb{Z}^s$. Consider the linear program $P = (x, A, b, c)$ in the form* (♠) *with upper bounds $u \in \mathbb{Z}_{\geq 0} \cup \{\infty\}$. An optimal fractional solution to P can be found in*

$$2^{\mathcal{O}(s^2)} q^{1+o(1)}$$

operations, or alternatively in $2^{\mathcal{O}(s^2)} \log^s(q)$ operations on each of q processors.

Corollary 2.16. *Consider the linear program $P = (x, A, b, c)$ in the form* (♠) *with upper bounds $u \in \mathbb{Z}_{\geq 0} \cup \{\infty\}$, where A is an n-fold matrix as in* (1.2)*. An optimal fractional solution to P can be found in*

$$2^{\mathcal{O}(r^2 + rs^2)} (nq)^{1+o(1)}$$

arithmetic operations, or alternatively in $2^{\mathcal{O}(r^2+rs^2)} \log^{\mathcal{O}(rs)}(nq)$ arithmetic operations on nq processors.

Corollary 2.17. *Let $P = (x, A, b, c)$ be a linear program in the form* (♠) *on t variables. Suppose A has dual depth $\mathsf{depth}_{\mathsf{D}}(A) = d$. Then, an optimal fractional solution to P can be found in*

$$2^{\mathcal{O}(d2^d)} t^{1+o(1)}$$

arithmetic operations, or alternatively with $2^{O(2^d)} \log(h)^{2^{d+1}-2}$ arithmetic operations on t processors.

Proof. We induct on the dual depth $d = \mathsf{depth}_{\mathsf{D}}(A)$. For $d = 1$, A consists of a single constraint and Corollary 2.15 with $s = 1$ gives a running time of $2^{\mathcal{O}(1)} \log(t)$ on each of t processors. For $\mathsf{depth}_{\mathsf{D}}(A) = d \geq 2$, the matrix A is in n-fold structure with $r = 1$ and for all polyhedra $Q_i = \{x_i : D_i x_i = b_i\}$ the dual depth of D_i is at most $d - 1$. Using the recursion hypothesis and Theorem 2.1 we get the desired result. □

3 Efficient Sequential and Parallel Algorithms for Multistage Stochastic Integer Programming using Proximity

This chapter contains a overworked version of [CEP+21] which is joint work with Friedrich Eisenbrand, Michał Pilipczuk, Moritz Venzin and Robert Weismantel.

3.1 Introduction

We consider integer programming problems where the constraint matrix is *2-stage stochastic* or *multistage stochastic* (see Section 1.2.1). Our aim is to exploit specific structural properties of the constraint matrix to develop efficient algorithms. Recall that A is two-stage stochastic or (p,q)*-stochastic* if after deleting the first p columns the matrix can be decomposed into blocks with at most q columns each. *Multistage stochastic* integer programming is a generalization of the two-stage variant obtained by allowing further recursive levels in the block structure. This recursive structure can be explained by the primal treedepth of a matrix.

The goal is to get a fixed parameter tractable algorithms for integer programming parameterized by $\mathsf{td}_{\mathsf{P}}(A)$ and $\|A\|_\infty$. A weak fixed parameter tractable algorithm for the considered parameterization follows implicitly from the work of Aschenbrenner and Hemmecke [AH07]. The first to explicitly observe the applicability of primal treedepth to the design of fpt algorithms for integer programming were Ganian and Ordyniak [GO18], although their algorithm also treats $\|b\|_\infty$ as a parameter besides $\mathsf{td}_{\mathsf{P}}(A)$ and $\|A\|_\infty$. A major development was brought by Koutecký et al. [KLO18], who gave the first strong fixed parameter tractable algorithm, with running time $f(\mathsf{td}_{\mathsf{P}}(A), \|A\|_\infty) \cdot t^3 \log^2(t)$, where t is the number of columns of A. We refer the reader to the joint manuscript of Eisenbrand et al. [EHK+19], which comprehensively presents the recent developments in the theory of block-structured integer programming. Corollaries 93 and 96 there discuss the cases of 2-stage stochastic and multistage stochastic integer programming.

Our contribution

We advance the state-of-the-art of fixed parameter tractable algorithms for 2-stage stochastic and multistage stochastic integer programming problems by proving the following. The number of rows of the constraint matrix A is denoted by d, and the primal treedepth of A is denoted by t.

(i) We give an $f(d, \|A\|_\infty) \cdot t \log^{\mathcal{O}(2^d)}(t)$-time algorithm for integer programmingin the strong sense, where f is a computable function (Theorem 3.5). This improves upon the currently fastest strong fixed parameter tractable algorithm by Koutecký et al. [KLO18] that is nearly cubic in n.

(ii) We provide a $2^{((p+q)\|A\|_\infty)^{\mathcal{O}(p(p+q))}} \cdot t \log^{\mathcal{O}(pq)}(t)$-time algorithm for (p,q)-stochastic integer programming. This improves upon the currently fastest algorithm that runs in time $2^{(2\|A\|_\infty)^{\mathcal{O}(p^2q+pq^2)}} \cdot t^{\mathcal{O}(1)}$ [EHK+19, Kle22], both in terms of the parametric dependence and in terms of the polynomial factor in the running time.

The algorithmic contributions (i) and (ii) rely on the following proximity result for integer programs with low primal treedepth. This result can be regarded as the core contribution of this chapter, and we believe that it uncovers an important connection between the primal treedepth of A and the solution space of (♠).

(iii) (Proximity) For each optimal solution $x^\star$ to the linear relaxation of (♠), there is an optimal integral solution $x^\diamond$ such that $\|x^\diamond - x^\star\|_\infty$ is bounded by a computable function of $\mathsf{td}_\mathsf{P}(A)$ and $\|A\|_\infty$. This is proved in Lemma 3.1.

This proximity result provides a very simple template for designing fixed parameter tractable algorithms for multistage stochastic integer linear programming. Let us explain it for the case of (p,q)-stochastic integer linear programming. After one has found an optimal fractional solution $x^\star$ of the linear relaxation of (♠), one only has to enumerate the $(2 \cdot f(d, \|A\|_\infty) + 1)^p$ many possible integer assignments for the p stage 1 variables that are within the allowed distance, where $f(d, \|A\|_\infty)$ is the proximity bound provided by (iii). For each of these assignments, the integer program (♠) decomposes into $\mathcal{O}(t)$ independent sub-problems, each with at most s variables. This results in a $f(p,q,\|A\|_\infty) \cdot t$-time algorithm (excluding the time needed for solving the linear relaxation). For multistage stochastic integer programming, this argument has to be applied recursively.

As for solving the linear relaxation, note that to obtain results (i) and (ii) we need to be able to solve linear programs with low primal treedepth in near-linear fixed parameter tractable time. This is a non-trivial task. We rely on a result of Chapter 2 which shows that the dual of (♠) can be solved in time $t \log^{\mathcal{O}(2^d)}(t)$. By linear programming duality, this provides an algorithm for finding the optimum value of the linear relaxation of (♠) within the required

complexity, but for applying the approach presented above, we need to actually compute an optimum fractional solution to (♠). While it is likely that the approach in Chapter 2 can be modified so that it outputs such a solution as well, we give a self-contained argument using complementary slackness that applies the results in Chapter 2 only as a black-box.

The approach in Chapter 2 is parallelizable, in the sense that the algorithm for solving the linear relaxation of (♠) can be implemented on t processors so that the running time is $\log^{\mathcal{O}(2^d)}(t)$, assuming the constraint matrix A is suitably organized on input. As the simple enumeration technique sketched above also can be easily applied in parallel, we obtain the following counterpart of (i) and (ii).

(iv) In both cases (i) and (ii), we provide algorithms running in time $f(d,\|A\|_\infty)\cdot\log^{\mathcal{O}(2^d)}(t)$ and $2^{((p+q)\|A\|_\infty)^{\mathcal{O}(p(p+q))}}\cdot\log^{\mathcal{O}(pq)}(t)$, respectively, on t processors. For (i) we assume that the constraint matrix is suitably organized on input. .

The proof of (iii) relies on a structural lemma of Klein [Kle22], which allows us to bound the ℓ_∞-norm of the projections of Graver basis elements to the space of stage 1 variables. In the language of convex geometry, the lemma says the following: if the intersection of integer cones $C_1,\dots,C_m\subseteq\mathbb{Z}^d$ is non-empty and for $i=1,\dots,m$ each generator of C_i has ℓ_∞-norm at most Δ, then there is an integer vector $b\in\bigcap_{i=1}^m C_i$ satisfying $\|b\|_\infty\le 2^{\mathcal{O}(d\Delta)^d}$. In fact, the original bound of Klein [Kle22] is doubly exponential in d^2. We provide a new proof improving this to a doubly exponential dependence on $d\log d$ only. A direct implication of this is the improvement in the parametric factor reported in (ii). We also consider some further relaxations of the statement which are important in the proof of (iii).

Further related work

The algorithm proposed by Koutecký et al. [KLO18] for multistage stochastic programming relies on *iterative augmentation* using elements of the *Graver basis*, see also [LHOW08, Onn10b, HOR13]. The idea of the augmentation framework is to iteratively improve the current solution along directions given by Graver basis elements. For multistage stochastic programs, the ℓ_∞-norm of a Graver basis element of the constraint matrix A is bounded by $g(\mathsf{td}_{\mathsf{P}}(A),\|A\|_\infty)$ for some computable function g. This makes the iterative augmentation applicable in this setting. However, this framework is inherently sequential.

Let us note that Koutecký et al. [KLO18] relied on bounds on the function g above due to Aschenbrenner and Hemmecke [AH07], which only guaranteed computability. Better and explicit bounds on g were later given by Klein [Kle22] and further improved by Klein and Reuter [KR22]. Roughly speaking, the first proof of Klein [Kle22] shows that $g(d,a)$ is at most d-fold exponential, which was then improved to $2^{(d\|A\|_\infty)^{\mathcal{O}(d^{3d+1})}}$ by Klein and Reuter [KR22].

On a related note, Jansen et al. [JKL21] gave a $2^{2^{o(s)}}\cdot t^{\mathcal{O}(1)}$ lower bound for $(1,s)$-stochastic IPs in which all coefficients of the constraint matrix are bounded by a constant in absolute

values. This is assuming the Exponential Time Hypothesis. Thus, for $(1, s)$-stochastic integer programming with bounded coefficients, our result (ii) is almost tight.

While robust and elegant, iterative augmentation requires further arguments to accelerate the convergence to an optimal solution in order to guarantee a good running time. As presented in [EHK+19], to overcome this issue one can either involve the bitlength of the input numbers in measuring the complexity, thus resorting to weak fixed parameter tractable algorithms, or reduce this bitlength using technical arguments. For instance, an integer program (♠) can be solved in time $f(d, \|A\|_\infty) \cdot t^{1+o(1)} \cdot \log^d(\|c\|_\infty)$, where $d = \mathrm{td}_{\mathsf{P}}(A)$. However, to the best of our knowledge, before this work there was no strong fixed parameter tractable algorithm achieving sub-quadratic running time dependence on t, even in the setting of two-stage stochastic integer programming.

We would also like to note that, Dong et al. [DLY21] proposed a sophisticated interior-point algorithm to approximately solve linear programs whose constraint matrices have primal treewidth d in time $\widetilde{\mathcal{O}}(td^2 \cdot \log(1/\varepsilon))$, where ε is an accuracy parameter. Note here that the primal treewidth is bounded by the primal treedepth, so this algorithm in principle could be applied to the linear relaxation of (♠). There are two caveats: the algorithm of [DLY21] provides only an approximate solution, and it is unclear whether it can be parallelized. For these reasons we rely on the algorithm in Chapter 2 through dualization, but exploring the applicability of the work of Dong et al. [DLY21] in our context is an exciting perspective for future work.

Structure of the chapter

We outline our algorithms in Section 3.2, deferring the implementation of the necessary ingredients to the later sections. In Section 3.3 we present the new, stronger proof of the structural result of Klein [Kle22], while in Section 3.4 we use it to establish the proximity result, that is item (iii). Solving the linear relaxation is discussed in Section 3.5.

3.2 Algorithms

As discussed, our algorithms for stochastic integer programming follow from a combination of two ingredients: proximity results for stochastic integer programs, and algorithms for solving their linear relaxations. These ingredients will be proven in the subsequent sections, while in this section, we state them formally and argue how the claimed results follow.

As for proximity, we show that in stochastic integer linear programs, for every optimal fractional solution there is always an optimal integral solution that is close, with respect to the ℓ_∞-norm. Precisely, the following results are proven in Section 3.4.

Lemma 3.1. *There exists a computable function $f\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ with the following property. Suppose $P = (x, A, b, c)$ is a linear program in the form (♠). Then, for every optimal fractional*

solution $x^\star \in \mathrm{Sol}^{\mathbb{R}}(P)$*, there exists an optimal integral solution* $x^\diamond \in \mathrm{Sol}^{\mathbb{Z}}(P)$ *satisfying*

$$\|x^\diamond - x^\star\|_\infty \leq f(\mathsf{depth}_{\mathsf{P}}(A), \|A\|_\infty).$$

Lemma 3.2. *Suppose that* $P = (x, A, b, c)$ *is a linear program in the form* (♠)*, where* A *is* (p, q)*-stochastic for some positive integers* p *and* q*. Then, for every optimal fractional solution* $x^\star \in \mathrm{Sol}^{\mathbb{R}}(P)$*, there exists an optimal integral solution* $x^\diamond \in \mathrm{Sol}^{\mathbb{Z}}(P)$ *satisfying*

$$\|x^\diamond - x^\star\|_\infty \leq 2^{\mathcal{O}(p(p+q)\|A\|_\infty)^{p(p+q)}}.$$

Note that an (p, q)-stochastic matrix has primal depth at most $p + q$, so Lemma 3.2 could be seen as a special case of Lemma 3.1. However, Lemma 3.2 provides a better upper bound on the proximity of (p, q)-stochastic matrices.

As for solving linear relaxations, in Section 3.5 we show the following.

Lemma 3.3. *Suppose we are given a linear program* $P = (x, A, b, c)$ *in the form* (♠)*. Let* t *be the number of rows of* A*. Then, one can using* t *processors and requiring* $\log^{\mathcal{O}(2^{\mathsf{depth}_{\mathsf{P}}(A)})} t$ *operations on each processor, compute an optimal fractional solution to* P*.*

Lemma 3.4. *Suppose we are given a* (p, q)*-stochastic linear program* $P = (x, A, b, c)$ *in the form* (♠)*. Let* t *be the number of rows of* A*. Then, one can using* t *processors and requiring* $2^{\mathcal{O}(p^2+pq^2)} \cdot \log^{\mathcal{O}(pq)} t$ *operations on each processor, compute an optimal fractional solution to* P*.*

Again, Lemma 3.4 differs from Lemma 3.3 by considering a more restricted class of matrices, i.e. (p, q)-stochastic, but providing better complexity bounds.

We now combine the tools presented above to show the following theorems.

Theorem 3.5. *There is a computable function* $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ *such that the following holds. Suppose we are given a linear program* $P = (x, A, b, c)$ *in the form* (♠)*. Let* t *be the number of rows of* A*. Then, one can using* t *processors and requiring* $f(\mathsf{depth}_{\mathsf{P}}(A), \|A\|_\infty) \cdot \log^{\mathcal{O}(2^{\mathsf{depth}_{\mathsf{P}}(A)})} t$ *operations on each processor, compute an optimal integral solution to* P*.*

Proof. Consider the following recursive algorithm.

Each recursive step of the algorithm depends on whether the constraint matrix A is block decomposable or not. By Lemma 1.1, this can be checked in time $\mathcal{O}\big(\mathsf{depth}_{\mathsf{P}}(A) \log(\mathsf{depth}_{\mathsf{P}}(A)\, t)\big)$.

Suppose first that A is block decomposable. Say that it decomposes into the blocks $D_1, \ldots, D_n$ ($n \geq 2$) which are computed while checking for the block decomposition. Then, P can be decomposed into independent integer linear programs $P_1, \ldots, P_n$ with constraint matrices $D_1, \ldots, D_n$ so that an optimal integral solution to P can be obtained by concatenating optimal

integral solutions to $P_1, \ldots, P_t$. Therefore, it suffices to solve programs $P_1, \ldots, P_n$ recursively and in parallel, by assigning s_i processors to program P_i, where s_i is the number of rows of D_i.

Suppose now that A is not block-decomposable. Using Lemma 3.3, we find an optimal fractional solution $x^\star \in \mathrm{Sol}^{\mathbb{R}}(P)$ in time $\log^{\mathcal{O}(2^{\mathsf{depth}_\mathsf{P}(A)})} t$. By Lemma 3.1, there exists an optimal integral solution $x^\diamond \in \mathrm{Sol}^{\mathbb{Z}}(P)$ such that $\|x^\diamond - x^\star\|_\infty \leq \rho$, where ρ is a constant that depends in a computable manner on $\mathsf{depth}_\mathsf{P}(A)$ and $\|A\|_\infty$. In particular, if we denote the first variable of x by x_1, then there exists an optimal integral solution $x^\diamond \in \mathrm{Sol}^{\mathbb{Z}}(P)$ satisfying $|x_1^\diamond - x_1^\star| \leq \rho$.

Since A is not block-decomposable, we can write $A = (a_1\ A')$, where a_1 is the first column of A and A' is a matrix with $\mathsf{depth}_\mathsf{P}(A') < \mathsf{depth}_\mathsf{P}(A)$. For every $\xi \in [x_1^\star - \rho, x_1^\star + \rho] \cap \mathbb{Z}_{\geq 0}$, consider the linear program $P'(\xi)$ defined as

$$\begin{aligned} &\max\ (c')^\intercal x' \\ &A'x' = b - \xi \cdot a_1 \\ &x' \geq 0 \end{aligned}$$

where c' and x' are c and x with the first entry removed, respectively. From the observation of the previous paragraph it follows that the set of optimal integral solutions to P is

$$\mathrm{opt}^{\mathbb{Z}}(P) = \max\{\, c_1\xi + \mathrm{opt}^{\mathbb{Z}}(P'(\xi)) \,:\, \xi \in [x_1^\star - \rho, x_1^\star + \rho] \cap \mathbb{Z}_{\geq 0} \,\},$$

where c_1 is the first entry of c. Therefore, this allows to first solve all programs $P'(\xi)$ for $\xi \in [x_1^\star - \rho, x_1^\star + \rho] \cap \mathbb{Z}_{\geq 0}$ recursively one by one to obtain respective optimal solutions $x'(\xi)$. Then we may output the solution minimizing $c^\intercal x$ among all solutions of the form $x = \binom{\xi}{x'(\xi)}$ with $\xi \in [x_1^\star - \rho, x_1^\star + \rho] \cap \mathbb{Z}_{\geq 0}$.

The correctness of the algorithm follows directly from Lemma 3.1. As for the running time, observe that when treating a linear program with a block decomposable constraint matrix, we recursively and in parallel solve programs with constraint matrices that are not block decomposable. On the other hand, when treating a linear program with a non block decomposable constraint matrix, we recursively solve at most $2\rho + 1$ programs in a sequential manner, each with a strictly smaller depth. Hence, if by $T[t, \Delta, d]$ we denote the (parallel) running time for programs with t rows, depth at most d, and all coefficients bounded in absolute value by Δ, then $T[t, \Delta, d]$ satisfies the recursive inequality:

$$T[t, \Delta, d] \leq \log^{\mathcal{O}(2^d)} t + (2\rho + 1) \cdot T[t, \Delta, d-1].$$

This recursion solves to

$$T[t, \Delta, d] \leq (2\rho + 1)^d \cdot \log^{\mathcal{O}(2^d)} t.$$

Since ρ is bounded by a computable function of $\mathsf{depth}_\mathsf{P}(A)$ and $\|A\|_\infty$, the claimed running time bound follows. □

Theorem 3.6. *Suppose we are given an (p,q)-stochastic linear program $P = (x, A, b, c)$ in the form (♠). Let t be the number of rows of A. Then, one can, using t processors and requiring $2^{((p+q)\|A\|_\infty)^{\mathcal{O}(p(p+q))}} \cdot \log^{\mathcal{O}(pq)} t$ on each processor, compute an optimal integral solution to P.*

Proof. Here the same algorithm as in the proof of Theorem 3.5 is applied, except that the usage of Lemma 3.3 is replaced with Lemma 3.4, and the proximity bounds are provided by Lemma 3.2 instead of Lemma 3.1. Since a (p,q)-stochastic matrix has primal depth at most $p+q$, the same time complexity analysis shows that the running time is bounded by

$$T[t, \|A\|_\infty, p, q] \leq (2\rho+1)^{p+q} \cdot 2^{\mathcal{O}(p^2+pq^2)} \cdot \log^{\mathcal{O}(pq)} t,$$

where $\rho \leq 2^{\mathcal{O}(p(p+q)\|A\|_\infty)^{p(p+q)}}$ is the proximity bound provided by Lemma 3.2. Thus, the total running time is bounded by $2^{((p+q)\|A\|_\infty)^{\mathcal{O}(p(p+q))}} \cdot \log^{\mathcal{O}(pq)} t$, as claimed. □

3.3 A stronger Klein Bound

In this section, we recall a structural result of Klein [Kle22] and prove a stronger variant, which we need for our proximity bounds in the next section. Formally, we prove the following theorem.

Theorem 3.7 (Stronger Klein Bound)**.** *Let $T_1, \ldots, T_n \subseteq \mathbb{Z}^d$ be multisets of integer vectors with ℓ_∞-norm at most Δ such that their respective sums are almost the same in the following sense: there is some $b \in \mathbb{Z}^d$ and a positive integer ε such that*

$$\Big\| \sum_{v \in T_i} v - b \Big\|_\infty < \varepsilon \qquad \text{for all } i \in \{1, \ldots, n\}.$$

There exists a function $f(d,\Delta) \in 2^{\mathcal{O}(d\Delta)^d}$ such that the following property holds. Assuming $\|b\|_\infty > \varepsilon \cdot f(d,\Delta)$, one can find nonempty sub-multisets $S_i \subseteq T_i$ for all $i \in \{1, \ldots, n\}$, and a vector $b' \in \mathbb{Z}^d$ satisfying $\|b'\|_\infty \leq f(d,\Delta)$, such that

$$\sum_{v \in S_i} v = b' \qquad \text{for all } i \in \{1, \ldots, n\}.$$

Before proceeding to the proof, let us discuss the aspects in which Theorem 3.7 strengthens the original formulation of Klein [Kle22, Lemma 2]. First, the formulation of Klein required all the vectors to be non-negative. Second, we allow the sums of respective multisets to differ slightly, which is governed by the slack parameter ε. In the original setting of Klein all sums need to be exactly equal. In our setting this corresponds to ε to be equal to 1. Finally, the argument of Klein yields a bound on the function $f(d,\Delta)$ that is doubly exponential in d^2, our proof improves this dependence to doubly exponential in $d\log(d)$. The second aspect will be essential in the proof of the proximity bound, while the last is primarily used for improving the parametric factor in the running time of our algorithms.

Note that by now, Theorem 3.7 is further generalized by Klein and Reuter in [KR22]. Intuitively, they expand the theorem to deal with multiple stages of a stochastic linear program at once instead of a single stage at a time. This allows for better running times for multistage stochastic linear programs.

The remainder of this section is devoted to the proof of Theorem 3.7. We need a few definitions at this point. The *cone* spanned by vectors $u_1, \dots, u_k \in \mathbb{Q}^d$ is the set of all non-negative linear combinations of $u_1, \dots, u_k$. In other words,

$$\operatorname{cone}(u_1, \dots, u_k) := \left\{ \sum_{i=1}^{k} \lambda_i v_i \colon \lambda_i \geq 0,\ i = 1, \dots, k \right\}.$$

The *integer cone* spanned by these vectors is the set of all non-negative *integer* linear combinations of $u_1, \dots, u_k$ and is described by

$$\operatorname{int.cone}(u_1, \dots, u_k) := \left\{ \sum_{i=1}^{k} \lambda_i v_i \colon \lambda_i \in \mathbb{Z}_{\geq 0},\ i = 1, \dots, k \right\}.$$

We also use the corresponding matrix notation given by

$$\operatorname{cone}(A) = \{Ax \colon x \in \mathbb{R}^k_{\geq 0}\} \quad \text{and} \quad \operatorname{int.cone}(A) = \{Ax \colon x \in \mathbb{Z}^k_{\geq 0}\},$$

for a matrix $A \in \mathbb{Q}^{d \times k}$. The following lemma is the crucial observation behind our proof of the improved Klein bound.

Lemma 3.8. *Let $A_1, \dots, A_\ell \in \mathbb{Z}^{d \times d}$ be invertible integer matrices such that $\|A_i\|_\infty \leq \Delta$ for each $i \in \{1, \dots, \ell\}$ and*

$$\mathcal{C} := \bigcap_{i=1}^{\ell} \operatorname{cone}(A_i) \neq \emptyset. \tag{3.1}$$

Then the following assertions hold

(L1) Let M be the least common multiple of the determinants of matrices A_i. Then $M \leq 3^{(d\Delta)^d}$.

(L2) For each integer vector $v \in \mathcal{C} \cap \mathbb{Z}^d$ one has $Mv \in \mathcal{I}$, where

$$\mathcal{I} = \bigcap_{i=1}^{\ell} \operatorname{int.cone}(A_i).$$

(L3) There exist integer vectors $v_1, \dots, v_t \in \mathcal{I}$ such that $\|v_j\|_\infty \leq 2^{\mathcal{O}(d\Delta)^d}$ for each $j \in \{1, \dots, t\}$ and

$$\operatorname{cone}(v_1, \dots, v_t) = \mathcal{C}.$$

Proof. By the Hadamard bound, $|\det(A_i)| \leq (d\Delta)^d$ for each $i \in \{1,\dots,\ell\}$. Hence, the least common multiple of the determinants of matrices A_i is bounded by the least common multiple of all the integers in the range $\{1,\dots,(d\Delta)^d\}$, which in turn is bounded by $3^{(d\Delta)^d}$ [Han72, Theorem 1]. This implies (L1).

We now move to assertion (L2). Consider any $i \in \{1,\dots,\ell\}$. Since $v \in \mathrm{cone}(A_i) \cap \mathbb{Z}^d$, there exists $x_i \in \mathbb{Q}^d_{\geq 0}$ with $v = A_i x_i$. In fact, one has $x_i = {A_i}^{-1} v$. Then Cramer's rule implies that

$$x_i = y_i / |\det(A_i)| \qquad \text{for some } y_i \in \mathbb{Z}^d_{\geq 0}.$$

This shows that if we denote

$$M := \mathrm{lcm}\left(\{|\det(A_i)| \colon i \in \{1,\dots,\ell\}\}\right),$$

then it holds that

$$M \cdot v \in \mathrm{int.cone}(A_i) \qquad \text{for each } i \in \{1,\dots,\ell\}.$$

Which establishes (L2).

By standard arguments, there exist integer vectors $w_1,\dots,w_t \in \mathcal{C} \cap \mathbb{Z}^d$ with ℓ_∞-norm bounded by $(d\Delta)^{d^2}$ generating the cone $\mathcal{C}$. This is a simple consequence of the Farkas-Minkowski-Weyl Theorem and its proof in [Sch98, Corollary 7.1a]. We briefly sketch the argument. The cones $\mathrm{cone}(A_i)$ are finitely generated and admit an *inequality description* of the form $\{c_j^\intercal x \geq 0\}_{j \in I}$. Note that $c_j \in \mathbb{Z}^d$, due to the Hadamard bound, we may assume that $\|c_j\|_\infty \leq (d\Delta)^{d/2}$. Cone $\mathcal{C}$ can then be obtained by conjunction of all the inequalities describing the cones $\mathrm{cone}(A_i)$. It follows that $\mathcal{C}$ is polyhedral, and thus finitely generated. A generator w of $\mathcal{C}$ satisfies $d-1$ of the inequalities of the form $c_i^\intercal x \geq 0$ with equality, i.e., w is orthogonal to $d-1$ vectors c_k. This implies that, w is in the kernel of some matrix with $d-1$ rows consisting of vectors $c_k^\intercal \in \mathbb{Z}^{1\times d}$. Take an integral generator w. Using the Hadamard bound and the fact that $\|c_k\|_\infty \leq (d\Delta)^{d/2}$, the ℓ_∞-norm of w is bounded by

$$\|w\|_\infty \leq \left(d(d\Delta)^{d/2}\right)^{d/2} \leq (d\Delta)^{d^2}.$$

.

Finally, by (L1) and (L2), we see that the vectors

$$v_1 := Mw_1, \qquad v_2 := Mw_2, \qquad \dots \qquad v_t := Mw_t$$

satisfy (L3). □

With Lemma 3.8 established, we may proceed to the main part of the proof.

Proof of Theorem 3.7. The proof follows the lines of Klein [Kle22]. But instead of using the Steinitz Lemma (see Theorem 1.2), we apply Lemma 3.8 and use the Minkowski-Weyl Theorem [Sch98, Theorem 8.5] to deal with non negative entries. In matrix notation, Theorem 3.7 can be restated as follows.

> *Let $D := (2\Delta+1)^d$ and let $B \in \mathbb{Z}^{d\times D}$ be a matrix whose columns are all possible integer vectors of ℓ_∞-norm at most Δ. Let $m_1,\dots,m_n \in \mathbb{Z}^D_{\geq 0}$ be non negative integer vectors such that the vectors Bm_i are almost equal, that is, one has*
>
> $$\|Bm_i - b\|_\infty < \varepsilon \qquad \text{for each } i \in \{1,\dots,n\},$$
>
> *for some integer vector $b \in \mathbb{Z}^d$ and a positive integer ε. Then supposing $\|b\|_\infty > \varepsilon \cdot 2^{\mathcal{O}(d\Delta)^d}$, there exist nonzero vectors $m'_1,\dots,m'_n \in \mathbb{Z}^D_{\geq 0}$ and $b' \in \mathbb{Z}^d$ such that for each $i \in \{1,\dots,n\}$,*
>
> *(K1)* $0 \leq m'_i \leq m_i$,
>
> *(K2)* $Bm'_i = b'$, *and*
>
> *(K3)* $\|b'\|_\infty \leq 2^{\mathcal{O}(d\Delta)^d}$.

We focus on proving this formulation. To this end, for each $i \in \{1,\dots,n\}$ we choose some vector $r_i \in \mathbb{Z}^D_{\geq 0}$ such that

$$B(m_i + r_i) = b.$$

We may assume $\|r_i\|_\infty \leq \varepsilon$, for instance by putting nonzero values in r_i only at entries corresponding to the columns $\{\pm e_1,\dots,\pm e_d\}$ of B, where $\{e_1,\dots,e_d\}$ is the standard base of $\mathbb{Z}^d$.

Now, set $z_i := m_i + r_i$. Then the vectors z_i belong to the following (unbounded) polyhedron

$$Q := \{x \in \mathbb{R}^D \mid Bx = b \text{ and } x \geq 0\}.$$

By the Minkowski-Weyl Theorem [Sch98, Theorem 8.5], it follows that Q can be written as

$$Q = \operatorname{conv}\big(\{u_1,\dots,u_k\}\big) + \operatorname{cone}\big(\{c_1,\dots,c_\ell\}\big) \tag{3.2}$$

for some vectors $u_1,\dots,u_k,c_1,\dots,c_\ell \in \mathbb{Z}^D$. Note that since $Q \subseteq \mathbb{R}^D_{\geq 0}$, the vectors $u_1,\dots,u_k$ and $c_1,\dots,c_\ell$ are all non-negative. Further, it holds that $Bu_j = b$ for all $j \in \{1,\dots,k\}$ and $Bc_h = 0$ for all $h \in \{1,\dots,\ell\}$. Observe also that we may assume the vectors u_j to be vertex solutions to the linear program defining Q. Hence, each vector u_j has at most d nonzero entries, and there is an invertible submatrix $B_j \in \mathbb{Z}^{d\times d}$ (the basis of u_j) consisting of d columns of B such that all nonzero entries of u_j are at the coordinates corresponding to the columns of B_j. In particular $\widetilde{u}_j = B_j^{-1}b$, where $\widetilde{u}_j$ is u_j projected onto the coordinates corresponding to the columns of B_j.

Fix $i \in \{1,\dots,n\}$. By (3.2) vector z_i can be written as

$$z_i = \sum_{j=1}^{k} \lambda_j u_j + \sum_{h=1}^{\ell} \mu_h c_h, \quad \text{where } \sum_{j=1}^{k} \lambda_j = 1 \text{ and } \lambda_j, \mu_h \in \mathbb{R}_{\geq 0}. \tag{3.3}$$

From Carathéodory's theorem, see [Sch98, Corollary 7.1i], it follows that the coefficients $\lambda_1,\dots,\lambda_k$ can be chosen in a way that there exists an index $j \in \{1,\dots,k\}$ with $\lambda_j \geq 1/(d+1)$. Denote this index by $j(i)$. Since all involved vectors and scalars in (3.3) are non-negative, we conclude that

$$0 \leq \frac{u_{j(i)}}{(d+1)} \leq z_i. \tag{3.4}$$

Now, we will argue that there exists a vector $c \in \mathbb{Z}^d$ and nonzero vectors $u'_1,\dots u'_k \in \mathbb{Z}^D_{\geq 0}$ such that

$$Bu'_j = c \qquad \text{for all } j \in \{1,\dots,k\} \tag{3.5}$$

and

$$u'_{j(i)} \leq m_i \qquad \text{for all } i \in \{1,\dots,n\} \tag{3.6}$$

and

$$\|c\|_\infty \leq 2^{\mathcal{O}(d\Delta)^d}. \tag{3.7}$$

We remark here that at the end of the proof we will set $m'_i := u'_{j(i)}$ for all $i \in \{1,\dots,n\}$ and $b' := c$. Observe that then, assertions (K1), (K2) and (K3) will immediately follow from (3.5), (3.6) and (3.7) respectively.

Since $Bu_j = B_j\widetilde{u}_j = b$, it follows that $\mathcal{C} := \bigcap_{j=1}^{k} \operatorname{cone}(B_j) \neq \emptyset$. By Lemma 3.8, assertion (L3), there exist nonzero integer vectors $v_1,\dots,v_t \in \bigcap_{j=1}^{k} \operatorname{int.cone}(B_j)$ of ℓ_∞-norm bounded by $2^{\mathcal{O}(d\Delta)^d}$ such that

$$\operatorname{cone}(v_1,\dots,v_t) = \mathcal{C}.$$

Since $b/(d+1) \in \mathcal{C}$, by Carathéodory's theorem, we can pick at most d vectors of $\{v_1,\dots,v_t\}$, say $v_1,\dots,v_d$, such that

$$\frac{b}{d+1} = \sum_{j=1}^{d} \alpha_j v_j \qquad \text{for some } \alpha_j \geq 0,\ j \in \{1,\dots,d\}.$$

Now use the assumption on $\|b\|_\infty$. Specifically, assume that

$$\|b\|_\infty > (d+1)d \cdot 2\varepsilon \cdot \max_i \|v_i\|_\infty = \varepsilon \cdot 2^{\mathcal{O}(d\Delta)^d}.$$

Observe that there exists an index $j \in \{1,\dots,d\}$ such that $\alpha_j > 2\varepsilon$. Without loss of generality

suppose $j = 1$. Then we can write

$$\frac{b}{d+1} = 2\varepsilon v_1 + \underbrace{\left((\alpha_1 - 2\varepsilon)v_1 + \sum_{j=2}^{d} \alpha_j v_j\right)}_{=:q}.$$

Since $v_1 \in \text{int.cone}(B_j)$, for each $j \in \{1,\dots,k\}$ there exists a vector $y_j \in \mathbb{Z}^d_{\geq 0}$ such that $B_j y_j = v_1$. Also, since q is in the cone $\mathcal{C}$, there exist vectors $x_j \in \mathbb{R}^d_{\geq 0}$ such that $B_j x_j = q$. Thus, for each $j \in \{1,\dots,k\}$ we have

$$\frac{B_j \widetilde{u}_j}{d+1} = \frac{b}{d+1} = 2\varepsilon v_1 + q = B_j(2\varepsilon y_j + x_j)$$

Since the matrices B_j are invertible, this implies that

$$\frac{\widetilde{u}_j}{d+1} = 2\varepsilon y_j + x_j. \tag{3.8}$$

Now, set $c = v_1$ and for $j \in \{1,\dots,k\}$ define $u'_j \in \mathbb{Z}^D_{\geq 0}$ as follows

- each entry of u'_j that corresponds to a column of B_j is set to the corresponding entry of y_j;
- every other entry of u'_j is set to 0.

Note that thus, the vectors u'_j are nonzero, because $v_1 = B_j y_j$ and v_1 is nonzero. We also have $Bu'_j = B_j y_j = v_1$, which implies that (3.5) holds. Finally, since $c = v_1$, it follows that $\|c\|_\infty \leq 2^{\mathcal{O}(d\Delta)^d}$, which is (3.7).

It remains to prove (3.6). Note that by (3.4) and (3.8) for all $i \in \{1,\dots,n\}$ we have

$$0 \leq 2\varepsilon y_{j(i)} \leq 2\varepsilon y_{j(i)} + x_{j(i)} = \frac{\widetilde{u}_{j(i)}}{d+1}$$

and

$$\frac{u_{j(i)}}{d+1} \leq z_i = m_i + r_i.$$

Since $\|r_i\|_\infty \leq \varepsilon$ and $y_{j(i)}$ is nonzero, from the above inequalities it follows that $u'_{j(i)} \leq m_i$. This establishes (3.6) and concludes the proof. □

3.4 Proximity

The goal of this section is to prove Lemma 3.1 and Lemma 3.2. Specifically, we bound the distance between an optimal fractional solution and an optimal integral solution in the case where the constraint matrix has bounded primal treedepth or is (p,q)-stochastic. To facilitate the discussion of proximity, let us introduce the following definition.

Definition 3.9. Let $P = (x, A, b, c)$ be a linear program in the form (♠). The *proximity* of P, denoted $\mathsf{proximity}_\infty(P)$, is the infimum of reals $\rho \geq 0$ satisfying the following: for every fractional solution $x^\star \in \mathrm{Sol}^{\mathbb{R}}(P)$ and integral solution $x^\square \in \mathrm{Sol}^{\mathbb{Z}}(P)$, there is an integral solution $x^\diamond \in \mathrm{Sol}^{\mathbb{Z}}(P)$ such that

$$\|x^\diamond - x^\star\|_\infty \leq \rho \qquad \text{and} \qquad x^\diamond - x^\star \sqsubseteq x^\square - x^\star.$$

We remind that $\sqsubseteq$ is the conformal partial order defined in Section 1.2.2. The condition $x^\diamond - x^\star \sqsubseteq x^\square - x^\star$ is equivalent to saying that $x^\diamond$ is contained in the axis parallel box spanned by $x^\star$ and $x^\square$, see Figure 3.1. Intuitively, $x^\diamond$ is an integral solution that is close to $x^\star$ in the ℓ_∞-distance while being placed "in the same direction" as $x^\square$.

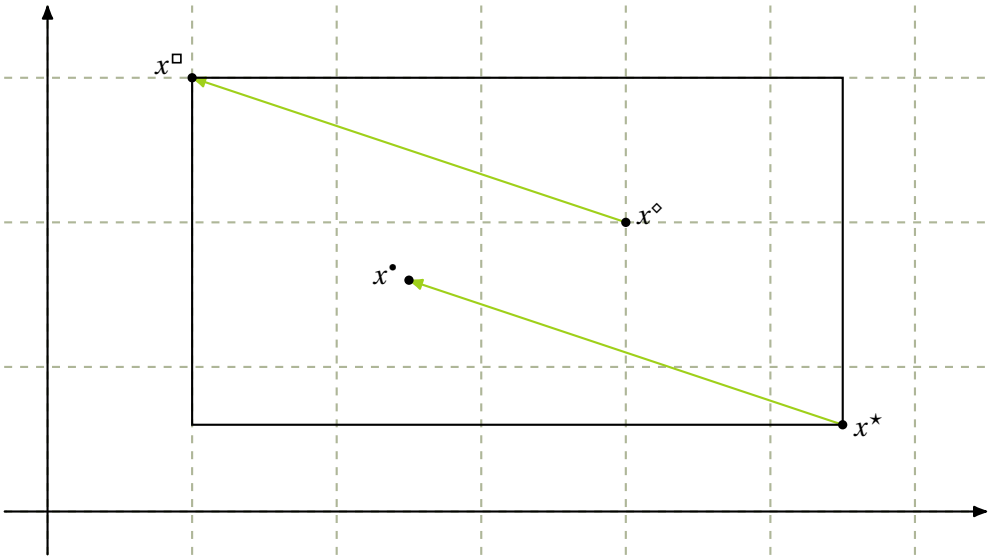

Figure 3.1: Example of vector $x^\diamond$ with the property $x^\diamond - x^\star \sqsubseteq x^\square - x^\star$. Shifting $x^\star$ by $x^\square - x^\diamond$ gives an other vector $x^\bullet$ with the same property.

Previously, the notion of proximity was mostly defined as the maximum distance of any optimal fractional solutions to its closest optimal integral solutions, see for instance [CGST86, EW20b]. Our notion of proximity does not depend on the optimization goal, it is a geometric quantity associated only with the polytope $\mathrm{Sol}^{\mathbb{R}}(P)$. However, this new notion can also be used to bound the distance of optimal fractional solutions to optimal integral solutions, as the next lemma explains.

Lemma 3.10. *Suppose $P = (x, A, b, c)$ is a linear program in the form (♠). Then for every optimal fractional solution $x^\star$ to P there exists an optimal integral solution $x^\diamond$ to P satisfying*

$$\|x^\diamond - x^\star\|_\infty \leq \mathsf{proximity}_\infty(P).$$

Proof. Consider any optimal integral solution $x^\square$ to P. By the definition of proximity, there is an integral solution $x^\diamond$ to P such that $\|x^\diamond - x^\star\|_\infty \leq \mathsf{proximity}_\infty(P)$ and $x^\diamond - x^\star \sqsubseteq x^\square - x^\star$. From

the optimality of $x^\square$ we get that

$$c^\intercal(x^\square - x^\diamond) \geq 0.$$

Let $x^\bullet := x^\star + x^\square - x^\diamond$. This is a fractional solution to P, because $Ax^\bullet = A(x^\star + x^\square - x^\diamond) = b$ and a straightforward coordinate-wise verification shows that

$$x^\star \geq 0,\ x^\square \geq 0,\ x^\diamond - x^\star \sqsubseteq x^\square - x^\star \quad \text{implies that} \quad x^\bullet \geq 0.$$

The optimality of $x^\star$ then gives that

$$c^\intercal(x^\square - x^\diamond) = c^\intercal((x^\star + x^\square - x^\diamond) - x^\star) = c^\intercal(x^\bullet - x^\star) \leq 0.$$

This implies that $c^\intercal x^\square = c^\intercal x^\diamond$, hence $x^\diamond$ is also optimal. □

For our main technical result, we need some additional notation. Suppose that A is a matrix admitting the stochastic decomposition ($\diamondsuit$). Let $x_0, x_1, \ldots, x_n$ be the partition of the vector of variables x so that x_0 corresponds to the columns of matrices $C_1, \ldots, C_n$, while x_i corresponds to the columns of D_i, for each $i \in \{1, \ldots, n\}$. Partition c into $c_0, c_1, \ldots, c_n$ in the same fashion, and partition b into $b_1, \ldots, b_n$ so that b_i corresponds to the rows of C_i and D_i. In this representation, the program P takes the form:

$$\begin{aligned} \max \sum_{i=0}^{n} c_i^\intercal x_i & \\ C_i x_0 + D_i x_i = b_i & \qquad \text{for all } i \in \{1, \ldots, n\}, \\ x_i \geq 0 & \qquad \text{for all } i \in \{0, 1, \ldots, n\}. \end{aligned}$$

For each $i \in \{1, \ldots, n\}$, let $E_i := (C_i\ D_i)$ and consider the linear program

$$P_i = \left(\begin{pmatrix} x_0 \\ x_i \end{pmatrix}, E_i, b_i, 0 \right),$$

that is, the linear program

$$\begin{aligned} \max\ & 0 \\ A_i x_0 + B_i x_i &= b_i, \\ x_0 \geq 0, \quad x_i &\geq 0. \end{aligned}$$

Note that

$$\begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_t \end{pmatrix} \in \mathrm{Sol}^{\mathbb{R}}(P) \qquad \text{if and only if} \qquad \begin{pmatrix} x_0 \\ x_i \end{pmatrix} \in \mathrm{Sol}^{\mathbb{R}}(P_i) \text{ for all } i \in \{1, \ldots, n\}.$$

We are now ready to state the main technical result of this section. Intuitively, it provides a single inductive step in the proof of Lemma 3.1 and reduces Lemma 3.2 to the case of matrices with a bounded number of columns.

Theorem 3.11 (Composition Theorem)**.** *Suppose $P = (x, A, b, c)$ is a linear program in the form* (♠)*, where A admits a stochastic decomposition* (♢)*. Adopt the notation presented above and let k be the number of columns of each of the matrices $C_1, \dots, C_n$. Further, let*

$$\gamma \coloneqq \max_{1 \le i \le n} g_\infty(E_i) \qquad \text{and} \qquad \rho \coloneqq \max_{1 \le i \le n} \mathsf{proximity}_\infty(P_i).$$

Then

$$\mathsf{proximity}_\infty(P) \le 3k\gamma\rho \cdot f(k, \gamma)$$

where $f(k, \gamma)$ is the bound provided by Theorem 3.7.

Note that by substituting $f(k, \gamma)$ with the bound provided by Theorem 3.7, we obtain that

$$\mathsf{proximity}_\infty(P) \le \rho \cdot 2^{\mathcal{O}(k\gamma)^k}.$$

Before we prove Theorem 3.11, let us observe the following two consequences of it. As a base case, we give a bound on the proximity of a standard integer program defined by an integer matrix with m columns. This is then first used to bound the proximity of an integer program defined by an (p, q)-stochastic matrix. The second consequence is a bound on the proximity of a integer program depending on the primal treedepth of the matrix defining it.

Lemma 3.12. *Let $P = (x, A, b, c)$ be a linear program in the form* (♠) *where A has m columns. Then*

$$\mathsf{proximity}_\infty(P) \le (m\|A\|_\infty)^{m+1}.$$

Proof. We apply a classical theorem of Cook et al. [CGST86] to our notion of proximity. Let $x^\star$ be a fractional solution and $x^\square$ an integral solution to P. Consider the following (integer) linear program:

$$\begin{aligned} \max\ & c^\mathsf{T} x \\ & Ax = b \\ & x - x^\star \sqsubseteq x^\square - x^\star. \end{aligned}$$

The constraint $x - x^\star \sqsubseteq x^\square - x^\star$ can be expressed as a conjunction of constraints of the form $x_i^\star \le x_i \le x_i^\square$ or $x_i^\square \le x_i \le x^\star$ for $i \in \{1, \dots, m\}$, depending on whether $x_i^\star \le x_i^\square$ or $x_i^\square \le x_i^\star$. Thus, the constraint matrix still has m columns and its coefficients are bounded by $\|A\|_\infty$. By the Hadamard bound it follows that its largest sub-determinant is bounded by $(m\|A\|_\infty)^m$. By [CGST86, Theorem 1] we conclude that there is an integral solution $x^\diamond$ such that $\|x^\diamond - x^\star\|_\infty \le m(m\|A\|_\infty)^m$ and $x^\diamond - x^\star \sqsubseteq x^\square - x^\star$. □

Corollary 3.13. *Let $P = (x, A, b, c)$ be a linear program in the form (♠), where A is (p, q)-stochastic. Then*

$$\mathsf{proximity}_\infty(P) \le 2^{\mathcal{O}(p(p+q)\|A\|_\infty)^{p(p+q)}}.$$

Proof. By assumption, matrix A admits a decomposition of the form (♢), where each block C_i has p columns and each block D_i has at most q columns. Adopting the notation introduced before Theorem 3.11, we see that each matrix $E_i = (C_i\ D_i)$ has at most $p+q$ columns. Applying Lemma 3.12 to P_i, we get

$$\mathsf{proximity}_\infty(P_i) \le ((p+q)\|A\|_\infty)^{p+q+1}.$$

Further, by Corollary 1.4 we have

$$g_\infty(E_i) \le (2(p+q)\|A\|_\infty + 1)^{p+q}.$$

Combine these two bounds using Theorem 3.11 to get the claimed bound on $\mathsf{proximity}_\infty(A)$. □

Corollary 3.14. *There is a computable function $h\colon \mathbb{N}\times\mathbb{N} \to \mathbb{N}$ such that for every linear program $P = (x, A, b, c)$ in the form (♠), we have*

$$\mathsf{proximity}_\infty(P) \le h(\mathsf{td}_\mathsf{P}(A), \|A\|_\infty).$$

Proof. We use induction on a more general problem. Suppose $P = (x, A, b, c)$ is a linear program in the form (♠), where A has the following property: removing the first k columns turns A into a matrix of primal depth at most ℓ. We would like to prove that

$$\mathsf{proximity}_\infty(A) \le \widehat{h}(k, \ell, \|A\|_\infty)$$

for some computable function $\widehat{h}$. The corollary then follows by considering the case $k = 0$, that is, setting $h(d, \|A\|_\infty) = \widehat{h}(0, d, \|A\|_\infty)$.

To prove the general statement we proceed by induction on ℓ, starting with $\ell = 0$. Then A is a matrix with k columns, and, as discussed in Lemma 3.12, we can fix a function

$$\widehat{h}(k, 0, \|A\|_\infty) \in \mathcal{O}(k\|A\|_\infty)^{k+1}.$$

Let us proceed to the induction step for $\ell > 0$. Since removing the first k columns turns A into a matrix of primal depth at most ℓ, it follows that A has a stochastic decomposition (♢), where the matrices C_i have k columns each and the matrices D_i have primal depth at most ℓ. We may further assume that matrices D_i are not block-decomposable, hence each matrix D_i becomes a matrix of primal depth at most $\ell - 1$ after removing its first column. This implies that each matrix $E_i = (C_i\ D_i)$ has the following property: removing the first $k+1$ columns

turns it into a matrix of primal depth at most $\ell - 1$. Theorem 1.5 implies that

$$g_\infty(E_i) \le f(\mathsf{depth}_P(E_i), \|E_i\|_\infty) \le f(k+\ell, \|A\|_\infty)$$

for a computable function f, while the induction assumption gives

$$\mathsf{proximity}_\infty(P_i) \le \widehat{h}(k+1, \ell-1, \|A\|_\infty),$$

where the programs P_i are defined as in the paragraph before Theorem 3.11. We may now combine these two bounds using Theorem 3.11 to get a bound on $\widehat{h}(k,\ell,\|A\|_\infty)$, expressed in terms of $f(k+\ell,\|A\|_\infty)$ and $\widehat{h}(k+1,\ell-1,\|A\|_\infty)$. □

Now, Lemmas 3.1 and 3.2 follow by combining Lemma 3.10 with Corollaries 3.13 and 3.14, respectively.

3.4.1 Proof of Theorem 3.11

As mentioned before, the proof of Theorem 3.11 relies heavily on Theorem 3.7: the strengthening of the structural lemma of Klein [Kle22] that was discussed in Section 3.3.

Proof of Theorem 3.11. Consider any $x^\star \in \mathrm{Sol}^{\mathbb{R}}(P)$ and $x^\square \in \mathrm{Sol}^{\mathbb{Z}}(P)$. Let $x^\diamond \in \mathrm{Sol}^{\mathbb{Z}}(P)$ be an integral solution such that $x^\diamond - x^\star \sqsubseteq x^\square - x^\star$ and subject to the conditions $\|x^\diamond - x^\star\|_1$ is minimized. Our goal is to show that then $\|x^\diamond - x^\star\|_\infty \le 3k\gamma\rho \cdot f(k,\gamma)$, where $f(\cdot,\cdot)$ is the function given by Theorem 3.7.

Observe that if there existed a non-zero vector $u \in \ker^{\mathbb{Z}}(A)$ such that $u \sqsubseteq x^\star - x^\diamond$, then we would have that $x^\diamond + u \in \mathrm{Sol}^{\mathbb{Z}}(P)$, $(x^\diamond + u) - x^\star \sqsubseteq x^\diamond - x^\star \sqsubseteq x^\square - x^\star$, and the ℓ_1 distance from $x^\star$ to $x^\diamond + u$ would be strictly smaller than to $x^\diamond$. This would contradict the choice of $x^\diamond$. Therefore, it is sufficient to show the following: if $\|x^\diamond - x^\star\|_\infty$ is larger than $3k\gamma\rho \cdot f(k,\gamma)$, then there exists a non-zero vector $u \in \ker^{\mathbb{Z}}(A)$ such that $u \sqsubseteq x^\star - x^\diamond$.

Consider any $i \in \{1,\dots,n\}$ and denote the restrictions of $x^\star$ and $x^\diamond$ to the variables of P_i as follows:

$$\widetilde{x}_i^\star := \begin{pmatrix} x_0^\star \\ x_i^\star \end{pmatrix} \in \mathrm{Sol}^{\mathbb{R}}(P_i) \qquad \text{and} \qquad \widetilde{x}_i^\diamond := \begin{pmatrix} x_0^\diamond \\ x_i^\diamond \end{pmatrix} \in \mathrm{Sol}^{\mathbb{Z}}(P_i).$$

By the definition of proximity, there is an integral solution

$$\widetilde{x}_i \in \mathrm{Sol}^{\mathbb{Z}}(P_i)$$

such that

$$\|\widetilde{x}_i - \widetilde{x}_i^\star\|_\infty \le \mathsf{proximity}_\infty(P_i) \le \rho \qquad \text{and} \qquad \widetilde{x}_i - \widetilde{x}_i^\star \sqsubseteq \widetilde{x}_i^\diamond - \widetilde{x}_i^\star.$$

Since $\widetilde{x}_i$ and $\widetilde{x}_i^\diamond$ are both integral solutions to P_i, we have $\widetilde{x}_i - \widetilde{x}_i^\diamond \in \ker^{\mathbb{Z}}(C_i\ D_i)$ and we can decompose this vector into a multiset G_i of Graver elements. That is, G_i is a multiset consisting

of sign compatible (i.e., belong to the same orthant) elements of $\mathcal{G}(E_i)$ with

$$\widetilde{x}_i - \widetilde{x}_i^{\diamond} = \sum_{g \in G_i} g.$$

Note that the first k entries of vectors $\widetilde{x}_1, \ldots, \widetilde{x}_n$ correspond to the same k variables of P, but they may differ for different $i \in \{1, \ldots, n\}$.

For a vector w, let $\pi(w)$ be the projection onto the first k entries of w. Let $\pi(G_i)$ be the multiset that includes a copy of $\pi(g)$ for each $g \in G_i$. By the definition of $\widetilde{x}_i^{\star}$ and $\widetilde{x}_i^{\diamond}$, we have $\pi(\widetilde{x}_i^{\star}) = \pi(\widetilde{x}_j^{\star})$ and $\pi(\widetilde{x}_i^{\diamond}) = \pi(\widetilde{x}_j^{\diamond})$ for all $i, j \in \{1, \ldots, n\}$. From this we get

$$\begin{aligned}
\Big\| \sum_{x \in \pi(G_i)} x - \pi(\widetilde{x}_1^{\star} - \widetilde{x}_1^{\diamond}) \Big\|_{\infty} &= \|\pi(\widetilde{x}_i) - \pi(\widetilde{x}_i^{\diamond}) - \pi(\widetilde{x}_1^{\star}) + \pi(\widetilde{x}_1^{\diamond})\|_{\infty} \\
&= \|\pi(\widetilde{x}_i) - \pi(\widetilde{x}_i^{\star})\|_{\infty} \\
&= \|\widetilde{x}_i - \widetilde{x}_i^{\star}\|_{\infty} \\
&\leq \rho,
\end{aligned}$$

for each $i \in \{1, \ldots, n\}$. Thus, Theorem 3.7 is applicable for $d = k$, $\Delta = \gamma$, and $\varepsilon = \rho$. Note here that for each $i \in \{1, \ldots, n\}$ and $g \in G_i$, we have $\|g\|_{\infty} \leq \gamma$. In the following we distinguish two cases.

Suppose first that

$$\|\pi(\widetilde{x}_1^{\star} - \widetilde{x}_1^{\diamond})\|_{\infty} > \rho \cdot f(k, \gamma).$$

By Theorem 3.7, there exist nonempty submultisets $S_1 \subseteq \pi(G_1), \ldots, S_n \subseteq \pi(G_n)$ such that

$$\sum_{x \in S_i} x = \sum_{x \in S_j} x \qquad \text{for all } i, j \in \{1, \ldots, n\}.$$

Define a vector u in the following way. For all $i \in \{1, \ldots, n\}$, let $\widehat{G}_i \subseteq G_i$ be submultisets with $\pi(\widehat{G}_i) = S_i$ and set

$$\widetilde{u}_i \coloneqq \sum_{g \in \widehat{G}_i} g \in \ker^{\mathbb{Z}}(E_i).$$

Observe that vectors $\pi(\widetilde{u}_i)$ are equal for all $i \in \{1, \ldots, n\}$. This allows us to define u as the vector obtained by combining all the $\widetilde{u}_i$, so that projecting u to the variables of P_i yields $\widetilde{u}_i$, for each $i \in \{1, \ldots, n\}$. Note that since multisets $\widehat{G}_i$ are nonempty, u is a non-zero vector. Also $u \in \ker^{\mathbb{Z}}(A)$, since $\widetilde{u}_i \in \ker^{\mathbb{Z}}(E_i)$ for all $i \in \{1, \ldots, n\}$. Further, we have $u \sqsubseteq x^{\star} - x^{\diamond}$, because for all $i \in \{1, \ldots, n\}$,

$$\widetilde{u}_i = \sum_{g \in \widehat{G}_i} g \sqsubseteq \widetilde{x}_i - \widetilde{x}_i^{\diamond} \sqsubseteq \widetilde{x}_i^{\star} - \widetilde{x}_i^{\diamond}.$$

Thus, u satisfies all the requested properties.

We move to the second case: suppose that

$$\|\pi(\widetilde{x}_1^\star - \widetilde{x}_1^\diamond)\|_\infty \leq \rho \cdot f(k,\gamma).$$

Since we have $\|\pi(\widetilde{x}_i - \widetilde{x}_i^\diamond) - \pi(\widetilde{x}_1^\star - \widetilde{x}_1^\diamond)\|_\infty \leq \rho$ for all $i \in \{1,\ldots,n\}$, we have

$$\|\pi(\widetilde{x}_i - \widetilde{x}_i^\diamond)\|_\infty \leq \rho \cdot f(k,\gamma) + \rho \leq 2\rho \cdot f(k,\gamma) \qquad \text{for all } i \in \{1,\ldots,n\}.$$

Suppose for a moment that for some $i \in \{1,\ldots,n\}$, there exists an element $g \in G_i$ with $\pi(g) = 0$. Then by putting zeros on all the other coordinates, we can extend g to a vector $u \in \ker^{\mathbb{Z}}(A)$ which satisfies $u \sqsubseteq x^\star - x^\diamond$. As g is non-zero, so is u, hence u satisfies all the requested properties. Hence, from now on we may assume that no multiset G_i contains an element g with $\pi(g) = 0$.

Thus, we have that for all $i \in \{1,\ldots,n\}$, the multiset $\pi(G_i)$ consists of non-zero, sign compatible, integral vectors. It follows that

$$|G_i| = |\pi(G_i)| \leq \Big\| \sum_{x \in \pi(G_i)} x \Big\|_1 \leq k \Big\| \sum_{x \in \pi(G_i)} x \Big\|_\infty = k\|\pi(\widetilde{x}_i - \widetilde{x}_i^\diamond)\|_\infty \leq 2k\rho \cdot f(k,\gamma).$$

Since $\|g\|_\infty \leq \gamma$ for every element $g \in G_i$, we infer that

$$\|\widetilde{x}_i - \widetilde{x}_i^\diamond\|_\infty \leq \Big\| \sum_{g \in G_i} g \Big\|_\infty \leq \gamma |G_i| \leq 2k\gamma\rho \cdot f(k,\gamma).$$

By combining this with $\|\widetilde{x}_i - \widetilde{x}_i^\star\|_\infty \leq \rho$, we get

$$\|\widetilde{x}_i^\diamond - \widetilde{x}_i^\star\|_\infty \leq \|\widetilde{x}_i^\diamond - \widetilde{x}_i\|_\infty + \|\widetilde{x}_i - \widetilde{x}_i^\star\|_\infty \leq 2k\gamma\rho \cdot f(k,\gamma) + \rho \leq 3k\gamma\rho \cdot f(k,\gamma).$$

This implies that $\|x^\diamond - x^\star\| \leq 3k\gamma\rho \cdot f(k,\gamma)$. □

3.5 Solving the Linear Relaxation

In this section we prove Lemmas 3.3 and 3.4. For this we rely on results from Section 2.2, where the dual problem was considered.

In the following, we focus on the proof of Lemma 3.3. The proof of Lemma 3.4 follows from the same line of reasoning, so we only discuss necessary differences at the end. Throughout this section, we only work with linear programming without any integrality constraints, so for brevity we drop adjectives “fractional” in the notation.

For convenience, we work with linear programs in the following form:

$$\begin{aligned} \max\ & c^\mathsf{T} x \\ & Ax \leq b \\ & x \geq 0 \end{aligned} \tag{♣}$$

Note that every linear program in the form (♠) can be reduced to a linear program in the form (♣) by replacing each equality with two inequalities. This reduction preserves the primal depth of the constraint matrix, as well as being (p,q)-stochastic.

Thus, from now on let us fix a linear program $P = (x, A, b, c)$ in the form (♣). Our goal is to compute an optimal solution to P. Let t be the number of rows of A and $d := \mathsf{depth}_\mathsf{P}(A)$. We may assume that A has no columns with only zero entries, hence A has at most dt columns.

The reason for using form (♣) is that P admits a simple formulation of the dual linear program. Namely, the dual of P is the following linear program P^T:

$$\begin{aligned} \max\ & b^\mathsf{T} y \\ & A^\mathsf{T} y \leq c \\ & y \leq 0 \end{aligned}$$

We observe that with the help of Corollary 2.17, we can efficiently solve P^T.

Lemma 3.15. *One can compute an optimal solution $y^\star$ to P^T in time $\log^{\mathcal{O}(2^d)} t$, using t processors.*

Proof. By negating the variables and introducing a vector of slack variables z, one for every constraint in P^T, solving P^T is equivalent to solving the following linear program $\overline{P}^\mathsf{T}$

$$\begin{aligned} \min\ & b^\mathsf{T} y \\ & -A^\mathsf{T} y + Iz = c \\ & y \geq 0, z \geq 0 \end{aligned}$$

More precisely, optimal solutions of P^T can be obtained from optimal solutions of $\overline{P}^\mathsf{T}$ by dropping the z variables and negating the y variables. Note that

$$\mathsf{depth}_\mathsf{D}\begin{pmatrix} -A^\mathsf{T} & I \end{pmatrix} = \mathsf{depth}_\mathsf{P}(A) = d.$$

Since $\overline{P}^\mathsf{T}$ is in the form (♠) and its constraint matrix has at most $t + dt$ columns, we may use Corollary 2.17 to find an optimal solution to $\overline{P}^\mathsf{T}$ in time $\log^{2^{\mathcal{O}(d)}}(t)$. Consequently, within the same asymptotic running time we can find an optimal solution $y^\star$ to P^T. □

By applying the algorithm of Lemma 3.15, we may assume that we have an optimal solution $y^\star$ to the dual program $P^\intercal$. Classic linear programming duality tells us that the optimum values of the programs P and $P^\intercal$ are equal. In other words, if we denote

$$\lambda \coloneqq b^\intercal y^\star,$$

then $\lambda = \mathrm{opt}^{\mathbb{R}}(P) = \mathrm{opt}^{\mathbb{R}}(P^\intercal)$. Note that λ can be computed from $y^\star$ in time $\mathcal{O}(\log t)$. However, we are interested in computing not only the optimum value of a solution to P — which is λ — but we would like to actually find some optimal solution.

To this end, we will exploit the knowledge of $y^\star$ trough the complementary slackness conditions. Let us denote the consecutive variables of x as $x_1, \ldots, x_m$, where m is the number of columns of A, and similarly enumerate the entries of y, b, and c. Also, let the consecutive columns of A be $a_{\circ,1}, \ldots, a_{\circ,m}$ and the consecutive rows of A be $a_{1,\circ}^\intercal, \ldots, a_{t,\circ}^\intercal$. The following claim captures the assertions that can be inferred from the complementary slackness conditions.

Claim 3.16. *There exists an optimal solution $x^\star$ to P satisfying the following properties:*

(S1) For every $i \in \{1, \ldots, t\}$ such that $y_i^\star < 0$, we have $a_{i,\circ}^\intercal x^\star = b_i$.
(S2) For every $j \in \{1, \ldots, m\}$ such that $a_{\circ,j}^\intercal y^\star < c_j$, we have $x_j^\star = 0$.

Let

$$X \coloneqq \{i \colon y_i^\star < 0\} \subseteq \{1, \ldots, t\} \qquad \text{and} \qquad Y \coloneqq \{j \colon a_j^\intercal y^\star < c_j\} \subseteq \{1, \ldots, m\}$$

be the sets of indices to which the implications of Claim 3.16 apply. Before we continue, let us discuss computing X and Y in the PRAM model using t processors. Obviously, X can be computed in time $\mathcal{O}(1)$. As for Y, we claim that it can be computed in time $d^{\mathcal{O}(1)} \cdot \log t$. Observe that computing the inner products $a_{\circ,j}^\intercal y^\star$ for all $j \in \{1, \ldots, m\}$ boils down to computing m sums, where the jth sum ranges over the list of non-zero entries in column $a_{\circ,j}$. Such lists can be computed in time $d^{\mathcal{O}(1)} \cdot \log t$ by sorting the list of non-zero entries of A in lexicographic order (first by the column index and then by the row index), and then splitting it appropriately. Since the total length of the lists is at most $d\,t$, their sums can be computed in time $d^{\mathcal{O}(1)} \cdot \log t$ on t processors by assigning to each list a number of processors proportional to its length.

Thus, we may assume that the sets X and Y are computed. Let $\overline{X} = \{1, \ldots, t\} \setminus X$. We introduce the following notation.

- Let $\widetilde{x}$, $\widetilde{c}$, $\widetilde{y}$, $\widetilde{b}$ be obtained from x, c, y, b by removing all the entries with indices in Y, Y, $\overline{X}$, and $\overline{X}$, respectively.
- Let $\widetilde{\widetilde{x}}$ be the vector of the remaining variables of x, i.e., those with indices in Y.
- Let $\widetilde{A}$ be the matrix obtained from A by removing all rows with indices in $\overline{X}$ and all columns with indices in Y.

The notation is extended naturally to solutions $x^\star$, $y^\star$, etc. Note that all these objects can be

computed in time $d^{\mathcal{O}(1)}$ using t processors.

Observe that

$$\widetilde{A}^{\mathsf{T}}\widetilde{y}^{\star} = \widetilde{c}, \tag{3.9}$$

as in the solution $y^{\star}$ to P^{T}, all constraints with indices outside of Y are tight by the definition of Y, while the entries of $y^{\star}$ outside of $\widetilde{y}^{\star}$ are zeros anyway. Further, Claim 3.16 can be rewritten as follows.

Claim 3.17. *There exists an optimal solution $x^{\star}$ to P such that*

$$\widetilde{A}\widetilde{x}^{\star} = \widetilde{b} \qquad \textit{and} \qquad \widetilde{\widetilde{x}}^{\star} = 0. \tag{3.10}$$

The next lemma explains the main gain provided by the complementary slackness conditions: if a solution to P satisfies the equations given in Claim 3.17, then it automatically is an optimal solution.

Lemma 3.18. *Suppose a vector $x^{\star} \in \mathbb{R}^m_{\geq 0}$ satisfies* (3.10)*. Then $c^{\mathsf{T}}x^{\star} = \lambda$.*

Proof. Observe that

$$\begin{aligned}
c^{\mathsf{T}}x^{\star} &= \widetilde{c}^{\mathsf{T}}\widetilde{x}^{\star} && \text{by } \widetilde{\widetilde{x}}^{\star} = 0\\
&= \left(\widetilde{A}^{\mathsf{T}}\widetilde{y}^{\star}\right)^{\mathsf{T}}\widetilde{x}^{\star} && \text{by (3.9)}\\
&= (\widetilde{y}^{\star})^{\mathsf{T}}\widetilde{A}\widetilde{x}^{\star}\\
&= (\widetilde{y}^{\star})^{\mathsf{T}}\widetilde{b} && \text{by } \widetilde{A}\widetilde{x}^{\star} = \widetilde{b}\\
&= \widetilde{b}^{\mathsf{T}}\widetilde{y}^{\star} = \lambda, && \text{as the value is a scalar}
\end{aligned}$$

as claimed. □

Now, consider the following linear program $\widehat{P}$ and recall that x_1 denotes the first variable of x.

$$\begin{aligned}
&\min\ x_1 &&\\
&Ax \leq b, && \widetilde{A}\widetilde{x} = \widetilde{b},\\
&x \geq 0, && \widetilde{\widetilde{x}} = 0.
\end{aligned}$$

By Claim 3.17, there exists an optimal solution to P which is also a feasible solution to $\widehat{P}$. On the other hand, by Lemma 3.18, every feasible solution to $\widehat{P}$ is actually an optimal solution to P. Therefore, there exists an optimal solution $x^{\star}$ to P that satisfies $x^{\star}_1 = \mathrm{opt}^{\mathbb{R}}(\widehat{P})$.

Now observe that the value $\mathrm{opt}^{\mathbb{R}}(\widehat{P})$ can be computed in time $\log^{\mathcal{O}(2^d)}(t)$ using the same approach as the one used in Lemma 3.15. Namely, we eliminate all the variables of $\widetilde{\widetilde{x}}$ from $\widehat{P}$ by just substituting them with zeroes, and replace each equality from $\widetilde{A}\widetilde{x} = \widetilde{b}$ with two inequalities. In this way, we obtain an equivalent linear program in the form (♣) with at most

$3n$ constraints and whose constraint matrix has primal depth at most d. Using the approach from Lemma 3.15 we compute an optimal solution to the dual of this program, whose value coincides with $\text{opt}^{\mathbb{R}}(\widehat{P})$.

To summarize, we established the following claim.

Claim 3.19. *In time* $\log^{\mathcal{O}(2^d)}(t)$ *we may compute the value* $\xi = \text{opt}^{\mathbb{R}}(\widehat{P})$*, with the following property: there exists an optimal solution* $x^\star$ *to* P *such that* $x^\star_1 = \xi$*.*

We now use Claim 3.19 in the following recursive algorithm for finding an optimal solution to P:

- If the constraint matrix A is block decomposable, say $D_1, \ldots, D_n$ ($n \geq 2$) are the blocks of the block decomposition of A, then decompose P into n independent programs $P_1, \ldots, P_n$ with constraint matrices $D_1, \ldots, D_n$, respectively. Solve these programs recursively in parallel, by assigning to each program P_i the number of processors equal to the number of rows of D_i. Then combine the obtained optimal solutions to $P_1, \ldots, P_n$ into an optimal solution to P.

- If the constraint matrix A is not block decomposable, then it can be written as $(a_{\circ,1}\ A')$, where $a_{\circ,1}$ is the first column of A and A' is a matrix such that $\mathsf{depth}_{\mathsf{P}}(A') < \mathsf{depth}_{\mathsf{P}}(A)$. Use Claim 3.19 to find, in time $\log^{\mathcal{O}(2^d)}(t)$, a value ξ such that there exists an optimal solution to P setting the first variable to ξ. Now, consider the linear program P' defined as

$$\begin{aligned} \min\ & c'^{\mathsf{T}} x' \\ & A' x' \leq b - \xi \cdot a_{\circ,1} \\ & x' \geq 0 \end{aligned}$$

  where c' and x' are c and x with the first entry removed, respectively. Apply the algorithm recursively to P', noting that its constraint matrix A' has a strictly smaller primal depth than A. Finally, an optimal solution to P can be obtained from the computed optimal solution to P' by assigning value ξ to the first variable.

The correctness of this algorithm follows from Claim 3.19 in a straightforward manner. As for the running time, observe that when the algorithm considers a linear program with a constraint matrix which is not block decomposable, it recurses on a linear program with a strictly smaller primal depth. On the other hand, when the algorithm considers a linear program with a block decomposable constraint matrix, it recurses on several linear programs whose constraint matrices are not block decomposable. It follows that if the initial linear program P has primal depth d, then the recursion has depth at most $2d$. As each level of the recursion is done in parallel in time $\log^{\mathcal{O}(2^d)}(t)$, the total running time of $\log^{\mathcal{O}(2^d)}(t)$ follows.

This concludes the proof of Lemma 3.3. The proof of Lemma 3.4 is done in exactly the same manner, except that the usage of Corollary 2.17 is replaced with Corollary 2.16, noting that the linear programs in question are (p,q)-stochastic. Also, the recursion has depth $2(p+q)$ instead of $2d$.

Part II Geometric Independent Sets

4 Parameterized Approximation for Maximum Weight Independent Set of Axis-Parallel Rectangles and Segments

This chapter contains a overworked version of [CPW22], which is joint work with Michał Pilipczuk and Karol Węgrzycki.

4.1 Introduction

In the *Maximum Weight Independent Set of Rectangles* (MWISR) problem, we are given a set $\mathcal{D}$ consisting of n axis-parallel rectangles in the plane, and a weight function $\omega\colon \mathcal{D} \to \mathbb{R}$. A feasible solution $\mathcal{S} \subseteq \mathcal{D}$ to the MWISR problem consists of a set of multiple disjoint rectangles, i.e., for any two different rectangles $R, R' \in \mathcal{S}$ in the solution, we have $R \cap R' = \emptyset$; we also call such a solution an *independent set*. The objective is to find a feasible solution of maximum total weight. In this chapter, we consider a parameterized setting of the problem. We use parameter $k \in \mathbb{N}$ to denote the *cardinality* of the solution. The maximum possible weight of an independent set in $\mathcal{D}$ whose cardinality is at most k, is denoted by $\mathrm{opt}_k(\mathcal{D})$.

MWISR is a fundamental problem in geometric optimization. It naturally arises in various applications, such as map labeling [AvKS98, DF92], data mining [FMMT01], routing [LNO02], or unsplittable flow routing [BSW14]. MWISR is well-known to be NP-hard [FPT82], and it admits a QPTAS [AW13]. The currently best approximation factor achievable in polynomial time is $\mathcal{O}(\log\log(n))$ [CW21]. From the parameterized perspective, it is known that the problem is W[1]-hard when parameterized by k, the number of rectangles in the solution, even in the unweighted setting and when all the rectangles are squares [Mar05]. Therefore, it is unlikely that there is an exact algorithm with a running time of the form $f(k)n^{\mathcal{O}(1)}$ for some computable function f, even in this restricted setting. In particular, this also excludes any $(1-\varepsilon)$-approximation algorithm running in $f(\varepsilon)n^{\mathcal{O}(1)}$ time [Baz95, CT97]. We note that in the case of weighted squares, there is a PTAS with a running time of the form $n^{g(\varepsilon)}$ [EJS05], where g is a computable function.

Approximating MWISR becomes easier in the unweighted setting (i.e. all costs are set to 1). With this restriction, constant factor approximation algorithms for MWISR are known [Mit21, GKM+22], and there is a QPTAS with a better running time [CE16]. Grandoni et al. [GKW19] were the first to consider parameterized approximation for the MWISR problem, although in the unweighted setting. They gave a parameterized approximation scheme for unweighted MWISR running in $k^{\mathcal{O}(k/\varepsilon^8)} n^{\mathcal{O}(1/\varepsilon^8)}$ time. As an open problem, they asked if one can also design a parameterized approximation scheme in the weighted setting. The question therefore is: Does Maximum Independent Set of Rectangles admit a parameterized approximation scheme in the weighted setting?

Our contribution

In this chapter we partially answer the open question of Grandoni et al. [GKW19] by proving the following result:

Theorem 4.1. *Suppose $\mathcal{D}$ is a set of axis-parallel rectangles in the plane with positive weights. Then given $k \in \mathbb{N}$ and $\varepsilon > 0$, one can in time*

$$2^{\mathcal{O}(k \log(k/\varepsilon))} |\mathcal{D}|^{\mathcal{O}(1/\varepsilon)}$$

find an independent set in $\mathcal{D}$ of weight at least $(1-\varepsilon)\operatorname{opt}_k(\mathcal{D})$.

Note that there is a caveat in Theorem 4.1: the returned solution may actually be of cardinality larger than k, but there is a guarantee that it will be an independent set. This is what we mean by a partial resolution of the question of Grandoni et al. [GKW19]: ideally, we would like the algorithm to return a solution of weight at least $(1-\varepsilon)\operatorname{opt}_k(\mathcal{D})$ *and* of cardinality at most k. At this point we are able to give such an algorithm only in the restricted case of axis-parallel segments (see Theorem 4.2 below), but we postpone this discussion and focus now on Theorem 4.1. Observe here that the issue of solutions with cardinality larger than k becomes immaterial in the unweighted case. Hence Theorem 4.1 applied to the unweighted setting solves the problem considered by Grandoni et al. [GKW19] and actually improves their running time.

We briefly describe the technical ideas behind the proof of Theorem 4.1. Similarly to Grandoni et al. [GKW19], the starting point is a polynomial-time construction of a grid such that each rectangle in $\mathcal{D}$ contains at least one gridpoint. However, in order to take care of the weights, our grid is of size $(2k^2/\varepsilon) \times (2k^2/\varepsilon)$. Moreover, already in this step, we may return an independent set with weight at least $(1-\varepsilon)\operatorname{opt}_k$ consisting of more than k rectangles. This is the only step where the algorithm may return more than k rectangles.

After this step, the similarities to the algorithm of Grandoni et al. [GKW19] end. We introduce the notion of the *combinatorial type* of a solution. This is simply a mapping from each rectangle in the solution to the set of all gridpoints contained in it. Observe that since the size of the grid is bounded by a function of k and ε, we can afford to guess (by branching into all

possibilities) the combinatorial type of an optimum solution in $f(k,\varepsilon)$ time. Note that there may be many different rectangles matching the type of a rectangle from the optimum solution. However, it is possible that such a rectangle overlaps with the neighboring rectangles (and violates independence). Therefore, we need to define constraints preventing rectangles from overlapping. For this, we construct an instance of the *Arity*-2 *Valued Constraint Satisfaction Problem* (2-VCSP) based on the guessed combinatorial type (see Section 1.3.1 for a definition of 2-VCSP).

Next, we observe that this instance induces a graph that is almost planar, hence we may apply a variant of Baker's shifting technique [Bak94]. This allows us to divide the instance into many independent instances of 2-VCSP while removing only $\varepsilon \cdot \mathrm{opt}_k$ weight from the optimum solution. Moreover, each of these independent instances induces a graph of bounded treewidth, and hence can be solved exactly in $|\mathcal{D}|^{\mathcal{O}(1/\varepsilon)}$ time. This concludes a short sketch of our approach.

Let us return to the apparent issue that our algorithm may return a solution with cardinality larger than k. This may happen in the very first step of the procedure, during the construction of the grid. By employing a completely different technique, we can circumvent this problem in the restricted setting of axis-parallel segments and prove the following result.

Theorem 4.2. *Suppose $\mathcal{D}$ is a set of axis-parallel segments in the plane with positive weights. Then given k and $\varepsilon > 0$, one can in time*

$$2^{\mathcal{O}(k^2 \log(k/\varepsilon))}|\mathcal{D}|^{\mathcal{O}(1)}$$

find an independent set in $\mathcal{D}$ of cardinality at most k and weight at least $(1-\varepsilon)\,\mathrm{opt}_k(\mathcal{D})$.

Kára and Kratochvíl [KK06] and Marx [Mar06] independently observed that the problem of finding a maximum cardinality independent set of axis-parallel segments admits an fpt algorithm. However, their approach heavily relies on the fact that the problem is unweighted. In this setting our approach is different: In fact, we show that finding a maximum weight independent set of axis-parallel segments admits an algorithm with running time $W^{\mathcal{O}(k^2)}|\mathcal{D}|^{\mathcal{O}(1)}$, where W is the number of distinct weights present among the segments.

We proceed with an outline of the proof of Theorem 4.2 and highlight some technical ideas. First, we modify the instance such that the number of different weights is bounded. This is done through guessing the largest weight of a rectangle in an optimum solution and rounding the weights down. This is the only place where we lose accuracy on the optimal solution. In other words, the algorithm is fixed-parameter tractable in k and the number of distinct weights $W = (k/\varepsilon)^{\mathcal{O}(1)}$.

With this assumption, we then construct a grid with $\mathcal{O}(k^2)$ lines hitting every segment of the instance. We say that the grid is *nice* with respect to a segment I, if I contains a grid point; equivalently, I is nice if it is hit by two orthogonal lines of the grid. Observe that the constructed grid is not necessarily nice for every segment of the instance. We adapt

the previously introduced notion of the *combinatorial type* in order to also accommodate segments which do not contain a grid point. This is done by mapping the segment to its four neighboring grid lines instead of the grid points contained inside the segment. Further, the weight of the segment is added to its combinatorial type. Similarly to before, the combinatorial type of a segment only depends on the grid size and the number of distinct weights. This allows to guess (by branching into all possibilities) the combinatorial type of the optimum solution $\mathcal{S}$ in $k^{\mathcal{O}(1)} \cdot W$ time.

The goal is to construct a grid which is nice with respect to all segments of an optimal solution $\mathcal{S}$. For this, we start by guessing the combinatorial type of all nice segments of an optimal solution $\mathcal{S}$. Then, we incrementally guess the combinatorial type of the next heaviest segment in $\mathcal{S}$ for which the grid is not yet nice. For each such combinatorial type, we find all possible candidate segments and add at most k lines to the grid G. This ensures that a correct candidate segments is hit both directions. Repeating this procedure at most k times we end up with a grid which is nice with respect to all the segments of $\mathcal{S}$.

Given such a grid, it remains to guess the combinatorial type of all segments in $\mathcal{S}$ and solve resulting instance. This can be done either greedily or by observing that the problem can be modeled as a 2-CSP instance whose constraint graph is a union of paths. Both these cases work due to the fact that the segments only interact with each other when they lie on the same grid line.

Structure of the chapter

Section 4.2 discusses the case of weighted axis-parallel rectangles. That is, we proof Theorem 4.1. We then consider the case of axis-parallel segments and prove Theorem 4.2, in Section 4.3.

4.2 Axis-Parallel Rectangles

In this section we prove Theorem 4.1. For this, let $\mathcal{D}$ be a set of weighted axis-parallel rectangles and $\omega\colon \mathcal{D} \to \mathbb{R}$ a weight function on the rectangles in $\mathcal{D}$. Note that we can assume all weights to be positive, since we can always drop rectangles with negative weight from the solution. Thus only improving the found solution. By $\mathrm{opt}_k(\mathcal{D})$ we denote the maximum possible weight of a set consisting of at most k disjoint rectangles in $\mathcal{D}$. An *optimal solution* $\mathcal{S}$ is a set $\mathcal{S} \subseteq \mathcal{D}$ of cardinality at most k satisfying $\omega(\mathcal{S}) = \mathrm{opt}_k(\mathcal{D})$.

We start with a simple preprocessing on $\mathcal{D}$. First, we guess a rectangle $R_{\max} \in \mathcal{S}$ with maximum weight among all rectangles of $\mathcal{S}$. This can be done with an extra overhead of $\mathcal{O}(n)$ in the running time. Observe that $\omega(R_{\max}) \geq \mathrm{opt}_k(\mathcal{D})/k$. Further, we remove from $\mathcal{D}$ every rectangle of weight larger than $\omega(R_{\max})$ and every rectangle of weight not exceeding $\varepsilon\omega(R_{\max})/k$; denote the resulting instance by $\mathcal{D}'$. Observe that this operation does not decrease the optimum

significantly, as none of the former rectangles and at most k of the latter rectangles could be used in $\mathcal{S}$. More precisely, we have

$$\mathrm{opt}_k(\mathcal{D}') \geq \mathrm{opt}_k(\mathcal{D}) - k \cdot \frac{\varepsilon \cdot \omega(R_{\max})}{k} \geq (1-\varepsilon)\,\mathrm{opt}_k(\mathcal{D}).$$

After this preprocessing, the optimum value decreased by at most $\varepsilon \cdot \mathrm{opt}_k(\mathcal{D})$. This concludes the description of preprocessing. Hence, without loss of generality, we assume our instance is $\mathcal{D} := \mathcal{D}'$.

4.2.1 Constructing a Grid

Recall that a grid is a set of horizontal and vertical grid lines. Such a grid G is *good* for a set of axis-parallel rectangles $\mathcal{D}$, if for every rectangle $R \in \mathcal{D}$ there is a grid point of G contained in R.

As mentioned in Section 4.1, our search for an optimal solution pivots around a bounded size grid which is good for the optimum solution S_{opt}. The construction of this grid is encapsulated in the following lemma.

Lemma 4.3. *Suppose we are given set $\mathcal{D}$ of axis-parallel rectangles with weight function ω and an integer $k \in \mathbb{N}$. Let $\Delta(\mathcal{D})$ be the ratio between lowest and highest weight in $\mathcal{D}$ and suppose $\Delta(\mathcal{D}) \geq \varepsilon / k$ for some $\varepsilon > 0$. Then, in polynomial time, one can either*

- *compute a grid G of size $|G| \leq \frac{2k^2}{\varepsilon}$ that is good for $\mathcal{S} \subseteq \mathcal{D}$, or*
- *return an independent set $\mathcal{I} \subseteq \mathcal{D}$ with $\omega(\mathcal{I}) \geq \mathrm{opt}_k(\mathcal{D})$.*

Proof. We construct the grid G by first constructing the vertical lines of G, and then with basically the same procedure we add the horizontal lines of G. For the construction of the vertical lines, we iteratively pick vertically disjoint rectangles in a greedy fashion. For every rectangle $R \in \mathcal{D}$, select a point $p_R \in R$ very close to the top-right corner of R. We start with $\mathcal{D}_1 := \mathcal{D}$. In iteration $i \in \mathbb{N}$, we select a rectangle $R_i^{\mathrm{ver}} \in \mathcal{D}_i$ for which $p_i := p_{R_i^{\mathrm{ver}}}$ is the leftmost among rectangles of $\mathcal{D}_i$. In case of ties, select any of the tying rectangles. Then, add the vertical line ℓ_i^{ver} which contains p_i to the grid. Next, delete every rectangle from $\mathcal{D}_i$ intersecting ℓ_i^{ver}, thus obtaining the next set $\mathcal{D}_{i+1}$. We repeat this procedure until no more rectangles are left in $\mathcal{D}_i$. To finish the construction of G, repeat the above algorithm in the orthogonal direction, thus selecting vertically disjoint rectangles R_i^{hor} and adding to G horizontal lines ℓ_i^{hor}. This concludes the construction of G; see Figure 4.1 for an illustration.

Trivially, the above algorithm runs in polynomial time. Moreover, it returns a good grid since every rectangle in $\mathcal{D}$ is intersected by some horizontal and some vertical line from G. So if $|G| \leq \frac{2k^2}{\varepsilon}$, we can just return G as the output of the algorithm.

It remains to show that if $|G| > \frac{2k^2}{\varepsilon}$, then we can find an independent set of weight at least $\mathrm{opt}_k(\mathcal{D})$. Assuming that $|G| > \frac{2k^2}{\varepsilon}$, either the vertical or the horizontal run of the greedy algorithm returned more than $\frac{k^2}{\varepsilon}$ lines. Without loss of generality assume that the vertical run

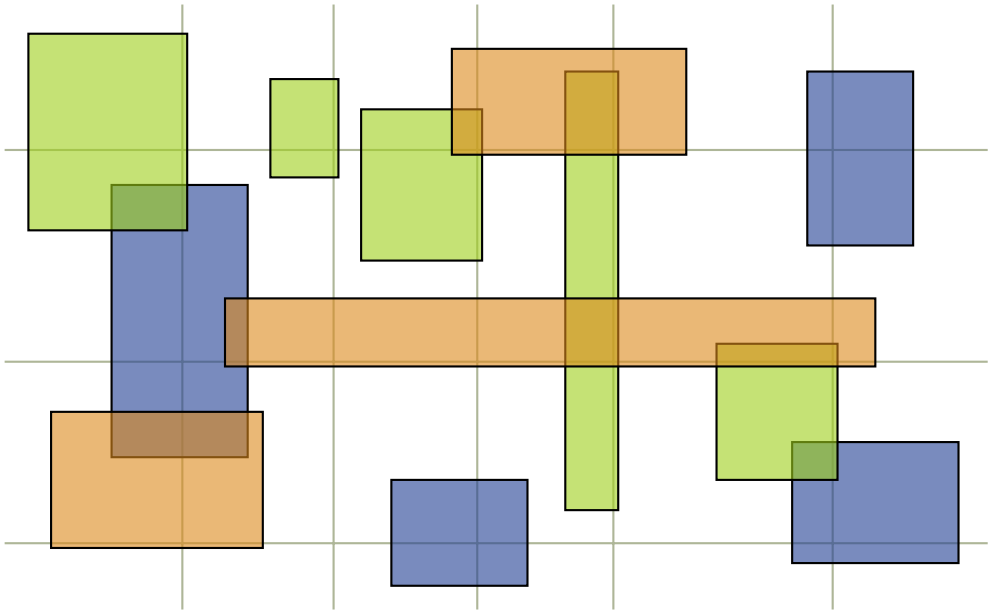

Figure 4.1: The grid constructed after applying the greedy procedure. Rectangles R_i^{ver} are green-filled, and rectangles R_i^{hor} are orange-filled. Observe that every rectangle is hit by at least one grid point.

constructed rectangles $R_1^{\text{ver}}, \ldots, R_m^{\text{ver}}$ for some $m > \frac{k^2}{\varepsilon}$. Observe that these rectangles form an independent set, because after iteration $i \in \{1, \ldots, m\}$ all rectangles with left side to the left of ℓ_i are removed. Since we assumed that the ratio between lowest and highest weight of a rectangle in $\mathcal{D}$ is at least ε/k, we may estimate the weight of $\{R_1^{\text{ver}}, \ldots, R_m^{\text{ver}}\}$ as follows:

$$\sum_{i=1}^{m} \omega(R_i^{\text{ver}}) \geq m \cdot \frac{\varepsilon \cdot \omega(R_{\max})}{k} \geq k \cdot \omega(R_{\max}) \geq \text{opt}_k(\mathcal{D}),$$

where $R_{\max}$ is the rectangle of highest weight in $\mathcal{D}$. Therefore, the rectangles $R_1^{\text{ver}}, \ldots, R_m^{\text{ver}}$ form a feasible output for the second point of the lemma statement. □

The first step of the algorithm is to run the procedure of Lemma 4.3. If this procedure returns an independent set of weight at least $\text{opt}_k(\mathcal{D})$, we just return it as a valid output and terminate the algorithm. Otherwise, from now on we may assume that we have constructed a grid G of size at most $2k^2/\varepsilon$ and that this grid is good for $\mathcal{S}$.

4.2.2 Combinatorial Types

Next, we define the notion of the combinatorial type of a rectangle with respect to a grid. This can be understood as a rough description of the position of the rectangle with respect to the grid.

Let G be a grid. For an axis-parallel rectangle R, we define the *combinatorial type* $T(R)$ of R with respect to G as

$$T_G(R) := R \cap \text{points}(G).$$

In other words, $T_G(R)$ is the set of grid points of G contained in R. For a set $\mathcal{S}$ of axis-parallel rectangles, the *combinatorial type* of $\mathcal{S}$ is $T_G(\mathcal{S})$, that is, the image of $\mathcal{S}$ under T_G. By Λ_k^G we denote the set of all possible combinatorial types with respect to G of sets $\mathcal{S}$ consisting of at most k axis-parallel rectangles. Observe that if a grid is small, there are only few combinatorial types on it.

Lemma 4.4. *For every grid G and positive integer k, we have $|\Lambda_k^G| \leq 2^{\mathcal{O}(k \log |G|)}$. Moreover, given G and k, Λ_k^G can be constructed in time $2^{\mathcal{O}(k \log |G|)}$.*

Proof. The combinatorial type of any axis-parallel rectangle R can be completely characterized by four lines (or lack thereof) in G: the left-most and the right-most vertical line of G intersecting R, and the top-most and the bottom-most horizontal line of G intersecting R. Hence, there are at most $(|G|+1)^4$ candidates for the combinatorial type of a single rectangle. It follows that the number of combinatorial types coming from sets of at most k rectangles is bounded by

$$1 + (|G|+1)^4 + (|G|+1)^8 + \cdots + (|G|+1)^{8k} \in 2^{\mathcal{O}(k \log |G|)}.$$

To construct Λ_k^G in time $2^{\mathcal{O}(k \log |G|)}$, just enumerate all possibilities as above. □

The next step of the algorithm is as follows. Recall that we work with a grid G of size at most $2k^2/\varepsilon$ that is good for $\mathcal{S}$. By Lemma 4.4, we can compute Λ_k^G in time $2^{\mathcal{O}(k \log(k/\varepsilon))}$ and we have $|\Lambda_k^G| \leq 2^{\mathcal{O}(k \log(k/\varepsilon))}$. Hence, by paying a $2^{\mathcal{O}(k \log(k/\varepsilon))}$ overhead in the time complexity, we can guess $\mathcal{T} := T_G(\text{opt}_k(G))$, that is, the combinatorial type of the optimum solution. From now on assume that the combinatorial type $\mathcal{T}$ is fixed. Since $\mathcal{S}$ is an independent set and G is good for $\mathcal{S}$, we may assume that sets in $\mathcal{T}$ are pairwise disjoint and nonempty.

4.2.3 Reduction to 2-VCSP

We say that a rectangle $R \in \mathcal{D}$ *matches* a subset of grid points $A \subseteq \text{points}(G)$ if $T_G(R) = A$, that is, $R \cap \text{points}(G) = A$. By $\mathcal{D}_A \subseteq \mathcal{D}$ we denote the set of rectangles from $\mathcal{D}$ matching A.

Based on the combinatorial type $\mathcal{T}$ we define an instance $I_{\mathcal{T}}$ of 2-VCSP as follows. The set of variables is $\mathcal{T}$. For every $A \in \mathcal{T}$, the domain of A is $\mathcal{D}_A \cup \{\bot\}$. That is, the set of rectangles from $\mathcal{D}$ matching A plus a special symbol $\bot$ denoting that non of the rectangle matching A are taken to the solution. Also, for every $A \in \mathcal{T}$ we add a unary constraint c_A on A with associated revenue function $f_{c_A} : \mathcal{D}_A \cup \{\bot\} \to \mathbb{R}$ defined as $f_{c_A}(R) = \omega(R)$ for each $R \in \mathcal{D}_A$ and $f_{c_A}(\bot) = 0$. In the definition of 2-VCSP only binary constraints are allowed, but a unary constraint can be modeled by a binary constraint which binds a variable with itself.

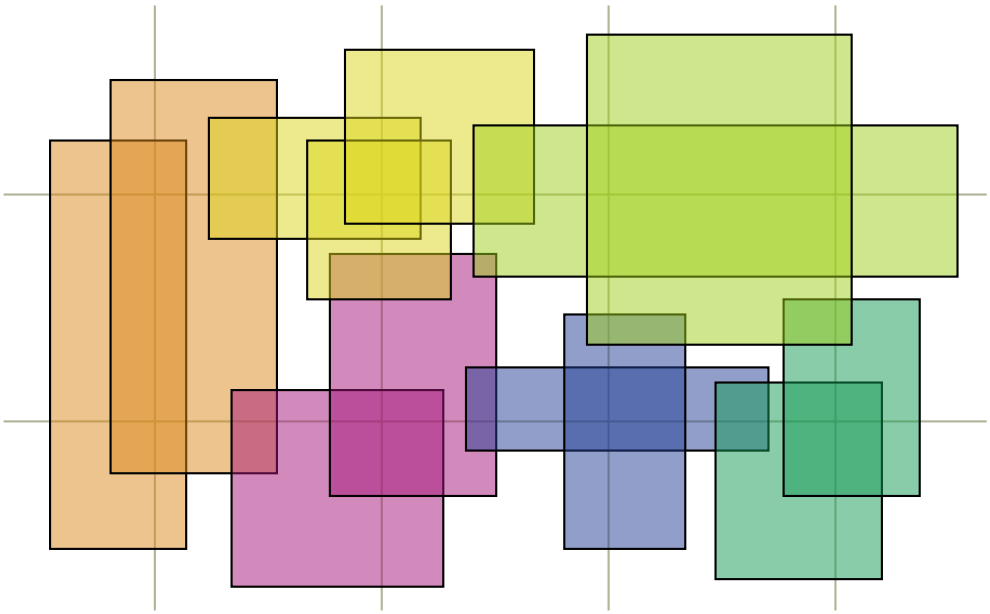

Figure 4.2: The instance after guessing the combinatorial type $\mathcal{T}$. Rectangles matching the same type $A \in \mathcal{T}$ are filled with the same color. Each variable $A \in \mathcal{T}$ corresponds to a different rectangle in the optimum solution. In this figure, the variables are shown as different colors. The domain $\mathcal{D}_A$ of a variable $A \in \mathcal{T}$ consists of all rectangles in the corresponding color.

It remains to define binary constraints binding pairs of distinct variables in $I_{\mathcal{T}}$. Two distinct grid points of G are *adjacent* if they lie on the same or on consecutive horizontal lines of G, and on the same or on consecutive vertical lines of G. We put a binary constraint $c_{A,B}$ binding variables $A, B \in \mathcal{T}$ if there exist adjacent grid points $p \in A$ and $q \in B$. The revenue function for $c_{A,B}$ is defined as follows: for $R_A \in \mathcal{D}_A \cup \{\perp\}$ and $R_B \in \mathcal{T}_B \cup \{\perp\}$, we set

$$f_{c_{A,B}}(R_A, R_B) = \begin{cases} -\infty & \text{if } R_A \in \mathcal{D}_A \text{ and } R_B \in \mathcal{D}_B \text{ intersect;} \\ 0 & \text{otherwise.} \end{cases}.$$

In other words, $c_{A,B}$ is a hard constraint: we require that the rectangles assigned to A and B are disjoint (or one of them is nonexistent), as otherwise the revenue is $-\infty$. This concludes the construction of the instance of $I_{\mathcal{T}}$; clearly, it can be done in polynomial time.

The instance $I_{\mathcal{T}}$ is constructed so that it corresponds to the problem of selecting disjoint rectangles from $\mathcal{D}$ which match the combinatorial type $\mathcal{T}$. This is formalized in the following statement.

Claim 4.5. *If $\mathcal{S} \subseteq \mathcal{D}$ is an independent set of rectangles with combinatorial type $\mathcal{T}$, then there exists a solution to $I_{\mathcal{T}}$ with revenue equal to $\omega(\mathcal{S})$. Conversely, if there exists a solution to $I_{\mathcal{T}}$ with revenue $r \geq 0$, then there exists an independent set $\mathcal{S} \subseteq \mathcal{D}$ of weight r and cardinality at most k.*

Proof. For the first implication, we construct an assignment $u\colon \mathcal{T} \to \mathcal{D}$. For each $A \in \mathcal{T}$, set $u(A)$ to be the unique rectangle $R \in \mathcal{S}$ for which $T_G(R) = A$. To see that the revenue of u is equal to $\omega(\mathcal{S})$, note that for every $A \in \mathcal{T}$ the unary constraint c_A yields revenue $\omega(u(A))$, while all binary constraints yield revenue 0, because the rectangles are pairwise disjoint.

For the second implication, let $\mathcal{S} \subseteq \mathcal{D}$ be the image of the optimal assignment u, where $\perp$ is removed. Clearly, $|\mathcal{S}| \leq |\mathcal{T}| \leq k$. Since u yields a non-negative revenue, all binary constraints must give revenue 0, hence $\omega(\mathcal{S})$ is equal to the revenue r of u. It remains to argue that $\mathcal{S}$ is an independent set. For this, take any distinct $A, B \in \mathcal{T}$; we need to argue that in case when rectangles $R_A := u(A)$ and $R_B := u(B)$ are both not equal to $\perp$, they are disjoint. For the sake of contradiction, suppose that R_A and R_B have some common point x. Let Q be the cell of the grid G containing x. Since $x \in R_A$ and A is nonempty, A must contain at least one corner of Q, say p. Recall that A is nonempty by the assumption of G being good for $\mathrm{opt}_k(\mathcal{D})$. Similarly, B contains a corner of Q, say q. Note that p and q are adjacent grid points, hence in $I_{\mathcal{T}}$ there is a constraint $c_{A,B}$ which yields a revenue of $-\infty$ in the case when the rectangles assigned to A and B intersect. By assumption, this is the case in u and we obtain a contradiction with the assumption $r \geq 0$. □

4.2.4 Almost Planarity of the Gaifman Graph

Let H be the Gaifman graph of $I_{\mathcal{T}}$; see Figure 4.3 for an example. Recall that the vertex set of H is $\mathcal{T}$, and distinct $A, B \in \mathcal{T}$ are considered adjacent in H if and only if there is a grid cell Q of G such that both A and B contain a corner of Q. Without loss of generality assume that H is connected, as otherwise we solve a $I_{\mathcal{T}}$ by treating each connected component separately and joining the solutions.

Note that the graph H is not necessarily planar, as there might be crossings within cells; this happens when all four corners belong to different elements of $\mathcal{T}$. However, the intuition is that the crossings within cells are the only problem, hence H is almost planar. We would like to apply Baker's technique [Bak94] on H, which is only possible directly applicable to planar graphs. In this setting we still apply the technique with the caveat that we need to be careful about the aforementioned crossings. For this, the following construction will be useful.

Call a grid cell Q a *cross* if Q has four corners and all those four corners belong to pairwise different elements of $\mathcal{T}$. Note that these elements form a 4-clique in H. Construct a graph $H^\bullet$ from H as follows: For every cross Q, add Q to the vertex set and make it adjacent to all four elements of $\mathcal{T}$ containing the corners of Q. Further, remove the two diagonal edges, that is the edge between the two elements of $\mathcal{T}$ containing the top-left, respectively the bottom-right corner of Q and the edge between the two elements of $\mathcal{T}$ containing the top-right respectively bottom-left corner of Q.

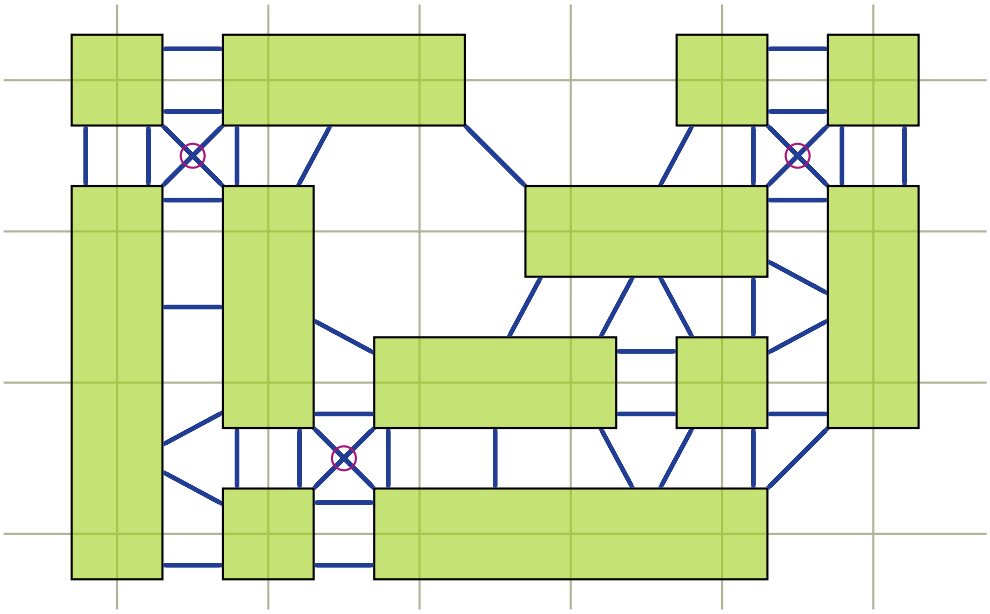

Figure 4.3: The Gaifman graph H of the constructed 2-VCSP instance $I_{\mathcal{T}}$. The vertices are depicted by green rectangles and the edges by thick blue segments. The graph $H^\bullet$ is constructed from H by introducing a new vertex at the intersection of every crossing. Hence, we add the vertices depicted by a red circle in the figure.

The reader may imagine $H^\bullet$ as obtained from H by introducing a new vertex at the intersection of diagonals within every cross Q; this new vertex is identified with Q. See Figure 4.3. We have the following simple observation.

Claim 4.6. *The graph $H^\bullet$ is planar.*

Proof. Let $H_0^\bullet$ be the graph consisting of the grid points of G where two grid points are adjacent if they are consecutive on the same line of G, additionally, a new vertex is added for every cell of G which it adjacent to all the corners of this cell. Clearly, $H_0^\bullet$ is planar. Now, $H^\bullet$ can be obtained from $H_0^\bullet$ as follows:

- contract every $A \in \mathcal{T}$ to a single vertex;
- remove every element of $\text{points}(G) \setminus \bigcup \mathcal{T}$; and
- for every grid cell Q of G that is not a cross, either contract the vertex corresponding to Q onto any of its neighbors, or remove it if it has no neighbors.

So $H^\bullet$ is a minor of a planar graph, hence it is planar as well. □

For a graph G, we denote the distance between two vertices u, v of G by $\text{dist}_G(u, v)$. We also have the following simple claim on the distances in $H^\bullet$ in relation to H.

Claim 4.7. *For all $A, B \in \mathcal{T}$, $\text{dist}_{H^\bullet}(A, B) \leq 2 \cdot \text{dist}_H(A, B)$.*

Proof. By repeated use of triangle inequality along a shortest path connecting A and B, it suffices to argue the following: if A and B are adjacent in H, then they are at distance at most 2 in $H^\bullet$. For this, observe that either A and B are still adjacent in $H^\bullet$, or they contain two opposite corners of some cross Q, which becomes their common neighbor in $H^\bullet$. □

We now apply Baker's technique. Select any $A \in \mathcal{T}$ and partition $\mathcal{T}$ into layers according to the distance in H from A: for a non-negative integer t, layer $\mathcal{L}_t$ consists of all those vertices $B \in \mathcal{T}$ for which $\mathrm{dist}_H(A,B) = t$. Note that layers $\mathcal{L}_t$ form a partition of $\mathcal{T}$ due to the assumption that H is connected. The following observation is crucial.

Lemma 4.8. *For all integers $0 \le i < j$, the treewidth of the sub-graph $H[\mathcal{L}_i \cup \mathcal{L}_{i+1} \cup \cdots \cup \mathcal{L}_j]$ is bounded by $\mathcal{O}(j-i)$.*

Proof. Assume $i > 0$, at the end we will quickly comment on how the case $i = 0$ is resolved in essentially the same way. Let $\mathcal{W} \subseteq V(H^\bullet)$ be the union of all layers $\mathcal{L}_t$ with $i \le t \le j$, plus all crosses Q which are adjacent in $H^\bullet$ to any element of those layers. Further, let $\mathcal{K} \subseteq V(H^\bullet)$ be the union of all layers $\mathcal{L}_t$ with $0 \le t < i$, plus all crosses Q which are adjacent in $H^\bullet$ to any element of those layers, except those are already included in $\mathcal{W}$. In this way, $\mathcal{K}$ and $\mathcal{W}$ are disjoint. Further, observe that from the definition of the sets $\mathcal{L}_t$ as distance layers from A it follows that both $H^\bullet[\mathcal{K}]$ and $H^\bullet[\mathcal{K} \cup \mathcal{W}]$ are connected.

Let H' be the graph obtained from $H^\bullet[\mathcal{K} \cup \mathcal{W}]$ by contracting the sub-graph $H^\bullet[\mathcal{K}]$ into a single vertex; call it A'. As a minor of a planar graph, H' is again planar. By the definition of the layers, for every $B \in \mathcal{L}_i \cup \cdots \cup \mathcal{L}_j$ there exists a $C \in \mathcal{L}_{i-1}$ such that $\mathrm{dist}_H(C,B) \le j - i + 1$. By Claim 4.7, we have $\mathrm{dist}_{H^\bullet}(C,B) \le 2(j-i+1)$, implying that $\mathrm{dist}_{H'}(A',B) \le 2(j-i+1)$. Since every cross included in $\mathcal{W}$ is adjacent to some B as above, we conclude that H' is a connected planar graph of radius at most $2(j-i+1)+1 = 2(j-i)+3$. By standard bounds linking treewidth with radius in planar graphs (see for instance [RS84, 2.7]), we conclude that H' has treewidth at most $6(j-i)+10$.

It remains to connect the treewidth of H' with the treewidth of $H'' := H[\mathcal{L}_i \cup \mathcal{L}_{i+1} \cup \cdots \cup \mathcal{L}_j]$. For this, let (T, bag) be a tree decomposition of H' of width at most $6(j-i)+10$. We turn (T, bag) into a tree decomposition (T, bag') of H'' as follows. For every cross $Q \in \mathcal{W}$, let N_Q be the set of at most four neighbors of Q in H'. Then (T, bag') is obtained by first removing A' from all bags, and then, for every cross $Q \in \mathcal{W}$, replacing Q with N_Q in all bags of (T, bag) containing Q. With the following observation, it is straight forward to verify that (T, bag') is a tree decomposition of H''. Every pair $B, B' \in \mathcal{W}$ which is adjacent in H'' but not in H' has a cross $Q \in \mathcal{W}$ as a common neighbor. Observe that B and B' are elements of N_Q and for every $Q \in \mathcal{W}$, all elements of N_Q are adjacent to Q and thus replace Q in its bag. Finally, in the transformation described above the cardinalities of bags grow by a multiplicative factor of at most 4, hence the width of (T, bag') is at most $24(j-i)+43 \in \mathcal{O}(j-i)$.

This finishes the proof for the case $i > 0$. If $i = 0$, we perform the same reasoning except that we do not contract $\mathcal{K}$ (which now is empty). Thus, we simply work with $H' = H^\bullet[\mathcal{W}]$, and this graph has radius at most $2(j-i) = 2j$ by Claim 4.7. The rest of the reasoning applies verbatim. □

4.2.5 Proof of Theorem 4.1

We are now ready to finish the proof of Theorem 4.1. Recall that the steps performed so far were as follows:

- We guessed a rectangle $R_{\max} \in \mathcal{S}$ of maximum weight from the optimal solution and removed all rectangles of weight larger than $\omega(R_{\max})$ or smaller than $\varepsilon \cdot \omega(R_{\max})/k$. This incurred a loss of at most $\varepsilon \cdot \mathrm{opt}_k(\mathcal{D})$ on the optimum.
- We applied the algorithm of Lemma 4.3. This way, we either find an independent set with a suitably large weight, or construct a grid G of size $|G| \leq 2k^2/\varepsilon$.
- We used Lemma 4.4 to guess the combinatorial type $\mathcal{T}$ of an optimum solution, by branching into $2^{\mathcal{O}(k\log(k/\varepsilon))}$ possibilities.
- We constructed a 2-VCSP instance $I_{\mathcal{T}}$ corresponding to the combinatorial type $\mathcal{T}$.

By Claim 4.5, it remains to find a solution to $I_{\mathcal{T}}$ which yields a revenue of at least $(1-\varepsilon)\,\mathrm{opt}(I_{\mathcal{T}})$, where $\mathrm{opt}(I_{\mathcal{T}})$ is the maximum possible revenue in $I_{\mathcal{T}}$. Note that by retracing previous steps, this will result in finding a solution to the original instance of MWISR with weight at least $(1-2\varepsilon)\,\mathrm{opt}_k(\mathcal{D})$, so at the end we need to apply the reasoning to ε scaled by a factor of $1/2$.

As argued before, we may assume that the Gaifman graph H of $I_{\mathcal{T}}$ is connected. We partition $\mathcal{T}$ into layers $\{\mathcal{L}_t \colon t = 0,1,2,\ldots\}$ as in the previous section. Let $\ell := \lceil 1/\varepsilon \rceil$, and define

$$\mathcal{M}_r := \bigcup_{t \equiv r \bmod \ell} \mathcal{L}_t \qquad \text{for all } r \in \{0,1,\ldots,\ell-1\}.$$

Note that $\{\mathcal{M}_r \colon r \in \{0,1,\ldots,\ell-1\}\}$ is a partition of $\mathcal{T}$.

Let u be an optimal solution of $I_{\mathcal{T}}$. Since it is always possible to assign $\perp$ to every element of $\mathcal{T}$, we have $f(u) \geq 0$, in particular all binary constraints are satisfied under f. Therefore, $f(u) = \sum_{r=0}^{\ell-1} f(u|_{\mathcal{M}_r})$. Since $\ell \geq 1/\varepsilon$, there exists $r_0 \in \{0,1,\ldots,\ell-1\}$ such that $f(u|_{\mathcal{M}_{r_0}}) \leq \varepsilon \cdot f(u)$. The algorithm guesses the value of r_0 by branching into ℓ possibilities.

Let $I'_{\mathcal{T}}$ be the 2-VCSP instance obtained from $I_{\mathcal{T}}$ by deleting all variables contained in $\mathcal{M}_{r_0}$. Observe that we have $\mathrm{opt}(I'_{\mathcal{T}}) \geq (1-\varepsilon) \cdot \mathrm{opt}(I_{\mathcal{T}})$, since u restricted to the variables of $I'_{\mathcal{T}}$ yields revenue at least $(1-\varepsilon) \cdot \mathrm{opt}(I_{\mathcal{T}})$. Further, every solution to $I'_{\mathcal{T}}$ can be lifted to a solution of $I_{\mathcal{T}}$ with the same revenue by mapping all variables of $\mathcal{M}_{r_0}$ to $\perp$. Hence, it suffices to find an optimal solution of the instance $I'_{\mathcal{T}}$.

For this, observe that the Gaifman graph of $I'_{\mathcal{T}}$ is equal to $H - \mathcal{M}_{r_0}$. This graph is the disjoint union of several sub-graphs, each contained in the union of at most $\ell-1$ consecutive layers $\mathcal{L}_t$.

By Lemma 4.8 we infer that the treewidth of $H - \mathcal{M}_{r_0}$ is bounded by $\mathcal{O}(\ell) = \mathcal{O}(1/\varepsilon)$. Now, we apply Theorem 1.7 to find an optimal solution of $I'_{\mathcal{T}}$ in time $|\mathcal{D}|^{\mathcal{O}(1/\varepsilon)} \cdot k^{\mathcal{O}(1)}$. Together with the previous guessing steps, this gives a total time complexity of $2^{\mathcal{O}(k\log(k/\varepsilon))} \cdot |\mathcal{D}|^{\mathcal{O}(1/\varepsilon)}$, as wanted. This concludes the proof of Theorem 4.1.

4.3 Axis-Parallel Segments

In this section we prove Theorem 4.1. We use the same notation as in Section 4.2: $\mathcal{D}$ is the given set of axis-parallel segments, $\omega\colon \mathcal{D} \to$ is the weight function on $\mathcal{D}$, and $\mathrm{opt}_k(\mathcal{D},\omega)$ is the maximum possible ω-weight of a set of at most k disjoint segments in $\mathcal{D}$; we may drop ω in the notation if the weight function is clear from the context. Also, whenever $\mathcal{D}$, ω, and k are clear from the context, then by an *optimum solution* we mean a set of pairwise disjoint segments $\mathcal{S} \subseteq \mathcal{D}$ such that $|\mathcal{S}| \leq k$ and $\omega(\mathcal{S}) = \mathrm{opt}_k(\mathcal{D})$.

4.3.1 Reducing the Number of Distinct Weights

We first apply the same preprocessing as in Section 4.2: we guess a segment $R_{\max} \in \mathcal{S}$ of maximum weight and remove from $\mathcal{D}$ all segments of weight larger than $\omega(R_{\max})$ or smaller than $\varepsilon \cdot \omega(R_{\max})/k$. Let $\mathcal{D}' \subseteq \mathcal{D}$ be the obtained set of segments. As argued in Section 4.2, we have

$$\mathrm{opt}_k(\mathcal{D}',\omega) \geq (1-\varepsilon) \cdot \mathrm{opt}_k(\mathcal{D},\omega).$$

As the next preprocessing step, we round all weights down to the set

$$M := \{\omega(R_{\max})(1-\varepsilon)^i \colon i \in \{0,1,\ldots,\lceil \log_{1-\varepsilon}(\varepsilon/k)\rceil\}\}.$$

That is, we define the new weight function $\omega'\colon \mathcal{D}' \to$ by setting $\omega'(R)$ to be the largest element of M not exceeding $\omega(R)$. Since the weight of every segment is scaled down by a multiplicative factor of at most $1-\varepsilon$, we have

$$\mathrm{opt}_k(\mathcal{D}',\omega') \geq (1-\varepsilon) \cdot \mathrm{opt}_k(\mathcal{D}',\omega) \geq (1-\varepsilon)^2 \cdot \mathrm{opt}_k(\mathcal{D},\omega) \geq (1-2\varepsilon) \cdot \mathrm{opt}_k(\mathcal{D},\omega).$$

Thus, the two preprocessing steps above reduce solving the instance $(\mathcal{D},\omega)$ to solving the instance $(\mathcal{D}',\omega')$, at the cost of losing $2\varepsilon \cdot \mathrm{opt}_k(\mathcal{D})$ on the optimum and a $|\mathcal{D}|^{\mathcal{O}(1)}$ overhead in the time complexity. Observe that in $(\mathcal{D}',\omega')$, the number of distinct weights of rectangles is bounded by $|M| \leq \mathcal{O}(\varepsilon \log(k/\varepsilon))$. We show in the sequel, that the MWISR problem for axis-parallel segments can be solved in fixed-parameter time when parameterized by both k and the number of distinct weights.

Theorem 4.9. *Suppose $\mathcal{D}$ is a set of axis-parallel segments in the plane and ω is a positive weight function on $\mathcal{D}$. Let W be the number of distinct weights assigned by ω. Then given k, in time $(kW)^{\mathcal{O}(k^2)} \cdot |\mathcal{D}|^{\mathcal{O}(1)}$ one can find an optimum solution.*

Note that Theorem 4.2 follows from combining Theorem 4.9 with the preprocessing described above applied to ε scaled by a factor of $1/2$. Therefore, from now on we focus on proving Theorem 4.9.

4.3.2 Constructing a Grid

Let $\preceq$ be any total order on $\mathcal{D}$ ordering the segments by non-decreasing weights, that is, $\omega(R) < \omega(R')$ entails $R \prec R'$. Extend $\preceq$ to subsets of $\mathcal{D}$ in a lexicographic manner: $\mathcal{S} \prec \mathcal{S}'$ if $\mathcal{S}$ is lexicographically smaller than $\mathcal{S}'$ where both sets are sorted according to $\preceq$. Let $\mathcal{S}_{\max}$ be the $\preceq$-maximum optimum solution.

The next step is to guess the $\preceq$-minimum segment $R_{\min}$ of $\mathcal{S}_{\max}$, by branching into $|\mathcal{D}|$ options. Having this done, we safely remove from $\mathcal{D}$ all segments R satisfying $R \prec R_{\min}$. Since $\mathcal{S}_{\max}$ is the $\preceq$-maximum optimum solution, we achieve the following property: every optimum solution contains the $\preceq$-smallest segment of $\mathcal{D}$. We proceed further with this assumption.

We show that under this assumption, there exist a grid of size at most k which hits every segment from $\mathcal{D}$. Here and later on, a segment is *hit* by a line if they intersect, and a segment is *hit* by a grid if it is hit by a line of this grid.

Claim 4.10. *Suppose every optimum solution contains the $\preceq$-minimum segment of $\mathcal{D}$. Then there exists a grid G of size at most k such that every segment in $\mathcal{D}$ is hit by G.*

Proof. Let R_0 be the $\preceq$-minimum segment of $\mathcal{D}$ and let $\mathcal{S}$ be any optimum solution. Let G be the grid comprising of, for every segment $R \in \mathcal{S}$, the line containing R. Clearly, we have $|G| \leq |\mathcal{S}| \leq k$. Suppose for the sake of contradiction, that there is a segment $R \in \mathcal{D}$ which is not hit by G. Clearly $R \neq R_0$, because $R_0 \in \mathcal{S}$ by assumption. Consider $\mathcal{S}' := \mathcal{S} - \{R_0\} \cup \{R\}$ and note that $\mathcal{S}'$ is an independent set, because all segments of $\mathcal{S}$ are contained in lines of G, while R is disjoint with all those lines. Since $R_0 \prec R$, we have $\omega(R_0) \leq \omega(R)$, hence $\omega(\mathcal{S}') \geq \omega(\mathcal{S})$. So $\mathcal{S}'$ is an optimum solution that does not contain R_0, a contradiction. □

Note that the proof of Claim 4.10 is non-constructive, as the definition of the grid depends on an unknown optimum solution $\mathcal{S}$. However, we can give a polynomial-time $\mathcal{O}(k)$-approximation algorithm for finding a grid which hits all segments in $\mathcal{D}$.

Lemma 4.11. *There exists a polynomial time algorithm which, given a set $\mathcal{D}$ of axis-parallel segments in the plane and an integer k, either correctly concludes that there is no grid of size at most k hitting all segments of $\mathcal{D}$, or finds such a grid of size $\mathcal{O}(k^2)$.*

Proof. We construct a grid G as follows. Swipe a vertical line from left to right across $\mathcal{D}$ until the first moment when the segments lying entirely to the left of the line can not be hit by k horizontal lines anymore. Let x_1 be the position of the line at this moment; in other words, x_1 is the least real such that the segments of $\mathcal{D}$ entirely contained in $(-\infty, x_1] \times \mathbb{R}$ cannot be

hit with k horizontal lines. We set $x_1 = \infty$ in case the whole $\mathcal{D}$ can be covered with at most k horizontal lines. By minimality of x_1, the segments entirely contained in $(-\infty, x_1] \times \mathbb{R}$ can be covered by $k+1$ lines: the k horizontal lines required to cover segments in $(-\infty, x_1)$, plus one vertical line at x_1 in case $x_1 \neq \infty$. We add all those $k+1$ lines to G, delete from $\mathcal{D}$ all segments hit by those lines, and repeat the procedure until no more segments are left in $\mathcal{D}$. This way we obtain numbers $x_1 \leq x_2 \leq \cdots \leq x_\ell$ and a grid G of size at most $(k+1)\ell$, where ℓ is the number of iterations of the procedure. See Figure 4.4 for an illustration.

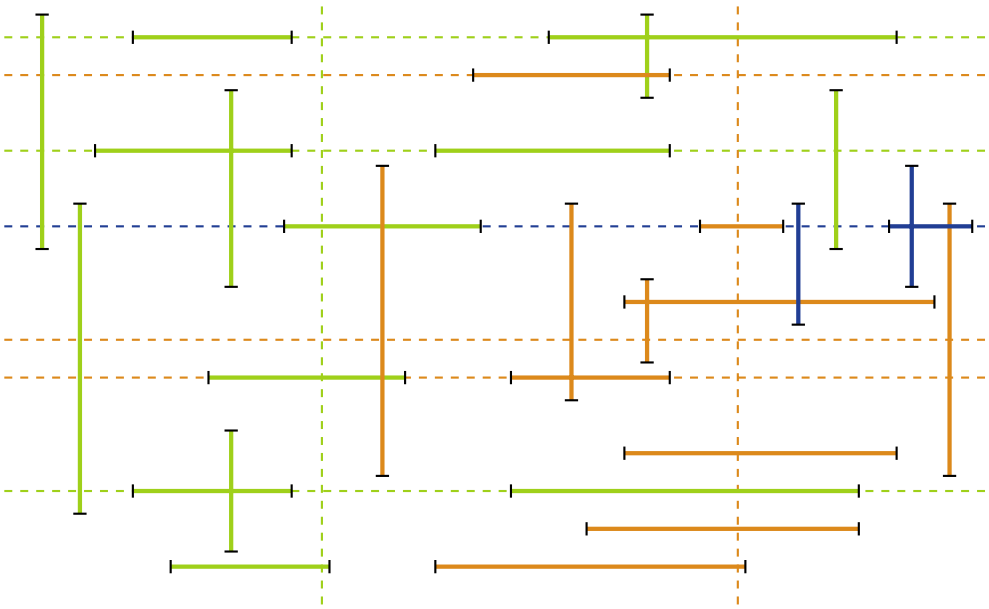

Figure 4.4: Example of the grid construction for $k = 3$. Subsequently, green, orange and then blue segments are removed in consecutive iterations. In each iteration we scan the segments from left to right until $k+1$ horizontal lines are needed to cover the already seen segments. In the last iteration at most k horizontal lines are selected. Lines added to G are dashed.

Clearly, G hits all segments in $\mathcal{D}$. So if $\ell \leq k+1$, then $|G| \leq (k+1)^2 = \mathcal{O}(k^2)$ and the algorithm can provide G as a valid output. We now argue that if $\ell > k+1$, then the algorithm may safely conclude that there is no grid of size at most k which hits all segments of $\mathcal{D}$. For the sake of a contradiction, suppose there is such a grid G'. For $i \in \{1, \ldots, \ell\}$, let $\mathcal{D}_i$ be the set of all segments entirely contained in $(x_{i-1}, x_i] \times \mathbb{R}$, where we set $x_0 = -\infty$. It is easy to see that $\mathcal{D}_i$ is precisely the set of segments for which the algorithm in iteration i decided that it cannot be hit by at most k horizontal lines. Hence, for each $i \in \{1, \ldots, \ell\}$, G' must contain at least one vertical line hitting at least one segment in $\mathcal{D}_i$. The x-coordinate of this vertical line must belong to the interval $(x_{i-1}, x_i]$, so these vertical lines must be pairwise different. We conclude that $|G'| \geq \ell > k$, a contradiction.

It remains to argue how to implement the algorithm such that it runs in polynomial time. Observe that for a set of segments $\mathcal{D}' \subseteq \mathcal{D}$, the minimum number of horizontal lines needed to hit all the segments of $\mathcal{D}'$ can be computed as follows: Project all segments of $\mathcal{D}'$ on the vertical

axis, and find the minimum number of points hitting the obtained set of intervals. Some of the intervals are single points; these are projected horizontal segments. This, in turn, can be done in time $\mathcal{O}(|\mathcal{D}'|\log|\mathcal{D}'|)$ using a standard greedy strategy. It is now straightforward to use this sub-procedure to execute the construction of G described above in polynomial time. □

We now combine Claim 4.10 and Lemma 4.11 as follows. Run the algorithm of Lemma 4.11 on $\mathcal{D}$ with parameter k. If the algorithm concludes that there is no grid of size at most k hitting all segments of $\mathcal{D}$, then by Claim 4.10 we can terminate the current branch, as clearly one of the previous guesses was incorrect. Otherwise, we obtain a grid G of size $\mathcal{O}(k^2)$ which hits every segment of $\mathcal{D}$. With this grid we proceed to the next steps.

For brevity of presentation, by adding four lines to G we may assume that all segments of $\mathcal{D}$ are contained in the interior of the rectangle delimited by the left-most and the right-most vertical line of G and the top-most and the bottom-most horizontal line of G. We will also say that a grid with this property *encloses* $\mathcal{D}$.

4.3.3 Constructing a Nice Grid

We use the same notion of niceness as in Section 4.2. That is, a grid G is *nice* with respect to a segment R, if R contains at least one grid point of G; in other words, R is intersected by both a horizontal and a vertical line in G. We will also say that R *respects* the grid G. The *ugliness* of a grid G with respect to some optimum solution $\mathcal{S}$ is the number of segments in $\mathcal{S}$ which do not respect G. Then the *ugliness* of G is the minimum over all optimum solutions $\mathcal{S}$ of the ugliness of G with respect to $\mathcal{S}$. This way, a grid is *nice* if its ugliness is 0, or equivalently, there exists an optimum solution $\mathcal{S}$ such that G is nice with respect to all the segments in $\mathcal{S}$.

We first observe that if we manage to construct a grid of ugliness 0, then finding an optimum solution reduces to solving an instance of 2-CSP. For this, it will be convenient to again rely on a suitably defined notion of a combinatorial type of a segment with respect to a grid.

Consider a grid G enclosing $\mathcal{D}$. For a segment $R \in \mathcal{D}$, the *combinatorial type* of R with respect to G is the 6-tuple consisting of:

- The boolean value indicating whether R is horizontal or vertical.
- The weight $\omega(R)$.
- The right-most line $\ell_{\leftarrow}$ of G such that R entirely lies strictly to the right of $\ell_{\leftarrow}$.
- The left-most line $\ell_{\rightarrow}$ of G such that R entirely lies strictly to the left of $\ell_{\rightarrow}$.
- The bottom-most line $\ell_{\uparrow}$ of G such that R entirely lies strictly below $\ell_{\uparrow}$.
- The top-most line $\ell_{\downarrow}$ of G such that R entirely lies strictly above $\ell_{\downarrow}$.

In other words, $(\ell_{\leftarrow}, \ell_{\rightarrow}, \ell_{\uparrow}, \ell_{\downarrow})$ contain the sides of the inclusion-wise minimal rectangle R' delimited by the lines from G whose interior contains R. Note that the set of grid points of G contained in R is equal to the set of grid points contained in the interior of R'. Assuming G is

clear from the context, for a type t we will denote this set of grid points by points(t). Observe that the number of different combinatorial types with respect to G is bounded by $2W|G|^4$, where W is the number of distinct weights assigned by ω.

Lemma 4.12. *Given a finite set $\mathcal{D}$ of axis-parallel segments in the plane, a positive weight function ω on $\mathcal{D}$, a positive integer k, and a grid G enclosing $\mathcal{D}$ with a guarantee that the ugliness of G is 0. Then an optimum solution for $(\mathcal{D},\omega,k)$ can be found in time $(W\cdot|G|)^{\mathcal{O}(k)}\cdot|\mathcal{D}|^{\mathcal{O}(1)}$.*

Proof. Fix any optimum solution $\mathcal{S}$ such that G is nice with respect to $\mathcal{S}$. We guess, by branching into all possibilities, the combinatorial types with respect to G, of all the segments in $\mathcal{S}$. Since there are at most $2W|G|^4$ different combinatorial types, this results in $(W\cdot|G|)^{\mathcal{O}(k)}$ branches. Let the guessed set of combinatorial types be $\mathcal{T}$. Since G is supposed to be nice with respect to $\mathcal{S}$, we may assume the sets in $\{\text{points}(t)\colon t\in\mathcal{T}\}$ to be nonempty and pairwise disjoint; otherwise the branch can be discarded.

We construct an auxiliary 2-CSP instance $I_{\mathcal{T}}$ which models the choice of segments in $\mathcal{S}$. The set of variables is $\mathcal{T}$. For every type $t\in\mathcal{T}$, the domain $\mathcal{D}_t$ consists of all segments from $\mathcal{D}$ whose combinatorial type is t. The constraints are as follows:

- If $t,t'\in\mathcal{T}$ are distinct types of horizontal segments, and points(t) and points(t') are two adjacent intervals of grid points on the same horizontal line of G, then we put a constraint between t and t' which among $\mathcal{D}_t\times\mathcal{D}_{t'}$, only allows pairs of disjoint segments.
- Analogous constraints are put for distinct types $t,t'\in\mathcal{T}$ of vertical segments for which points(t) and points(t') are adjacent intervals on the same vertical line.

It is straightforward to verify that solutions to $I_{\mathcal{T}}$ correspond in a one-to-one fashion to the independent sets in $\mathcal{D}$ for which the set of combinatorial types is $\mathcal{T}$. Moreover, observe that the Gaifman graph of $I_{\mathcal{T}}$ is a disjoint union of paths. Every path $t_1-\cdots-t_p$ corresponds to a sequence points$(t_1),\ldots,$points(t_p) of intervals on the same grid line such that points(t_i) is adjacent to points(t_{i+1}) for $i\in\{1,\ldots,p-1\}$. Therefore, it suffices to solve $I_{\mathcal{T}}$ optimally, which can be done in time $|\mathcal{D}|^{\mathcal{O}(1)}$ using, for instance, Theorem 1.7. □

Lemma 4.12 suggests that we should aim to construct a grid with zero ugliness. So far, the grid G constructed in the previous section may have positive ugliness: some segments of $\mathcal{D}$ may be intersected by just one, and not two orthogonal lines, and there is no reason why an optimum solution should not contain any such segments. Our goal is to reduce the ugliness of the grid by further branching steps. The branching strategy is captured in the following lemma.

Lemma 4.13. *Given a finite set $\mathcal{D}$ of axis-parallel segments in the plane, a positive weight function ω on $\mathcal{D}$, a positive integer k, and a grid G which hits all segments of $\mathcal{D}$ and encloses $\mathcal{D}$. Let W be the number of different weights assigned by ω. Then one can construct, in time $(|G|\cdot W)^{\mathcal{O}(k)}\cdot|\mathcal{D}|^{\mathcal{O}(1)}$, a family $\mathcal{G}$ of grids with the following properties:*

(i) $|\mathcal{G}| \leq (|G| \cdot W)^{\mathcal{O}(k)}$;
(ii) *for each* $G' \in \mathcal{G}$, *we have* $G' \supseteq G$ *and* $|G' \setminus G| \leq k$; *and*
(iii) *if the ugliness of* G *is positive, then there exists* $G' \in \mathcal{G}$ *whose ugliness is strictly smaller than the ugliness of* G.

Proof. Fix an optimum solution $\mathcal{S}$ such that the ugliness of G with respect to $\mathcal{S}$ is minimum possible. Assume that this ugliness is positive, since otherwise (iii) always holds and any family $\mathcal{G}$ satisfying (i) and (ii) is a valid output. We construct $\mathcal{G}$ by a branching algorithm which, intuitively, guesses a bounded amount of information about $\mathcal{S}$ and augments G according to the guess, such that the augmented grid is nice with respect to at least one more segment of $\mathcal{S}$. Thus, different members of $\mathcal{G}$ correspond to different guesses on the structure of $\mathcal{S}$.

Let $\mathcal{N}$ be the set of all segments in $\mathcal{S}$ which respect G. As the ugliness of G is positive, $\mathcal{S} \setminus \mathcal{N}$ is nonempty. Let $R_{\max}$ be a maximum weight segment of $\mathcal{S} \setminus \mathcal{N}$; in case there are several with the same maximum weight, pick any of them.

The algorithm guesses, by branching into all possibilities, the combinatorial types of all segments in $\mathcal{N} \cup \{R_{\max}\}$; this results in at most $(1 + 2W|G|^4)^k \leq (|G| \cdot W)^{\mathcal{O}(k)}$ branches. For every guess we shall construct one grid $G' \supseteq G$ included in $\mathcal{G}$. Therefore, we fix one guess and proceed to the description of G'.

By symmetry assume that $R_{\max}$ is vertical. Let $\mathcal{T}$ be the (already guessed) set of combinatorial types of segments from $\mathcal{N}$, and let $t_{\max} = (\mathsf{vertical}, w, \ell_{\leftarrow}, \ell_{\rightarrow}, \ell_{\uparrow}, \ell_{\downarrow})$ be the (already guessed) combinatorial type of $R_{\max}$. Similar to the proof of Lemma 4.12, we assume that the sets $\{\mathrm{points}(t) \colon t \in \mathcal{T}\}$ are nonempty and pairwise disjoint, as otherwise the guess can be safely discarded as incorrect. Note that each set $\mathrm{points}(t)$ is an interval consisting of consecutive grid points on a single line of G.

Let B be the rectangle delimited by $(\ell_{\leftarrow}, \ell_{\rightarrow}, \ell_{\uparrow}, \ell_{\downarrow})$. Since G is not nice with respect to $R_{\max}$, the interior of B does not contain any grid point of G. There are two cases to consider:

Case 1: $\ell_{\leftarrow}$ and $\ell_{\rightarrow}$ are consecutive vertical lines of G. This is equivalent to $R_{\max}$ lying in the interior of the vertical strip between $\ell_{\leftarrow}$ and $\ell_{\rightarrow}$. In particular $R_{\max}$ is not contained in any line of G. Note that since $R_{\max}$ is hit by G, which is true about every segment of $\mathcal{D}$, the two horizontal lines $\ell_{\uparrow}$ and $\ell_{\downarrow}$ are non-consecutive in G. So B is the union of two or more vertically adjacent grid cells of G.
Case 2: $\ell_{\leftarrow}$ and $\ell_{\rightarrow}$ are non-consecutive vertical lines of G. Since $R_{\max}$ is vertical, there must exist exactly one line of G between $\ell_{\leftarrow}$ and $\ell_{\rightarrow}$, say ℓ, and ℓ must contain $R_{\max}$. Note that since $R_{\max}$ contains no grid point of G, $\ell_{\uparrow}$ and $\ell_{\downarrow}$ must be two consecutive horizontal lines of G. So B is the union of two horizontally adjacent grid cells of G.

We consider these two cases separately. See Figure 4.5 for an illustration.

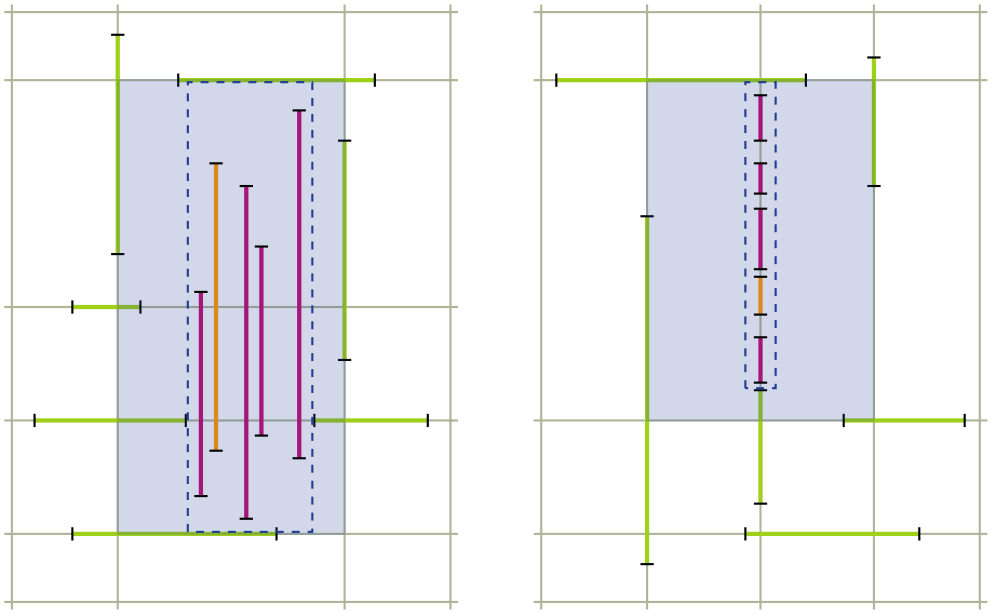

Figure 4.5: Illustration of Case 1 and Case 2 in the proof of Lemma 4.13. The green segments depict the segments in $\mathcal{N}$, so they already contain at least one grid-point. The orange segment is the candidate segment $R_{\max}$, it has maximum weight among all those segments in the optimal solution which do not contain a grid-point. The box B is represented by a blue region. We greedily find a maximum size region inside B containing candidate segments, depicted in red, with the same combinatorial type as $R_{\max}$. If there are more than k independent candidates, we can return an optimal solution, since $R_{\max}$ has maximum weight. Otherwise we can add fewer than k grid-lines to the current grid, such that all possible candidates are hit by a newly added grid-line.

Case 1: The segment $R_{\max}$ does not lie on a grid line. We construct an auxiliary 2-CSP instance $I_{\mathcal{T}}$ corresponding to the choice of segments in $\mathcal{N}$, exactly as in the proof of Lemma 4.12. That is, the set of variables is $\mathcal{T}$, and the constraints are as described in the proof of Lemma 4.12. Again, solutions to $I_{\mathcal{T}}$ are in a one-to-one correspondence to those independent sets in $\mathcal{D}$ whose set of combinatorial types is $\mathcal{T}$. Also, the Gaifman graph H of $I_{\mathcal{T}}$ is a disjoint union of paths, where each path corresponds to a sequence of adjacent intervals of grid-points contained in a single grid-line of G.

The idea is to compute a solution u to $I_{\mathcal{T}}$ which leaves "the most space" for the placement of $R_{\max}$. For every connected component C of H, we do the following. Recall that C is a path. Enumerate the consecutive variables on C as $t_1, \ldots, t_p$. Let $\text{points}(C) := \bigcup_{i=1}^{p} \text{points}(t_i)$; then $\text{points}(C)$ is an interval of grid points on one line of G, say ℓ. We again consider two cases.

First, if ℓ is vertical, or $\text{points}(C)$ does not contain any grid point lying in the interior of a side of B; compute any solution within C, say using the algorithm of Theorem 1.7.

Second, we consider the case where ℓ is horizontal and $\text{points}(C)$ contains some grid point lying in the interior of a side of B. Note that the intersection of ℓ with B is a segment. Let $x_{\leftarrow} \in \ell_{\leftarrow}$ and $x_{\rightarrow} \in \ell_{\rightarrow}$ be the endpoints of this segment. Then $x_{\leftarrow}$ and $x_{\rightarrow}$ are two horizontally

adjacent grid points of G which lie in the interior of the left and right side of B, respectively. The set points(C) contains one or both of $x_{\leftarrow}$ and $x_{\rightarrow}$. For concreteness, assume for now that points(C) contains both $x_{\leftarrow}$ and $x_{\rightarrow}$; the other cases are simpler and will be discussed later. Assume that there is no $i \in \{1,\dots,p\}$ such that points(t_i) contains both $x_{\leftarrow}$ and $x_{\rightarrow}$, because then the corresponding segment of $\mathcal{N}$ would necessarily intersect $R_{\max}$. So if this occurs, we can discard the branch as incorrect. Up to reversing indexing if necessary, there exists an index $i \in \{1,\dots,p-1\}$ such that $x_{\leftarrow} \in$ points(t_i) and $x_{\rightarrow} \in$ points(t_{i+1}). We compute a solution within C greedily as follows:

- First, process the variables $t_1,\dots,t_i$ in this order. When considering t_j, assign the segment whose right endpoint is the leftmost possible among the available segments of $\mathcal{D}_{t_j}$, that is, disjoint with the segment assigned to t_{j-1}, for $j > 1$.
- Second, apply a symmetric greedy procedure to the variables $t_p, t_{p-1},\dots,t_{i+1}$ in this order, picking the available segment with the rightmost possible left endpoint.

In case any of $x_{\leftarrow}$ or $x_{\rightarrow}$ does not belong to points(C), only one of the above greedy procedures is applied.

If $I_{\mathcal{T}}$ has a solution, the algorithm described above clearly succeeds in finding some solution u to $I_{\mathcal{T}}$. Since we assume $I_{\mathcal{T}}$ to have a solution, witnessed by $\mathcal{N}$, we may terminate the branch as incorrect in case no solution to $I_{\mathcal{T}}$ is found. Let $\mathcal{N}' = u(\mathcal{T})$ be the independent set of segments found by the algorithm above. It is straightforward to see that the greedy choice of solutions within the components of H justifies the following claim.

Claim 4.14. *It holds that* $\mathrm{int}(B) \cap \mathcal{N}' \subseteq \mathrm{int}(B) \cap \mathcal{N}$*, where* $\mathrm{int}(B)$ *denotes the interior of* B*. Consequently,* $R_{\max}$ *is disjoint with every segment in* $\mathcal{N}'$*.*

Now, let $\mathcal{R} \subseteq \mathcal{D}$ be the set of all segments in $\mathcal{D}$ whose combinatorial type is $t_{\max}$ and which are disjoint with all segments in $\mathcal{N}'$. By Claim 4.14, we necessarily have $R_{\max} \in \mathcal{R}$. Let L be the set comprising of all vertical lines containing some segment $R \in \mathcal{R}$.

- If $|L| < k - |\mathcal{N}|$, then we add the grid $G' := G \cup L$ to $\mathcal{G}$.
- If $|L| \geq k - |\mathcal{N}|$, then we add the grid $G' := G \cup L'$ to $\mathcal{G}$, where L' is any subset of L with size $k - |\mathcal{N}|$.

It remains to argue that in both cases, the ugliness of G' is strictly smaller than the ugliness of of G.

In the case $|L| < k - |\mathcal{N}|$, it suffices to note that since $R_{\max}$ is contained on some line of L, the grid $G' = G \cup L$ is nice with respect to $R_{\max}$, while by assumption, G is not nice with respect to $R_{\max}$.

Consider now the case $|L| \geq k - |\mathcal{N}|$. For every line $\ell \in L'$, pick any segment $R_\ell \in \mathcal{R}$ which lies on ℓ. Let $\mathcal{L} := \{R_\ell : \ell \in L'\}$. Note that the segments of $\mathcal{L}$ are pairwise disjoint due to lying on different vertical lines, and they are also disjoint from all the segments of $\mathcal{N}'$ by the definition of $\mathcal{R}$. So $\mathcal{N}' \cup \mathcal{L}$ is an independent set of segments, and has size k. Furthermore, since the combinatorial type also features the weight of a segment, and $R_{\max}$ was chosen to be the heaviest segment within $\mathcal{S} \setminus \mathcal{N}$, we have $\omega(\mathcal{N}') = \omega(\mathcal{N})$ and $\omega(\mathcal{L}) \geq \omega(\mathcal{S} \setminus \mathcal{N})$. It follows that

$\omega(\mathcal{N}' \cup \mathcal{L}) \geq \omega(\mathcal{S})$, hence $\mathcal{N}' \cup \mathcal{L}$ is also an optimum solution. But $G' = G \cup L'$ is nice with respect to all the segments of $\mathcal{N}' \cup \mathcal{L}$, so the ugliness of G' is 0.

Case 2: The segment $R_{\max}$ lies on a grid line. This case works in a very similar fashion as the previous one, hence we only outline the differences here.

Recall that here B consists of two horizontally adjacent cells of G. Let S be the common side of those cells; then our guess on the combinatorial type $t_{\max}$ of $R_{\max}$ says that $R_{\max}$ should be contained in the interior of S.

We construct an instance $I_{\mathcal{T}}$ of 2-CSP in exactly the same manner as in Case 1. We solve it using a similar greedy procedure, so that the space left for placing $R_{\max}$ within the interior of s is maximized. Here, there will be at most one connected component of the Gaifman graph H of $I_{\mathcal{T}}$ where a greedy strategy is applied; this is the vertical component C such that points(C) contains one or both endpoints of S. Let $\mathcal{N}'$ be the obtained solution to $I_{\mathcal{T}}$. The analogue of Claim 4.14 now says the following:

Claim 4.15. *It holds that* $\mathrm{int}(S) \cap \mathcal{N}' \subseteq \mathrm{int}(S) \cap \mathcal{N}$*, hence* $R_{\max}$ *is disjoint with every segment in* $\mathcal{N}'$*.*

Consequently, if we denote $S' := \mathrm{int}(S) \setminus \mathcal{N}'$, then S' is an open segment which contains $R_{\max}$. Now let $\mathcal{R}$ be the set of all segments from $\mathcal{D}$ contained in S' and whose weight is equal to the guessed weight of $R_{\max}$. Since all segments of $\mathcal{R}$ lie on the same line, using a polynomial-time top-to-bottom greedy sweep we may find a maximum independent set of segments within $\mathcal{R}$; call it $\mathcal{L}$. Let L be the set horizontal lines passing through the lower endpoints of the segments in $\mathcal{L}$. Note that by construction of $\mathcal{L}$, L hits all segments in $\mathcal{R}$. We again consider two sub-cases:

- If $|\mathcal{L}| = |L| < k - |\mathcal{N}|$, then we add the grid $G' = G \cup L$ to $\mathcal{G}$.
- If $|\mathcal{L}| = |L| \geq k - |\mathcal{N}|$, then we add the grid $G' = G \cup L'$ to $\mathcal{G}$, where L' is any, arbitrarily chosen, subset of L with size $k - |\mathcal{N}|$.

A reasoning analogous to Case 1 shows the following. In the first sub-case, G' is nice with respect to $R_{\max}$, hence the ugliness of G' is strictly smaller than that of G. In the second case, $\mathcal{N}' \cup \mathcal{L}$ is an optimum solution and G' is nice with respect to $\mathcal{N}' \cup \mathcal{L}$, hence the ugliness of G' is 0.

In both Case 1 and Case 2 we constructed a grid $G' \supseteq G$ with $|G' \setminus G| \leq k$ whose ugliness is strictly smaller than the ugliness of G. We conclude the proof by taking $\mathcal{G}$ to be the set of all grids G' constructed in this manner. □

Finally, Lemma 4.13 can be applied in a recursive manner to obtain a nice grid.

Lemma 4.16. *Given a finite set* $\mathcal{D}$ *of axis-parallel segments in the plane, a positive weight function* ω *on* $\mathcal{D}$*, a positive integer* k*, and a grid* G *hitting all segments of* $\mathcal{D}$ *enclosing* $\mathcal{D}$*. Let* W *be the number of different weights assigned by* ω*. Then one can in time* $(k \cdot W \cdot |G|)^{\mathcal{O}(k^2)} \cdot |\mathcal{D}|^{\mathcal{O}(1)}$ *construct a family* $\mathcal{G}$ *of grids such that:*

(i) $|\mathcal{G}| \leq (k \cdot W \cdot |G|)^{\mathcal{O}(k^2)}$;
(ii) *for each* $G' \in \mathcal{G}$, *we have* $G' \supseteq G$ *and* $|G' \setminus G| \leq k^2$; *and*
(iii) $\mathcal{G}$ *contains at least one grid of ugliness* 0.

Proof. Starting with $\mathcal{G}_0 := \{G\}$, we iteratively construct families of grids $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_k$ as follows: to construct $\mathcal{G}_i$ from $\mathcal{G}_{i-1}$, replace each grid $G \in \mathcal{G}_{i-1}$ with the family of grids $\mathcal{G}(G)$ obtained by applying Lemma 4.13 to G. A straightforward induction using properties (i) and (ii) of Lemma 4.13 shows that:

- $|\mathcal{G}_i| \leq (k \cdot W \cdot |G|)^{\mathcal{O}(ik)}$,
- for each $G' \in \mathcal{G}_i$ it holds that $G' \supseteq G$ and $|G' \setminus G| \leq ik$, and the construction of $\mathcal{G}_i$ takes $(k \cdot W \cdot |G|)^{\mathcal{O}(ik)} \cdot |\mathcal{D}|^{\mathcal{O}(1)}$ time.

Moreover, by property (iii) of Lemma 4.13, if the minimum ugliness among grids in $\mathcal{G}_{i-1}$ is positive, then the minimum ugliness among the grids in $\mathcal{G}_i$ is strictly smaller than that in $\mathcal{G}_{i-1}$. Since the ugliness of G is at most k, it follows that $\mathcal{G} := \mathcal{G}_k$ satisfies all the required properties. □

4.3.4 Proof of Theorem 4.9

We are now ready to assemble all the tools and prove Theorem 4.9. As discussed in Section 4.3.2, by preprocessing the instance and branching into $\mathcal{O}(|\mathcal{D}|)$ possibilities, we construct a grid of size $\mathcal{O}(k^2)$ such that every segment in $\mathcal{D}$ is hit by G. Adding four lines to G ensures that G encloses $\mathcal{D}$. Then we apply Lemma 4.16 to G, and construct a family of grids $\mathcal{G}$ which contains at least one grid with ugliness 0. It now remains to apply Lemma 4.12 to each grid in $\mathcal{G}$ and output the heaviest of the obtained solutions.

Following directly from the guarantees provided by Lemmas 4.12 and 4.16, this algorithm runs in time $(kW)^{\mathcal{O}(k^2)} \cdot |\mathcal{D}|^{\mathcal{O}(1)}$.

5 Independence Number of Intersection Graphs of Axis-Parallel Segments

This chapter contains a overworked version of [CCPW22], which is joint work with Marco Caoduro, Michał Pilipczuk and Karol Węgrzycki.

5.1 Introduction

For a graph G, the *independence number* $\alpha(G)$ is the maximum size of an independent set in G. An independent set of a graph G is a subset of pairwise non-adjacent vertices. Both lower and upper bounds on the independence number were intensively studied in various graph classes, including interval graphs [GLL82], planar graphs [ALMM08], and triangle-free graphs [She91]. In this chapter, we study the independence number in classes of geometric intersection graphs. For a set $\mathcal{S}$ of geometric objects, the independence number $\alpha(\mathcal{S})$ is naturally defined as the maximum size of a subset of pairwise disjoint objects in $\mathcal{S}$.

The *chromatic number* $\chi(G)$ is the minimum number of colors needed to color the vertices of G in a way that no two adjacent vertices have the same color. A simple lower bound on the independence number can be often obtained by studying the chromatic number and using the obvious inequality $\alpha(G) \geq n/\chi(G)$, where n is the number of vertices in G. This strategy does not always provide optimum lower bounds, which is also the case in the geometric setting we consider in this chapter.

Specifically, we consider intersection graphs of axis-parallel segments in the plane where no three segments intersect at a single point. For simplicity, we will denote this class of graphs by $\mathcal{G}_{\text{seg}}$. Observe that we have $\chi(G) \leq 4$ for every $G \in \mathcal{G}_{\text{seg}}$, because we can use two colors to properly color the horizontal segments, and another two for the vertical segments. Hence, if $G \in \mathcal{G}_{\text{seg}}$ has n vertices, then $\alpha(G) \geq n/4$.

Our contribution

Our two main results, presented below, prove that the simple lower bound $\alpha(G) \geq n/4$ can be improved by an additive term of the order $\sqrt{n}$, but no further improvement is possible.

Theorem 5.1. *Let G be a graph in $\mathcal{G}_{\text{seg}}$ with n vertices. Then the independence number of G is at least*

$$\alpha(G) \geq \frac{n}{4} + c_1\sqrt{n},$$

for some absolute constant c_1.

Theorem 5.2. *For any $n \in \mathbb{N}$ there exists a graph G in $\mathcal{G}_{\text{seg}}$ on n vertices with independence number*

$$\alpha(G) \leq \frac{n}{4} + c_2\sqrt{n},$$

for some absolute constant c_2.

The independence number is often studied in relation to the clique covering number. A *clique* in a graph is a set of pairwise adjacent vertices, and the *clique covering number* $\theta(G)$ of a graph G is defined as the minimum size of a partition of the vertex set of G into cliques. For any graph G, the clique covering number $\theta(G)$ is a natural upper bound on the independence number $\alpha(G)$. Indeed, an independent set contains at most one vertex from each clique. This implies that for any graph G, the ratio $\theta(G)/\alpha(G)$ is at least one. Giving an upper bound on this ratio is a question that was studied for several classes of intersection graphs, e.g. [GL85], [KN08]. In this topic, the main open question concerns the relation between the independence number and the clique covering number in intersection graphs of axis-parallel rectangles.

Conjecture 5.3 (Wegner 1965,[Weg65])**.** *Let G be the intersection graph of a set of axis-parallel rectangles in the plane. Then*

$$\theta(G) \leq 2\alpha(G) - 1.$$

For an intersection graph G of axis-parallel rectangles the best known bound on the clique covering number is $\theta(G) = \mathcal{O}\big(\alpha(G)\log^2(\log(\alpha(G)))\big)$ by Correa et. al. [CFPS15]. In particular, no linear upper bounds are known. Even, obtaining good lower bounds on the maximum ratio θ/α was wide open problem until very recently. For nearly thirty years after Wegner formulated his conjecture, the largest known ratio remained 3/2, obtained by taking five axis-parallel rectangles forming a cycle. In 1993, Fon-Der-Flaass and Kostochka presented a family of axis-parallel rectangles with clique cover number 5 and independence number 3 [FK93]. Only in 2015, Jelínek constructed families of rectangles with ratio θ/α arbitrarily close to 2, showing that the constant of 2 in Wegner's conjecture cannot be improved[1].

One consequence of our results is that the bound in Wegner's conjecture can be slightly improved for triangle-free intersection graphs of axis-parallel segments. More precisely, we have the following corollary of Theorem 5.1.

[1] The former construction is attributed to Jelínek in [CFPS15, Acknowledgment].

Corollary 5.4. *Let G be a triangle-free intersection graph of axis-parallel segments in the plane. Then*

$$\theta(G) \leq 2\alpha(G) - c_1\sqrt{n}.$$

We point out that the families of axis-parallel rectangles constructed by Jelínek also have a triangle-free intersection graph [CFPS15, Acknowledgment]. An analysis of the construction yields that they in fact satisfy $\theta(G) = 2\alpha(G) - 4$. Thus Corollary 5.4 provides a nice separation of triangle-free intersection graphs of axis-parallel segments from those of axis-parallel rectangles.

On the other hand, our results also give a proof that the ratio of 2 in Wegner's conjecture, cannot be improved even in this highly restricted case of axis-parallel segments, even with the assumption of triangle-freeness. More precisely, we have the following corollary of the full version of Theorem 5.2 (see Section 5.3).

Corollary 5.5. *For any $\varepsilon > 0$, there exists a graph G in $\mathcal{G}_{\text{seg}}$ such that*

$$\theta(G) \geq (2-\varepsilon)\alpha(G).$$

Corollary 5.5 can be further strengthened to the fractional setting, implying a lower bound on the integrality gap of the standard linear programming relaxation of the independent set problem. Namely, consider the *fractional independence number* of a graph G, denoted $\alpha^\star(G)$, which is defined similarly to $\alpha(G)$. The difference being that every vertex u can be included in the solution with a fractional multiplicity $x_u \in [0,1]$, and the constraints are $x_u + x_v \leq 1$ for every edge uv of G. Similarly, in the *fractional clique cover number* $\theta^\star(G)$ every clique K in G can be included in the cover with a fractional multiplicity $y_K \in [0,1]$, and the constraints are $\sum_{K:\, v\in K} y_K \geq 1$ for every vertex v. In triangle-free graphs the linear programs defining $\alpha^\star(G)$ and $\theta^\star(G)$ are dual to each other, hence

$$\alpha(G) \leq \alpha^\star(G) = \theta^\star(G) \leq \theta(G) \qquad \text{for every triangle-free graph } G.$$

The proof of Corollary 5.5, based on the full version of Theorem 5.2, yields the following result.

Corollary 5.6. *For any $\varepsilon > 0$, there exists a graph G in $\mathcal{G}_{\text{seg}}$ such that*

$$\alpha^\star(G) \geq (2-\varepsilon)\alpha(G).$$

Consequently, the integrality gap of the standard linear programming relaxation of the maximum independent set problem in graphs from $\mathcal{G}_{\text{seg}}$ is not smaller than 2.

We note that recently, Gálvez et al. gave a polynomial-time $(2+\varepsilon)$-approximation algorithm for the maximum independent set problem in intersection graphs of axis-parallel rectangles [GKM+22]. Thus, Corollary 5.6 shows that one cannot improve upon the approximation ratio of 2 by only relying on the standard linear programming relaxation, even in the case of

axis-parallel segments. Note that in this case, obtaining a 2-approximation algorithm is very easy: restricting attention to either horizontal or vertical segments reduces the problem to the setting of interval graphs, where it is polynomial-time solvable.

Structure of the chapter

In Section 5.2 we give the constructions of independent sets for any graph in the class of graphs $\mathcal{G}_{\text{seg}}$ used to prove Theorem 5.1. In Section 5.3 we then construct a family of graphs to prove Theorem 5.2. This construction is based on a classical result of Erdős and Szekeres [ES35].

5.2 The Lower Bound: Proof of Theorem 5.1

The goal of this section is to prove Theorem 5.1. For this, we examine a graph $G \in \mathcal{G}_{\text{seg}}$, and exhibit three different independent sets in G by constructing three different subsets of disjoint segments. A trade-off between these three independent sets then results in a lower bound.

The set of geometric objects S is called a *representation* of its intersection graph $G(S)$ Note that a graph can have multiple representations. Our proof starts with some observations on possible sets of segments representing a graph in the class $\mathcal{G}_{\text{seg}}$. Let G be a graph in $\mathcal{G}_{\text{seg}}$ with n vertices and let $\mathcal{S}$ be a representation of G. Thus, $\mathcal{S}$ consists of axis-parallel segments, no three of which meet at one point. We may assume that in $\mathcal{S}$ every two parallel segments which intersect meet at a single point, called the *meeting point*. If two segments do not meet at a single point, we can choose any common point and shorten both segments up to this common point. Since no three segments of $\mathcal{S}$ meet at one point, all intersections are preserved and the modified set of segments is still a representation of G. Further, we may assume that if two orthogonal segments intersect, their intersection point lies in the interior of both of them. Indeed, otherwise we could slightly extend one or both of these segments around the meeting point. Finally, we may assume that the segments of $\mathcal{S}$ lie on a grid of size $\ell_{\text{hor}} \times \ell_{\text{ver}}$ such that the segments lying on the same grid line induce a path in the intersection graph. Indeed, if on a single grid line the segments induce a disjoint union of several paths, then we can move these paths slightly such that they are realized on separate grid lines. A representation $\mathcal{S}$ of G with the properties described above is called *favorable*.

To give constructions for the different subsets of pairwise disjoint segments in a favorable representation $\mathcal{S}$, we first need some notation. Say the grid associated with $\mathcal{S}$ is of size $\ell_{\text{hor}} \times \ell_{\text{ver}}$. We partition the grid lines into lines of *even type* and lines of *odd type* depending if there is a even or odd number of segments lying on them. Let ℓ_{even} be the number of grid lines of even type and ℓ_{odd} the number of grid lines of odd type. This is naturally extended to the set $\mathcal{S}$ by partitioning the segments according to which type of grid line they lie on. Let s_{even} be the total number of segments which lie on a grid line of even type and s_{odd} the total number of segments which lie on a grid line of odd type. Further, we denote the maximum

number of segments on a single grid line by t.

The following three lemmas correspond each to a different set of pairwise disjoint segments in $\mathcal{S}$. In all three lemmas, we assume $\mathcal{S}$ to be a favorable representation of a graph in $\mathcal{G}_{\text{seg}}$ with n vertices.

Lemma 5.7. *There exists a subset of $\mathcal{S}$ consisting of $\frac{n}{4} + \frac{\ell_{\text{odd}}}{4}$ pairwise disjoint segments.*

Lemma 5.8. *There exists a subset of $\mathcal{S}$ consisting of $\frac{n}{4} + \frac{t}{4}$ pairwise disjoint segments.*

Lemma 5.9. *There exists a subset of $\mathcal{S}$ consisting of $\frac{n}{4} + \frac{\sqrt{2s_{\text{even}}}}{4} - \frac{\ell_{\text{odd}}}{4}$ pairwise disjoint segments.*

Before proving these lemmas, we use them to conclude Theorem 5.1.

Proof of Theorem 5.1. Let G be a graph in the class $\mathcal{G}_{\text{seg}}$ with n vertices and let $\mathcal{S}$ be a favorable representation of G. A subset of pairwise disjoint segments in $\mathcal{S}$ corresponds to an independent set in G of the same size. We distinguish three cases. If $\ell_{\text{odd}} \geq \sqrt{n}/c$ for some constant c, by Lemma 5.7 G has an independent set of size at least

$$\frac{n}{4} + \frac{1}{4c} \cdot \sqrt{n}.$$

If $\ell_{\text{odd}} \leq \sqrt{n}/c$ and $s_{\text{even}} \geq 2n/c^2$, by Lemma 5.9 G has an independent set of size at least

$$\begin{aligned} \frac{n}{4} + \frac{\sqrt{2s_{\text{even}}}}{4} - \frac{\ell_{\text{odd}}}{4} &\geq \frac{n}{4} + \frac{\sqrt{4n}}{4c} - \frac{\sqrt{n}}{4c} \\ &\geq \frac{n}{4} + \frac{1}{4c} \cdot \sqrt{n}. \end{aligned}$$

If $\ell_{\text{odd}} \leq \sqrt{n}/c$ and $s_{\text{even}} \leq 2n/c^2$, we get $s_{\text{odd}} \geq n(1-2/c^2)$ using $s_{\text{even}} + s_{\text{odd}} = n$. Then the maximum number of segments t lying on a single line is at least

$$t \geq \frac{s_{\text{odd}}}{\ell_{\text{odd}}} \geq \frac{n(1-2/c^2)}{\sqrt{n}/c} = \frac{c^2-2}{c} \cdot \sqrt{n}.$$

By Lemma 5.8 we get an independent set of G of size at least

$$\frac{n}{4} + \frac{c^2-2}{4c} \cdot \sqrt{n}.$$

Setting $c = \sqrt{3}$ gives the desired result: there is always an independent set of size at least $\frac{n}{4} + \frac{1}{4\sqrt{3}} \cdot \sqrt{n}$. □

Before proving the three lemmas, we quickly discuss how to prove Corollary 5.4 with Theorem 5.1.

Proof of Corollary 5.4. Consider a graph $G \in \mathcal{G}_{\text{seg}}$ minimizing the difference $2\alpha(G) - \theta(G)$ and whose sub-graphs all have a strictly larger difference. Note that removing by a vertex x from

G, the independence number and the clique covering number of $G-x$ either equals that of G or decreases by 1. Since $G-x$ is a sub-graph of G, by assumption, $2\alpha(G-x)-\theta(G-x) > 2\alpha(G)-\theta(G)$. This implies that for all vertices x of G, the clique covering number decreases by 1. Since a clique of G is either an edge or a single vertex, a clique covering consists of a maximum matching of G and a set of singletons. So removing a vertex x only decreases the clique covering number if there exists a maximum matching of G, that does not match x. Thus by Gallai's Lemma [LP09], for any vertex x, the graph $G-x$ has a perfect matching. This means that $\theta(G-x) = (n-1)/2$, where n is the number of vertices of G. Using Theorem 5.1, we get

$$2\alpha(G)-\theta(G) \geq \frac{n}{2} + 2c_1\sqrt{n} - \frac{n+1}{2} \geq c_1\sqrt{n}. \qquad \square$$

The remainder of this section is dedicated to proof the three lemmas.

Proof of Lemma 5.7. This construction exploits grid lines of odd type, so those with an odd number of segments on them. For each grid line, select every second segment lying on that line, starting from the leftmost for horizontal lines, and the topmost for vertical lines. If the grid line is of even type, exactly half of the segments are selected. If the grid line is of odd type, the selected number of segments is half rounded up. This corresponds to selecting exactly half of all segments and adding $1/2$ for each grid line of odd type. So in total,

$$\frac{n}{2} + \frac{\ell_{\text{odd}}}{2}$$

segments are selected.

By construction of this subset, two segments are only intersecting if one is horizontal and the other one is vertical. Thus, the set can be partitioned into horizontal and vertical segments with both parts only containing pairwise disjoint segments. By the pigeonhole principle, one of the two parts contains at least half of the selected segments. $\square$

Proof of Lemma 5.8. This construction exploits a single grid line with many segments on it. Let g_{hor} be a horizontal grid line with the maximum number of segments t_{hor} lying on it, and let s_{ver} be the total number of vertical segments. Consider the subset $\mathcal{S}_{\text{hor}} \subseteq \mathcal{S}$ consisting of all segments lying on g_{hor} and all vertical segments. Analogously define g_{ver}, t_{ver}, s_{hor}, and $\mathcal{S}_{\text{ver}}$. Now we choose the larger set among $\mathcal{S}_{\text{ver}}$ and $\mathcal{S}_{\text{hor}}$. The size of this set is

$$\begin{aligned}\max\{s_{\text{hor}} + t_{\text{ver}}, s_{\text{ver}} + t_{\text{hor}}\} &\geq \frac{s_{\text{hor}} + t_{\text{ver}} + s_{\text{ver}} + t_{\text{hor}}}{2} \\ &\geq \frac{n+t}{2}\end{aligned}$$

For the second inequality, we use assertions $s_{\text{hor}} + s_{\text{ver}} = n$ and $t = \max\{t_{\text{hor}}, t_{\text{ver}}\}$.

We now observe that the intersection graphs of both sets S_{ver} and S_{hor} are bipartite. Indeed,

any cycle in the intersection graph has to contain at least two horizontal segments lying on two different horizontal grid lines, and two vertical segments lying on two different vertical grid lines. But S_{ver} contains horizontal segments from only one horizontal grid line, while S_{hor} contains vertical segments from only one vertical grid line. In a bipartite graph the vertices can be partitioned into two independent sets A and B, one of which contains at least half of the vertices. Hence, the larger of the two sets S_{ver} and S_{hor} contains an independent set of size at least $\frac{n+t}{4}$. □

The proof of Lemma 5.9 heavily depends on the following classic theorem of Erdős and Szekeres, here rephrased in the plane setting. We say that a sequence of points in the plane is *non-decreasing* if both their first and second coordinates are non-decreasing along the sequence; it is *non-increasing* if the first coordinate is non-decreasing along the sequence while the second coordinate is non-increasing.

Theorem 5.10 (Erdős, Szekeres [ES35])**.** *Given n distinct points in the plane, it is always possible to choose at least $\sqrt{n}$ of those points and arrange them into a sequence which is either non-increasing or non-decreasing.*

Proof of Lemma 5.9. This construction exploits grid lines of even type, so those with an even number of segments on them. With the help of Theorem 5.10 we first construct a poly-line cutting through the segments. Then we use this poly-line to define two sets of pairwise disjoint segments in $\mathcal{S}$, one of which has the desired size.

Recall that meeting points are the points in which two parallel segments of $\mathcal{S}$ intersect. A meeting point on a grid line naturally partitions the segments lying on this line into two parts: those to the left of it and to the right of it (for horizontal lines), or those above it and below it (for vertical lines). Call a meeting point a *candidate point* if both those parts have odd cardinalities. Note that thus, candidate points only occur on grid lines with an even number of segments. Further, in total there are $s_{\text{even}}/2$ candidate points.

By Theorem 5.10, there exists either a non-increasing or a non-decreasing sequence of $\sqrt{s_{\text{even}}/2}$ candidate points. Suppose without loss of generality that the sequence is non-increasing and of maximum possible length. We call *cutting points* those candidate points which occur in the sequence and denote the number of cutting points by C. Observe that $C \geq \sqrt{s_{\text{even}}/2}$. For every two consecutive cutting points, connect them with a segment. Then consider two half-lines with negative inclinations, one ending at the first cutting point and one starting at the last cutting point. This gives a poly-line intersecting all grid lines. We call this path the *cut*.

Using the cut, we construct two sets of segments $\mathcal{S}_{\text{blue}}$ and $\mathcal{S}_{\text{orange}}$; see Figure 5.1.

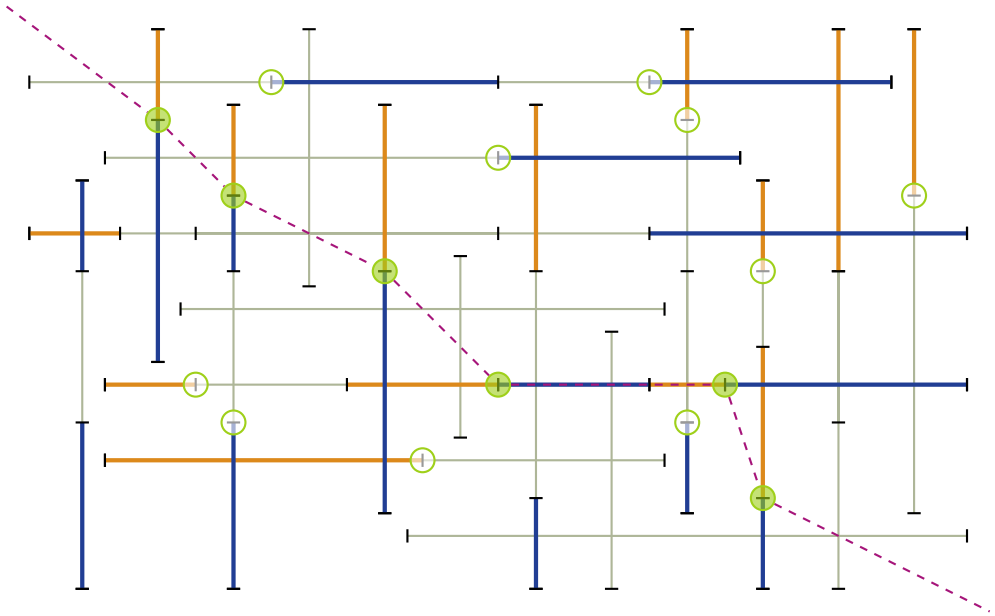

Figure 5.1: Selection of the two independent sets in the proof of Lemma 5.9. The endpoints of each segment, and in consequence all meeting points, are indicated by a perpendicular dash. Candidate points are highlighted by a green circle, where the circle is filled on cutting points. The red dashed line is the cut. Blue and orange segments are those chosen in one of the sets $\mathcal{S}_{\text{blue}}$ and $\mathcal{S}_{\text{orange}}$, respectively.

Construction of $\mathcal{S}_{\text{blue}}$: The set $\mathcal{S}_{\text{blue}}$ is constructed as follows. For each vertical grid line, start from the segment with the lowest endpoint and choose every second segment with the upper endpoint on the cut or below. Next, for each horizontal grid line, start from the segment with the right-most endpoint and choose every second segment with the left endpoint on the cut or to the right.

Construction of $\mathcal{S}_{\text{orange}}$: The set $\mathcal{S}_{\text{orange}}$ is defined symmetrically to $\mathcal{S}_{\text{blue}}$. Namely, for each vertical grid line, start from the segment with the highest endpoint and choose every second segment with the lower endpoint on the cut or above. For each horizontal grid line, start from the segment with the left-most endpoint and choose every second segment with the right endpoint on the cut or to the left.

If the sequence would be non-decreasing, the choice strategy for horizontal segments would be inverted between $\mathcal{S}_{\text{blue}}$ and $\mathcal{S}_{\text{orange}}$.

We argue that the segments of $\mathcal{S}_{\text{blue}}$ are pairwise disjoint. Note that the segments lying on the bottom-left side of the cut are vertical and pairwise disjoint by the construction. Those segments lying on the top-right side of the cut are horizontal and also pairwise disjoint. It remains to argue that there is no pair consisting of a vertical segment and a horizontal from $\mathcal{S}_{\text{blue}}$ which intersect at a point lying on the cut. Recall that since the representation is favorable, such an intersection point would lie in the interiors of both segments. This

would imply that either the vertical segment would have the top endpoint strictly above the cut, or the horizontal segment would have the left endpoint strictly to the left of the cut. A contradiction with the construction of $\mathcal{S}_{\text{blue}}$. A symmetric argument applied to $\mathcal{S}_{\text{orange}}$ shows that the segments of $\mathcal{S}_{\text{orange}}$ are also pairwise disjoint.

It remains to show that $\mathcal{S}_{\text{blue}} \cup \mathcal{S}_{\text{orange}}$ has at least $\frac{n}{2} + \frac{\sqrt{2s_{\text{even}}}}{2} - \frac{\ell_{\text{odd}}}{2}$ segments.

Consider a grid line of even type, thus with an even number of segments on it. For each candidate point on this line which is not a cutting point, exactly one segment containing this candidate point is in $\mathcal{S}_{\text{blue}} \cup \mathcal{S}_{\text{orange}}$. However, for each cutting point on this line, both segments meeting at this cutting point are included in $\mathcal{S}_{\text{blue}} \cup \mathcal{S}_{\text{orange}}$, as there is an odd number of segments on either side of the cutting point. This means that on each grid line of even type, the total number of segments included in $\mathcal{S}_{\text{blue}} \cup \mathcal{S}_{\text{orange}}$ is exactly half of all segments, plus one segment for each cutting point on the grid line.

Consider now a grid line of odd type, thus with an odd number of segments on it. The sets $\mathcal{S}_{\text{blue}}$ and $\mathcal{S}_{\text{orange}}$ contain every second segment starting from the outermost ones. Without the cut, this would include half of the segments lying on the line rounded up. Since there is an odd number of segments on the grid line, the cut crosses it at a single point. Due to this, at most one segment is removed from $\mathcal{S}_{\text{blue}} \cup \mathcal{S}_{\text{orange}}$. Meaning that among the segments lying a grid line of odd type, at least half rounded down are included in $\mathcal{S}_{\text{blue}} \cup \mathcal{S}_{\text{orange}}$. Thus, we lose at most $1/2$ for each grid line of odd type.

Together, this gives that $\mathcal{S}_{\text{blue}} \cup \mathcal{S}_{\text{orange}}$ contains at least

$$\frac{s_{\text{even}}}{2} + C + \frac{s_{\text{odd}}}{2} - \frac{\ell_{\text{odd}}}{2} \geq \frac{n}{2} + \frac{\sqrt{2s_{\text{even}}}}{2} - \frac{\ell_{\text{odd}}}{2}$$

segments. By choosing the larger of the two sets, we obtain an independent set of the desired size. □

5.3 The Upper Bound: Proof of Theorem 5.2

In this section, we construct families of axis-parallel segments whose intersection graphs satisfying the requirements of Theorem 5.2. In fact, we prove the following stronger statement.

Theorem 5.11 (Full version of Theorem 5.2)**.** *For any integer $k \geq 1$, there exists a graph G_k in $\mathcal{G}_{\text{seg}}$ on $4k^2$ vertices with clique covering number $\theta(G_k) = 2k^2$, fractional independence number $\alpha^\star(G_k) = 2k^2$, and independence number*

$$\alpha(G_k) = k^2 + 3k - 2.$$

Note that Corollaries 5.5 and 5.6 follow from Theorem 5.11 by considering $G = G_k$ for k large enough depending on $1/\varepsilon$. The remainder of this section is devoted to the proof of Theorem 5.11.

Fix an integer $k \geq 1$. We construct a set of $4k^2$ axis-parallel segments $\mathcal{M}_k$. The set $\mathcal{M}_k$ is constructed using k sets with $4k$ segments each; these sets will be called *k-boxes*. A *k-box* is a set of $4k$ axis-parallel segments distributed on k horizontal and k vertical grid lines, each with exactly two segments lying on it. For every grid line, the two segments intersect in a single point, which we call *meeting point*. In the construction of a k-box, the meeting points are arranged in a diagonal from the top left to the bottom right, see the case $k = 6$ in Figure 5.2. The *up segments* (respectively *down segments*) of a k-box are the segments lying vertically above (respectively below) a meeting point. Similarly, define the *left* and *right segments* of a k-box.

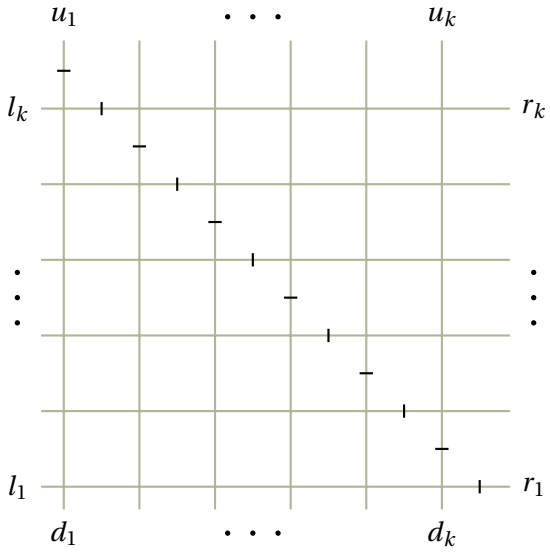

Figure 5.2: A 6-box. The meeting points are represented by a perpendicular dash. Thus, every line contains two segments of the box, whose only intersection is the meeting point on this line.

To construct $\mathcal{M}_k$, consider a large square and place k different k-boxes $\{\mathcal{B}_i\}_{i=1}^{k}$ along its diagonal from the bottom left to the top right. Then, prolong each segment away from the meeting point until it touches a side of the square, see Figure 5.3. The construction results in the set $\mathcal{M}_k$ consisting of $4k^2$ segments. We note that $\mathcal{M}_k$ is a favorable representation of its intersection graph in the sense introduced in Section 5.2. Also, perhaps not surprisingly, the construction is inspired by a tight example for the Erdős-Szekeres Theorem (Theorem 5.10). Note that it also proves tightness of the bound provided by Lemma 5.9.

We are left with verifying the asserted properties of $\mathcal{M}_k$. For this, we introduce some notation and definitions. Let $\mathcal{I}$ be a set of pairwise disjoint segments in $\mathcal{M}_k$. A k-box $\mathcal{B}_i$ of $\mathcal{M}_k$ is said to be *interesting* for $\mathcal{I}$ if $\mathcal{B}_i \cap \mathcal{I}$ contains either at least one down segment and one right

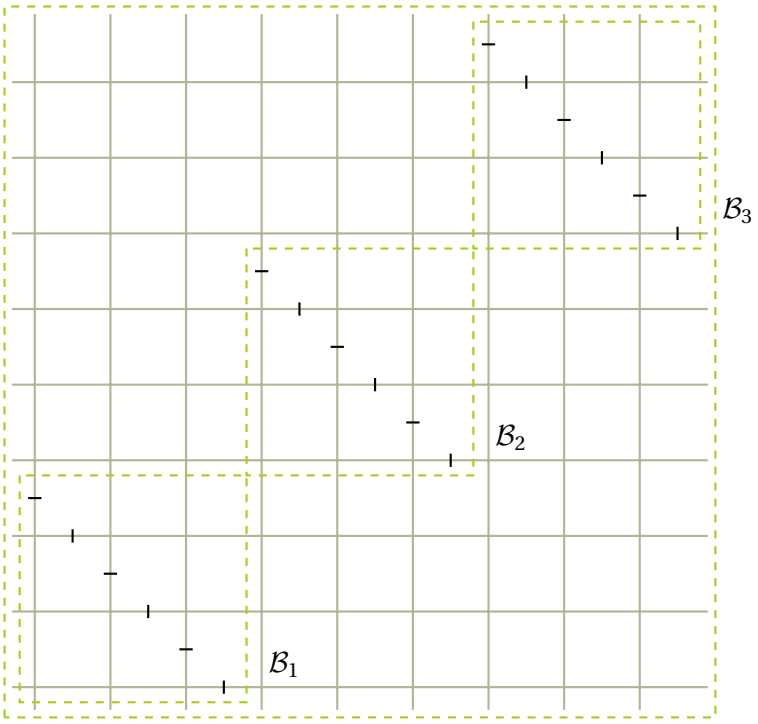

Figure 5.3: The set $\mathcal{M}_3$. The meeting points are represented by perpendicular dashes. The dashed green lines indicate bounds of the large square and the three 3-boxes.

segment, or at least one up segment and one left segment. Otherwise, the k-box is *boring* for $\mathcal{I}$. Distinguishing between interesting and boring boxes allows more precise estimates on the maximum possible cardinality of $\mathcal{I}$.

In the next two lemmas, we consider $\mathcal{I}$ to be a set of pairwise disjoint segments in $\mathcal{M}_k$.

Lemma 5.12. *For any k-box $\mathcal{B}$ in $\mathcal{M}_k$, it holds that $|\mathcal{B} \cap \mathcal{I}| \leq 2k$. Moreover, if $\mathcal{B}$ is boring for $\mathcal{I}$, then $|\mathcal{B} \cap \mathcal{I}| \leq k+1$.*

Proof. The first statement holds because $\mathcal{I}$ contains at most one segment per line, and there are $2k$ lines in a box: k vertical and k horizontal lines.

Assume now that $\mathcal{B}$ is a box which is boring for $\mathcal{I}$. Enumerate the up and down segments of $\mathcal{B}$ from left to right as $U = \{u_1, \ldots, u_k\}$ and $D = \{d_1, \ldots, d_k\}$, respectively; further, enumerate the right and left segments from top to bottom as $R = \{r_1, \ldots, r_k\}$ and $L = \{l_1, \ldots, l_k\}$, respectively; see Figure 5.2. If all segments of $\mathcal{B} \cap \mathcal{I}$ are pairwise parallel (that is, they are either all vertical or all horizontal), then $|\mathcal{B} \cap \mathcal{I}| \leq k$ since $\mathcal{I}$ can contain only one segment per line. There are two cases left to check: $\mathcal{B}$ either contains only up and right segments, or only down and left segments. Observe that $U \cup R$ can be partitioned into $k+1$ parts as follows: u_1 and r_k are singleton parts, the remaining segments are partitioned into $k-1$ pairs of intersecting segments given by $\{u_{i+1}, r_i\}_{i=1}^{k-1}$. Similarly, $D \cup L$ can be partitioned into k pairs of intersecting

segments $\{d_i, l_i\}_{i=1}^{k}$. The independent set $\mathcal{I}$ can contain at most one segment from each part of these partitions. Hence, $|\mathcal{B} \cap \mathcal{I}| \leq k+1$ in both cases. □

Lemma 5.13. *There are at most two boxes which are interesting for $\mathcal{I}$.*

Proof. We show that there is at most one interesting box with at least one up and one left segment included in $\mathcal{I}$. A symmetric argument then shows that there is at most one interesting box with at least one down and one right segment included in $\mathcal{I}$, implying that there are at most two interesting boxes in total.

For the sake of contradiction, assume $\mathcal{M}_k$ has two distinct interesting boxes $\mathcal{B}, \mathcal{B}'$ of the first kind. Then, either an up segment of $\mathcal{B} \cap \mathcal{I}$ intersects a left segment of $\mathcal{B}' \cap \mathcal{I}$, or vice-versa. This contradicts the fact that all segments of $\mathcal{I}$ are pairwise disjoint. □

With Lemmas 5.12 and 5.13 in place, we are in position to finish the proof of Theorem 5.11. Let G_k be the intersection graph of $\mathcal{M}_k$. By construction, the set $\mathcal{M}_k$ consists of $4k^2$ axis-parallel segments and G_k is in $\mathcal{G}_{\text{seg}}$.

First, we compute the clique covering number and the fractional independence number of G_k. Observe that G_k is triangle-free, hence every clique in G_k is of size at most 2. It follows that every clique covering of G_k is of size at least $\frac{|\mathcal{M}_k|}{2} = 2k^2$, that is, $\theta(G_k) \leq 2k^2$. On the other hand, taking every vertex of G_k with multiplicity $1/2$ gives a fractional independent set of size $\frac{|\mathcal{M}_k|}{2} = 2k^2$, implying that $\alpha^\star(G_k) \geq 2k^2$. Since $\theta(H) \geq \alpha^\star(H)$ for every triangle-free graph H, we conclude that

$$\theta(G_k) = \alpha^\star(G_k) = 2k^2.$$

It remains to prove that $\alpha(G_k) = k^2 + 3k - 2$. We give a set of pairwise disjoint segments in $\mathcal{M}_k$, corresponding to an independent set in G_k. This shows that $\alpha(G_k) \geq k^2 + 3k - 2$. The set of segments in $\mathcal{M}_k$ consists of:

(i) the left and up segments of $\mathcal{B}_1$,
(ii) the right and down segments of $\mathcal{B}_2$, and
(iii) the right segments and the topmost up segment of B_i, for each $3 \leq i \leq k$.

This is a set of pairwise disjoint segments in $\mathcal{M}_k$ and it contains $2\cdot(2k)+(k-2)(k+1) = k^2+3k-2$ segments.

To show that $\alpha(G_k) \leq k^2 + 3k - 2$ we apply Lemma 5.12 and Lemma 5.13 to obtain that, for any set $\mathcal{I}$ of pairwise disjoint segments in $\mathcal{M}_k$,

$$|\mathcal{I}| = |\mathcal{M}_k \cap \mathcal{I}| = \sum_{i=1}^{k} |\mathcal{B}_i \cap \mathcal{I}| \leq 2 \cdot (2k) + (k-2)(k+1) = k^2 + 3k - 2.$$

Which concludes the proof of Theorem 5.11.

Bibliography

[AH07] Matthias Aschenbrenner and Raymond Hemmecke. Finiteness theorems in stochastic integer programming. *Foundations of Computational Mathematics*, 7(2):183–227, 2007.

[ALMM08] Vladimir Alekseev, Vadim Lozin, Dmitriy Malyshev, and Martin Milanič. The maximum independent set problem in planar graphs. In *Proceedings of the 33rd International Symposium on Mathematical Foundations of Computer Science*, volume 5162, pages 96–107. Springer, 2008.

[AvKS98] Pankaj K. Agarwal, Marc J. van Kreveld, and Subhash Suri. Label placement by maximum independent set in rectangles. *Computational Geometry*, 11(3-4):209–218, 1998.

[AW13] Anna Adamaszek and Andreas Wiese. Approximation Schemes for Maximum Weight Independent Set of Rectangles. In *Proceedings of the 54th Annual Symposium on Foundations of Computer Science*, pages 400–409. IEEE Computer Society, 2013.

[AWZ17] Stephan Artmann, Robert Weismantel, and Rico Zenklusen. A strongly polynomial algorithm for bimodular integer linear programming. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, page 1206–1219. ACM, 2017.

[Bak94] Brenda S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. ACM*, 41(1):153–180, 1994.

[Baz95] Cristina Bazgan. Schémas d'approximation et complexité paramétrée. *Rapport de stage de DEA d'Informatiquea Orsay*, 300, 1995. In French.

[BFP+72] Manuel Blum, Robert W. Floyd, Vaughan R. Pratt, Ronald L. Rivest, and Robert Endre Tarjan. Linear time bounds for median computations. In *Proceedings of the fourth annual ACM symposium on Theory of computing*, pages 119–124. ACM, 1972.

[BSW14] Paul S. Bonsma, Jens Schulz, and Andreas Wiese. A constant-factor approximation algorithm for unsplittable flow on paths. *SIAM J. Comput.*, 43(2):767–799, 2014.

[BV14] Stephen P. Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2014.

[CCPW22] Marco Caoduro, Jana Cslovjecsek, Michał Pilipczuk, and Karol Węgrzycki. Independence number of intersection graphs of axis-parallel segments. *CoRR*, abs/2205.15189, 2022. (Submitted to Journal of Combinatorial Geometry).

[CE16] Julia Chuzhoy and Alina Ene. On Approximating Maximum Independent Set of Rectangles. In *Proceedings of the 57th Annual Symposium on Foundations of Computer Science*, pages 820–829. IEEE Computer Society, 2016.

[CEH+21] Jana Cslovjecsek, Friedrich Eisenbrand, Christoph Hunkenschröder, Lars Rohwedder, and Robert Weismantel. Block-structured integer and linear programming in strongly polynomial and near linear time. In *Proceedings of the 2021 Symposium on Discrete Algorithms*, pages 1666–1681. SIAM, 2021.

[CEP+21] Jana Cslovjecsek, Friedrich Eisenbrand, Michał Pilipczuk, Moritz Venzin, and Robert Weismantel. Efficient sequential and parallel algorithms for multistage stochastic integer programming using proximity. In *Proceedings of the 29th Annual European Symposium on Algorithms*, volume 204, pages 33:1–33:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[CFPS15] José R. Correa, Laurent Feuilloley, Pablo Pérez-Lantero, and José A. Soto. Independent and hitting sets of rectangles intersecting a diagonal line: algorithms and complexity. *Discrete & Computational Geometry*, 53(2):344–365, 2015.

[CGST86] William J. Cook, A. M. H. Gerards, Alexander Schrijver, and Éva Tardos. Sensitivity theorems in integer linear programming. *Math. Program.*, 34(3):251–264, 1986.

[Cla86] Kenneth L. Clarkson. Linear programming in $O(n \times 3^{d^2})$ time. *Inf. Process. Lett.*, 22(1):21–24, 1986.

[CM18] Lin Chen and Dániel Marx. Covering a tree with rooted subtrees - parameterized and approximation algorithms. In *Proceedings of the Twenty-Ninth Annual Symposium on Discrete Algorithms*, pages 2801–2820. SIAM, 2018.

[CMYZ17] Lin Chen, Dániel Marx, Deshi Ye, and Guochuan Zhang. Parameterized and approximation results for scheduling with a low rank processing time matrix. In *Proceedings of the 34th Symposium on Theoretical Aspects of Computer Science*, volume 66, pages 22:1–22:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[CPW22] Jana Cslovjecsek, Michał Pilipczuk, and Karol Węgrzycki. Parameterized approximation for maximum weight independent set of rectangles and segments. (Submitted to Symposium on Computational Geometry), 2022.

[CRZ20] Clément Carbonnel, Miguel Romero, and Stanislav Zivný. Point-Width and Max-CSPs. *ACM Trans. Algorithms*, 16(4):54:1–54:28, 2020.

[CT97] Marco Cesati and Luca Trevisan. On the efficiency of polynomial time approximation schemes. *Information Processing Letters*, 64(4):165–171, 1997.

[CW21] Parinya Chalermsook and Bartosz Walczak. Coloring and maximum weight independent set of rectangles. In *Proceedings of the 2021 Symposium on Discrete Algorithms*, pages 860–868. SIAM, 2021.

[DDvtH16] Pål Grønås Drange, Markus S. Dregi, and Pim van 't Hof. On the computational complexity of vertex integrity and component order connectivity. *Algorithmica*, 76(4):1181–1202, 2016.

[DF92] Jeffrey S. Doerschler and Herbert Freeman. A rule-based system for dense-map name placement. *Commun. ACM*, 35(1):68–79, 1992.

[Dic13] Leonard Eugene Dickson. Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. *American Journal of Mathematics*, 35(4):413–422, 1913.

[DLY21] Sally Dong, Yin Tat Lee, and Guanghao Ye. A nearly-linear time algorithm for linear programs with small treewidth: A multiscale representation of robust central path. In *Proceedings of the 53rd Annual Symposium on Theory of Computing*, pages 1784–1797. ACM, 2021.

[Dye86] Martin E. Dyer. On a multidimensional search technique and its application to the euclidean one-centre problem. *SIAM J. Comput.*, 15(3):725–738, 1986.

[EHK18] Friedrich Eisenbrand, Christoph Hunkenschröder, and Kim-Manuel Klein. Faster algorithms for integer programs with block structure. In *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming*, volume 107, pages 49:1–49:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[EHK+19] Friedrich Eisenbrand, Christoph Hunkenschröder, Kim-Manuel Klein, Martin Koutecký, Asaf Levin, and Shmuel Onn. An algorithmic theory of integer programming. *CoRR*, abs/1904.01361, 2019.

[EJS05] Thomas Erlebach, Klaus Jansen, and Eike Seidel. Polynomial-Time Approximation Schemes for Geometric Intersection Graphs. *SIAM J. Comput.*, 34(6):1302–1323, 2005.

[ES35] Paul Erdös and George Szekeres. A combinatorial problem in geometry. *Compositio mathematica*, 2:463–470, 1935.

[EW20a] Friedrich Eisenbrand and Robert Weismantel. Proximity results and faster algorithms for integer programming using the steinitz lemma. *ACM Trans. Algorithms*, 16(1):5:1–5:14, 2020.

[EW20b] Friedrich Eisenbrand and Robert Weismantel. Proximity results and faster algorithms for integer programming using the steinitz lemma. *ACM Trans. Algorithms*, 16(1):5:1–5:14, 2020.

[FK93] Dmitry Fon-Der-Flaass and Alexandr V. Kostochka. Covering boxes by points. *Discret. Math.*, 120(1-3):269–275, 1993.

[FMMT01] Takeshi Fukuda, Yasuhiko Morimoto, Shinichi Morishita, and Takeshi Tokuyama. Data mining with optimized two-dimensional association rules. *ACM Trans. Database Syst.*, 26(2):179–213, 2001.

[FPT82] Robert J. Fowler, Mike Paterson, and Steven L. Tanimoto. Optimal packing and covering in the plane are NP-complete. *Networks*, 12(4):459–467, 1982.

[Fre90] Eugene C. Freuder. Complexity of k-tree structured constraint satisfaction problems. In *Proceedings of the 8th National Conference on Artificial Intelligence*, pages 4–9. AAAI Press / The MIT Press, 1990.

[GHR+95] Raymond Greenlaw, H Hoover, Walter Ruzzo, et al. *Limits to parallel computation: P-completeness theory*. Oxford University Press on Demand, 1995.

[GKM+22] Waldo Gálvez, Arindam Khan, Mathieu Mari, Tobias Mömke, Madhusudhan Reddy Pittu, and Andreas Wiese. A 3-approximation algorithm for Maximum Independent Set of Rectangles. In *Proceedings of the 2022 Symposium on Discrete Algorithms*, pages 894–905. SIAM, 2022.

[GKW19] Fabrizio Grandoni, Stefan Kratsch, and Andreas Wiese. Parameterized approximation schemes for Independent Set of Rectangles and Geometric Knapsack. In *Proceedings of the 27th Annual European Symposium on Algorithms*, volume 144, pages 53:1–53:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[GL85] András Gyárfás and Jenö Lehel. Covering and coloring problems for relatives of intervals. *Discret. Math.*, 55(2):167–180, 1985.

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

[GO18] Robert Ganian and Sebastian Ordyniak. The complexity landscape of decompositional parameters for ILP. *Artif. Intell.*, 257:61–71, 2018.

[GS80] Victor S Grinberg and Sergey V Sevast'yanov. Value of the steinitz constant. *Functional Analysis and Its Applications*, 14(2):125–126, 1980.

[Han72] Denis Hanson. On the product of the primes. *Canadian Mathematical Bulletin*, 15(1):33–37, 1972.

[HK10] Alan J. Hoffman and Joseph B. Kruskal. Integral boundary points of convex polyhedra. In *50 Years of Integer Programming 1958-2008 - From the Early Years to the State-of-the-Art*, pages 49–76. Springer, 2010.

[HOR13] Raymond Hemmecke, Shmuel Onn, and Lyubov Romanchuk. N-fold integer programming in cubic time. *Math. Program.*, 137(1-2):325–341, 2013.

[HS03] Raymond Hemmecke and Rüdiger Schultz. Decomposition of test sets in stochastic integer programming. *Math. Program.*, 94(2-3):323–341, 2003.

[JKL21] Klaus Jansen, Kim-Manuel Klein, and Alexandra Lassota. The double exponential runtime is tight for 2-stage stochastic ILPs. In *Proceedings of the 22nd International Conference on Integer Programming and Combinatorial Optimization*, volume 12707, pages 297–310. Springer, 2021.

[JKMR21] Klaus Jansen, Kim-Manuel Klein, Marten Maack, and Malin Rau. Empowering the configuration-ip: new PTAS results for scheduling with setup times. *Mathematical Programming*, pages 1–35, 2021.

[JLR20] Klaus Jansen, Alexandra Lassota, and Lars Rohwedder. Near-linear time algorithm for n-fold ilps via color coding. *SIAM J. Discret. Math.*, 34(4):2282–2299, 2020.

[JR19] Klaus Jansen and Lars Rohwedder. On integer programming and convolution. In *Proceedings of the 10th Innovations in Theoretical Computer Science Conference*, volume 124, pages 43:1–43:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[Kan87] Ravi Kannan. Minkowskiś convex body theorem and integer programming. *Math. Oper. Res.*, 12(3):415–440, 1987.

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations*, pages 85–103, Boston, MA, 1972. Plenum Press, New York.

[KK06] Jan Kára and Jan Kratochvíl. Fixed parameter tractability of independent set in segment intersection graphs. In *Proceedings of the Second International Workshop on Parameterized and Exact Computation*, volume 4169, pages 166–174. Springer, 2006.

[KK18] Dusan Knop and Martin Koutecký. Scheduling meets n-fold integer programming. *Journal of Scheduling*, 21(5):493–503, 2018.

[KKL+19] Dusan Knop, Martin Koutecký, Asaf Levin, Matthias Mnich, and Shmuel Onn. Multitype integer monoid optimization and applications. *CoRR*, abs/1909.07326, 2019.

[KKM20a] Dusan Knop, Martin Koutecký, and Matthias Mnich. Combinatorial n-fold integer programming and applications. *Math. Program.*, 184(1):1–34, 2020.

[KKM20b] Dusan Knop, Martin Koutecký, and Matthias Mnich. Voting and bribing in single-exponential time. *ACM Trans. Economics and Comput.*, 8(3):12:1–12:28, 2020.

[Kle22] Kim-Manuel Klein. About the complexity of two-stage stochastic IPs. *Math. Program.*, 192(1):319–337, 2022.

[KLO18] Martin Koutecký, Asaf Levin, and Shmuel Onn. A parameterized strongly polynomial algorithm for block structured integer programs. In *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming*, volume 107, pages 85:1–85:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[KN90] Jan Kratochvíl and Jaroslav Nešetřil. Independent set and clique problems intersection-defined classes of graphs. *Commentationes Mathematicae Universitatis Carolinae*, 31(1):85–93, 1990.

[KN08] Seog-Jin Kim and Kittikorn Nakprasit. Coloring the complements of intersection graphs of geometric figures. *Discret. Math.*, 308(20):4589–4594, 2008.

[KPW20] Dusan Knop, Michał Pilipczuk, and Marcin Wrochna. Tight complexity lower bounds for integer linear programming with few constraints. *ACM Trans. Comput. Theory*, 12(3):19:1–19:19, 2020.

[KR22] Kim-Manuel Klein and Janina Reuter. Collapsing the tower - on the complexity of multistage stochastic ips. In *Proceedings of the 2022 Symposium on Discrete Algorithms*, pages 348–358. SIAM, 2022.

[LHOW08] Jesús A. De Loera, Raymond Hemmecke, Shmuel Onn, and Robert Weismantel. N-fold integer programming. *Discret. Optim.*, 5(2):231–241, 2008.

[LJ83] Hendrik W. Lenstra Jr. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8(4):538–548, 1983.

[LNO02] Liane Lewin-Eytan, Joseph Naor, and Ariel Orda. Routing and admission control in networks with advance reservations. In *Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization*, volume 2462, pages 215–228. Springer, 2002.

[LP09] László Lovász and Michael D Plummer. *Matching theory*, volume 367. American Mathematical Soc., 2009.

[Mar05] Dániel Marx. Efficient approximation schemes for geometric problems? In *Proceedings of the 13th Annual European Symposium on Algorithms*, volume 3669, pages 448–459. Springer, 2005.

[Mar06] Dániel Marx. Parameterized Complexity of Independence and Domination on Geometric Graphs. In *Proceedings of the Second International Workshop on Parameterized and Exact Computation, Second International Workshop*, volume 4169, pages 154–165. Springer, 2006.

[Meg84] Nimrod Megiddo. Linear programming in linear time when the dimension is fixed. *J. ACM*, 31(1):114–127, 1984.

[Mit21] Joseph S. B. Mitchell. Approximating maximum independent set for rectangles in the plane. In *Proceedings of the 62nd Annual Symposium on Foundations of Computer Science*, pages 339–350. IEEE, 2021.

[NPT92] Carolyn Haibt Norton, Serge A. Plotkin, and Éva Tardos. Using separation algorithms in fixed dimension. *J. Algorithms*, 13(1):79–98, 1992.

[Onn10a] Shmuel Onn. Nonlinear discrete optimization. *Zurich Lectures in Advanced Mathematics, European Mathematical Society*, page 75, 2010.

[Onn10b] Shmuel Onn. *Nonlinear Discrete Optimization: An Algorithmic Theory*. European Mathematical Society Publishing House, 2010.

[Pap81] Christos H. Papadimitriou. On the complexity of integer programming. *J. ACM*, 28(4):765–768, 1981.

[RRVS14] Felix Reidl, Peter Rossmanith, Fernando Sánchez Villaamil, and Somnath Sikdar. A faster parameterized algorithm for treedepth. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming*, volume 8572, pages 931–942. Springer, 2014.

[RS84] Neil Robertson and Paul D. Seymour. Graph minors. III. Planar tree-width. *J. Comb. Theory, Ser. B*, 36(1):49–64, 1984.

[RS95] Neil Robertson and Paul D. Seymour. Graph minors. XIII. The disjoint paths problem. *J. Comb. Theory, Ser. B*, 63(1):65–110, 1995.

[RWZ21] Miguel Romero, Marcin Wrochna, and Stanislav Zivný. Treewidth-pliability and ptas for max-csps. In *Proceedings of the 2021 Symposium on Discrete Algorithms*, pages 473–483. SIAM, 2021.

[Sch98] Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, 1998.

[She91] James Shearer. A note on the independence number of triangle-free graphs, II. *Journal of Combinatorial Theory, Series B*, 53(2):300–307, 1991.

[Ste13] Ernst Steinitz. Bedingt konvergente reihen und konvexe systeme. *Journal der Reine und Angewandte Mathematik*, 143:128–175, 1913.

[Weg65] G Wegner. Über eine kombinatorisch-geometrische frage von hadwiger und debrunner. *Israel Journal of mathematics*, 3(4):187–198, 1965.

Jana Cslovjecsek

disopt.epfl.ch/jana-cslovjecsek | Discrete Optimization Group
jana.cslovjecsek@epfl.ch | EPFL MA B1 533
ORCID: 0000-0002-4214-8842 | CH-1015 Lausanne

Research Interests

Sparse Integer Programming
Combinatorial and Discrete Optimization
Computational Geometry

Education

since 2018 PhD student
École Polytechnique Fédérale de Lausanne,
Supervisor: Friedrich Eisenbrand

2018 Master of Science in Mathematics
École Polytechnique Fédérale de Lausanne

Thesis Titel: Popularity meets Pareto optimality in two sided matchings
Supervisors: Friedrich Eisenbrand, Yuri Faenza
Written at Columbia University in the City of New York

2016 Bachelor of Science in Mathematics,
École Polytechnique Fédérale de Lausanne,

Thesis Titel: Knots and their relation to braids
Supervisor: Senja Dominique, Kathryn Hess Bellwald

Teaching

2016, 2020, 2021 Algèbre Linéaire Avancée II, Prof. Friedrich Eisenbrand, teaching assistant

2019, 2020 Discrete Optimization, Prof. Friedrich Eisenbrand/Prof. Adam Marcus, main assistant

2019, 2020 Graph Theory, Prof. Friedrich Eisenbrand/Dr. Riccardo Maffucci, main assistant

Languages

German, native
English, fluent
French, fluent

Publications and Work in Progress

[1] Marco Caoduro, Jana Cslovjecsek, Michal Pilipczuk, and Karol Wegrzycki. Independence number of intersection graphs of axis-parallel segments. *CoRR*, abs/2205.15189, 2022.

[2] Jana Cslovjecsek, Friedrich Eisenbrand, Christoph Hunkenschröder, Lars Rohwedder, and Robert Weismantel. Block-structured integer and linear programming in strongly polynomial and near linear time. In *Proceedings of the 2021 Symposium on Discrete Algorithms*, pages 1666–1681. SIAM, 2021.

[3] Jana Cslovjecsek, Friedrich Eisenbrand, Michal Pilipczuk, Moritz Venzin, and Robert Weismantel. Efficient sequential and parallel algorithms for multistage stochastic integer programming using proximity. In *Proceedings of the 29th Annual European Symposium on Algorithms*, volume 204 of *LIPIcs*, pages 33:1–33:14, 2021.

[4] Jana Cslovjecsek, Friedrich Eisenbrand, Michal Pilipczuk, Moritz Venzin, and Robert Weismantel. Efficient sequential and parallel algorithms for multistage stochastic integer programming using proximity. In *Proceedings of the 29th Annual European Symposium on Algorithms*, volume 204 of *LIPIcs*, pages 33:1–33:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[5] Jana Cslovjecsek, Romanos-Diogenes Malikiosis, Márton Naszódi, and Matthias Schymura. Computing the covering radius of a polytope with an application to lonely runners. *Comb.*, 42(4):463–490, 2022.

[6] Jana Cslovjecsek, Michal Pilipczuk, and Karol Wegrzycki. n^{ϵ}-approximation for independent set of objects. (in write up), 2022.

[7] Jana Cslovjecsek, Michal Pilipczuk, and Karol Wegrzycki. Parameterized approximation for maximum weight independent set of rectangles and segments. (in write up), 2022.