UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

# MATCHING AND COVERING WITH BOXES

TESIS PARA OPTAR AL GRADO DE
DOCTOR EN CIENCIAS, MENCIÓN COMPUTACIÓN

**JAVIEL ROJAS LEDESMA**

PROFESORES GUÍAS:
JÉRÉMY BARBAY
PABLO PÉREZ LANTERO

MIEMBROS DE LA COMISIÓN:
BENJAMIN BUSTOS CÁRDENAS
NANCY HITSCHFELD KAHLER
DAVID KIRKPATRICK

Este trabajo fue financiado por CONICYT-PCHA/Doctorado Nacional/2013-63130209,
y los proyectos CONICYT Fondecyt/Regular nos 1120054, 1170366 y 1160543

SANTIAGO DE CHILE
2018

# Resumen

El estudio de las interacciones entre cajas multi-dimensionales (es decir, hiperrectángulos $d$-dimensionales alineados a los ejes) ha encontrado aplicaciones en distintas áreas, incluyendo geometría computacional, bases de datos, teoría de grafos y redes. Esta tesis considera varias preguntas abiertas sobre este tema, enfocándose en tres materias fundamentales: el cálculo de emparejamientos de un conjunto de puntos con cajas, la detección de redundancias en la región cubierta por un conjunto de cajas, y el cálculo de distintas medidas de dicha región. Se estudia la complejidad computacional de tres grupos respectivos de problemas, tanto en el peor caso, como dentro del marco de análisis adaptativos.

Primero se consideran problemas sobre el cálculo de distintas medidas de la región del espacio cubierta por un conjunto $\mathcal{B}$ de cajas. Se introduce el problema de calcular la distribución de profundidad de $\mathcal{B}$, que generaliza el cálculo de su medida de Klee y su profundidad máxima, respectivamente. Se describen distintos algoritmos para calcular la distribución de profundidad de un conjunto de cajas, y se prueban cotas computacionales superiores refinadas para los problemas de calcular la medida de Klee y la profundidad máxima de $\mathcal{B}$, respectivamente, considerando distintas medidas de dificultad de las instancias de estos problemas. Además, se demuestran distintas cotas inferiores condicionales para el problema de calcular la distribución de profundidad, que ayudan a entender su relación con otros problemas fundamentales en la computación.

Luego, se estudian distintos problemas sobre el cálculo de emparejamientos de pares de puntos coloreados en un conjunto finito mediante cajas. Un emparejamiento con cajas de un conjunto finito $\mathcal{S}$ de puntos, es un conjunto de cajas cerradas, disjuntas dos a dos, y tales que cada caja contiene exactamente dos puntos de $\mathcal{S}$. Los problemas que esta tesis considera difieren entre sí en restricciones tales como que las cajas deban emparejar solo a puntos del mismo color (llamados emparejamientos monocromáticos) o contener solo puntos de distintos colores (llamados emparejamientos bicromáticos), o restricciones sobre el conjunto de puntos, por ejemplo, que se requiera que estén en posición general. Se muestra que algunos de estos problemas son difíciles de resolver en tiempo polinomial, pero que sus soluciones óptimas se pueden aproximar hasta factores constantes en tiempo polinomial.

Finalmente, se consideran problemas sobre la eliminación de redundancias en la región del espacio cubierta por un conjunto de cajas multi-dimensionales. Se estudia el problema de encontrar un kernel de cobertura de tamaño mínimo, que consiste en, dado un conjunto $\mathcal{B}$ de cajas $d$-dimensionales, encontrar un subconjunto de $\mathcal{B}$ de tamaño mínimo que cubra la misma región que $\mathcal{B}$. Este problema es NP-difícil, pero como muchos problemas NP-difícil sobre grafos, se puede resolver en tiempo polinomial bajo distintas restricciones sobre el grafo inducido por $\mathcal{B}$. Esta tesis considera varias clases de grafos, y muestra que el problema de encontrar un kernel de cobertura de tamaño mínimo sigue siendo NP-difícil incluso para instancias severamente restringidas; y proporciona dos algoritmos de aproximación en tiempo polinomial para este problema.

# Abstract

The study of the interactions between multidimensional boxes (i.e., axis aligned $d$-dimensional hyperrectangles) has found important applications in distinct areas, including computational geometry, databases, graph theory and networks. We deal with some open questions on this matter, focusing on three main issues: finding matchings of a set of points with boxes, finding redundancies in the region covered by a set of boxes, and computing distinct measures of this region. We consider three corresponding groups of problems, and study their computational complexity, in both the worst-case and adaptive analysis frameworks.

We first consider problems on computing distinct measures of the region covered by a set $\mathcal{B}$ of boxes. We introduce the computation of the depth distribution of $\mathcal{B}$, which generalizes the computation of its Klee's measure and maximum depth, respectively. We describe distinct algorithms to compute the depth distribution of a set of boxes, and introduce refined computational upper bounds for the Klee's Measure and Maximum Depth problems, respectively, by considering distinct measures of difficulty of the input of these problems. We introduce various conditional lower bounds for the computational complexity of the Depth Distribution problem. These lower bounds not only provide insights on how fast the depth distribution can be computed, but also clarify the relation between the Depth Distribution problem and other fundamental problems in computer science.

Then, we consider problems on matching pairs of points in a given finite set with boxes in distinct settings. A matching of a set $\mathcal{S}$ of colored points with boxes is a set of axis-aligned closed rectangles, pair-wise disjoint, and such that each box contains exactly two points of $\mathcal{S}$. The problems we consider differ between each other in restrictions such as that boxes must match only points of the same color (monochromatic) or match only points of distinct colors (bichromatic), or restrictions over the point-set as, for instance, when the points are required to be in general position. We show that some of these problems are *hard* to solve in polynomial time, but that their optimal solution can be approximated up to constant factors in polynomial time.

Finally, we study problems on finding redundancies in the region covered by a set of multidimensional boxes. We consider the Minimum Coverage Kernel problem: given a set $\mathcal{B}$ of $d$-dimensional boxes, find a subset of $\mathcal{B}$ of minimum size covering the same region as $\mathcal{B}$. This problem is NP-hard, but as for many NP-hard problems on graphs, the problem becomes solvable in polynomial time under restrictions on the graph induced by the pairwise intersections of the boxes in $\mathcal{B}$. We consider various classes of graphs, show that the Minimum Coverage Kernel problem remains NP-hard even for severely restricted instances, and provide two polynomial-time approximation algorithms for this problem.

*A mi familia, la de siempre, y la nueva.*

# Acknowledgements

I would like to thank my supervisors Jérémy Barbay and Pablo Pérez Lantero for guiding and encouraging me throughout this long process. I appreciate all their contributions of time, ideas, and funding to make this experience productive and stimulating. But above all, I am grateful for their friendship and help since the moment in which the idea of doing this PhD (a hemisphere away from home) was being conceived.

I would like to thank the Academic Committee of the Department of Computer Science of the University of Chile, and to the rest of my thesis committee: Prof. Benjamin Bustos, Prof. Nancy Hitschfeld, and Prof. David Kirkpatrick, for their encouragement and insightful comments, which incentivized me to widen my research from various perspectives. I want to thank Carlos Ochoa and Luis Evaristo Caraballo, who co-authored part of the work presented in the fourth chapter of this thesis, for their contributions and stimulating discussions.

My sincere thanks to the amazing people at the Department, particularly to Angélica Aguirre and Sandra Gáez for their advice and patience throughout all these years. Special thanks also to Raimil Cruz, Guillermo de la Torre, Carlos Ochoa, and Eduardo Merino for creating such a good atmosphere there.

I must thank my family, specially my parents Daniel and Marisela, for their infinite encouragement and support along the way, and my lovely wife, who has provided more support than I could have ever asked for. I am profoundly grateful to Zulima, Taly, Silvita, María Eliana, Tencha and Manolo for all their love and invaluable help.

Finally, this research would have been impossible without the financial support of CONICYT-Chile through their PhD. Scholarship Program (CONICYT-PCHA/2013-63130209). I am also grateful to the Department of Computer Science of the University of Chile for the financial help to travel to various conferences.

# Contents

# List of Figures

# Chapter 1

# Introduction

The combinatorial complexity of the intersections occurring in a set of $d$-boxes (i.e., axis-aligned $d$-dimensional hyperrectangles) is so rich that many central problems in computational geometry can be stated as computing different properties or measures of these intersections. We extend the knowledge on this combinatorial complexity by tackling some of the open problems in the field, and finding new relations between central problems in computer science with problems on boxes, both well studied and new. This chapter describes the scope of this research, the open problems we consider, and the concrete contributions of this thesis.

The study of the interactions between boxes in a set occurs in various forms (e.g., measuring their union [44, 45], reporting their intersections [30, 96], stabbing them with points [4, 49, 119], using them to match points [2, 32, 124] or cover polygons [93, 97], etc.), and in several fields (e.g., databases [1, 115], graph theory [44, 67], combinatorial optimization [124], networks [17], etc.). In databases for instance, Abo Khamis et al. [1] described solutions for the problem of evaluating join queries by representing relations as a set of boxes, and reducing join queries to finding the points of the space which are not inside the union of these boxes (a problem closely related to the computation of the volume of the union of a set of boxes [45, 87, 114]). Answering range queries in a database (i.e., outputting the tuples which fall within a given range of the attributes) can be done by representing the data tuples as points in a high-dimensional space, representing the queries as boxes in the same space, and using data structures to find which points are covered by the query boxes [5, 49, 117]. In graph theory, it is possible to detect cliques of fixed size $k$ in a graph by representing the edges as $k$-dimensional boxes and finding whether there is any point left uncovered by the boxes within a properly chosen domain box [44]. Several scheduling problems can be solved via reductions to instances of the problem of covering a set of points with the minimum number of elements in a given set of boxes [20], closely related the problem of counting the number of points in a given set covered by an input set of boxes [127]. We study the combinatorial complexity of the interactions occurring in a set of boxes, considering the computational complexity of various problems on these interactions, both in the traditional framework of the worst-case analysis over instances of fixed size, and in the adaptive analysis framework for various parameters.

The worst-case analysis, the most common approach, can sometimes be too pessimistic: in several practical scenarios, worst case instances appears rarely. In an attempt to circumvent this, adaptive analysis measures the performance as a function not just of the input size, but of other parameters that capture, in various ways, the easiness or difficulty of the input instance. In the context of Sorting multisets, for instance, it should be easier to sort a list of elements which is given almost sorted, than a list of elements in a completely random order. Thus, several measures of *pre-sortedness* have been proposed, and there are algorithms sensitive to these measures [64, 109, 108]. This approach has been applied with success to classical problems such as Searching in a sorted array [27, 31], Sorting a permutation [51, 98, 104, 109, 108], computing the Maxima and the Convex Hull of a set of points [6, 11, 85, 86, 123], and many other problems [1, 21, 22, 25, 26, 58, 89].

This thesis focuses on three groups of problems: computing different measures of the region covered by a set of boxes (Section 1.1); finding matchings of a set of points with boxes (Section 1.2); and finding redundancies in the region covered by a set of boxes (Section 1.3). We briefly describe these groups of problems, together with some of their applications, and the open questions we consider. Then, we describe in Section 1.4 our results, and the structure of this document.

## 1.1 Measures on a Set of Boxes

Consider a set $\mathcal{B}$ of $n$ $d$-boxes, for fixed (constant) $d$. We focus on the computation of two measures on $\mathcal{B}$: the Klee's measure, and the maximum depth. The Klee's measure of $\mathcal{B}$ is the size of the region covered by $\mathcal{B}$, more formally, the hypervolume of the union of the boxes in $\mathcal{B}$ (see Figure 1.1.b for an illustration):

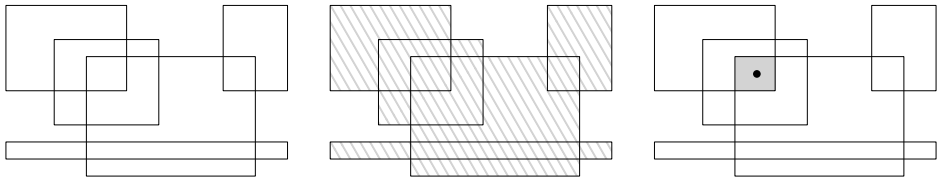
> Pb Klee's Measure problem: Given a set $\mathcal{B}$ of $n$ boxes in $\mathbb{R}^d$, compute the hypervolume of the union of the boxes in $\mathcal{B}$ (namely, the *Klee's measure* of $\mathcal{B}$).

Originally suggested on the line by Klee [87], its computation is well studied in higher dimensions [37, 38, 44, 45, 114], and can be performed in time within $\mathcal{O}(n^{d/2})$, using a paradigm called "Simplify, Divide and Conquer" introduced by Chan [45] in 2013. The Maximum Depth problem is very similar. The *depth* of a point $p \in \mathbb{R}^d$ with respect to a set of boxes $\mathcal{B}$ is the number of boxes in $\mathcal{B}$ that contain $p$. The Maximum Depth problem consists in finding a point with maximum depth with respect to a given set of boxes (see Figure 1.1.c for an illustration):

> Pb Maximum Depth problem: Given a set $\mathcal{B}$ of $n$ boxes in $\mathbb{R}^d$, compute the maximum depth among the points in $\mathbb{R}^d$ with respect to $\mathcal{B}$ (namely, the *maximum depth* of $\mathcal{B}$).

The computational complexity of the Maximum Depth problem is similar to that of the Klee's Measure problem, converging to the same complexity within $\mathcal{O}(n^{d/2})$ [45].

We study three main issues on these problems. The first direction is about the relation between the Klee's Measure and Maximum Depth problems. The known algorithms



(a) (b) (c)

Figure 1.1: The Klee's measure and maximum depth of a set of boxes. (a) A set of five boxes in the euclidean plane. (b) The Klee's measure of the set is the area of the dashed region. (c) Any point in the grayed region is covered by three boxes, hence it has depth three, which is the maximum depth of the set.

to compute these two measures are all strikingly similar, to the point that Chan [45] states that all known techniques used for computing the Klee's measure can be applied to the computation of the maximum depth. That would suggest a reduction from one to the other, but those two measures are completely distinct: the Klee's measure is a volume whose value might be a real number, while the maximum depth is a cardinality whose value is an integer in the range $[1..n]$. Given this, it is only natural to ask whether there is any way to formalize the close relationship between these two measures.

**Question 1.** Can the close relationship between Maximum Depth and Klee's Measure problems hinted by Chan [45] be formalized in some way?

The two other research directions originate from a result described by Chan [44]. He showed that both problems have the parameterized k-Clique problem as a special case, which implies that if the dimension is considered part of the input (i.e., it is not a constant), then the problem is $W[1]$-hard [59].

Pb k-Clique problem: Given a graph $\mathcal{G}$ of $n$ vertices, and a value $k \in [1..n]$, find whether there is a clique in $G$ of size $k$.

More importantly, Chan [44] conjectured that any *combinatorial* algorithm computing the Klee's measure or the maximum depth of a set of $n$ boxes requires within $\Omega(n^{d/2})$ operations in the worst case. This comes from the fact that any combinatorial algorithm computing these measures in time within $O(n^{\frac{d}{2}-\varepsilon})$ would imply ground-breaking results for the k-Clique problem. As a consequence of the conjectured lower bound, recent works have been focused on the study of special cases which can be solved faster than $\Omega(n^{d/2})$, like for instance when all the boxes are *orthants* [45], $\alpha$-*fat* boxes [37], or cubes [38]. However, it remains unknown whether there are measures separating *easy* instances for these problems from the *hard* ones *gradually* (as for Maxima [85, 6] for instance):

**Question 2.** Are there any measures, gradually separating *easy* instances of the Klee's Measure or the Maximum Depth problems from the *hard* ones, and are there algorithms which can take advantage from these measures?

For the k-Clique problem, the running time of the fastest combinatorial algorithms can be beaten by more than polynomial factors using Matrix Multiplication algorithms [15]. Thus, Chan [44] asked whether similar improvements are possible for the Klee's Measure or the Maximum Depth problems, or whether these problems are explicitly related:

> ? **Question 3.** Can algorithms for fast multiplication of matrices or polynomials be used for or inspire new algorithms for the Klee's Measure or the Maximum Depth problems? Is there any explicit relation between these four problems?

We study these problems, and give partial answers to these questions in Chapter 3.

## 1.2 Matching Points with Boxes

We also consider problems on matching points with two dimensional boxes. In general, problems on matchings have an important role in combinatorial graph theory, both for theoretical and applied aspects [16]. A matching of a finite set $\mathcal{S}$ of points with elements in a class $\mathcal{C}$ of geometric objects is a collection $M \subseteq \mathcal{C}$ such that each element of $M$ contains exactly two points of $\mathcal{S}$ and every point of $\mathcal{S}$ lies in at most one element of $M$. A geometric matching is called *strong* if the geometric objects are disjoint, and *perfect* if every point of $\mathcal{S}$ belongs to some element of $M$. The problem of finding geometric matchings was introduced by Ábrego et al. [2]; and the classes of geometric objects considered so far include segments [16, 32], triangles [19], squares [2], disks [2], and rectangles [33, 124]. Matching with these geometric objects is strongly connected to map labeling, the problem of annotating graphical features on maps and other diagrams [8, 32].

We consider a generalization where every point in $\mathcal{S}$ is colored either red or blue. In this setting, a matching is *monochromatic* if every matching object contains two points of the same color, and *bichromatic* if every matching object contains two points of different colors. In this context we study the following problems (see Figure 1.2 for an illustration):

> Pb Maximum Monochromatic Rectangle Matching (MonoMRM): Given a set $\mathcal{S} = R \cup B$ of red and blue points in $\mathbb{R}^2$, find a monochromatic strong matching of $\mathcal{S}$ with the maximum number of boxes.

> Pb Maximum Bichromatic Rectangle Matching (BicMRM): Given a set $\mathcal{S} = R \cup B$ of red and blue points in $\mathbb{R}^2$, find a bichromatic strong matching of $\mathcal{S}$ with the maximum number of boxes.

The study of the computational complexity of these problems here is particularly motivated by a question left open by Bereg et al. [32] who studied the computation of matchings of uncolored points with rectangles (i.e., the special case of MonoMRM where all the points have the same color). They showed that such sets of $n$ points admits a matching of at least $2\lfloor n/3 \rfloor$ of the points, and described an algorithm to find it in time within $\mathcal{O}(n \log n)$. However, the size of this matching can be far from optimal: there are instances where all the points can be matched while their algorithm matches only $2\lfloor n/3 \rfloor$ of the points. They were able to prove

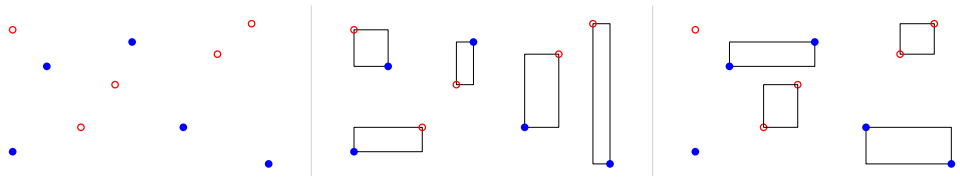



Figure 1.2: Strong monochromatic and bichromatic matchings of points with boxes. (a) A set of colored points. (b) A maximum-size strong bichromatic matching of the points. (c) A maximum-size strong monochromatic matching of the points.

that when the matching rectangles are restricted to squares, and the point-set is not required to be in a general position, then it is NP-hard to decide whether a perfect matching exists or not. However, they left open whether maximum size matchings with rectangles could be computed in polynomial time in this uncolored setting:

? **Question 4.** Can MonoMRM be solved in polynomial time, in the general and uncolored cases? Can BicMRM be solved in polynomial time?

The MonoMRM and BicMRM problems are special cases of the Maximum Independent Set of Rectangles (MISR) problem, a classical NP-hard problem in computational geometry and combinatorics [3, 7, 41, 42, 67, 77, 120]:

Pb Maximum Independent Set of Rectangles (MISR): Given a set $\mathcal{B}$ of $n$ axis-parallel rectangles (equivalently, 2-boxes), find a subset of pairwise disjoint rectangles of maximum cardinality.

Any $\alpha-$approximation algorithm for the MISR directly implies an $\alpha-$approximation algorithm for both MonoMRM and BicMRM. The general MISR admits a polynomial-time approximation algorithm, which with high probability produces an independent set of rectangles with within $\Omega(\frac{1}{\log\log n})$ times the number of rectangles in an optimal solution, $n$ being the number of rectangles in the input [41, 42]. However, the existence of a $\mathcal{O}(1)$-approximation algorithm, or a PTAS, for the MISR problem, is still open:

? **Question 5.** If we are unlikely to find efficient algorithms for the MonoMRM and BicMRM problems (e.g., if they are NP-hard), are there polynomial-time constant-factor approximation algorithms or PTAS for these problems?

There are instances of BicMRM which are known to be solvable in polynomial time. Soto and Telha [124] considered the special case where the matching rectangles are restricted to have a red point as bottom-left corner and a blue point as top-right corner, and showed that it can be solved in polynomial time. In general, it was unknown whether optimal monochromatic and bichromatic matchings of points with rectangles can be computed or approximated in polynomial time, but if it was the case of this being hard, it was also unknown whether there are more general classes of instances than the one described by Soto and Telha [124] which can be solved optimally in polynomial time:

**Question 6.** If MonoMRM and BicMRM are NP-hard, under which restricted settings can these problems be solved optimally in polynomial time?

We study these problems, and give answers to these questions in Chapter 4.

## 1.3 Removing Redundancies in a Set of Boxes

Finally, we study problems on finding redundancies in the region covered by a set of multidimensional boxes. Given a set $P$ of $n$ points, and a set $\mathcal{B}$ of $m$ boxes (i.e. axis-aligned closed hyper-rectangles) in $d$-dimensional space, the Box Cover problem consists in finding a set $\mathcal{C} \subseteq \mathcal{B}$ of minimum size such that $\mathcal{C}$ covers $P$. A special case is the Orthogonal Polygon Covering problem: given an orthogonal polygon $\mathcal{P}$ with $n$ edges, find a set of boxes $\mathcal{C}$ of minimum size whose union covers $\mathcal{P}$.

Pb Box Cover: Given a set $\mathcal{S}$ of $n$ points in $\mathbb{R}^d$ and a set $\mathcal{B}$ of $m$ boxes in $\mathbb{R}^d$, compute the subset $C$ of $\mathcal{B}$ of minimum size covering $\mathcal{S}$.

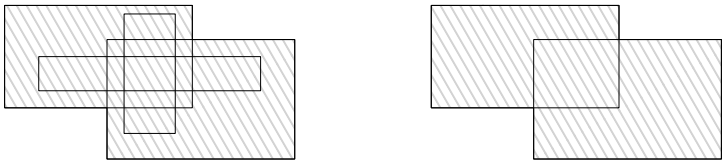
Pb Orthogonal Polygon Covering: given an orthogonal polygon $\mathcal{P}$ (i.e., a polygon with all its edges either vertical or horizontal), find a set $\mathcal{B}$ of boxes whose union is exactly $\mathcal{P}$, and such that the size of $\mathcal{B}$ is as small possible.

The Orthogonal Polygon Covering can be reduced in polynomial time to the Box Cover problem [67], and both are known to be NP-hard [55, 67]. However, their known approximabilities in polynomial time are different: while Box Cover can be approximated up to a factor within $\mathcal{O}(\log \mathtt{OPT})$, where $\mathtt{OPT}$ is the size of an optimal solution [39, 50]; Orthogonal Polygon Covering can be approximated up to a factor within $\mathcal{O}(\sqrt{\log n})$ [93]. Attempting to better understand what makes these problems hard, and why there is such a gap in their approximabilities, we introduce the notion of coverage kernels, and study their computational complexity.

Given a set $\mathcal{B}$ of $n$ $d$-dimensional boxes, a *coverage kernel* of $\mathcal{B}$ is a subset $\mathcal{K} \subseteq \mathcal{B}$ covering the same region as $\mathcal{B}$, and a minimum coverage kernel of $\mathcal{B}$ is a coverage kernel of minimum size. The computation of a minimum coverage kernel (namely, the Minimum Coverage Kernel problem) is intermediate between the Orthogonal Polygon Covering and the Box Cover problems: the first problem can be reduced in polynomial time to the Minimum Coverage Kernel problem, and this in turn can be reduced to the Box Cover problem (we explore this relation in details in Section 2.4).

The Minimum Coverage Kernel problem has found applications (under slight variations) in the compression of access control lists in networks [56], and in obtaining concise descriptions of structured sets in databases [95, 118]. Since Orthogonal Polygon Covering is NP-hard, the same holds for the Minimum Coverage Kernel problem.

Pb Minimum Coverage Kernel problem: Given a set $\mathcal{B}$ of $n$ boxes in $\mathbb{R}^d$, compute a minimum-size subset $C$ of $\mathcal{B}$ covering the same region as $\mathcal{B}$.



(a) (b)

Figure 1.3: A minimum coverage kernel of a set of boxes. (a) A set of four boxes covering the dashed region. (b) A subset of minimum size covering the same dashed region, namely, a minimum coverage kernel.

We are interested in the exact computation and approximability of Minimum Coverage Kernel in various restricted settings:

**Question 7.** Under which restrictions is the exact computation of Minimum Coverage Kernel still NP-hard?

**Question 8.** How precisely can one approximate the Minimum Coverage Kernel problem polynomial time?

When the interactions among the boxes in a set $\mathcal{B}$ are simple (e.g., when all the boxes are disjoint), a minimum coverage kernel of $\mathcal{B}$ can be computed in polynomial time. A natural way to capture the complexity of these interactions is through the intersection graph. The intersection graph of $\mathcal{B}$ is the un-directed graph with a vertex for each box, and in which two vertices are adjacent if and only the respective boxes intersect. When the intersection graph is a tree, for instance, each box of $\mathcal{B}$ is either completely covered by another, or present in any coverage kernel of $\mathcal{B}$, and thus a minimum coverage kernel can be computed efficiently. For NP-hard problem on graphs, a common approach to understand when does it become "easy" is to study restricted classes of graphs, in the hope to define some form of "boundary classes" of the input, separating "easy" from "hard" instances [13]. Based on this, we study the hardness of the problem under restricted classes of the intersection graph of the input.

## 1.4 Thesis Structure and Contributions

We provide partial answers to the questions mentioned above, and give new elements which extend the state of the art on their solutions. We list below the most relevant results, along with a description of the structure of this thesis. Chapter 2 introduces the the basic concepts, and briefly surveys the body of work relevant to this research. We present the results described above for each group of problems in Chapters 3, 4, and 5, respectively. Finally, in Chapter 6 we summarize the results and conclusions obtained in this research, discuss their impact, and describe possible future work directions.

**Chapter 3.** *Depth Distribution of a Set of Boxes.* We study the questions for the first group of problems in this chapter. We describe a first step towards a formalization of the close relation between Klee's Measure and Maximum Depth, in the form of a new problem: the computation of the Depth Distribution of a set $\mathcal{B}$ of boxes.

> Pb Depth Distribution problem: Given a set $\mathcal{B}$ of $n$ boxes in $\mathbb{R}^d$, compute the vector $(V_1, V_2, \ldots, V_n)$ of the volumes of the regions covered by exactly $1, 2, \ldots, n$ boxes of $\mathcal{B}$, respectively.

This natural problem allows to better understand the similarities between the Maximum Depth and the Klee's Measure problems, and has interesting applications and results of its own. The computation of the Depth Distribution of a set $\mathcal{B}$ of boxes generalizes both the computation of the Klee's measure and the maximum depth of $\mathcal{B}$, respectively. We focus on the computational complexity of the Depth Distribution problem. In Section 3.2, we provide upper bounds for this computational complexity. In the worst case over instances of fixed input size $n$ and dimension $d \geq 2$, we describe an algorithm with running time within $\mathcal{O}(n^{\frac{d+1}{2}} \log n)$, using space within $\mathcal{O}(n \log n)$ (Theorem 3.3). We refine the time bound for various measures of difficulty of the input, such as the profile of the input (Theorem 3.4) and treewidth and degeneracy of the intersection graph induced by the boxes (Theorem 3.7), by describing algorithms which compute the depth distribution in running time sensitive to the value of these difficulty measures.

In Section 3.3 we study lower bounds for the running time of computing the Depth Distribution. We show that the Depth Distribution problem generalizes not only Klee's Measure and Maximum Depth problems, but surprisingly many others apparently unrelated to sets of boxes. For this, we prove that the classical Integer Multiplication and Matrix Multiplication problems are special cases of Depth Distribution (Theorem 3.11). This allows us to provide a conditional lower bound for the computational complexity of the Depth Distribution problem of within $\Omega(n^{\frac{\omega}{2}})$, where $\omega$ is the Matrix Multiplication exponent [70]. This lower bound permits to argue that, unfortunately, the generalization of Klee's Measure and the Maximum Depth problems comes at a price: one cannot compute the depth distribution in the same asymptotic running time than the Klee's measure or the maximum depth.

**Part of these results were presented at the 23rd International Computing and Combinatorics Conference (COCOON'17) [24] under the title *"Depth Distribution in High Dimension"*. An extended version of this article was invited to an special issue of the journal Algorithmica, which is currently under review.**

**Chapter 4.** *Matching Colored Points with Boxes.* In this chapter, we consider the problems on matching points with rectangles. In Section 4.2, we study the hardness of the MonoMRM and BicMRM problems. We show that, as intuition indicates, both problems are NP-hard. These NP-hardness results follow by showing that deciding the existence of a perfect matching is NP-complete in each case. More importantly, we show that the hardness condition remains under restricted settings: we show that MonoMRM remains NP-hard even if the matching

rectangles are restricted to axis-aligned segments (Theorem 4.1), or the points are in general position (Theorem 4.3), or all the points have the same color (Theorem 4.5). The NP-hardness result under the uncolored setting answers a question posed by Bereg et al. [32] almost 10 years ago. For BicMRM, we show that the problem remains NP-hard even if the points are in general position (Theorem 4.6).

In Section 4.3 we complement these results by providing approximation algorithms for the MonoMRM and BicMRM problems. We extend the class of instances of MonoMRM and BicMRM known to be solvable in polynomial time (Lemma 4.7), generalizing a result by Soto and Telha [124]. We use this result as the core of two polynomial time 4-approximation algorithms for MonoMRM (Theorem 4.10) and BicMRM (Theorem 4.11), respectively.

**These results were published in 2017 in the Journal of Combinatorial Optimization (JOCO) [40] under the title "*Matching colored points with rectangles*".**

**Chapter 5:** *Covering Regions with Boxes.* In this chapter we study the problems related to removing redundancies in the region covered by a set of boxes. In Section 5.2, we study the Minimum Coverage Kernel problem under three restrictions of the intersection graph, commonly considered for other problems [13]: planarity of the graph, bounded clique-number, and bounded vertex-degree. We show that the problem remains NP-hard even when the intersection graph of the boxes has clique-number at most 4, and the maximum vertex-degree is at most 8 (Theorem 5.1). For the Box Cover problem we show that it remains NP-hard even under the severely restricted setting where the intersection graph of the boxes is planar, its clique-number is at most 2 (i.e., the graph is triangle-free), the maximum vertex-degree is at most 3, and every point is contained in at most two boxes (Theorem 5.2).

In Section 5.3, we complement these hardness results with two approximation algorithms for the Minimum Coverage Kernel problem, both running in polynomial time. We describe a $\mathcal{O}(\log n)$-approximation algorithm which runs in time within $\mathcal{O}(\mathtt{OPT} \cdot n^{\frac{d+1}{2}} \log^2 n)$ (Theorem 5.7); and a randomized algorithm computing a $\mathcal{O}(\log \mathtt{OPT})$-approximation in expected time within $\mathcal{O}(\mathtt{OPT} \cdot n^{\frac{d+1}{2}} \log^2 n)$, with high probability (at least $1 - \frac{1}{n^{\Omega(1)}}$) (Theorem 5.8). Our main contribution in this matter is not the existence of polynomial-time approximation algorithms (which can be inferred from results on the Box Cover problem), but rather a new data structure (Theorem 5.6) which significantly improves the running time of computing those approximations. This is relevant in applications where a minimum coverage kernel needs to be computed repeatedly [10, 56, 95, 118].

**These results were presented at the 24th International Computing and Combinatorics Conference (COCOON'18) [23] under the title *"Computing Coverage Kernels Under Restricted Settings"*.**

# Chapter 2

# Foundations

The problems considered in this thesis are related to three fundamental and extensively studied combinatorial optimization problems: Set Cover, Maximum Clique, and Maximum Independent Set. In this chapter, we trace the path from these problems to the special cases studied in this research, give a brief overview of the previous works on them, and introduce several concepts essential to this thesis.

## 2.1 Three Fundamental Problems

The decision versions of the Set Cover, Maximum Clique, and Maximum Independent Set problems were part of Karp's famous list of 21 problems. Karp [82] described polynomial time reductions from the Boolean Satisfiability problem (SAT) to each of these 21 problems, thus showing that they are all NP-hard. This demonstrated that many natural computational problems occurring throughout computer science might be computationally intractable, and it drove interest in the study of NP-completeness and the P=NP question.

The Set Cover, Maximum Clique, and Maximum Independent Set problems are closely related, and different known results on them are globally relevant to the more specialized results described in chapters 3 to 5. The optimization versions of these problems are formally defined as follows:

Pb Maximum Independent Set: Given an undirected graph $G$, find a subset of pairwise non-adjacent vertices (namely, an **independent set**) of **maximum** cardinality.

Pb Maximum Clique: Given an undirected graph $G$, find a subset of mutually adjacent vertices (namely, a **clique**) of **maximum** cardinality.

Pb Set Cover: Given a set system $\Sigma = (\mathcal{T}, \mathcal{R})$, where $\mathcal{T}$ is a set of $n$ elements of some universe, and $\mathcal{R}$ is a subset of $m$ elements of the power set $2^{\mathcal{T}}$, find a subset $C \subseteq \mathcal{R}$ of **minimum** cardinality such that $C$ **covers** $\mathcal{T}$ (namely, a **set cover**).

An example of the tight relation between these problems is the fact that in a graph $G$ of $n$ vertices there is an independent set of size $k$ if and only if the complement of $G$ has a clique of size $(n-k)$[1]. Similarly, if given an $n$-vertex graph $G=(V,E)$, we let $\mathcal{T}$, and $S$ be the sets $\mathcal{T}=E$ and $S=\{E(u) \mid u \in V\}$, respectively, where $E(u)$ denotes the edges of $E$ incident on $u$, then there is a cover of $\mathcal{T}$ of size $k$ if and only if $G$ has an independent set of size $(n-k)$.

While these problems in their general settings might seem equally hard, specially Maximum Independent Set and Maximum Clique which are *complementary*, the results known on their hardness, to this day, indicate that they are considerably distinct when restricted to special cases, or when trying to approximate their solutions through polynomial-time algorithms. For instance, consider the special cases of the Maximum Independent Set and Maximum Clique problems in which the input graph $G$ is the intersection graph of a given set of $d$-boxes:

**Definition 2.1** (Intersection Graph of a Set of Boxes)**.** *Let $\mathcal{B}$ be a set of $n$ $d$-boxes. The* intersection graph *of $\mathcal{B}$ is the $n$-vertex undirected graph $G=(V,E)$ in which there is a vertex in $V$ for each box of $\mathcal{B}$, and there is an edge $(u,v) \in E, u \neq v$, if and only if the boxes corresponding to $u$ and $v$ in $\mathcal{B}$ intersect.*

While the restriction of the Maximum Independent Set problem to $d$-boxes (i.e., the Maximum Independent Set of Rectangles problem) remains NP-hard even in $d=2$ dimensions [67], the analogous special case of the Maximum Clique problem (i.e., the Maximum Depth problem) can be optimally solved in polynomial time in any constant dimension $d$ [45]. The Box Cover problem, an analogous special case of Set Cover, is also known to remain NP-hard in any constant dimension $d>1$ [67]. Since these special cases are central to this thesis, we review them in their own sections (see Sections 2.2, 2.3, and 2.4, respectively). Something similar happens when trying to approximate their optimal solutions by means of polynomial-time algorithms:

**Definition 2.2** (Approximation Algorithm [53])**.** *An algorithm $\mathcal{A}$ is an $f(n)$-*approximation algorithm *for a minimization (resp. maximization) problem $P$ if given any instance of $P$ with size $n$, which has an optimal solution of size $\mathtt{OPT}$, $\mathcal{A}$ outputs a solution of size at most $f(n) \cdot \mathtt{OPT}$ (resp. at least $\frac{1}{f(n)} \cdot \mathtt{OPT}$). We say that an $f(n)$-approximation algorithm is an algorithm with* approximation factor $f(n)$.

For the Maximum Independent Set and Maximum Clique problems the current best polynomial-time approximation algorithm achieves an $\mathcal{O}(n/(\log n)^2)$-factor [34]. Besides, it is known that for these problems there are no polynomial-time $(n^{1-\varepsilon})$-approximation algorithms unless P = NP [130]. Note, however, that there is a considerable gap between the best factor of approximation known to this day, and this inapproximability bound. For the Set Cover problem, much stronger results are known: a simple greedy algorithm due to Johnson [78] and Lovász [101] yields a $(\ln n)$-approximation, and this approximation cannot be improved by more than constant factors unless P = NP [65, 103].

[1]The complement of a graph $G=(V,E)$ is the graph $\overline{G}=(V,\overline{E})$ where there is an edge $(u,v) \in \overline{E}$ if and only if $(u,v) \notin E$.

For many of the practical applications of these problems, it is enough to solve them on restricted classes of instances, which often turn out to be more tractable, as the special case of Maximum Clique commented before. In the upcoming sections we overview three such restricted classes of the Set Cover, Maximum Clique, and Maximum Independent Set problems, where the instances can be represented through $d$-boxes, for constant $d$.

## 2.2 Matchings and Maximum Independent Sets

The Maximum Independent Set of Rectangles (MISR) problem is a fundamental problem in computational geometry and combinatorics, which has been extensively studied [3, 7, 41, 42, 67, 77, 120]. The input of MISR is a set $\mathcal{B}$ of $n$ axis-parallel rectangles, and the goal is to find a subset of pairwise non-intersecting rectangles of maximum cardinality.

**Remark.** *Through this thesis we will refer to axis-aligned rectangles (i.e., 2-boxes) simply as "rectangles", which sounds more natural. There is no ambiguity in doing so given that we only consider axis-aligned rectangles and hyper-rectangles in our work.*

Clearly, MISR problem is equivalent to the special case of the Maximum Independent Set problem where the input is restricted to the intersection graph (see Definition 2.1) of a given set of $n$ rectangles. Fowler et al. [67], and Imai and Asano [77], showed that the MISR problem is NP-hard. Consequently, efforts have been put in designing approximation algorithms for this problem, and polynomial-time exact algorithms for special cases. Some of those special cases are known to remain hard [54, 67, 92], while others become solvable in polynomial time [71, 102, 124]. Among these special cases are the problems of finding monochromatic and bichromatic matchings of a set of colored points with rectangles, as we detail next.

### 2.2.1 Finding Matchings via Independent Sets

In this section, we formalize the relation between the MISR problem and the problem of finding matchings of points with rectangles. First, let us introduce some necessary definitions.

**Definition 2.3** (Matching of a set of points with rectangles)**.** *A* matching *of a set $\mathcal{S}$ of points in $\mathbb{R}^2$ with rectangles is a set $M$ of rectangles such that each element of $M$ contains exactly two points of $\mathcal{S}$, and every point of $\mathcal{S}$ lies in at most one rectangle of $M$. Such a matching is called* strong *if the rectangles in $M$ are pairwise disjoint, and* perfect *if every point of $\mathcal{S}$ belongs to some rectangle of $M$.*

**Definition 2.4** (Chromatic matchings)**.** *Let $\mathcal{S} = R \cup B$ be a set of points colored either red or blue, where $R$ and $B$ are the sets of red and blue points, respectively. A matching $M$ of $\mathcal{S}$ is called* ***monochromatic*** *if every rectangle in $M$ has its two points equally colored, and* ***bichromatic*** *if every rectangle in $M$ has its two points of different color.*

Pb Maximum Monochromatic Rectangle Matching (MonoMRM) problem: Given a set $\mathcal{S} = R \cup B$ of red and blue points in $\mathbb{R}^2$, find a monochromatic strong matching of $\mathcal{S}$ with the maximum number of axis-aligned rectangles.

Pb Maximum Bichromatic Rectangle Matching (BicMRM) problem: Given a set $\mathcal{S} = R \cup B$ of red and blue points in $\mathbb{R}^2$, find a bichromatic strong matching of $\mathcal{S}$ with the maximum number of axis-aligned rectangles.

Let $\mathcal{S}$ be a set of points in $\mathbb{R}^2$. For every point $p$ of $S$, let $x(p)$, $y(p)$, and $\mathsf{color}(p)$ denote the $x$-coordinate, the $y$-coordinate, and the color of $p$, respectively. Given two points $a$ and $b$ of the plane with $x(a) < x(b)$, or $x(a) = x(b)$ and $y(a) < y(b)$, let $B(a, b)$ denote the smallest box containing $a$ and $b$. Consider the following two sets of axis-aligned rectangles:

$$\begin{aligned} \mathcal{R}(\mathcal{S}) &= \big\{B(p,q) \mid p,q \in S; \mathsf{color}(p) = \mathsf{color}(q);\ \text{and } B(p,q) \cap S = \{p,q\}\big\} \\ \overline{\mathcal{R}}(\mathcal{S}) &= \big\{B(p,q) \mid p,q \in S; \mathsf{color}(p) \neq \mathsf{color}(q);\ \text{and } B(p,q) \cap S = \{p,q\}\big\} \end{aligned}$$

Observe that the MonoMRM problem is equivalent to finding a maximum subset of $\mathcal{R}(\mathcal{S})$ of independent rectangles. Similarly, the BicMRM problem is equivalent to finding a maximum subset of $\overline{\mathcal{R}}(\mathcal{S})$ of independent rectangles. Thus, both the MonoMRM and BicMRM problems are special cases of the MISR problem. Naturally, the main question arising from this is whether the instances of the MonoMRM and BicMRM problems are simpler than those of MISR: are these problems solvable in polynomial time, or are their solutions simpler to approximate? Up to this work, not much was known in this direction: some special cases of the MonoMRM and BicMRM were known to be solvable in polynomial time, while others were known to be easier to approximate. For the general case, the only approximation algorithms known were those that these problems trivially "inherit" from the MISR problem. We review some of these approximation algorithms in Section 2.2.2, and in Section 2.2.3 we review the special cases of MonoMRM and BicMRM problems considered in the literature previous to our work.

### 2.2.2 Approximation Algorithms for Maximum Independent Set

Since the MSIR problem is NP-hard [67, 77], the design of approximation algorithms for this problem has received a lot of attention [8, 41, 42, 84, 113]. In this section, we review some of these results.

Different groups of authors have independently described $\mathcal{O}(\log n)$-approximation algorithms for the general MISR problem [8, 84, 113]. Moreover, it is known that this problem admits a polynomial-time approximation algorithm, which with high probability produces an independent set of rectangles with cardinality within $\Omega(\frac{1}{\log \log n}) \cdot \mathtt{OPT}$, where $\mathtt{OPT}$ is the size of an optimal solution [41, 42].

For some problems, it is possible to find approximate solutions that are arbitrarily close to the optimal solution:

**Definition 2.5** (PTAS)**.** *We say that an algorithm $\mathcal{A}$ is a* polynomial-time approximation scheme *(PTAS) for a minimization (resp. maximization) problem $P$ if for every fixed $\varepsilon > 0$ and for any instance of size $n$ , with an optimal solution of size $\mathit{OPT}$, the running time of $\mathcal{A}$ is polynomial in $n$, and it returns a solution of size at most $(1+\varepsilon) \cdot \mathit{OPT}$ (resp. at least $(1-\varepsilon) \cdot \mathit{OPT}$).*

Finding a PTAS, or even a constant-factor approximation algorithm for the MISR problem is still an intriguing open question. There exist, though, polynomial-time exact algorithms, constant-factor approximation algorithms, and PTAS's for various special cases of the MISR problem. They take advantage of properties which arise in the intersection graph of the rectangles, according to the types of intersections that occur in the set.

**Definition 2.6** (Piercing intersection of two rectangles)**.** *Given two rectangles $R_1$ and $R_2$, we say that $R_1$* pierces *$R_2$ if into the $x$-axis the orthogonal projection of $R_1$ contains the orthogonal projection of $R_2$, and into the $y$-axis the orthogonal projection of $R_2$ contains the orthogonal projection of $R_1$. We say that two intersecting rectangles* **pierce** *if one of them pierces the other one (see Figure 2.1a).*

For any set $\mathcal{B}$ of axis-aligned rectangles, let $G(\mathcal{B})$ denote the intersection graph of $\mathcal{B}$. Independently, Agarwal and Mustafa [7] and Lewin-Eytan et al. [99] showed that if every pair of intersecting rectangles pierce, then the MISR problem can be solved in polynomial time: in this case, the intersection graph of the rectangles is perfect[2], and using a classical result of Grötschel et al. [72], a maximum independent set of a perfect graph can be computed in polynomial time. Agarwal and Mustafa [7] generalized this fact showing that for any set $\mathcal{B}$, the spanning subgraph[3] $G' = (V, E')$ of the intersection graph $G(\mathcal{B}) = (V, E)$ with $E'$ being the edges corresponding to the piercing intersections, is also perfect.

**Remark.** *We use some of these results on piercing rectangles as part of two approximation algorithms for the MonoMRM and BicMRM problems in Section 4.3.*

In other direction, Chan [43] described a PTAS for the MISR problem when the rectangles have unit height. Also, a PTAS for rectangles of bounded aspect ratio was introduced by Chan [46] and by Erlebach et al. [63]. Finally, it is known that if the size of a maximum clique in the intersection graph is $q$, then there exists a $(4q)$-approximation algorithm [7, 99].

**Remark.** *Approximation algorithms for the* MISR *problem, sensitive to the size of a maximum clique, do not necessarily yield good approximations for the MonoMRM and BicMRM problems: for both one can build examples in which the size of the optimal solution is either big or small independently of the clique number of the intersection graphs of $\mathcal{R}(\mathcal{S})$ and $\overline{\mathcal{R}}(\mathcal{S})$.*

[2] A graph is *perfect* if the chromatic number of every induced subgraph equals the size of the largest clique of that subgraph.

[3] A graph $G' = (V', E')$ is a *spanning* subgraph of $G = (V, E)$ if $V' = V$ and $E' \subseteq E$.

### 2.2.3 Variants and Special Cases of Matchings with Rectangles

Several variants and special cases of the MonoMRM and BicMRM problems have being studied before, including when the rectangles are restricted to straight segments, or squares, or all the points have the same color. We briefly explore them below.

Dumitrescu and Steiger [61] considered the computation of strong monochromatic matchings of two-colored point sets in the plane with axis-aligned segments. The best results known for this problem are due to Dumitrescu and Kaye [60]: every two-colored point set $S = R \cup B$ of $n$ points admits a strong straight segment matching that matches $\frac{6}{7}n - O(1)$ points, which can be found in time within $O(n^2)$; furthermore, there exist sets of $n$ points such that every strong matching with straight segments matches at most $\frac{94}{95}n + O(1)$ points. The computational complexity of deciding whether a given two-colored point set admits a perfect monochromatic strong matching with straight segments is still an open problem [61].

Ahn et al. [12] studied the $(p, k)$-Rectangle Covering problem: given $n$ points in the plane, find $p$ pairwise-disjoint rectangles that, together, cover at least $n - k$ of the points, while minimizing the area of the largest rectangle. Among other results, they showed that given $n$ points and an input $p$, deciding whether the points can be covered with $p$ pairwise-disjoint rectangles, each of area at most one, is NP-complete. The arguments used in their proof rely on the fact that rectangles can cover either two or three points, those being not straightforward to generalize to prove that the MonoMRM problem is NP-hard.

**Remark.** *We show in Section 4.2 that the MonoMRM problem is NP-hard when the rectangles are restricted to axis-aligned segments, proving that their decision problem is NP-complete.*

Bereg et al. [32] studied the special case of the MonoMRM problem, where all the points have the same color. They showed that a set of $n$ points admits a monochromatic matching that matches at least $2\lfloor n/3 \rfloor$ of the points, and described an algorithm to find it in time within $\mathcal{O}(n \log n)$. Such a matching can be far from optimal though: there are instances where all the points can be matched while their algorithm matches only $2\lfloor n/3 \rfloor$ of the points. They were able to prove that if the point set is not in a general position, and the matching rectangles are restricted to squares, then it is NP-hard to find a matching of maximum size, but they left open whether maximum size matchings with rectangles could be computed in polynomial time in this setting.

**Remark.** *We show in Section 4.2 that the* MonoMRM *problem is NP-hard even when all the points have the same colors. Thus, the special case studied by Bereg et al. [32] cannot be solved in polynomial time, unless P = NP.*

Other special cases in the literature consider restrictions on the types of intersections between rectangles that can occur in the instance of the MonoMRM and BicMRM problems. Let $S$ be a set of points in $\mathbb{R}^2$, and consider the sets $\mathcal{R}(S)$ and $\overline{\mathcal{R}}(S)$ as previously defined in Section 2.2.2:

$$\begin{aligned} \mathcal{R}(\mathcal{S}) &= \big\{B(p,q) \mid p,q \in S; \mathsf{color}(p) = \mathsf{color}(q);\ \text{and } B(p,q) \cap S = \{p,q\}\big\} \\ \overline{\mathcal{R}}(\mathcal{S}) &= \big\{B(p,q) \mid p,q \in S; \mathsf{color}(p) \neq \mathsf{color}(q);\ \text{and } B(p,q) \cap S = \{p,q\}\big\} \end{aligned}$$

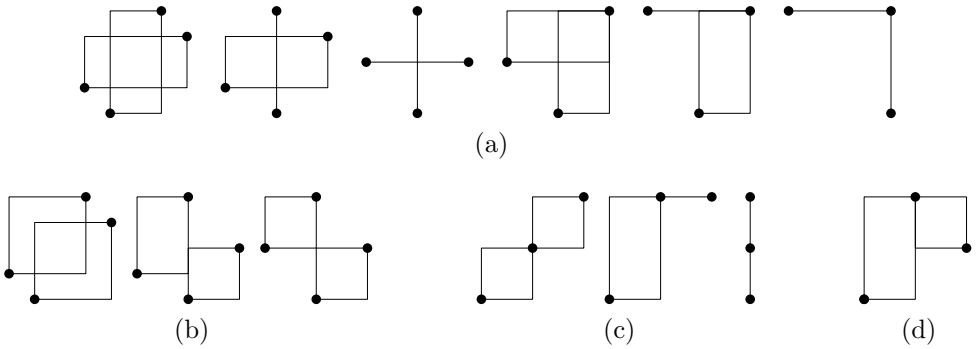



Figure 2.1: Up to symmetry, the four types of intersection: (a) piercing; (b) corner; (c) point; and (d) side.

Since every rectangle in these sets is defined by two points of $S$ being opposed vertices, is closed, and does not contain any other point of $S$; only four types of intersection can occur:

**Definition 2.7** (Types of intersections between rectangles)**.** *In the sets $\mathcal{R}(S)$ and $\overline{\mathcal{R}}(S)$ two intersecting rectangles realize only one of the following types of intersection:*

1. *a* piercing intersection *in which the two rectangles pierce (see Definition 2.6, and Figure 2.1a);*
2. *a* corner intersection *in which each rectangle contains exactly one of the corners of the other one, and these corners are not elements of $S$ (see Figure 2.1b);*
3. *a* point intersection *where the intersection of the rectangles is precisely an element of $S$ and it is not a piercing intersection (see Figure 2.1c); and*
4. *a* side intersection *which is the complement of the above three intersection types (see Figure 2.1d).*

Agarwal and Mustafa [7] and Lewin-Eytan et al. [99] showed that if every pair of intersecting rectangles in $\mathcal{R}(S)$ (resp. $\overline{\mathcal{R}}(S)$) pierce, then the MonoMRM (resp. BicMRM) problem can be solved in polynomial time.

Soto and Telha [124] studied the following problem to model cross-free matchings in two-directional orthogonal ray graphs (2-dorgs): Given finite point sets $X$ and $Y$ in the plane, find a maximum set of independent rectangles from the set $\mathcal{R}(X, Y)$ of the rectangles having an element of $X$ as bottom-left corner and an element of $Y$ as top-right corner. For $X = R$ and $Y = B$, where $S = R \cup B$, this problem is equivalent to the special case of the BicMRM problem where the rectangles are restricted to have a red point as bottom-left corner and a blue point as top-right corner. They showed that this special case can be solved optimally in polynomial time. For this, they use the next observations: (*i.*) in $\mathcal{R}(X, Y)$ only two types of intersections can occur: piercing and corner; and (*ii.*) there is a subset $\mathcal{R}_0 \subseteq \mathcal{R}(X, Y)$ with the same maximum independent set as $\mathcal{R}(X, Y)$, and where only piercing intersections occur.

Since in $\mathcal{R}_0$ only piercing intersections occur, its intersection graph is perfect, and thus a maximum independent set of $\mathcal{R}_0$ can be computed in polynomial time.

**Remark.** *In Section 4.3 we generalize the results of Agarwal and Mustafa [7], Lewin-Eytan et al. [99] and Soto and Telha [124]: we show that for a larger class of rectangle intersection graphs, where both piercing and corner intersections can occur, maximum independent sets can be computed in polynomial time. This new result is key for the approximation algorithms that we describe for the MonoMRM and BicMRM problems.*

While many of the special cases of the MISR problem mentioned in this section remain NP-hard, as the more general Maximum Independent Set problem, the special case of Maximum Clique analogous to MISR can be solved in polynomial time. We provide a detailed overview of these results in the next section.

## 2.3 Maximum Clique and the Klee's Measure

As mentioned before, the special case of Maximum Clique in which the graph is restricted to the intersection graph of a given set of boxes is precisely the Maximum Depth problem. Unlike the Maximum Independent Set of Rectangles, the Maximum Depth problem can be optimally solved in polynomial time [45].

The Maximum Depth problem is closely related to the Klee's Measure problem. Although no direct reduction between these problems is known, the techniques used so far to solve the Maximum Depth problem can all be used to solve the Klee's Measure problem [45], and the same is true for the lower bound arguments known so far. Many of the results known for these problems were originally introduced for the Klee's Measure problem, and then extended to the Maximum Depth problem. We retrace some of these results, but first we reformulate below the definition of these problems so that to include a domain box $\Gamma$ as an extra input parameter. These new formulations will be the ones consider throughout the remain of this thesis:

> Pb Klee's Measure problem: Given a set $\mathcal{B}$ of $n$ $d$-boxes, and a $d$-dimensional domain box $\Gamma$, compute the hypervolume of the union of the boxes in $\mathcal{B}$ within $\Gamma$.

> Pb Maximum Depth problem: Given a set $\mathcal{B}$ of $n$ $d$-boxes, and a $d$-dimensional domain box $\Gamma$, compute the maximum number of boxes intersecting at a same point within $\Gamma$ (equivalently, the size of a maximum clique in the intersection graph of $\mathcal{B}$).

The techniques used to compute the Klee's measure have evolved over time. After Klee's original solution [87] for the problem with intervals, Bentley [29] described a solution for the two dimensional case running in time within $\mathcal{O}(n \log n)$. Bentley's algorithm sweeps a vertical line from left to right across the rectangles, maintaining the intersection between the rectangles and the sweep line (a collection of intervals) in a *Segment Tree* (a data structure specially introduced by Bentley [29] for this problem, and that later found many other

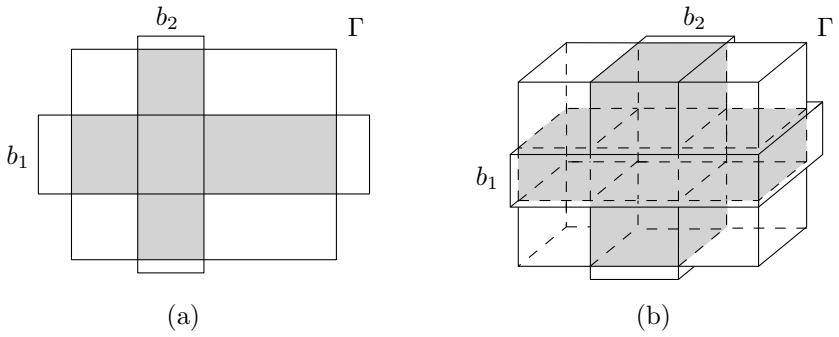



Figure 2.2: An illustration in dimensions 2 (a) and 3 (b) of two boxes $b_1, b_2$ equivalent to slabs when restricted to the box $\Gamma$. The Klee's measure of $\{b_1, b_2\}$ within $\Gamma$ is the area (resp. volume) of the shadowed region in (a) (resp. (b)).

applications). Whenever the sweep line hits a vertical edge of a rectangle, the segment tree is updated (in time within $\mathcal{O}(\log n)$), along with the measure of the swept area.

The trivial extension of Bentley's algorithm [29] to higher dimension yields an algorithm running in time within $\mathcal{O}(n^{d-1} \log n)$. After some minor progresses for higher dimensions [68], Overmars and Yap [114] described a solution running in time within $\mathcal{O}(n^{d/2} \log n)$. To achieve this, Overmars and Yap [114] made a key observation for the special case of the problem where all the boxes are slabs:

**Definition 2.8** (Slab)**.** *Let $b$ and $\Gamma$ be two $d$-boxes. The box $b$ is said to be a* slab *within $\Gamma$ if the intersection $b \cap \Gamma$ is of the form $\{(x_1, \ldots, x_d) \in \Gamma \mid \alpha \leq x_i \leq \beta\}$, for some dimension $i \in [1..d]$ (to which the slab is orthogonal) and some real values $\alpha, \beta$ (see Figure 2.2 for an illustration). Alternatively, we can say that a box $b$ is a slab within $\Gamma$ if no $(d-2)$-face of $b$ intersects $\Gamma$.*

Overmars and Yap [114] showed that, if all the boxes in $\mathcal{B}$ are slabs inside the domain box $\Gamma$, then the Klee's measure of $\mathcal{B}$ within $\Gamma$ can be computed in linear time (after the boxes have been pre-sorted in each dimension). Overmars and Yap's solution [114] remained the best solution for the Maximum Depth and Klee's Measure problems in $d$ dimensions for more than 20 years, until 2013 when Chan [45] presented a simpler and faster algorithm running in time within $\mathcal{O}(n^{d/2})$. Both approaches are central to some of the new results introduced in this thesis. Therefore, we describe them in sections 2.3.1 and 2.3.2, respectively.

### 2.3.1 A dynamic data structure for the Klee's Measure

Overmars and Yap's algorithm [114] is based on Bentley's technique [29]: solve the static problem in $d$ dimensions by combining a data structure for the dynamic version of the problem in $d-1$ dimensions with a plane sweep over the $d$-th dimension.

The algorithm starts by partitioning the domain $\Gamma$ into $\mathcal{O}(n^{d/2})$ rectangular cells in which the boxes in $\mathcal{B}$ are equivalent to slabs. This partition is done as follows: $(i.)$ split $\Gamma$ into $2\sqrt{n}$

sub-domains (or cells) by using hyperplanes orthogonal the $x_1$ axis such that the interior of each sub-domain contains at most $\sqrt{n}$ 1-faces[4]; (*ii.*) for each such sub-domain $s$ let $\mathcal{B}_s$ be the set of boxes that partially cover $s$, and split $\mathcal{B}_s$ into two subsets: $\mathcal{B}_s^1$ with the (at most $\sqrt{n}$) boxes that have a 1-face inside $s$, and $\mathcal{B}_s^2$ with the boxes that do not have any 1-face inside $s$; (*iii.*) split each sub-domain $s$ with respect to the $x_2$ axis using (at most $\sqrt{n}$) hyperplanes passing through each 2-face of a box in $\mathcal{B}_s^1$, and by every $\sqrt{n}$-th 2-face of a box in $\mathcal{B}_s^2$; (*iv.*) as a result, each sub-domain $s$ is split into $\mathcal{O}(\sqrt{n})$ new sub-domains containing no 1-face, and a number of 2-faces within $\mathcal{O}(\sqrt{n})$; (*v.*) for each such a new subdomain $s'$, let $\mathcal{B}_{s'}$ be the set of boxes that partially cover $s'$, and split again $\mathcal{B}_{s'}$ into two subsets: $\mathcal{B}_{s'}^{1,2}$ with the (at most $\sqrt{n}$) boxes that have a 2-face intersecting the interior of $s'$, and $\mathcal{B}_{s'}^3$ with the boxes that have neither a 1 or 2-face intersecting the interior of $s'$; (*vi.*) split $s'$ into $\mathcal{O}(\sqrt{n})$ new cells by using hyperplanes orthogonal to the $x_3$ axis passing through each 3-face of a box in $\mathcal{B}_{s'}^{1,2}$ an by every $\sqrt{n}$-th 3-face of a box in $\mathcal{B}_s^3$; (*vii.*) proceed analogously for the remaining dimensions. Since at each dimension the number of cells increases by a $\mathcal{O}(\sqrt{n})$-factor, the final partition has $\mathcal{O}(\sqrt{n}^d)$ cells. Over this partition one can build a tree with the following properties:

**Lemma 2.1** (Lemma 4.2 of Overmars and Yap [114])**.** *Let $\mathcal{B}$ be a set of $n$ $d$-boxes. There exists a binary partition tree for storing any subset of $\mathcal{B}$ such that*

- *The tree can be computed in time within $\mathcal{O}(n^{d/2})$, and it has $\mathcal{O}(n^{\frac{d}{2}})$ nodes;*
- *Each box is stored in $\mathcal{O}(n^{\frac{d-1}{2}})$ leafs of the tree;*
- *The boxes stored in a leaf are slabs within the cell corresponding to the node;*
- *Each leaf stores no more than $\mathcal{O}(\sqrt{n})$ boxes.*

This partition tree can be used to dynamically maintain the Klee's measure or the maximum depth of a given set of boxes. The key is that, since in the leaves of this tree all the boxes are slabs when restricted to the respective cell, the Klee's measure or the maximum depth of the boxes partially intersecting each cell can be maintained with the help of $d$ segment trees [29] per cell. These segment trees are used to keep track of the solutions to the $d$ one dimensional instances of the problem which arise from projecting each slab to its orthogonal dimension.

This partition of Overmars and Yap [114] inspired solutions for other related problems. For instance, in 2011, Yildiz et al. [128] studied a stochastic version of the Klee's Measure problem where a probability of being in a random sample is associated to each box in the set. The objective in this case is to compute the expected Klee's measure of a random sample. Note that this problem is more general, since, associating a probability $p = 1$ with each box, results in the original problem. Based on the results by Overmars and Yap [114], they described a new data structure called *Anonymous Segment Tree* for maintaining the expected volume of a dynamic set of probabilistic boxes with $\mathcal{O}(n^{(d-1)/2} \log n)$ update time. Again, by

[4]A $j$-face is a $j$-dimensional face of an input box; e.g. a 0-face is a vertex (i.e., a point), a 1-face is an edge, and a $(d-1)$-face is a facet.

transforming the static $d$-dimensional problem in a dynamic $(d-1)$-dimensional problem, they obtained a $\mathcal{O}(n^{d/2} \log n)$-time algorithm to compute the expected Klee's Measure.

**Remark.** *The analogous stochastic version of the* Maximum Depth *problem seems harder than that of the* Klee's Measure*, and it is open whether this version can even be solved in polynomial time. The difference lies in the fact that, given two random variables $X$ and $Y$, computing the expectation of the random variable $X + Y$ is in general a lot easier than computing the expectation of the random variable* $\max\{X, Y\}$.

**Remark.** *We introduce in Section 5.3 two approximation algorithms for the* Minimum Coverage Kernel *problem which are based on Overmars and Yap's approach [114].*

There are two main issues with the solution of Overmars and Yap [114] for the Maximum Depth and Klee's Measure problems: the partition tree uses space within $\mathcal{O}(n^{d/2})$; and less importantly, it is difficult to implement. Chan [45] introduced a solution which, not only improves the running time by a logarithmic factor, but tackles these two issues: it is simpler, and uses linear space. We describe this solution in the next section.

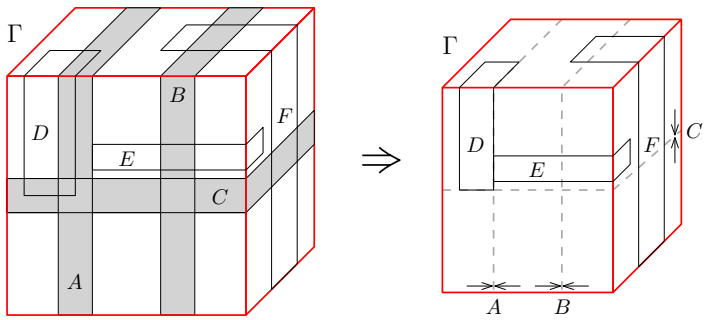
### 2.3.2 A divide and conquer algorithm for the Klee's Measure

In 2013, Chan [45] introduced a simple divide and conquer algorithm [45] to compute the Klee's measure of a set $\mathcal{B}$ of $n$ $d$-boxes within a domain $\Gamma$. This algorithm runs in time within $\mathcal{O}(n^{d/2})$, improving Overmars and Yap's approach [114] by a logarithmic factor. The simplicity of Chan's algorithm, and its running-time analysis, lies on three simple ideas: (*i.*) he changed the objective to computing the volume of the complement of $\mathcal{B}$ within $\Gamma$ (which is of course equivalent to the original problem); (*ii.*) he introduced weights for the $(d-2)$-faces of the boxes in $\mathcal{B}$, and used their weighted median to obtain the recursive problems; and (*iii.*) he described a simple way to handle the slabs and remove them from the input (a step he named *simplification*), ensuring that the number of boxes of each subproblem eventually decreases. The following is an outline of Chan's algorithm:

**Algorithm 2.1** `uncoveredVolume(`$\mathcal{B}, \Gamma$`)`

**Input:** A set $\mathcal{B}$ of $n$ boxes in $\mathbb{R}^d$, a $d$-dimensional domain box $\Gamma$
**Output:** The volume of the region in $\Gamma$ not covered by any box in $\mathcal{B}$

```
1: if |B| is constant then
2:     return the answer directly
3: else
4:     simplify the slabs of B in Γ
5:     partition Γ into two sub-cells Γ_L and Γ_R
6:     let B_L, B_R be the boxes of B intersecting Γ_L, Γ_R, respectively
7:     return uncoveredVolume(B_L, Γ_L) + uncoveredVolume(B_R, Γ_R)
```




(a) (b)

Figure 2.3: Simplification of slabs on a set of six boxes within a domain Γ. The boxes $A, B, C$ in (a) are slabs when restricted to Γ (they have no edges inside Γ). These boxes can be simplified by collapsing them as in (b).

**How to simplify:** Let $\mathcal{B}^*$ be the subset of boxes in $\mathcal{B}$ that are slabs when restricted to Γ. The region of Γ covered by $\mathcal{B}^*$ clearly does not belong to the complement of the union of the boxes in $\mathcal{B}$. This region can be "removed" from Γ by collapsing the slabs in $\mathcal{B}^*$ as illustrated in Figure 2.3 for three dimensions. This simplification can be done in $\mathcal{O}(n)$-time since it is equivalent to computing the union of $d$ sets of intervals, which can be done in linear time over a previously sorted input. Since a box is a slab in Γ if and only if it has no $(d-2)$-face intersecting Γ, after the simplification step all the boxes remaining in $\mathcal{B}$ must have at least one $(d-2)$-face intersecting Γ. Thus, the size of $\mathcal{B}$ after the simplification is at most the total number of $(d-2)$-faces intersecting Γ.

**How to Partition:** To obtain the recursive subproblems, a constant weight of $2^{\frac{i+j}{2}}$ is assigned to each ($d$-2)-face intersecting the domain, where $i, j \in [1..d]$ are the dimensions to which the face is orthogonal. Then, the domain is partitioned into two cells by the hyperplane $x_1 = m$, where $m$ is the weighted median of the ($d$-2)-faces orthogonal to the first dimension. After that, the axes are renumbered so that the axes $x_1, \ldots, x_d$ correspond to the new axes $x_d, x_1, \ldots, x_{d-1}$. The axis renumbering makes the weights of the ($d$-2)-faces orthogonal to dimensions $i, j \in [2..d]$ decrease by a factor of $2^{2/d}$. On the other side, the ($d$-2)-faces that are orthogonal to $x_1, x_j$ for some dimension $j \in [2..d]$ increase its weight by a factor of $2^{(d-2)/d}$. However, since Γ is partitioned by the weighted median of these faces, their total weight within each sub-domain decreases by a factor of $2^{(d-2)/d}/2 = 2^{2/d}$. This yields a decrease by a factor of $2^{2/d}$ in the total weight of the ($d$-2)-faces intersecting each sub-domain.

To analyze the running time and space used by the algorithm, note that each weight is a constant between 1 and 4, and that since the number of boxes intersecting Γ after the simplification is at most the number of $(d-2)$-faces intersecting Γ, the total weight within Γ is at most a constant factor far from the size $n$ of $\mathcal{B}$. Thus, the running time $T(n)$ of the algorithm follows the recurrence

$$T(n) \in 2T(n/2^{2/d}) + \mathcal{O}(n)$$

which solves to within $\mathcal{O}(n^{d/2})$ for any dimension $d \geq 3$. Similarly, the space used by the algorithm obeys the recurrence

$$S(n) \in S(n/2^{2/d}) + \mathcal{O}(n)$$

which implies that $S(n)$ is linear in $n$.

**Remark.** *We describe in Section 3.2 an algorithm for the* Depth Distribution *problem which takes features of both the approach of Overmars and Yap [114], and Chan's approach [45].*

Note that, as in Overmars and Yap's approach [114], Chan's algorithm [45] induces a partition of the domain $\Gamma$ into $\mathcal{O}(n^{d/2})$ cells, where all the boxes are slabs. In this case however, one cannot ensure that each cell partially intersects at most $\mathcal{O}(\sqrt{n})$ boxes, as for the partition described by Overmars and Yap [114].

Unfortunately, there are sets of $n$ $d$-boxes which require partitions of the space into a number of cells within $\Omega(n^{d/2})$ to ensure that every box in the set is equivalent to a slab when restricted to each cell. Hence, without a radically new technique, any algorithm based on this approach for the Maximum Depth or the Klee's Measure problems will run in time within $\Omega(n^{d/2})$. We discuss lower bounds for these problems in the next section.

### 2.3.3 Special cases and lower bounds

The only lower bound known for the computational complexity of the Klee's Measure problem, other than the trivial $\Omega(n)$, was proved by Fredman and Weide [68] in the linear decision tree model. They showed a bound within $\Omega(n \log n)$ by reducing the $\varepsilon$-Closeness problem to the Klee's Measure problem on intervals.

> Pb $\varepsilon$-Closeness problem: Given a set $\{x_1, x_2, \ldots, x_n\}$ of $n$ real numbers and a real value $\varepsilon \geq 0$, decide whether there are two elements $x_i, x_j$, $i \neq j$, such that $|x_i - x_j| \leq \varepsilon$.

The lower bound derives from the fact that in a set $\{x_1, x_2, \ldots, x_n\}$ of real numbers there are two elements at distance less than $\varepsilon$ if and only if the Klee's measure of the set of intervals $\big\{[x_1, x_1 + \varepsilon], [x_2, x_2 + \varepsilon], \ldots, [x_n, x_n + \varepsilon]\big\}$ is less than $\varepsilon n$. Note that this reduction can be simply transformed to obtain the same lower bound for the Maximum Depth problem: there are two elements in the set of real numbers at distance less than $\varepsilon$ if and only the maximum depth of the generated set of intervals is at least two.

These reductions were introduced in the linear decision tree model, which is very restrictive. Hence, lower bounds for the $\varepsilon$-Closeness proved later for less restrictive models (such as Ben-Or's results for algebraic computation trees [28]) can be transferred directly into lower bounds for the Klee's Measure and the Maximum Depth problems in such models. Finally, it worth mentioning that the lower bounds do not hold in more *powerful* models of computation such as the word-RAM [68].

In higher dimensions, Chan [45] conjectured that any *combinatorial* algorithm computing the Klee's measure requires within $\Omega(n^{d/2})$ operations, via a reduction from the parameterized k-Clique problem, in the worst case over instances of fixed size $n$. He described an analogous

argument for the MAXIMUM DEPTH problem. Furthermore, all the techniques used so far for this problem are based on binary partitions of the space. As mentioned, there are sets of $n$ boxes in $\mathbb{R}^d$ which require partitions of the space into a number of cells within $\Omega(n^{d/2})$ to ensure that every box in the set is equivalent to a slab when restricted to each cell. Hence, without a radically new technique, any algorithm based on this approach will require running time within $\Omega(n^{d/2})$. As a consequence, recent works have focused on the study of special cases which can be solved faster than $\Omega(n^{d/2})$. We enumerate below some of them:

- CUBE-KMP: The boxes are hypercubes. Bringmann [37] showed how to solve them in time within $\mathcal{O}(n^{(d+2)/3})$ for any constant $d \geq 2$. He also described simple reductions showing that the case of cubes is roughly the same as the case of "$\alpha$-fat boxes", where all side lengths of a box differ by at most a constant factor $\alpha$. Chan [45] described an algorithm running in time within $\mathcal{O}(n^{(d+1)/3} \operatorname{polylog} n)$ any $d \geq 3$, thus strictly subsuming Bringmann's result [37]. In three dimensions this bound is improved by the $\mathcal{O}(n \log^4 n)$ bound obtained by Agarwal [9].

- ORTHANTS: The boxes are orthants of the form $\{(x_1, \ldots, x_d) \in \mathbb{R}^d \mid (x_1 \leq \alpha_1) \wedge (x_2 \leq \alpha_2) \wedge \ldots \wedge (x_d \leq \alpha_d)\}$, where each $\alpha_i$ is a real number. Beume et al. [35] gave an $\mathcal{O}(n \log n)$ time algorithm for $d = 3$. The same authors showed an unconditional lower bound of $\Omega(n \log n)$ for $d > 1$. Chan [45] presented an $\mathcal{O}(n^{d/3} \operatorname{polylog} n)$ time algorithm based on the same ideas that he used for the general case, but with a more powerful simplification step.

- UNITCUBE-KMP: The boxes are hypercubes of the same side length. As this is a particular case of CUBE-KMP, all algorithms from above apply. The combinatorial complexity of the union of $n$ unit cubes is within $\mathcal{O}(n^{\lfloor d/2 \rfloor})$ [36]. Kaplan et al. [80] used this result to obtain an algorithm with running time within $\mathcal{O}(n^{\lfloor d/2 \rfloor} \operatorname{polylog} n)$, which is only better than the $\mathcal{O}(n^{(d+1)/3} \operatorname{polylog} n)$ bound of CUBE-KMP when $d = 3$. Chan [45] showed that this special case reduces to the ORTHANTS case, thus obtaining a solution running in time within $\mathcal{O}(n^{d/3} \operatorname{polylog} n)$. As for CUBE-KMP, Chan [45] and Bringmann [37] gave generalizations to $\alpha$-fat boxes with the same computational complexity.

- $k$-GROUNDED: The projection of the input boxes to the first $k$ dimensions is an ORTHANTS instance, where $0 \leq k \leq d$, the other coordinates are arbitrary. Yildiz and Suri [126] gave an algorithm to solve 2-GROUNDED running in time within $\mathcal{O}(n^{(d-1)/2} \log^2 n)$, for any $d \geq 3$.

**Remark.** *We study in Section 3.2.2 special cases of the* MAXIMUM DEPTH *and* KLEE'S MEASURE *problems from a different perspective: we show that there are measures which gradually separate* easy *instances of these problems from the* hard *ones, and that there are algorithms that can benefit from this.*

In the next section, we explore the relation between the SET COVER problem, the last of the three fundamental problems mentioned in Section 2.1, and the different problems on covering regions with boxes that we consider in this thesis.

## 2.4 From Covering Sets to Orthogonal Polygons

The Set Cover problem is hard to solve in polynomial time not only optimally, but even approximately up to $o(\log n)$-factor. Since $\mathcal{O}(\log n)$-approximation algorithms for this problem exist [78, 101], the hardness of the Set Cover problem is essentially resolved if $\mathsf{P} \neq \mathsf{NP}$. However, there are many special cases of interest for which the hardness of optimal solutions or approximations do not apply, especially in restricted settings that arise in geometric formulations of the problem. Several of those special cases have been studied in the context of the Geometric Set Cover problem:

> Pb Geometric Set Cover: Given a set system $\Sigma = (\mathcal{P}, \mathcal{R})$, where $\mathcal{P}$ is a set of $n$ $d$-dimensional points, and $\mathcal{R}$ a set of $m$ $d$-dimensional geometric objects (called ranges) belonging to the same family, compute a subset $\mathcal{C} \subseteq \mathcal{R}$ of minimum size covering $\mathcal{P}$.

Clearly, the Box Cover problem is the special case of Geometric Set Cover where the elements of $\mathcal{R}$ are $d$-boxes. Other families of geometric objects considered include lines [74, 94], segments [83], disks [76, 47, 50], squares [39, 48, 111]. Many of the formulations of Geometric Set Cover are known to remain $\mathsf{NP}$-hard even under very restricted settings, as when the geometric objects are forced to intersect in at most one element [94]; or when the geometric objects are segments and their orientations are fixed [66], or their intersection graph is triangle-free [79] or planar [83]; or when the geometric objects are unit-size squares and the point set $\mathcal{P}$ corresponds to the center of the squares [67].

These results help clarify the boundaries between instances which can be solved in polynomial time, and those which potentially cannot, and help to understand which features make these problems hard. An example where those boundaries are far from being clear is in the approximabilities of the Orthogonal Polygon Covering and Box Cover problems. Given that the Minimum Coverage Kernel problem is intermediate between these two other problems, the studyof its computational hardness has the potential to provide insights in this direction. To better understand the relation between the Orthogonal Polygon Covering, Box Cover and Minimum Coverage Kernel problems, we review in the next section the reductions between them.

### 2.4.1 Coverage Kernels: between Box and Polygon Coverings

We describe the reductions between the Orthogonal Polygon Covering, Box Cover and Minimum Coverage Kernel problems in the plane, since their generalizations to higher dimensions are straightforward. We start by showing that the Orthogonal Polygon Covering problem is a special case of the Minimum Coverage Kernel problem.

Let $\mathcal{P}$ be an orthogonal polygon with $n$ horizontal/vertical edges. Consider the grid formed by drawing infinitely long lines through each edge of $\mathcal{P}$ (see Figure 2.4.a for an illustration), and let $G$ be the set of $\mathcal{O}(n^2)$ points of this grid lying on the intersection of two lines. Create a set $\mathcal{B}$ of boxes as follows: for each pair of points in $G$, if the box having those two points as opposed vertices is contained in $\mathcal{P}$, then add it to $\mathcal{B}$ (see Figure 2.4.b.) Let $\mathcal{C}$

be any set of boxes covering $\mathcal{P}$. Note that for any box $c \in \mathcal{C}$, either the vertices of $c$ are in $G$, or $c$ can be extended horizontally and/or vertically (keeping $c$ contained in $\mathcal{P}$) until this property is met. Hence, there is at least one box in $\mathcal{B}$ that covers each $c \in \mathcal{C}$, respectively, and thus there is a subset $\mathcal{B}' \subseteq \mathcal{B}$ covering $\mathcal{P}$ with $|\mathcal{B}'| \leq |\mathcal{C}|$. Therefore, any minimum coverage kernel of $\mathcal{B}$ is also an optimal covering of $\mathcal{P}$ (thus, transferring the NP-hardness of the Orthogonal Polygon Covering problem [55] to Minimum Coverage Kernel).

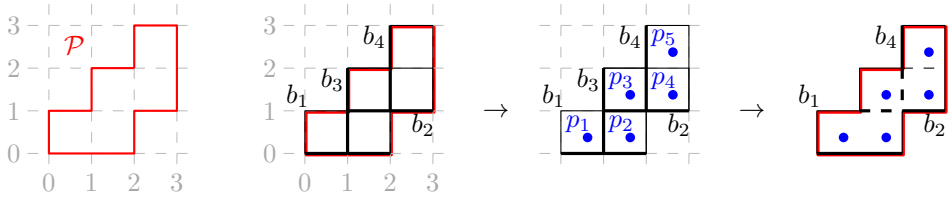



Figure 2.4: a) An orthogonal polygon $\mathcal{P}$. b) A set of boxes $\mathcal{B} = \{b_1 = [0,2]\times[0,1], b_2 = [1,3] \times [1,2], b_3 = [1,2] \times [0,2], b_4 = [2,3] \times [1,3]\}$ covering exactly $\mathcal{P}$, and such that in any cover of $\mathcal{P}$ with boxes, every box is either in $\mathcal{B}$, or fully covered by a box in $\mathcal{B}$. c) A set of points $\mathcal{D}(\mathcal{B}) = \{p_1, p_2, p_3, p_4, p_5\}$ such that any subset of $\mathcal{B}$ covering $\mathcal{D}(\mathcal{B})$, covers also $\mathcal{P}$. d) The subset $\{b_1, b_2, b_4\}$ is an optimal solution for the Orthogonal Polygon Covering problem on $\mathcal{P}$, the Minimum Coverage Kernel problem on $\mathcal{B}$, and the Box Cover problem on the set system $(\mathcal{D}(\mathcal{B}), \mathcal{B})$.

Now, let $\mathcal{B}$ be a set of $n$ rectangles, and consider the grid obtained by drawing infinite vertical and horizontal lines (hyperplanes in general) through the edges of each box in $\mathcal{B}$. This grid has within $\mathcal{O}(n^2)$ cells (which becomes $\mathcal{O}(n^d)$ when generalized to $d$ dimensions). Create a point-set $\mathcal{D}(\mathcal{B})$ as follows: for each cell $c$ which is completely inside a box in $\mathcal{B}$ we add to $\mathcal{D}(\mathcal{B})$ the middle point of $c$ (see Figure 2.4.c for an illustration). Note that a set $\mathcal{C} \subseteq \mathcal{B}$ covers $\mathcal{D}(\mathcal{B})$ if and only if $\mathcal{C}$ covers the same region as $\mathcal{B}$. We call such a point-set a coverage discretization of $\mathcal{B}$:

**Definition 2.9** (Coverage Discretization)**.** *Let $\mathcal{B}$ be a set of $d$-boxes. A* coverage discretization *of $\mathcal{B}$ (denoted as $\mathcal{D}(\mathcal{B})$) is a set of points $P$ such that any subset $\mathcal{C} \subset \mathcal{B}$ covers $\mathcal{D}(\mathcal{B})$ if and only if $\mathcal{C}$ covers the same region as $\mathcal{B}$ (namely, $\mathcal{C}$ is a* coverage kernel *of $\mathcal{B}$).*

Clearly, given any set of boxes $\mathcal{B}$, solving the Box Cover problem over the set system $(\mathcal{D}(\mathcal{B}), \mathcal{B})$ yields a solution for the Minimum Coverage Kernel problem. Note that a coverage discretization of a set $\mathcal{B}$ of $n$ $d$-boxes can be computed in polynomial time, and that its size is within $\mathcal{O}(n^d)$. Moreover, this bound is tight as there are sets of boxes which require a coverage discretization with $\Omega(n^d)$ points.

The relations described in this section between the Orthogonal Polygon Covering, Box Cover and Minimum Coverage Kernel problems have two main implications. Firstly, polynomial-time hardness results for the Minimum Coverage Kernel problem can be transferred to the Box Cover problem. In fact, we do this in Section 5.2, where we show

that Minimum Coverage Kernel remains NP-hard under severely restricted settings, and extend this result to the Box Cover problem in even more restricted settings. The other main implication is that polynomial-time approximation algorithms for the Box Cover problem can also be used for the Minimum Coverage Kernel problem. However, in scenarios where the boxes in $\mathcal{B}$ represent high dimensional data [56, 95, 118] and Coverage Kernels need to be computed repeatedly [10], using approximation algorithms for Box Cover can be unpractical: constructing $\mathcal{D}(\mathcal{B})$ requires time and space within $\Theta(n^d)$. We deal with this in Section 5.3, where we introduce a data structure to index $\mathcal{D}(\mathcal{B})$ without constructing it explicitly, and show how to use it to improve two existing approximation algorithms [39, 101] for the Box Cover problem (which we describe in the next section). This makes possible to obtain approximated solutions for the Minimum Coverage Kernel problem in the scenarios commented before.

### 2.4.2 Approximation algorithms for the Box Cover problem

As the Minimum Coverage Kernel problem is a special case of the Box Cover problem, all the approximation algorithms for the later one can be used to obtained approximated solutions for the Minimum Coverage Kernel problem. We review below two such algorithms.

The first algorithm we consider is the greedy $\mathcal{O}(\log n)$-approximation algorithm by Lovász [101] and Johnson [78]. This algorithm was described originally for the Set Cover problem, but it can be naturally described in terms of the Box Cover problem: iteratively pick the box which covers the most yet uncovered points, until there are no points left to cover.

**Algorithm 2.2** `Greedy-Set-Cover`$(\mathcal{T}, \mathcal{R})$

**Input:** A set system $(\mathcal{T}, \mathcal{R})$
**Output:** A subset $C \subseteq \mathcal{R}$ covering $\mathcal{T}$

```
1: U ← T, C ← ∅
2: while U ≠ ∅ do
3:     select an element r ∈ R that maximizes |U ∩ r|
4:     U ← U \ (U ∩ r)
5:     C ← C ∪ {r}
6: return C
```

Lovász [101] and Johnson [78] showed that this algorithm obtains a cover with at most $\mathtt{OPT} \cdot \ln n$ boxes, where $n$ is the number of points, and its running time is clearly polynomial in the size of the input. Note that, although the approximation factor depends on the number of points, this algorithm yields a $\mathcal{O}(\log n)$-approximation for the Minimum Coverage Kernel problem. This, because the number of points generated in the reduction to Box Cover is within $\mathcal{O}(n^d)$, $n$ being this time the number of boxes of the Minimum Coverage Kernel instance.

For the general Set Cover problem, the greedy algorithm is the best that one can hope for, unless P=NP. However, one would expect that in the case of the Box Cover

problem, better approximation factors can be obtained by exploiting the underlying geometry. Brönnimann and Goodrich [39] showed that indeed this is possible, and described a $\mathcal{O}(\log \mathtt{OPT})$-approximation algorithm running in polynomial time. Their algorithm employs and adapts a wide range of techniques, including the usage of $\varepsilon$-nets [75]:

**Definition 2.10** ($\varepsilon$-net)**.** *Let $\Sigma = (\mathcal{T}, \mathcal{R})$ be a set system, and let $\varepsilon \in (0, 1]$ be a parameter. An* $\varepsilon$-net *for $\Sigma$ is a subset $N \subseteq \mathcal{R}$ such that, for every $t \in \mathcal{T}$ that is contained in at least $\varepsilon|\mathcal{R}|$ elements of $\mathcal{R}$, there is one element in $N$ that contains $t$.*[5]

The concept of an $\varepsilon$-net can be naturally generalized to consider weights. Given a set system $\Sigma = (\mathcal{T}, \mathcal{R})$, and a weight function $\omega : \mathcal{R} \to \mathbb{R}_{\geq 0}$, let $\omega(S)$ denote the total weight of the elements in a subset $S \subset \mathcal{R}$; and for any $x \in \mathcal{T}$ let $\mathcal{R}_x = \{R \in \mathcal{R} \mid R \text{ contains } x\}$.

**Definition 2.11** ($\varepsilon$-heavy, $\varepsilon$-light)**.** *Let $\Sigma = (\mathcal{T}, \mathcal{R})$ be a set system, let $\omega : \mathcal{R} \to \mathbb{R}_{\geq 0}$ be a weight function and let $\varepsilon \in (0, 1]$ be a parameter. An element $t \in \mathcal{T}$ is said to be* $\varepsilon$-heavy *if $\omega(\mathcal{R}_t) \geq \varepsilon\omega(\mathcal{R})$, and* $\varepsilon$-light *otherwise*

**Definition 2.12** (Weighted $\varepsilon$-net)**.** *Let $\Sigma = (\mathcal{T}, \mathcal{R})$ be a set system, let $\omega : \mathcal{R} \to \mathbb{R}_{\geq 0}$ be a weight function and let $\varepsilon \in (0, 1]$ be a parameter. A* weighted $\varepsilon$-net *for $\Sigma$ and $\omega$ is a subset $N \subseteq \mathcal{R}$ such that $N \cap \mathcal{R}_t \neq \varnothing$ for every $\varepsilon$-heavy element $t \in \mathcal{T}$.*

Typically, $\varepsilon$-net algorithms are designed for the unweighted case [39]. However, simple methods allow to reduce the weighted case to an unweighted one. For instance, the following was described by Brönnimann and Goodrich [39]: first, let $m = |\mathcal{R}|$, and scale the weights such that $w(\mathcal{R}) = m$; then, take $\lfloor w(R) + 1 \rfloor$ copies of each range $R \in \mathcal{R}$. Note that the new set of ranges $\mathcal{R}'$ thus obtained contains all the elements of $\mathcal{R}$, and has cardinality at most $2m$. An $\varepsilon$-net for the set $\mathcal{R}'$ is also an $\varepsilon$-net for the original set $\mathcal{R}$ with weights $\omega$ [39, 105].

**Brönnimann and Goodrich's algorithm.** Let $\Sigma = (\mathcal{T}, \mathcal{R})$ be a set system, and let $\mathtt{OPT}$ denote the size of an optimal Set Cover for $\Sigma$. Let $k$ be an integer value such that $k/2 < \mathtt{OPT} < k$. Initialize the weight of each $R \in \mathcal{R}$ to 1, and repeat the following *weight-doubling step* until every element $t \in \mathcal{T}$ is $\frac{1}{2k}$-heavy: find an $\frac{1}{2k}$-light point $p$ and double the weights of all the elements of $\mathcal{R}$ containing $p$. When this process stops, return a $\frac{1}{2k}$-net $C$ with respect to the final weights as the approximated solution.

Since each point in $\mathcal{T}$ is $\frac{1}{2k}$-heavy, $C$ covers all the points of $\mathcal{T}$. Hence, if a $\frac{1}{2k}$-net of size $\mathcal{O}(kg(k))$ can be computed efficiently, this algorithm computes a solution of size $\mathcal{O}(kg(k))$ (i.e., a $g(\mathcal{O}(\mathsf{OPT}))$-approximation). The following lemma is the key to the performance of the algorithm:

**Lemma 2.2** (Lemma 3.4 of Brönnimann and Goodrich [39])**.** *If $\Sigma$ has a* Set Cover *of size at most $k$, then the algorithm performs at most $\mu_k = 4k \log(m/k)$ weight-doubling steps. The final weight of $\mathcal{R}$ is at most $m^4/k^3$.*

[5] The definition introduced by Haussler and Welzl [75] slightly differs from the one in Definition 2.10. However, it is simple to check that they are equivalent by considering their original formulation over the dual set system. The adaptation of Definition 2.10 is more natural for the Set Cover problem and its special cases, while the original definition is more natural for the dual Hitting Set problem.

By Lemma 2.2, if the algorithm does not terminate within $\mu_k$ steps, then one can conclude that $\Sigma$ does not have a Set Cover of size at most $k$ (i.e., $\mathtt{OPT} > 2k$). This allows to guess the correct $k$ via doubling search in $\mathcal{O}(\log \mathsf{OPT})$ stages (see Algorithm 2.3 for a detailed description). Since each step can be trivially implemented in time polynomial in $n + m$, and the weights can be represented with $\mathcal{O}(\log n)$ bits, the total running time of the algorithm is polynomial in $n + m$.

**Algorithm 2.3** `BG-Set-Cover`$(\mathcal{T}, \mathcal{R})$

**Input:** A set system $(\mathcal{T}, \mathcal{R})$
**Output:** A subset $C \subseteq \mathcal{R}$ covering $\mathcal{T}$

$k \leftarrow 1$, $C \leftarrow \varnothing$
**do**
  $k \leftarrow 2k$ ▷ (guess a new bound for $\mathtt{OPT}$)
  $\mu_k = \lceil 4k \log(m/k) \rceil$ ▷ (set the maximum number of steps to test the guess)
  **for** $R \in \mathcal{R}$ **do** $\omega(R) \leftarrow 1$
  **while** $\mu_k > 0$ **do**
    **if** there is an $\frac{1}{2k}$-light element $t \in \mathcal{T}$ **then**
      $\mu_k \leftarrow \mu_k - 1$
      **for** $R \in \mathcal{R}_t$ **do** $w(R) \leftarrow 2\omega(R)$ ▷ (weight-doubling step)
    **else** ▷ (all elements of $\mathcal{T}$ are $\frac{1}{2k}$-heavy)
      **find a** $\frac{1}{2k}$**-net** $N$ **with respect to** $\omega$ ▷ (and thus any $\frac{1}{2k}$-net covers $\mathcal{T}$)
      $C \leftarrow N$, $\mu_k \leftarrow 0$ ▷ (store the solution, and end both cycles)
**while** $C = \varnothing$
**return** $C$

Note that Brönnimann and Goodrich's algorithm [39] is a general method which is not bound to geometric instances of the Set Cover problem, but to the existence of algorithms computing good $\varepsilon$-nets. And it is precisely in this last part that geometry becomes relevant. Haussler and Welzl [75] proved that for a set system of $VC$-dimension[6] $\delta$, an $\varepsilon$-net of size $\mathcal{O}((\delta/\varepsilon)\log(\delta/\varepsilon))$ can be computed in polynomial time. The $VC$-dimension of a set system $(\mathcal{P}, \mathcal{B})$, where $\mathcal{P}$ and $\mathcal{B}$ are sets of $d$-dimensional points and boxes, respectively, is constant in any constant dimension $d$. Thus, the result of Haussler and Welzl [75] automatically yields a polynomial-time $\mathcal{O}(\log \mathsf{OPT})$-approximation algorithm. But the bounds on the size of $\varepsilon$-nets have been improved for several other geometric set systems. For instance, for set systems of points and halfspaces in $\mathbb{R}^2$ and $\mathbb{R}^3$, $\varepsilon$-nets of size $\mathcal{O}(1/\varepsilon)$ can be computed in polynomial time [106]; and for the set systems of the Box Cover problem in $d = 2, 3$ dimensions, Aronov et al. [18] gave a polynomial-time randomized algorithm to construct an $\varepsilon$-net of size within $\mathcal{O}(1/\varepsilon \log\log(1/\varepsilon))$.

**Remark.** *Both approximation algorithms described in this section for the* Box Cover *problem can be used to obtain approximation solutions for the* Minimum Coverage Kernel

[6]The *VC-dimension* of a set system $\Sigma = (\mathcal{T}, \mathcal{R})$ is the size of the largest subset $A \subseteq \mathcal{T}$ such that $|\{A \cap R \mid R \in \mathcal{R}\}| = 2^{|A|}$.

*problem. However, that requires explicitly constructing $\mathcal{D}(\mathcal{B})$, which takes time and space within $\Theta(n^d)$, and may be prohibitive in practice. We show how to avoid this construction in Section 5.3 by means of a new data structure to index $\mathcal{D}(\mathcal{B})$.*

## 2.5 Final Remarks

As seen, the problems considered in this thesis are all closely related, and their roots can be traced back to three fundamental optimization problems which are also tightly coupled. While a considerable research effort has been devoted to these problems, there are still a lot of open questions about them.

In the remaining of this thesis, we try to answer some of these open questions. We start in Chapter 3 by studying the striking similarities between the known algorithms and the lower bound arguments for the Klee's Measure and the Maximum Depth problems (described in Section 2.3) through a new and natural problem: the Depth Distribution problem. Then, in Chapter 4 we consider the computational complexity of the problems of finding maximum monochromatic and bichromatic matchings of points with rectangles (reviewed in Section 2.2). Finally, in Chapter 5, we consider the computational complexity of the Minimum Coverage Kernel problem, intermediate between the Box Cover and the Orthogonal Polygon Covering problems (as described in Section 2.4), attempting to understand the apparent differences in the approximabilities of the former two. Although there is almost a one to one relation (with respect to the topics considered) between the three main sections of this chapter, and the upcoming three chapters of this thesis, the definitions and results described within each section can be relevant to more than one of those chapters. For instance, we described in Section 2.3 two algorithms for the Klee's Measure and Maximum Depth problems which will be fundamental for new results introduced in two distinct chapters: in Chapter 3 we describe an algorithm for computing the Depth Distribution of a set of $d$-boxes based on Chan's algorithm [45] (detailed in Section 2.3.2), while in Chapter 5 we describe a data structure to approximate the solutions of the Minimum Coverage Kernel based on Overmars and Yap's partition tree [114] (detailed in Section 2.3.1).

# Chapter 3

# Depth Distribution of a Set of Boxes

In this chapter, we study the computation of the depth distribution of a set $\mathcal{B}$ of $n$ $d$-boxes. We introduce an algorithm which computes the depth distribution in time within $\mathcal{O}(n^{\frac{d+1}{2}} \log n)$, using space within $\mathcal{O}(n \log n)$, and describe a lower bound within $\Omega(n^{\frac{\omega}{2}})$ for this problem in two dimensions, where $\omega$ is the Matrix Multiplication exponent [70]. We also describe two variants of our algorithm which refine the computational upper bound of $\mathcal{O}(n^{\frac{d+1}{2}} \log n)$ for various measures of the difficulty on instances of the Depth Distribution problem.

## 3.1 Introduction

In Section 2.3 we described algorithms to compute the Klee's measure and the maximum depth of a set $\mathcal{B}$ of $n$ $d$-boxes. All known algorithms for these two measures are strikingly similar, and all the techniques used for solving the Klee's Measure problem can be applied with minor variations to solve the Maximum Depth problem. Even the arguments on their computational lower bounds are very similar. Although that could suggest a reduction from one to the other, as mentioned in Section 1.1, those two measures seem to be completely distinct. Given this, it is only natural to ask whether there is any way to formalize the relationship between these two measures, or whether the techniques used so far to compute them are actually exploring the certificate of a richer problem.

In a first step towards such a formalization, we introduce the notion of *depth distribution* of a set $\mathcal{B}$ of $n$ boxes $d$-boxes, defined as the vector formed by the $n$ values $(V_1, \ldots, V_n)$, where $V_i$ corresponds to the volume covered by exactly $i$ boxes from $\mathcal{B}$.

> Pb Depth Distribution problem: Given a set $\mathcal{B}$ of $n$ $d$-boxes, compute the vector $(V_1, V_2, \ldots, V_n)$ of the volumes of the regions covered by exactly $1, 2, \ldots, n$ boxes of $\mathcal{B}$, respectively.

The depth distribution of a set $\mathcal{B}$ can be interpreted as a probability distribution function (hence the name): if a point $p$ is selected uniformly at random from the region covered by $\mathcal{B}$, the probability that $p$ hits exactly $k$ boxes from $\mathcal{B}$ is $(V_k / \sum_{i=1}^{n} V_i)$, for all $k \in [1..n]$. In the context of a database, when receiving multidimensional range queries (e.g., about cars in a

two dimensional space measuring their safety and affordability), the depth distribution of the queries, as a probability distribution, yields valuable information to the database owner (e.g., a car dealer) about the distribution of the queries in the space of the data, to allow for informed decisions on it (e.g., to orient the future purchase of cars to resell based on the clients' desires, as expressed by their queries).

Note that the depth distribution **refines both** the Klee's measure and the maximum depth. It is a measure finer than the Klee's measure in the sense that the Klee's measure of a set $\mathcal{B}$ can be obtained in time linear in the size $n$ of $\mathcal{B}$ by summing the components of the depth distribution of $\mathcal{B}$. Similarly, the depth distribution is a measure finer than the maximum depth in the sense that the maximum depth of a set $\mathcal{B}$ can be obtained in linear time by finding the largest $i \in [1..n]$ such that $V_i \neq 0$.

**Computing the Depth Distribution.** The trivial approach of partitioning the space into cells that are completely contained within all the boxes that they intersect, results in a solution with prohibitive running time within $\mathcal{O}(n^{d+1})$. Simple variants of the techniques previously used to compute the Klee's measure [45, 114] result in a solution running in time within $\mathcal{O}(n^{d/2+1})$, using linear space, or a solution running in time within $\mathcal{O}(n^{(d+1)/2} \log n)$, but using space within $\mathcal{O}(n^{d/2} \log n)$.

**Our Results.** We provide computational upper bounds for the Depth Distribution problem. In Section 3.2, we describe an algorithm computing the depth distribution in time within $\mathcal{O}(n^{\frac{d+1}{2}} \log n)$, using space within $\mathcal{O}(n \log n)$. We refine the upper bound for various measures of difficulty of the input, such as the profile of the input and the treewidth of the intersection graph of the input set (Theorem 3.7), by describing algorithms which compute the depth distribution in time sensitive to the value of these measures.

In Section 3.3, we study lower bounds for the running time of computing the depth distribution. We show that the Depth Distribution problem generalizes not only Klee's Measure and the Maximum Depth problems, but surprisingly many others apparently unrelated to sets of boxes. For instance, we prove that the classical Matrix Multiplication problem is a special case of the Depth Distribution problem (Theorem 3.11), which extends the knowledge on an open question posed by Chan [44], and more importantly, it allows us to provide a lower bound for the computation of the depth distribution of $\Omega(n^{\frac{\omega}{2}})$, where $\omega$ is the Matrix Multiplication exponent. This lower bound allows to argue that the generalization of Klee's Measure and the Maximum Depth problems comes at a price: one cannot solve the Depth Distribution problem in the same asymptotic running time than the former two.

## 3.2 Algorithms Computing the Depth Distribution

We combine the techniques previously used to compute the Klee's measure [45, 114] into an algorithm which computes the depth distribution of a set of $n$ $d$-boxes in time within $\mathcal{O}(n^{\frac{d+1}{2}} \log n)$, using space within $\mathcal{O}(n \log n)$. We present this algorithm in Section 3.2.1, and

then in Section 3.2.2, we refine this upper bound for various measures of difficulty of the input instances of Depth Distribution.

### 3.2.1 Worst-case Analysis

We introduce an algorithm to compute the depth distribution inspired by a combination of the techniques introduced by Chan [45], and by Overmars and Yap [114], for the computation of the Klee's measure (described in Section 2.3). As in those approaches, the algorithm partitions the domain $\Gamma$ into $\mathcal{O}(n^{d/2})$ cells inside which every box in $\mathcal{B}$ is equivalent to a slab. After that, it computes the depth distribution within each cell, and combines those solutions into the final answer. Two main issues must be addressed: how to compute the depth distribution when the boxes are slabs, and how to partition the domain efficiently.

We first address the special case of slabs. We show in Lemma 3.1 that computing the depth distribution of a set of $n$ $d$-dimensional slabs within a domain $\Gamma$ can be done by using an algorithm to multiply polynomials of degree at most $n$.

**Lemma 3.1.** *Let $\mathcal{B}$ be a set of $n$ $d$-dimensional boxes whose intersection with a domain box $\Gamma$ are slabs. The computation of the depth distribution $(V_1, \ldots, V_n)$ of $\mathcal{B}$ within $\Gamma$ can be performed via a multiplication of $d$ polynomials of degree at most $n$.*

*Proof.* Assume w.l.o.g. that no box in $\mathcal{B}$ covers completely the domain $\Gamma$. For all $i \in [1..d]$, let $\mathcal{B}_i$ be the subset of slabs that are orthogonal to the $i$-th dimension, and let $(V_1^i, \ldots, V_n^i)$ be the depth distribution within $\Gamma$ of the intervals that result from projecting $\mathcal{B}_i$ into the $i$-th dimension. We associate a polynomial $P_i(x)$ of degree $n$ with each $\mathcal{B}_i$ as follows:

- let $\Gamma_i$ be the projection of the domain $\Gamma$ into the $i$-th dimension, and
- let $V_0^i$ be the length of the region of $\Gamma_i$ not covered by a box in $\mathcal{B}_i$ (i.e., $V_0^i = (|\Gamma_i| - \sum_{j=1}^{n} V_j^i)$); then
- $P_i(x) = \sum_{j=0}^{n} V_j^i \cdot x^j$.

Since any slab entirely covers the domain in all the dimensions but the one to which it is orthogonal, any point $p$ has depth $k$ in $\mathcal{B}$ if and only if it has depths $j_1$ in $\mathcal{B}_1$, $j_2$ in $\mathcal{B}_2$, ..., and $j_d$ in $\mathcal{B}_d$, such that $j_1 + j_2 + \ldots + j_d = k$. Thus, for all $k \in [0..n]$:

$$V_k = \sum_{\substack{0 \le j_1, \ldots, j_d \le n \\ j_1 + \ldots + j_d = k}} \left( \prod_{i=1}^{d} V_{j_i}^i \right),$$

which is precisely the $(k+1)$-th coefficient of $P_1(x) \cdot P_2(x) \cdot \ldots \cdot P_d(x)$. Thus, this product yields the depth distribution $(V_1, \ldots, V_n)$ of $\mathcal{B}$ in $\Gamma$. □

Using standard *Fast Fourier Transform* techniques, two polynomials can be multiplied in $O(n \log n)$-time (considering elementary arithmetic operations over the coefficients, such

as additions and multiplications, as primitive operations that consume constant time) [53]. Moreover, the depth distribution of a set of intervals (i.e., when $d = 1$) can be computed in linear time after sorting, by a simple scan-line algorithm, as can be done for the Klee's measure [45]. Thus, as a consequence of Lemma 3.1, when the boxes in $\mathcal{B}$ are slabs within a domain box $\Gamma$, the depth distribution of $\mathcal{B}$ inside $\Gamma$ can be computed in time within $\mathcal{O}(n \log n)$.

**Corollary 3.2.** *Let $\mathcal{B}$ be a set of $n$ $d$-dimensional boxes whose intersections with a $d$-dimensional box $\Gamma$ are slabs. The depth distribution of $\mathcal{B}$ inside $\Gamma$ can be computed in time within $\mathcal{O}(n \log n)$.*

A naive application of previous techniques for the KLEE'S MEASURE problem [45, 114] to the computation of depth distribution yields poor results:

- On one hand, combining the result described in Corollary 3.2 with the partition of the space and data structure described by Overmars and Yap [114] yields an algorithm to compute the depth distribution in time within $\mathcal{O}(n^{\frac{d+1}{2}} \log n)$, but using prohibitive space within $\Theta(n^{d/2} \log n)$.
- On the other hand, combining the result in Corollary 3.2 with Chan's partition of the space [45], yields an algorithm using space linear in the number of boxes, but running in time within $\Theta(n^{\frac{d}{2}+1} \log n)$ (i.e., paying an extra $\mathcal{O}(n^{\frac{1}{2}})$-factor for the reduction in space usage of Overmars and Yap [114]).

We combine these two approaches into a recursive algorithm which achieves the best features of both: it runs in time within $\mathcal{O}(n^{\frac{d+1}{2}} \log n)$, and uses $\mathcal{O}(n \log n)$-space. As in Chan's approach [45] we use a recursive simplify, divide and conquer algorithm, but we show that the running time is asymptotically the same as if the partition and data structure described by Overmars and Yap [114] were used (see Algorithm 3.1 for a detailed description). With each recursive step, the domain becomes smaller. When the domain is small enough such that all the boxes within it are slabs, we compute the depth distribution using Corollary 3.2 (base case, steps 1-4). We apply a simplification step before each recursive call removing the boxes that cover the entire domain and updating a value which keeps track of the number of such boxes (steps 6-8). To obtain the smaller domains for the recursive calls, we cut the current domain into two by a hyperplane, using for this the weighted median of the ($d$-2)-faces of the boxes orthogonal to the first dimensions (steps 9-12). We discuss the running time and space usage of this algorithm in Theorem 3.3.

**Theorem 3.3.** *Let $\mathcal{B}$ be a set of $n$ boxes in $\mathbb{R}^d$. The depth distribution of $\mathcal{B}$ can be computed in time within $\mathcal{O}(n^{\frac{d+1}{2}} \log n)$, using space within $\mathcal{O}(n \log n)$.*

*Proof.* First, we show that the running time $T(n)$ of Algorithm 3.1 is within $\mathcal{O}(n^{\frac{d+1}{2}} \log n)$. We can charge the number of boxes in the set to the number of ($d$-1)-faces intersecting the domain: if a box in $\mathcal{B}$ does not have a ($d$-1)-face intersecting the domain, then it covers the entire domain, and it would have been simplified (steps 6-8). Note that the ($d$-1)-faces orthogonal to dimension $x_1$ cannot intersect both the sub-domains $\Gamma_L$ and $\Gamma_R$ of the recursive

**Algorithm 3.1** `SDC-DDistribution`$(\mathcal{B}, \Gamma, c, (V_1, \ldots, V_n))$

**Input:** A set $\mathcal{B}$ of $n$ boxes in $\mathbb{R}^d$; a $d$-dimensional domain box $\Gamma$; the number $c$ of boxes not in $\mathcal{B}$ but in the original set that completely contain $\Gamma$; and a vector $(V_1, \ldots, V_n)$ representing the depth distribution computed so far.

1: **if** no box in $\mathcal{B}$ has a ($d$-2)-face intersecting $\Gamma$ (i.e., all the boxes are slabs) **then**
2: Compute the depth distribution $\left(V'_1, \ldots, V'_{|\mathcal{B}|}\right)$ of $\mathcal{B}$ within $\Gamma$ using Corollary 3.2
3: **for** $i \in [1..|\mathcal{B}|]$ **do**
4: $V_{i+c} \leftarrow V_{i+c} + V'_i$
5: **else**
6: Let $\mathcal{B}^0 \subseteq \mathcal{B}$ be the subset of boxes completely containing $\Gamma$
7: $c \leftarrow c + |\mathcal{B}^0|$
8: Let $\mathcal{B}' = \mathcal{B} \setminus \mathcal{B}^0$
9: Let $m$ be the weighted median of the ($d$-2)-faces orthogonal to $x_1$
10: Split $\Gamma$ into $\Gamma_L, \Gamma_R$ by the hyperplane $x_1 = m$
11: Rename the dimensions so that $x_1, \ldots, x_d$ becomes $x_2, \ldots, x_d, x_1$
12: Let $\mathcal{B}_L$ and $\mathcal{B}_R$ be the subsets of $\mathcal{B}'$ intersecting $\Gamma_L$ and $\Gamma_R$ respectively
13: Call `SDC-DDistribution`$(\mathcal{B}_\mathcal{L}, \Gamma_L, c, (V_1, \ldots, V_n))$
14: Call `SDC-DDistribution`$(\mathcal{B}_\mathcal{R}, \Gamma_R, c, (V_1, \ldots, V_n))$

calls at the same time (because the algorithm uses a hyperplane orthogonal to $x_1$ to split the domain into $\Gamma_L$ and $\Gamma_R$). Hence, although at the $d$-th level of the recursion there are $2^d$ recursive calls, any ($d$-1)-face can appear in at most $2^{d-1}$ of those. In general, for any $i$, there are at most $2^i$ recursive calls at the $i$-th level of recursion, but any ($d$-1)-face of the original set can intersect at most $2^{\lfloor i/d \rfloor (d-1)}$ of the cells corresponding to the domain of those calls. Hence, the total number of ($d$-1)-faces which "survive" until the $i$-th level of the recursion tree is within $\mathcal{O}(n2^{\lfloor i/d \rfloor (d-1)})$ (a similar argument was used by Overmars and Yap [114] to bound the running time of the data structure they introduced).

Let $h$ be the height of the recursion tree of Algorithm 3.1. Chan [45] showed that when the partition is done as in steps 9-11, $h$ is at most $\frac{d}{2} \log n + \mathcal{O}(1)$ (see Section 2.3.2 for details). We analyze separately the total cost $T_I(n)$ of the *interior nodes* of the recursion tree (i.e., the nodes corresponding to recursive calls which fail the base case condition in step 1) from the total cost $T_L(n)$ of the *leaves* of the recursion tree.

Since, the cost of each interior node is linear in the number of ($d$-1)-faces intersecting the corresponding domain, $T_I(n)$ is bounded by:

$$\begin{aligned}
T_I(n) &\in \sum_{i=1}^{h} \mathcal{O}\left(n \cdot 2^{\lfloor i/d \rfloor (d-1)}\right) \\
&\subseteq \mathcal{O}\left(n \sum_{i=1}^{h} 2^{i(d-1)/d}\right) \\
&\subseteq \mathcal{O}\left(n \cdot 2^{\frac{d-1}{d}h}\right) \\
&\subseteq \mathcal{O}\left(n \cdot 2^{\frac{d-1}{d}\frac{d}{2}\log n}\right) \qquad\qquad (\text{as } h \in \frac{d}{2}\log n + \mathcal{O}(1)) \\
&= \mathcal{O}\left(n \cdot n^{\frac{d-1}{2}}\right) = \mathcal{O}\left(n^{\frac{d+1}{2}}\right)
\end{aligned}$$

To analyze the total cost of the leaves of the recursion tree, first note that the total number $l$ of such recursive calls is within $\mathcal{O}(n^{d/2})$. Let $n_1, \ldots, n_l$ denote the number of ($d$-1)-faces in each of those recursive calls, respectively. Note that $T_L(n)$ is within $\mathcal{O}(\sum_{i=1}^{l} n_i \log n_i)$ because the result of Lemma 3.1 is used in step 1 of the algorithm. Besides, since the number of ($d$-1)-faces which survive until the $h$-th level of the recursion tree is within $\mathcal{O}(n^{\frac{d-1}{2}})$, $\sum_{i=1}^{l} n_i \in \mathcal{O}(n^{\frac{d+1}{2}})$. That bound, and the fact that $\log n_i \leq \log n$, for all $i \in [1..l]$, yields $T_L(n) \in \mathcal{O}(n^{\frac{d+1}{2}} \log n)$. As $T(n) = T_I(n) + T_L(n)$, the bound for the running time follows.

With respect to the space used by the algorithm, note that only one path in the recursion tree is active at any moment, and that at most $\mathcal{O}(n)$ extra space is needed within each recursive call. Since the height of the recursion tree is within $\mathcal{O}(\log n)$, the total space used by the algorithm is clearly within $\mathcal{O}(n \log n)$. □

Note that in the algorithm `SDC-DDistribution` the depth distribution is accumulated into a parameter. This is only to simplify the description and analysis of the algorithm, it does not impact its computational or space complexity. The initialization of the parameters of the algorithm `SDC-DDistribution` should be done as shown in Algorithm 3.2.

**Algorithm 3.2** `DDistribution`$(\mathcal{B}, \Gamma)$

**Input:** A set $\mathcal{B}$ of $n$ boxes in $\mathbb{R}^d$, a $d$-dimensional domain box $\Gamma$
**Output:** The depth distribution of $\mathcal{B}$ within $\Gamma$

1: $(V_1, V_2, \ldots, V_n) \leftarrow (0, 0, \ldots, 0)$
2: `SDC-DDistribution`$(\mathcal{B}, \Gamma, 0, (V_1, V_2, \ldots, V_n))$
3: return $(V_1, V_2, \ldots, V_n)$

The bound for the running time in Theorem 3.3 is worse than that of computing the Klee's measure (and Maximum Depth) by a factor within $\mathcal{O}(\sqrt{n} \log n)$, which raises the question of the optimality of the bound: we consider this matter in Section 3.3.

### 3.2.2 Adaptive Analysis

Even though the asymptotic complexity of $\mathcal{O}(n^{\frac{d+1}{2}} \log n)$ is the best we know so for the Depth Distribution problem in the worst case, the constructions for which that bound is met are rather artificial. From a practical perspective there are many instances which can be solved faster. While some of those "easy" instances can be mere particular cases, others could be hints of some hidden measures of difficulty of the Depth Distribution problem. We show that there are at least two such difficulty measures, gradually separating instances of the same size $n$ into various classes of difficulty. Informally, the first one measures how separable the boxes are by means of axis-aligned hyperplanes, while the second one measures how "complex" are the interactions between the boxes in the set.

#### A Profile-Sensitive Algorithm for the Depth Distribution problem

A special type of set of boxes commonly arising in practice is that where the set has bounded profile [57, 112]. The *i-th profile* $p_i$ of a set $\mathcal{B}$ of $d$-dimensional boxes is the maximum number of boxes intersected by any hyperplane orthogonal to the $i$-th dimension; and the *profile* $p$ of $\mathcal{B}$ is $p = \min_{i \in [1..d]}\{p_i\}$. D'Amore et al. [57] showed how to compute its value in linear time (after sorting the coordinates of the boxes in each dimension).

We show in the following lemma that the depth distribution can be computed in time sensitive to the profile of the input set.

**Theorem 3.4.** *Let $\mathcal{B}$ be a set of $n$ boxes in $\mathbb{R}^d$ with profile $p$, and $\Gamma$ be a $d$-dimensional domain box. The depth distribution of $\mathcal{B}$ within $\Gamma$ can be computed in time within $\mathcal{O}(n \log n + np^{\frac{d-1}{2}} \log p) \subseteq \mathcal{O}(n^{\frac{d+1}{2}} \log n)$.*

*Proof.* We describe an algorithm which partitions the domain $\Gamma$ into disjoint cells, computes the depth distribution within each cell, and combines the results into the final answer. For this, it sweeps the boxes with a hyperplane orthogonal to the dimension with smallest profile, and after every $2p$ endpoints of the boxes, it creates a new cell cutting the space with a hyperplane orthogonal to this dimension. This yields a partition of $\Gamma$ into $\mathcal{O}(n/p)$ cells, each intersecting at most $\mathcal{O}(p)$ boxes. Finally, the algorithm computes the depth distribution of $\mathcal{B}$ within each cell in time within $\mathcal{O}(p^{\frac{d+1}{2}} \log p)$, and obtains the depth distribution of $\mathcal{B}$ within $\Gamma$ by summing the respective components of the depth distribution within each slab. In total, this takes time within $\mathcal{O}(n \log n + np^{\frac{d-1}{2}} \log p)$. □

The theorem above automatically yields refined results for the computation of the Klee's measure and the maximum depth of a set of boxes $\mathcal{B}$. However, applying the technique in an *ad-hoc* way to these problems yields a slightly better bound:

**Corollary 3.5.** *Let $\mathcal{B}$ be a set of $n$ boxes in $\mathbb{R}^d$ with profile $p$, and $\Gamma$ be a $d$-dimensional domain box. The Klee's measure and* Maximum Depth *of $\mathcal{B}$ within $\Gamma$ can be computed in time within $\mathcal{O}(n \log n + np^{\frac{d-2}{2}}) \subseteq \mathcal{O}(n^{d/2})$.*

The algorithms from Theorem 3.4 and Corollary 3.5 asymptotically outperform previous ones in the sense that their running time is never worse than previous algorithms by more than a constant factor when the profile $p$ is within $\mathcal{O}(n^{1-\varepsilon})$, for some constant $\varepsilon > 0$.

An orthogonal approach is to consider how complex are the interactions between the boxes in the input set $\mathcal{B}$, analyzing, for instance, the intersection graph of $\mathcal{B}$. We study such a technique in the next section.

### Adaptivity to the Treewidth and Degeneracy of the Intersections Graph

The *treewidth* of a graph is a concept which captures how "close" to a tree the graph is. The treewidth was introduced independently several times under different names, and many graph problems that are NP-hard for general graphs can be solved in polynomial time for graphs with small treewidth (see Sections 10.4 and 10.5 of Kleinberg and Tardos's book [88] for nice overview). We describe below a technique to improve the running time of the depth distribution of sets of boxes whose intersection graphs have small treewidth.

A *$k$-degenerate* graph is an undirected graph in which every subgraph has a vertex of degree at most $k$ [100]. The *degeneracy* of a graph $G$ is the smallest value $k$ such that $G$ is $k$-degenerate. Every $k$-degenerate graph accepts an ordering of the vertices (called *degenerate ordering*) in which every vertex is connected with at most $k$ of the vertices that precede it.

In the following lemma we show that this ordering can be used to compute the depth distribution of a set $\mathcal{B}$ of $n$ boxes in running time sensitive to the degeneracy of the intersection graph of $\mathcal{B}$.

**Lemma 3.6.** *Let $\mathcal{B}$ be a set of $n$ $d$-boxes, let $\Gamma$ be a domain $d$-box, and let $k$ be the degeneracy of the intersection graph $G$ of $\mathcal{B}$. The depth distribution of $\mathcal{B}$ in $\Gamma$ can be computed in time within $\mathcal{O}(n \log^d n + e + nk^{\frac{d+1}{2}})$, where $e \in \mathcal{O}(n^2)$ is the number of edges of $G$.*

*Proof.* We describe an algorithm that runs in time within the bound in the lemma. The algorithm first computes the intersection graph $G$ of $\mathcal{B}$ in time within $\mathcal{O}(n \log^d n + e)$ [62], as well as the $k$-degeneracy of this graph and a degenerate ordering $O$ of the vertices in time within $\mathcal{O}(n+e)$ [107]. For $i \in [1..n]$ let $O[1..i]$ denote the first $i$ vertices of $O$, and $O[i]$ denote the $i$-th vertex of $O$. The algorithm then iterates over $O$ maintaining the invariant that, after the $i$-th step, the depth distribution within $\Gamma$ of the boxes corresponding to vertices in $O[1..i]$ has been correctly computed.

For any subset $U$ of vertices of $G$, let $\mathtt{DD}^{\Gamma}_{\mathcal{B}}(U)$ denote the depth distribution within $\Gamma$ of the boxes in $\mathcal{B}$ corresponding to the vertices in $U$. From $\mathtt{DD}^{\Gamma}_{\mathcal{B}}(O[1..i\text{-}1])$ (which the algorithm "knows" after the ($i$-1)-th iteration), $\mathtt{DD}^{\Gamma}_{\mathcal{B}}(O[1..i])$ can be obtained as follows: ($i.$) let $P$ be the subset of $O[1..i\text{-}1]$ adjacent to $O[i]$; ($ii.$) compute $\mathtt{DD}^{O[i]}_{\mathcal{B}}(P \cup \{O[i]\})$ in time within $\mathcal{O}(k^{\frac{d+1}{2}} \log k)$ using Algorithm 3.1 (note that the domain this time is $O[i]$ itself, instead of $\Gamma$); ($iii.$) add to $(\mathtt{DD}^{\Gamma}_{\mathcal{B}}(O[1..i]))_1$ the value of $(\mathtt{DD}^{O[i]}_{\mathcal{B}}(P \cup O[i]))_1$; and ($iv.$) for all $j = [2..k{+}1]$, subtract from $(\mathtt{DD}^{\Gamma}_{\mathcal{B}}(O[1..i]))_{j-1}$ the value of $(\mathtt{DD}^{O[i]}_{\mathcal{B}}(P \cup O[i]))_j$ and add it to $(\mathtt{DD}^{\Gamma}_{\mathcal{B}}(O[1..i\text{-}1]))_j$.

Since the updates to the depth distribution in each step take time within $\mathcal{O}(k^{\frac{d+1}{2}} \log k)$, and there are $n$ such steps, the lemma follows. □

The degeneracy $k$ of a graph $G$ is always at most the treewidth $t$ of $G$ [88] (i.e. $k \leq t$). Thus, the algorithm described above is sensitive to the treewidth of the intersection graph.

**Theorem 3.7.** *Let $\mathcal{B}$ be a set of $n$ boxes in $\mathbb{R}^d$, let $\Gamma$ be a $d$-dimensional domain box, and let $t$ be the treewidth of the intersection graph $G$ of the boxes in $\mathcal{B}$. The depth distribution of $\mathcal{B}$ within $\Gamma$ can be computed in time within $\mathcal{O}(n \log^d n + e + nt^{\frac{d+1}{2}})$, where $e \in \mathcal{O}(n^2)$ is the number of edges of $G$.*

Unlike the algorithm sensitive to the profile described in Section 3.2.2, this algorithm can run in time within $\mathcal{O}(n^{1+\frac{d+1}{2}})$ (e.g., when $k \in \Theta(n)$) in the worst-case, which is better than the $\mathcal{O}(n^{\frac{d+1}{2}})$ complexity of algorithm `SDC-DDistribution` only for values of the degeneracy $k$ within $\mathcal{O}(n^{1-\frac{2}{d}})$. A simple dove-tailing combination yields a solution whose asymptotic running time is the best of both.

As in the case of the algorithm sensitive to the profile of the input instance, applying the same technique in an *ad-hoc* fashion to the Klee's Measure and Maximum Depth problems yields improved solutions:

**Corollary 3.8.** *Let $\mathcal{B}$ be a set of $n$ boxes in $\mathbb{R}^d$, let $\Gamma$ be a $d$-dimensional box, and let $t$ be the treewidth of the intersection graph $G$ of $\mathcal{B}$. The Klee's measure and* Maximum Depth *of $\mathcal{B}$ within $\Gamma$ can be computed in time within $\mathcal{O}(n \log^d n + e + nt^{\frac{d}{2}})$, where $e \in \mathcal{O}(n^2)$ is the number of edges of $G$.*

The algorithms for the Depth Distribution problem described here, even those taking advantage of distinct measures of difficulty of the input, are computationally more expensive than their analogous for the Klee's Measure and the Maximum Depth problems. In the next section, we argue on why such a gap in the computational complexity of these problems might be unavoidable (without major breakthrough results).

## 3.3 Lower Bounds for the Depth Distribution problem

The only non-trivial lower bound known for the computational complexity of the Depth Distribution problem is $\Omega(n \log n)$ [68] (in the decision tree model with linear tests), which derives from the fact that this problem has the Klee's Measure problem as a special case (see Section 2.3.3 for details). Note, however, that this bound is known to be tight only when the input is a set of intervals (i.e., $d = 1$). For higher dimensions, the lower bound of $\Omega(n^{d/2})$ conjectured by Chan in 2008 [44] for the computational complexity of the Klee's Measure problem can be extended analogously to the computational complexity of the Depth Distribution problem. Again, note that there is a gap between this conjectured lower bound and the upper bound of $\mathcal{O}(n^{\frac{d+1}{2}} \log n)$ described in Section 3.2.

In this section we study whether the gap between the computational complexities of the Depth Distribution problem and the Klee's Measure and Maximum Depth problems can be eluded. Although, we do not know the answer to this question yet, we argue that it is in fact a very hard (and relevant) question about not only boxes, but other

central problems in Computer Science. We show that proving that such a gap can be eluded implies breakthrough results for two fundamental problems in computer science, the Integer Multiplication and Matrix Multiplication problems:

- First, we consider in Section 3.3.1 the case when all the boxes are slabs, and the input is given pre-sorted in each dimension. Both the Klee's measure and the maximum depth of such instances can be computed in linear time. We show that if there is an algorithm computing the depth distribution of such instances in linear time then there is also a linear-time algorithm for multiplying two $n$-bit numbers, which would contradict a conjecture of Schönhage and Strassen [122] for the Integer Multiplication problem, widely accepted by the community [69, 73].

- Then, in Section 3.3.2 we consider the planar case of the Depth Distribution problem. The Klee's measure and maximum depth of a set of $n$ rectangles can be computed in time within $\mathcal{O}(n \log n)$ [45]. We show that any algorithm computing the depth distribution in time within $\mathcal{O}(n \log n)$ can be transformed into an algorithm which multiplies two $n \times n$ matrices in time within $\mathcal{O}(n^2 \log n)$. Thus, any such algorithm for the Depth Distribution problem would solve a fundamental and long-standing question for the Matrix Multiplication problem [52, 70, 121, 125].

### 3.3.1 Conditional lower bound for the case of sorted slabs

Let $\Gamma$ be a domain $d$-box, and let $\mathcal{B}$ be a set of $n$ $d$-boxes which are all slabs within $\Gamma$. Suppose that, together with $\Gamma$ and $\mathcal{B}$, we are given as part of the input the relative order of the endpoints of the boxes in $\mathcal{B}$ in each dimension. It is well known that both the Klee's measure and maximum depth of $\mathcal{B}$ can be computed in linear time under these settings [45, 114]. We argue that the same is not true for the depth distribution: we prove that computing the depth distribution of $\mathcal{B}$ requires time within $\Omega(n \log n)$, unless two $n$-bit integers can be multiplied in time within $o(n \log n)$.

Computing the product of two $n$-bit integers, known as the Integer Multiplication problem, is an important fundamental problem in algorithmic number theory, algebra and theoretical computer science [91]. A naive approach leads to an algorithm that uses $\mathcal{O}(n^2)$ bit operations, but in 1963 Karatsuba and Ofman [81] showed how to reduce the number of operations to within $\mathcal{O}(n^{\log_2 3})$. In a major breakthrough, in 1971, Schönhage and Strassen [122] described an efficient algorithm for multiplying integers using fast polynomial multiplication, and running in time within $\mathcal{O}(n \cdot \log n \cdot \log\log n)$. Since then, the prevailing (and widely accepted) conjecture has always been that the computational complexity of this problem is within $\Theta(n \log n)$ [69, 73, 122].

We show that any algorithm $\mathcal{A}$ for the Depth Distribution problem over a set of pre-sorted slabs can be used to multiply two $n$-bit integers via a reduction using $\mathcal{O}(n)$ bit operations. We do this in two steps: first, in Lemma 3.9, we show that any two polynomials with non-negative integer coefficients can be multiplied using $\mathcal{A}$, and then in Theorem 3.10, using the fact that the product of two $n$-bits integers can be obtained by means of an

algorithm for polynomial multiplication, we prove the conditional lower bound. Since the Integer Multiplication problem is usually studied in the multi-tape Turing machine model (also referred to as the *bit-complexity* model) [116], we show that our reduction can be executed in linear time on a 3-tape Turing machine. Thus, the conditional lower bound described here also applies to any computational model with at least as much power as a multi-tape Turing machine (which is already a highly restrictive computational model).

**Lemma 3.9.** *Let $P(x), Q(x)$ be two $m$-degree polynomials with integer coefficients in $[0..(2^m - 1)]$, and let $n = 2m^2$ (i.e., $n$ is a bound for the size in bits of the representations of $P, Q$). There is a set $\mathcal{B}$ with $2m$ rectangles, and a rectangle $\Gamma$ such that:*

1. *All the rectangles, including $\Gamma$ have their endpoints in $[0..m2^m]$;*
2. *All the rectangles of $\mathcal{B}$ are slabs in $\Gamma$, and the relative order of their endpoints in each dimension is known;*
3. *Both the domain $\Gamma$, and the set $\mathcal{B}$, can be computed in time within $\mathcal{O}(n)$;*
4. *If the depth distribution of $\mathcal{B}$ within $\Gamma$ can be computed in time $T(n)$, then the polynomial $P(x) \cdot Q(x)$ can be computed in time within $\mathcal{O}(T(n) + n)$.*

*Proof.* Given $P(x)$ and $Q(x)$, we create a set $\mathcal{B}$ with $2m$ rectangles, and a domain rectangle $\Gamma$, such that from the depth distribution $(V_1, V_2, \ldots, V_{2m})$ of $\mathcal{B}$ within $\Gamma$, the coefficients of $P(x) \cdot Q(x)$ can be obtained via a simple linear time procedure. Let $p_i$ and $q_i$ denote the $i$-th coefficient of $P(x)$ and $Q(x)$, respectively, for all $i = [1..m]$. Choose $\mathcal{B}$ and $\Gamma$ as follows (see Figure 3.1 for an illustration):

1. Take $\Gamma$ as the rectangle $\{(x, y) \mid 0 \leq x \leq \sum_{j=0}^{m} p_i, 0 \leq y \leq \sum_{j=0}^{m} q_i\}$;
2. Take $\mathcal{B} = \mathcal{B}_p \cup \mathcal{B}_q$, where $\mathcal{B}_p$ and $\mathcal{B}_q$ are the sets of slabs (orthogonal to the $x$ and $y$ axes, respectively) defined as:

$$\mathcal{B}_p = \bigcup_{i=1}^{m} \left\{ b_i^p = \left\{ (x, y) \in \Gamma \mid 0 \leq x \leq \sum_{k=0}^{i} p_{m-k} \right\} \right\}$$
$$\mathcal{B}_q = \bigcup_{i=1}^{m} \left\{ b_i^q = \left\{ (x, y) \in \Gamma \mid 0 \leq y \leq \sum_{k=0}^{i} q_{m-k} \right\} \right\}.$$

Let $\Gamma_x$, $\Gamma_y$ denote the intervals resulting from projecting $\Gamma$ to the $x$ and $y$ axes, respectively, and let $|I|$ denote the length of any interval $I$. $\mathcal{B}_p$ is defined so that when its $m$ rectangles are projected to the $x$-axis, the length of the region in $\Gamma_x$ covered by exactly $k$ rectangles is precisely $p_k$, for all $k = [0..m]$. Thus, if we let $(V_1^x, \ldots, V_m^x)$ denote the depth distribution of the projection of $\mathcal{B}_p$ to the $x$-axis within $\Gamma_x$, and let $V_0^x = |\Gamma_x| - \sum_{i=1}^{m} V_i^x$, then $p_k = V_k^x$, for all $k = [0..m]$. Analogous observations can be made for $Q(x), \mathcal{B}_q, \Gamma_y$ and $(V_0^y, V_1^y, \ldots, V_m^y)$.

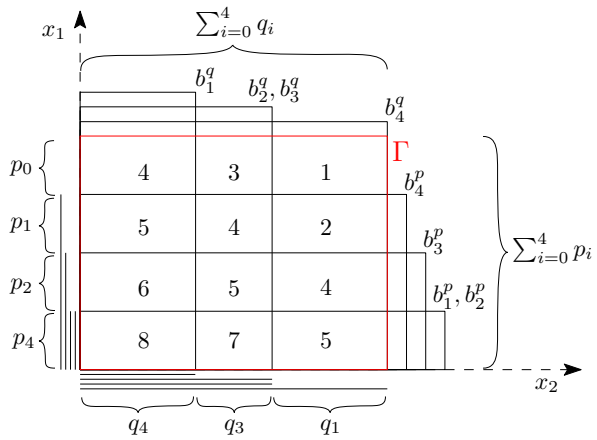



Figure 3.1: A product $P(n) \cdot Q(n)$ as an instance of Depth Distribution, for $P(n) = p_4n^4 + p_2n^2 + p_1n + p_0$, and $Q(n) = q_4n^4 + q_3n^3 + q_1n$. The set of rectangles generated is $\mathcal{B} = \mathcal{B}_p \cup \mathcal{B}_q$, where $\mathcal{B}_p = \{b_1^p, b_2^p, b_3^p, b_4^p\}$ and $\mathcal{B}_q = \{b_1^q, b_2^q, b_3^q, b_4^q\}$. The number within each cell indicates the depth of the corresponding region. The lines to the left of (resp. below) the $x_1$ axis (resp. the $x_2$ axis) illustrate the intervals that would result from projecting $\mathcal{B}_p$ (resp. $\mathcal{B}_q$) to $x_1$ (resp. to $x_2$). The numbers over curly brackets indicate the length of the region delimited by the brackets.

Let $c_k$ denote the $k$-th coefficient of $P(x) \cdot Q(x)$. Since any point $p$ has depth $k$ in $\mathcal{B}$ if and only if it has depth $i$ in $\mathcal{B}_p$ and $j$ in $\mathcal{B}_q$ such that $i + j = k$, for all $k \in [1..2m]$ we have:

$$V_k = \sum_{\substack{0 \le i,j \le m \\ i+j=k}} V_i^x \cdot V_j^y \quad = \sum_{\substack{0 \le i,j \le m \\ i+j=k}} p_i \cdot q_j \quad = \quad a_k$$

Thus, the $2m$ components of the depth distribution of $\mathcal{B}$ within $\Gamma$ are precisely the first $2m$ coefficients (in descending order of the exponents) of $P(x) \cdot Q(x)$. Besides, the constant coefficient of the product is given by $|\Gamma| - \sum_{i=1}^{m} V_i$, where $|\Gamma|$ is the volume of $\Gamma$. Therefore, the product $P(x) \cdot Q(x)$ can be obtained almost directly from the Depth Distribution of $\mathcal{B}$ and $\Gamma$.

Now, let us analyze the running time of this reduction in a 3-Tapes Turing Machine (the input and output tapes, and one additional tape to keep the partial sums). We assume the input tape contains first all the coefficients of $P(x)$ in descending order of the exponents, and then all the coefficients of $Q(x)$. To produce the set $\mathcal{B}_p$ the coefficients of $P(x)$ are read one by one, keeping track of the partial sum of the ones read so far in the additional tape. To process the $i$-th coefficient of $P(x)$ we add $p_i$ to the partial sum, and write to the output tape the box $b_i^p$. Note that, to represent each endpoint of a box in $\mathcal{B}$, $\lceil \log(m \cdot 2^m) \rceil$-bits suffice, and thus processing each coefficient of $P(x)$ costs within $\mathcal{O}(\log(m \cdot 2^m))$ operations. Therefore, the total operations required to produce $\mathcal{B}_p$ is within $\mathcal{O}(m \log(m \cdot 2^m)) \subseteq \mathcal{O}(m \log m + m^2) \subseteq \mathcal{O}(n)$. The same is true for $\mathcal{B}_q$, and therefore also for $\mathcal{B}$. Besides, $\Gamma$ can be computed also trivially with $\mathcal{O}(m)$ operations. Finally, let $\mathcal{A}$ be an algorithm which computes the depth distribution of $\mathcal{B}$ within $\Gamma$ in time $T(n)$. We execute $\mathcal{A}$ over $\mathcal{B}, \Gamma$, and produce the polynomial $P(x) \cdot Q(x)$

from the output of $\mathcal{A}$ as already described, which can be done with a total number of operations linear in size of the representation of $P(x) \cdot Q(x)$. To complete the proof, let us bound the bits required to represent $P(x) \cdot Q(x)$. Note that $P(x) \cdot Q(x)$ has at most $2m$ coefficients, and since each of those coefficients is the sum of at most $2^m$ coefficients of $P(x)$ and $Q(x)$, the number of bits required to represent it is within $\mathcal{O}(\log(2^m \cdot 2^m)) \subseteq \mathcal{O}(m)$. Therefore, the total number of bits in $P(x) \cdot Q(x)$ is within $\mathcal{O}(n)$. □

Since the product of two $n$-bit integers can be obtained from the product of two polynomials with non-negative integer coefficients, the result of Lemma 3.9 yields the following conditional lower bound for the Depth Distribution problem on pre-sorted slabs:

**Theorem 3.10.** *Let $\Gamma$ be a domain $d$-box, and let $\mathcal{B}$ be a set of $d$-boxes which are all slabs within $\Gamma$, and let $n$ be a the number of bits required to represent $\mathcal{B}$ and $\Gamma$. The depth distribution of $\mathcal{B}$ within $\Gamma$ cannot be computed in time within $o(n \log n)$ unless two $n$-bit integers can be multiplied using within $o(n \log n)$ bit operations.*

*Proof.* Suppose there exists an algorithm $\mathcal{A}$ which computes the depth distribution of $\mathcal{B}$ within $\Gamma$ using $T(n) \in o(n \log n)$ bit operations. Let $a, b$ be two $n$-bit integers. Assume that $n$ is a power of 2 (if not, we can add padding zeros until this condition is met), and let $m = \sqrt{n}$. The product $c = a \cdot b$ can be computed using $\mathcal{A}$ as follows:

1. Split $a$ and $b$ into $m$ blocks $\{a_1, \ldots, a_m\}$ and $\{b_1, \ldots, b_m\}$, respectively, such that each block has $m$ bits, and
$$a = \sum_{i=0}^{m-1} a_i 2^{im}, \quad b = \sum_{i=0}^{m-1} b_i 2^{im} \qquad \text{(equivalently, express } a, b \text{ in base } 2^m\text{)};$$
2. Let $P_a(x), P_b(x)$ be the $m$-degree polynomials defined as
$$P_a(x) = \sum_{i=0}^{m-1} a_i x^i, \quad P_b(x) = \sum_{i=0}^{m-1} b_i x^i \qquad \text{(note that } a = P_a(2^m), b = P_b(2^m)\text{)};$$
3. Compute $P_c(x) = P_a(x) \cdot P_b(x)$ using algorithm $\mathcal{A}$ as in Lemma 3.9;
4. Finally, perform carrying $\mod (2^m + 1)$ over the coefficients of $P_c(x)$ in ascending order of their exponents to obtain $c$.

This procedure, inspired by Schönhage and Strassen's algorithm [122], performs within $\mathcal{O}(T(n) + n)$ operations on a 3-tapes Turing machine. This is clear for the first two steps, and for the third one it directly derives from Lemma 3.9. For the fourth step, note that $P_c(x)$ has at most $2m$ coefficients, each of those being the sum of at most $2^m$ coefficients of $P_a(x)$ and $P_b(x)$. Thus, the number of bits required to represent each coefficient of $P_c(x)$ is within $\mathcal{O}(\log(2^m \cdot 2^m)) \subseteq \mathcal{O}(m)$, and the total number of bits in $P_c(x)$ is within $\mathcal{O}(n)$. □

Therefore, proving that the depth distribution of a set of sorted slabs can be computed in the same running time than its Klee's measure or its maximum depth, would contradict Schönhage and Strassen's conjecture [122] of the optimality of $\Theta(n \log n)$ for the complexity of the Integer Multiplication problem.

### 3.3.2 Conditional lower bound for sets of planar boxes

For general sets of boxes in the Euclidean plane, one can also argue that the DEPTH DISTRIBUTION problem is computationally harder than the KLEE'S MEASURE and the MAXIMUM DEPTH problems. Proving the opposite would imply breakthrough results in the long-standing problem of MATRIX MULTIPLICATION. We prove that any instance of MATRIX MULTIPLICATION can be solved by using an algorithm which computes the depth distribution of a set of planar boxes. For this, we make use of the following simple observation:

**Observation 3.1.** *Let $A, B$ be two $n \times n$ matrices of real numbers, and let $C_i$ denote the $n \times n$ matrix that results from multiplying the $n \times 1$ vector corresponding to the $i$-th column of $A$ with the $1 \times n$ vector corresponding to the $i$-th row of $B$. Then, $AB = \sum_{i=1}^{n} C_i$.*

We show in Theorem 3.11 that multiplying two $n \times n$ matrices can be done by transforming the input into a set of $\mathcal{O}(n^2)$ axis aligned rectangles, and computing the depth distribution of the resulting set. Moreover, this transformation can be done in linear time, thus, the theorem yields a conditional lower bound for the computation of the depth distribution.

**Theorem 3.11.** *Let $A, B$ be two $n \times n$ matrices of non-negative real numbers. There is a set $\mathcal{B}$ of planar boxes of size within $O(n^2)$, and a domain box $\Gamma$, such that the depth distribution of $\mathcal{B}$ within $\Gamma$ can be projected to obtain the value of the product $AB$.*

*Proof.* We create a *gadget* to represent each $C_i$. Within the $i$-th gadget, there will be a rectangular region for each coefficient of $C_i$ with the value of that coefficient as area (see Figure 3.2 for a general outlook of such instance). We arrange the boxes so that two of such rectangular regions have the same depth if and only if they represent the same respective coefficients of two different matrices $C_i$ and $C_{i'}$ (formally, they represent some coefficients $(C_i)_{j,k}$ and $(C_{i'})_{j',k'}$, respectively, such that $i \neq i', j = j'$, and $k = k'$).

We describe a set $\mathcal{B}$ of $5n^2$ planar boxes (one box for each of the $n^2$ coefficients of $A$, two boxes for each of the $n^2$ coefficients of $B$, and $2n^2$ additional boxes) such that, for each $i, j \in [1..n]$, the $(2ni + 2j + 1)$-th component of the depth distribution of $\mathcal{B}$ is equal to the component $AB_{i,j}$ of the product $AB$. Such a set can be constructed as follows (see Figure 3.3 on page 47 for a graphical representation of such an instance):

- Let the domain $\Gamma = \{(x, y) \mid 0 \leq x \leq \sum_i \sum_j B_{i,j}, 0 \leq y \leq \sum_i \max_j\{A_{i,j}\}\}$.
- For all $i \in [1..n]$ we create a gadget for $C_i$ that covers the entire domain in the $y$-direction, and that spans from $C_i^{start} = \sum_{j=1}^{i-1} \sum_{k=0}^{n} B_{j,k}$ to $C_i^{end} = C_i^{start} + \sum_{k=0}^{n} B_{i,k}$ in the $x$-direction.
- Within the gadget for $C_i$ we place one box for each $A_{j,i}$ and two boxes for each $B_{i,j}$, for $i, j \in [1..n]$, as follows: the one corresponding to $A_{j,i}$ will span $C_i$ entirely in the $x$-direction, and is bounded by $(\sum_{k=1}^{j} \max_{l=1}^{n}\{A_{k,l}\}) \leq y \leq (A_{j,i} + \sum_{k=1}^{j} \max_{l=1}^{n}\{A_{k,l}\})$ in the $y$-direction. For $B_{i,j}$ we place two identical boxes entirely spanning $C_i$ in the $y$-direction, and in the $x$-direction bounded by $(C_i^{start} + \sum_{k=1}^{j-1} B_{i,k}) \leq x \leq C_i^{end}$.

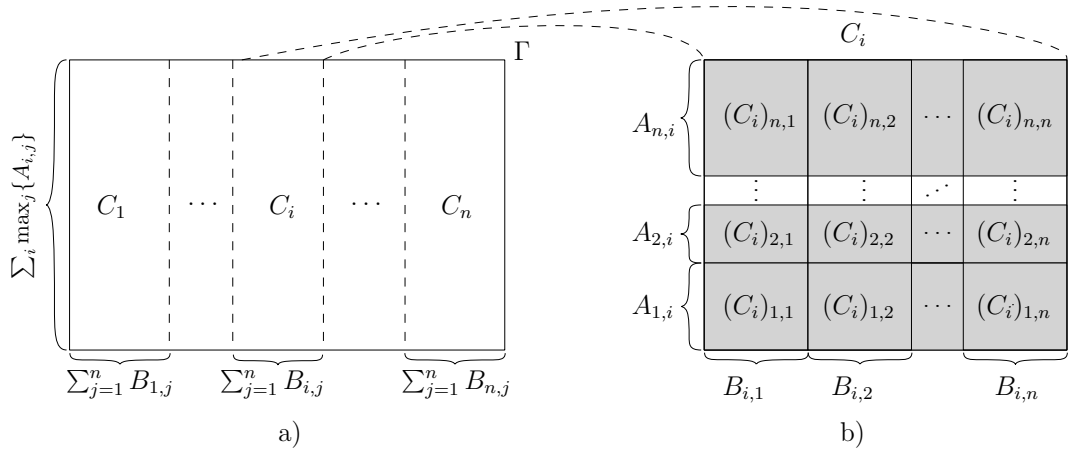



Figure 3.2: An outlook of the instance generated for the product $AB$: we add a gadget for each $C_1, \ldots, C_n$, within a domain $\Gamma$ as in a). In b), a representation of $C_i$ with $2n$ boxes, the volumes of the rectangular regions correspond to the coefficients of $C_i$ (the regions in a gadget must have different depths to avoid that their areas are added into a same component of the depth distribution).

- Finally, we add $2n^2$ boxes to ensure that rectangular regions corresponding to two coefficients $C_{i,j}$ and $C_{i,k}$ in different rows $j, k$ of a same $C_i$ do not share the same depth, for all $i, j, k \in [1..n]$ . For this, for all $j \in [1..n]$ we add $2n$ identical boxes entirely spanning the domain in the $x$-direction, and spanning from $(\sum_{k=1}^{j} \max_{l=1}^{n}\{A_{k,l}\})$ to $(\sum_i \max_j\{A_{i,j}\}))$ in the $y$-direction.

Note that in the instance generated, for $i, j \in [1..n]$:

- a region has odd depth if and only if its area is equal to some coefficient of any $C_i$;
- the regions corresponding to coefficients of the $i$-th rows have depth between $(2in + 3)$ and $(4in + 1)$;
- within the gadget for each $C_i$, the rectangular region with volume corresponding to the coefficient $C_{i,j}$ has depth $(2ni + 2j + 1)$, and no other rectangular region within the gadget has that depth;
- two regions have the same depth if and only if they represent the same respective coefficients of two matrices $C_i$ and $C_{i'}$.

The arguments above and the fact that, by definition of the depth distribution, the volumes of regions with the same depth are accumulated together, yield the result of the theorem. □

The optimal time to compute the product of two $n \times n$ matrices in an arithmetic circuit [70] is still an intriguing open question. It can naturally be computed using within $O(n^3)$ arithmetic operations. However, Strassen showed in 1969 that within $O(n^{2.81})$ arithmetic operations

are enough [125]. This gave rise to a new area of research, where the central question is to determine the value of the exponent of the computational complexity of square matrix multiplication, denoted $\omega$, and defined as the minimum value such that two $n \times n$ matrices can be multiplied using within $O(n^{\omega+\varepsilon})$ arithmetic operations for any $\varepsilon > 0$.

The result of Theorem 3.11 directly yields a conditional lower bound on the complexity of the Depth Distribution problem: in particular, if the depth distribution of $n$ boxes in the plane can be computed in time within $\mathcal{O}(n \log n)$, then Matrix Multiplication can be computed in time within $\mathcal{O}(n^2 \log n)$, implying that $\omega = 2$. However, this would be a great breakthrough in the area, the best known upper bound to date is approximately $\omega \leq 2.37$, when improvements in the last 30 years [52, 70, 121] have been in the range $[2.3728, 2.3754]$.

Note that although the reduction described in the proof of Theorem 3.11 works only for matrices with non-negative coefficients, this fact does not weaken the result in the theorem. The multiplication of two matrices with some negative coefficients can be obtained from the multiplication of two matrices with only positive coefficients, and such a transformation requires only time linear in the size of the matrices. Moreover, even for the multiplication of boolean matrices (i.e., matrices with all its coefficients either zero or one) the best bound known so far for $\omega$ is the same as for the general problem [129]. Since the set $\mathcal{B}$ of Theorem 3.11 can be obtained using within $\mathcal{O}(n)$ arithmetic operations in an arithmetic circuit, we obtain the following conditional lower bound:

**Corollary 3.12** (Conditional lower bound)**.** *Computing the depth distribution of a set $\mathcal{B}$ of $n$ $d$-dimensional boxes requires within $\Omega(n^{1+c})$ arithmetic operations, for some constant $c > 0$, unless two $n \times n$ matrices can be multiplied using a number of arithmetic operations within $\mathcal{O}(n^{2+\varepsilon})$, for any constant $\varepsilon > 0$*

In the next section we summarize the results described in this chapter and present some future directions of research on the Depth Distribution problem.

## 3.4 Discussion

The computation of the depth distribution generalizes not only that of the Klee's measure and the maximum depth, but it also generalizes many other central problems with no apparent relation to orthogonal boxes (such as Matrix Multiplication and all those this generalizes). As a measure, the depth distribution captures many of the features in common between the Klee's measure and the maximum depth, so that new upper bounds on the computation of the depth distribution will yield corresponding results for the computation of those two measures. Nevertheless, we know of no direct reduction from the Depth Distribution problem to Klee's measure or the Maximum Depth problems, and in fact we argued this might be unlikely. We discuss below some further issues to ponder about these measures.

**Discrete variants.** In practice, multidimensional range queries are applied to a database of multidimensional points. This yields discrete variants of each of the problems previously discussed [1, 127]. In the Discrete Klee's Measure problem, the input is composed of

not only a set $\mathcal{B}$ of $n$ boxes, but also of a set $S$ of $m$ points. The goal is now to compute not the volume of the union of the boxes, but the number (and/or the list) of points which are covered by those boxes. Similarly, one can define a discrete version of the MAXIMUM DEPTH problem (which points are covered by the maximum number of boxes) and of the DEPTH DISTRIBUTION problem (how many and which points are covered by exactly $i$ boxes, for $i \in [1..n]$). Interestingly enough, the computational complexity of these discrete variants is much less than that of their continuous versions when there are reasonably few points [127]: the discrete variant becomes hard only when there are many more points than boxes [1]. Nevertheless, "easy" configurations of the boxes also yield "easy" instances in the discrete case: it will be interesting to analyze the discrete variants of these problems according to the *profile* of the input set, and according to the *degeneracy* and *treewidth* of its intersection graph, introduced for the continuous versions (in Section 3.2.2).

**Tighter Bounds.** Chan [44] conjectured that any *combinatorial* algorithm which computes the Klee's measure or the maximum depth of a set of $n$ boxes in $\mathbb{R}^d$, requires running time within $\Omega(n^{d/2})$. This conjectured lower bound on the computation of the Klee's measure applies of course to the more general problem of computing the depth distribution. However, the running time of the algorithm described in 3.2.1 can be worse than that lower bound by a factor within $\mathcal{O}(\sqrt{n}\log n)$. It is unknown whether this gap between the asymptotic upper and lower bounds on the computational complexity of the DEPTH DISTRIBUTION problem can be closed, either by finding an improved algorithm, or by proving more restrictive lower bounds. The depth distribution of a set of boxes yields much more information than its Klee's measure (actually, a large part of this information can be ignored during the computation of the Klee's measure). This may hint that computing the depth distribution is actually asymptotically harder than the special cases, and that proving more restrictive lower bounds is plausible. We were able to argue in Section 3.2.2 that at least in two dimensions this is the case, but it is unclear whether (or how) that argument can be extended to higher dimensions.

**Special cases of the input.** As a consequence of the lower bound that Chan [44] conjectured, recent work on the computation of the Klee's measure have focused on the study of special cases which can be solved faster than $\mathcal{O}(n^{d/2})$ [37, 38, 45]. For instance, when all the boxes are *orthants* the Klee's measure can be computed in time within $O(n^{d/3}\operatorname{polylog} n)$ [45]; the Klee's measure of a set of hypercubes can be computed in time within $O(n^{(d+1)/3}\operatorname{polylog} n)$ [9, 37, 38, 45]; and the case of 2-*grounded* boxes can be solved in time within $O(n^{(d-1)/2}\log^2 n)$, for any dimension $d \geq 3$ [126]. Similar improvements for the computation of the depth distribution of those special cases are likely. An interesting question is whether in some of these specific cases, one can show that computing the depth distribution is asymptotically equally hard to computing the Klee's measure.

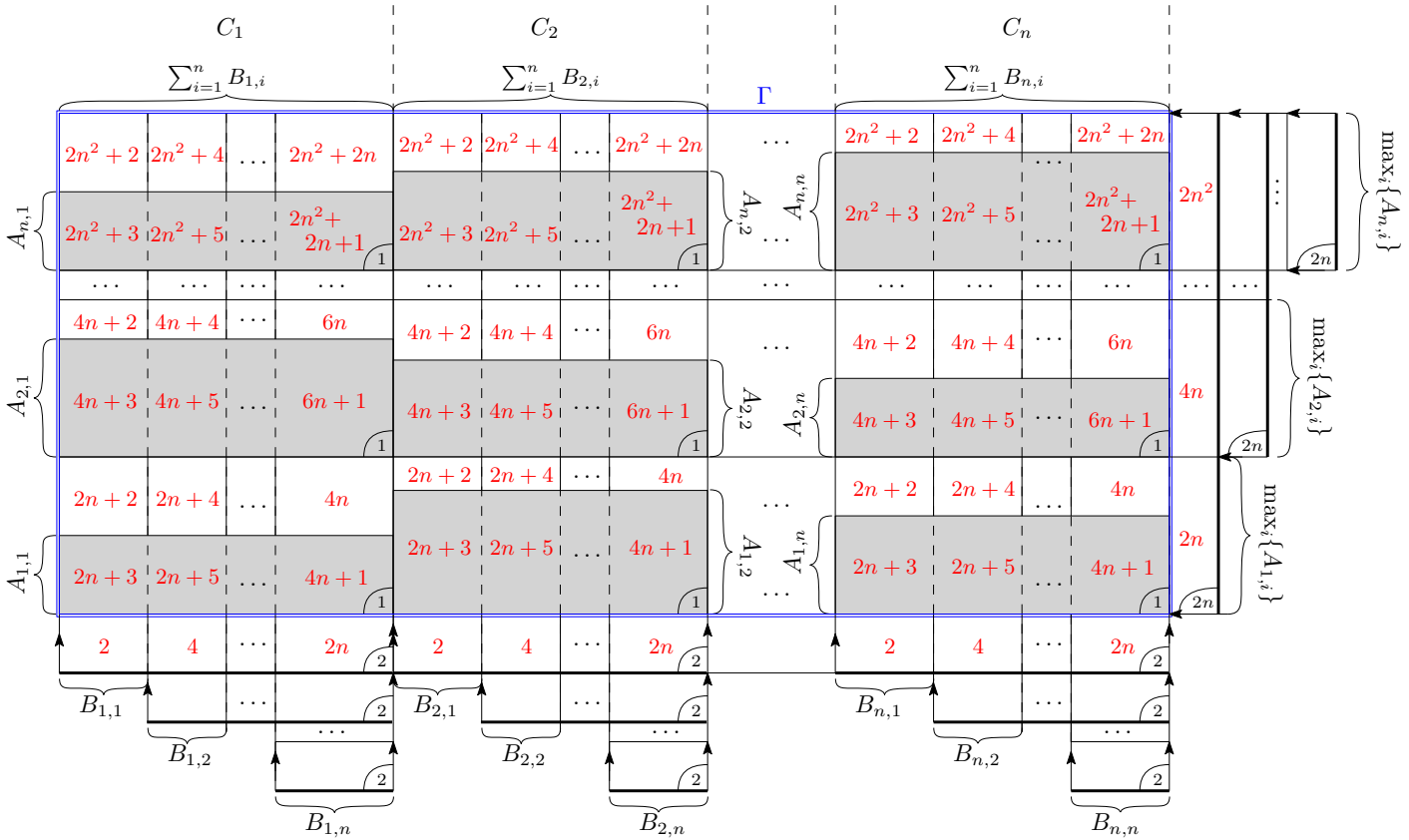



Figure 3.3: Illustration of an instance of Depth Distribution generated for the product $AB$. The values (in red) inside each rectangular denote the depth of the respective region they are in. The small arrows indicate that the boxes they delimit span the entire domain in the direction they point to. The small numbers in the corner of each box indicate the number of exact copies of the box added to the instance (intuitively, the weight of the box). The double-lined (blue) box is the domain $\Gamma$. Finally, the numbers over curly brackets indicate the length of the region delimited by the brackets.

# Chapter 4

# Matching Colored Points with Rectangles

In this chapter, we study the computation of monochromatic and bichromatic strong matchings of a set $S$ of colored points with axis-aligned rectangles. We provide two polynomial-time 4-approximation algorithms. We show that the problem of finding monochromatic matchings is NP-hard even if either the matching rectangles are restricted to axis-aligned segments or $S$ is in general position, (i.e., no two points of $S$ share the same $x$ or $y$ coordinate). We further show that in the bichromatic case, the problem is also NP-hard, even if $S$ is in general position. Additionally, we prove that it is NP-complete to decide the existence of a perfect matching with rectangles in the case where all the points have the same color.

## 4.1 Introduction

In Chapter 2 we introduced the Maximum Monochromatic Rectangle Matching (MonoMRM) and Maximum Bichromatic Rectangle Matching (BicMRM) problems, and explored the relation between these problems and the Maximum Independent Set of Rectangles (MISR) problem. We repeat here their definitions:

> Pb Maximum Monochromatic Rectangle Matching (MonoMRM): Given a set $S = R \cup B$ of red and blue points in $\mathbb{R}^2$, find a monochromatic strong matching of $S$ with the maximum number of axis-parallel rectangles.

> Pb Maximum Bichromatic Rectangle Matching (BicMRM): Given a set $S = R \cup B$ of red and blue points in $\mathbb{R}^2$, find a bichromatic strong matching of $S$ with the maximum number of axis-parallel rectangles.

Pb Maximum Independent Set of Rectangles (MISR): Given a set $\mathcal{B}$ of $n$ axis-parallel rectangles, find a subset of pairwise non-intersecting rectangles (namely, an **independent set**) of **maximum** cardinality.

Being both the MonoMRM and BicMRM problems special cases of the MISR problem (as seen in Section 2.2.1), the main question arising from this relation is whether the instances of the MonoMRM and BicMRM problems are simpler than those of MISR: are these problems solvable in polynomial time, or their solutions simpler to approximate?

There are indeed special cases of the MISR problem which are easier to either solve optimally or approximate. We described various such cases in Section 2.2.3, some of which are also special cases of the MonoMRM and BicMRM problems. However, in general it was unknown whether optimal monochromatic/bichromatic matchings of points with rectangles could be computed in polynomial time, or approximated by factors better than those achieved by approximation algorithms for the MISR problem (see Section 2.2.2 for an overview).

We show that both problems are NP-hard, even under various restricted settings: we show that the MonoMRM problem remains NP-hard even if the matching rectangles are restricted to axis-aligned segments (Theorem 4.1), or if the points are required to be in general position (Theorem 4.3), or if all the points have the same color (Theorem 4.5). The NP-hardness result under the uncolored setting solves an open question posed by Bereg et al. [32] almost 10 years ago. For BicMRM, we show that the problem remains NP-hard even if the points are in general position (Theorem 4.6).

In Section 4.3, we complement these results by providing approximation algorithms for the MonoMRM and BicMRM problems. We extend the class of instances of MonoMRM and BicMRM known to be solvable in polynomial time (Lemma 4.7), generalizing a result by Soto and Telha [124]. We use this result as the core of two polynomial time 4-approximation algorithms for MonoMRM (Theorem 4.10) and BicMRM (Theorem 4.11), respectively.

## 4.2 Hardness Results

We prove that the MonoMRM and BicMRM problems are NP-hard, even if further conditions are assumed. To this end, we consider the following decision problems:

Pb Perfect Monochromatic Rectangle Matching (MonoPRM): Given a set $S = R \cup B$ of red and blue points in $\mathbb{R}^2$, find whether there is a perfect monochromatic strong matching of $S$ with rectangles.

Pb Perfect Bichromatic Rectangle Matching (BicPRM): Given a set $S = R \cup B$ of red and blue points in $\mathbb{R}^2$, find whether there is a perfect bichromatic strong matching of $S$ with rectangles.

Proving that the MonoPRM and BicPRM problems are NP-complete implies that the MonoMRM and the BicMRM problems are NP-hard. First, in Section 4.2.1, we prove that the MonoPRM problem is NP-complete. Then, using a slightly modified version of this proof,

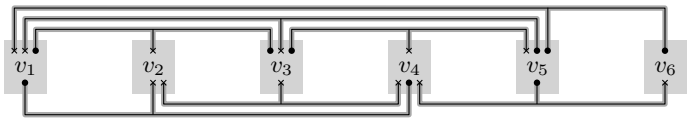



Figure 4.1: Planar embedding of the formula $\varphi = (v_1 \vee \overline{v_2} \vee v_3) \wedge (v_3 \vee \overline{v_4} \vee \overline{v_5}) \wedge (\overline{v_1} \vee \overline{v_3} \vee v_5) \wedge (v_1 \vee \overline{v_2} \vee v_4) \wedge (\overline{v_2} \vee \overline{v_3} \vee \overline{v_4}) \wedge (\overline{v_4} \vee v_5 \vee \overline{v_6}) \wedge (\overline{v_1} \vee v_5 \vee v_6)$. The crosses and dots at the end of the clause legs indicate that the connected variable appears in the clause negated or not, respectively.

we show in Section 4.2.2 that BicPRM problem is also NP-complete, and consider distinct restricted cases of both the MonoPRM and BicPRM problems.

### 4.2.1 Hardness of the MonoMRM problem

To prove that the MonoPRM problem is NP-hard (and thus that MonoMRM is NP-hard) we show that the Planar 1-in-3 SAT problem (a classical NP-complete problem [110]) can be solved by reducing its instances in polynomial time to instances of the MonoPRM problem.

> **Pb** Planar 1-in-3 SAT problem: Given a Boolean formula in 3-CNF whose incidence graph[1]is planar, find whether there is an assignment which satisfies exactly one literal of each clause in the formula.

Let $\varphi$ be a planar 3-SAT formula. The (planar) incidence graph of $\varphi$ can be represented in the plane as in Figure 4.1, where all variables lie on a horizontal line, and all clauses are represented by *non-intersecting* three-legged combs [90]. This representation can be computed in time polynomial in the size of $\varphi$ [90]. We refer to such a representation of $\varphi$ as the *planar embedding* of $\varphi$. Based on the planar embedding of $\varphi$ we construct a set $S$ of points in a polynomial-size grid, such that $S$ admits a perfect monochromatic strong matching with rectangles if and only if $\varphi$ is satisfiable. We proof this result in Theorem 4.1.

**Theorem 4.1.** *The MonoPRM problem is* NP*-complete, even in the special case where the matching objects are restricted to straight segments.*

*Proof.* Given a combinatorial matching of $S$, certifying that such a matching is monochromatic, strong, and perfect can be done in polynomial time. Then, the MonoPRM problem is in NP. We prove now that the MonoPRM problem is NP-hard.

Intuitively, given a planar 3-SAT formula $\varphi$, we construct a point-set $S = S_1 \cup S_2$, such that: (1) the elements of $S_2$ will be placed so that certain pairs of points in $S_1$ can only be matched using axis-aligned segments, and in a predefined way; (2) there always exists a perfect matching with segments for the elements of $S_2$ independently of $S_1$; and (3) there exists a perfect matching with segments for $S_1$ independently of $S_2$ if and only $\varphi$ is 1-in-3-satisfiable. For an overview of this construction of $S$, refer to Figure 4.2. We use variable gadgets (the dark-shaded rectangles called variable rectangles) and clause gadgets (the light-shaded orthogonal polygon representing the three-legged comb).

[1]The *incidence graph* is the bipartite graph with vertices the variables and the clauses, and there exists an edge between a variable and a clause if and only if the variable participates in the clause.

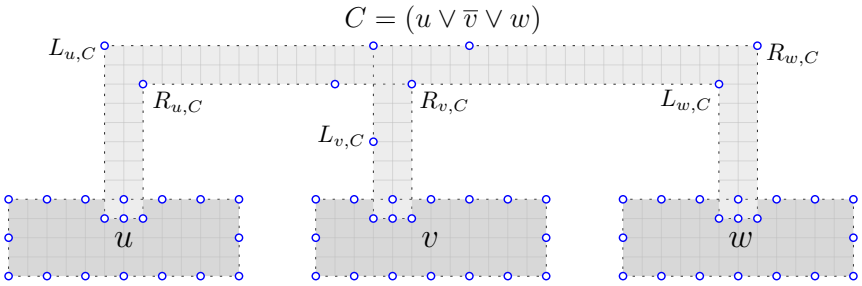



Figure 4.2: The variable gadgets and the clause gadgets. In the figure, each variable $u$, $v$, $w$ might participate in other clauses.

**Variable gadgets:** For each variable $v$, its rectangle $Q_v$ has height 4 and width $6 \cdot d(v)$, where $d(v)$ is the number of clauses in which $v$ appears. We assume that each variable appears in every clause at most once. Along the boundary of $Q_v$, starting from a vertex, we put blue points with integer coordinates so that every two successive points are at distance 2 from each other. We number consecutively in clockwise order these $4 + 6 \cdot d(v)$ points, starting from the top-left vertex of $Q_v$ which is numbered 1.

**Clause gadgets:** Let $C$ be a clause with variables $u$, $v$, and $w$, appearing in this order from left to right in the embedding of $\varphi$. Assume w.l.o.g. that the gadget of $C$ is above the horizontal line through the variables. Every leg of the gadget of $C$ overlaps the rectangles of its corresponding variable (denoted $x$) in a rectangle $Q_{x,C}$ of height 1 and width 2, so that the midpoint of the top side of $Q_{x,C}$ is a blue point in the boundary of $Q_x$. The overlapping satisfies that such a midpoint is numbered with an even number if and only if $x$ appears positive in $C$. We further put three blue points equally spaced at distance 1 in the bottom side of $Q_{x,C}$, and other nine blue points in the boundary of the gadget, as shown in Figure 4.2. Among these nine points, for $x \in \{u, v, w\}$, let $R_{x,C}$ denote the blue point in the right side of the vertical leg corresponding to $x$ in gadget of $C$, and $L_{x,C}$ the bottommost blue point in the left side (see Figure 4.2).

**Forcing convenient matchings of the blue points:** We add a polynomial number of red points to force that, for any pair $a \neq b$ of blue points, if $D(a, b)$ is not a segment of a dotted line (see Figure 4.2) then $D(a, b)$ contains a colored point in its interior[2]. Thus, two blue points $a$ and $b$ can be matched if and only if $D(a, b)$ is a segment of any dotted line and does not contain any other colored point (we call *forbidden matching rectangle* to any rectangle $D(a, b)$ not meeting these two restrictions). This can be done as follows: first, scale the blue point-set (multiplying by 2) so that every element has even $x$- and $y$-coordinates; then, add a quadratic number of red points, each with at least one odd coordinate so that the above condition can be ensured; finally, make a copy of the scaled red points, translated half a unit downwards, and scale again all the points (the blue and the red ones).

[2]If $D(a, b)$ is a box then its interior is the interior of the box. Otherwise, if $D(a, b)$ is a segment, then its interior is the set $D(a, b) \setminus \{a, b\}$.

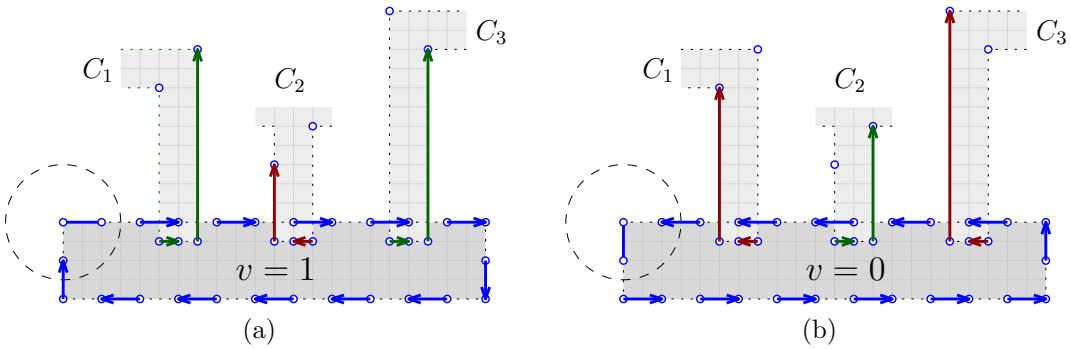



Figure 4.3: The variable $v$ appears positive in the clauses $C_1$ and $C_3$, and appears negative in the clause $C_2$. (a) If the blue point at the top-left vertex of $Q_v$ is matched to the right (i.e. $v = 1$), then each of $R_{v,C_1}$, $L_{v,C_2}$, and $R_{v,C_3}$ is matched with a point in $Q_v$, since $v = 1$ satisfies both $C_1$ and $C_3$, but not $C_2$. (b) If the blue point at the top-left vertex of $Q_v$ is matched downwards (i.e. $v = 0$), then each of $L_{v,C_1}$, $R_{v,C_2}$, and $L_{v,C_3}$ is matched with a point in $Q_v$, since $v = 0$ satisfies $C_2$, but neither $C_1$ nor $C_3$. In both (a) and (b), the arrows are matching segments. Each arrow represents the fact that the blue point at the source vertex needs to be matched with the blue point at the target one, due to the match inside the dashed circle which is the first one that was made.

**Intuition:** Consider the blue point at the top-left vertex of the rectangle $Q_v$ of variable $v$. This point can be matched only with either the blue point immediately to its right or the blue point immediately below. If we decide to match this point as in the first case (see Figure 4.3a), then we consider that $v = 1$. Otherwise, if we match as in the second case, we consider that $v = 0$ (see Figure 4.3b). Then, trying to find a maximum matching, this decision *propagates* a matching of the other blue points in the boundary of $Q_v$, as shown in Figures 4.3a and 4.3b. Furthermore, if the value of $v$ satisfies some clause $C$, it induces to match the point $R_{v,C}$ with the point in the bottom-right vertex of $Q_{v,C}$. Otherwise, the point $L_{v,C}$ is induced to be matched with the point in the bottom-left vertex of $Q_{v,C}$. Let $C$ be a clause with variables $u$, $v$, and $w$. It can be verified that all the blue points in the gadget of $C$ can be matched (throughout the matching that starts with the decisions made at the top-left vertices of $Q_u$, $Q_v$, and $Q_w$, which propagate to the other blue points) if and only if precisely one of $u$, $v$, and $w$ satisfies $C$. This statement is described in Figure 4.4.

**Reduction:** Based on the intuition, observe that in each variable $v$, the blue points along the boundary of $Q_v$ can be matched independently of the other points, and that they have two perfect strong matchings: the *1-matching* that matches the $i$th point with the $(i + 1)$th point for all odd $i$; and the *0-matching* that matches the $i$th point with the $(i + 1)$th one for all even $i$. In each clause $C$ in which $v$ appears, each of these two matchings induces a maximum strong matching on the blue points in the leg of the gadget of $C$ that overlaps $Q_v$, until reaching the points in the union of the three legs. We consider that variable $v = 1$ if we use the 1-matching, and consider $v = 0$ if the 0-matching is used. Let $C$ be a clause with variables $u$, $v$, and $w$; and draw perfect strong matchings on the blue points of the boundaries of $Q_u$, $Q_v$, and $Q_w$, respectively, giving values to $u$, $v$, and $w$. Notice that if exactly one

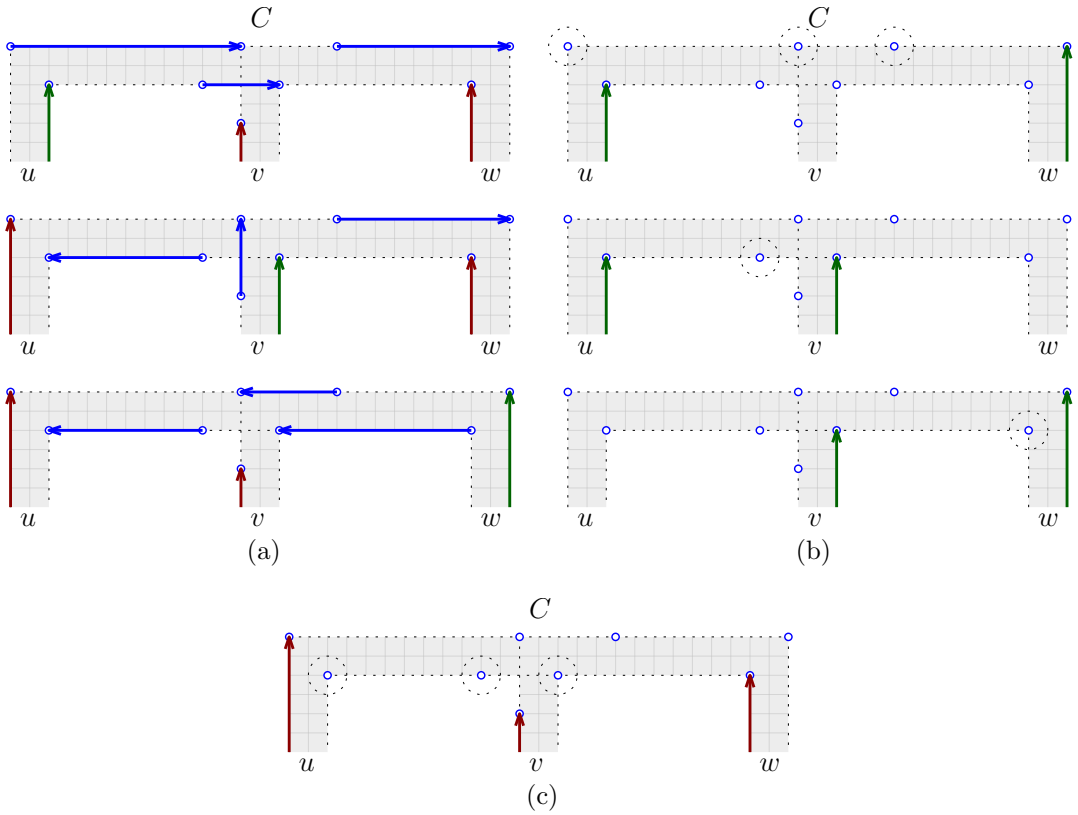



Figure 4.4: (a) If exactly one of $u$, $v$, and $w$ satisfies $C$, then all blue points in the gadget of $C$ can be matched. Note that $C$ is satisfied only by $u$ (resp. $v$, $w$) in the top (resp. middle, bottom) figure. (b) In each case (top, middle, and bottom) at least two variables among $u$, $v$, and $w$ satisfy $C$. Then, at least one of the blue points inside the dotted circles cannot be matched. (c) If none of $u$, $v$, and $w$ satisfies $C$, then one of the blue points inside a dotted circle cannot be matched.

among $u$, $v$, and $w$ makes $C$ positive, then the strong matching induced in the blue points of the gadget of $C$ is perfect (see Figure 4.5). Otherwise, if none or at least two among $u$, $v$, and $w$ make $C$ positive, then the strong matching induced on the blue points of the gadget of $C$ is not perfect since two blue points are unmatched (see Figure 4.6 and Figure 4.7). Finally, note that the red points admit a perfect strong matching with segments such that no segment contains a blue point. Therefore, we can ensure that the 3-SAT formula $\varphi$ can be satisfied if and only if the point-set $S$ admits a perfect strong matching with segments. □

### 4.2.2 Extension to BicMRM, and special cases of MonoMRM

Suppose now that the two-colored point-set $S$ is in general position. In what follows we show that the MonoPRM problem remains NP-complete under this assumption. To this end, we first perturb the two-colored point-set of the construction of the proof of Theorem 4.1 so that

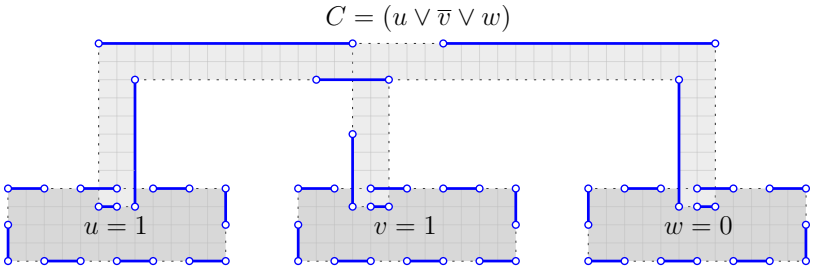



Figure 4.5: If $u = 1$, $v = 1$, and $w = 0$, then only $u$ satisfies $C$ and there exists a perfect strong matching on the blue points.

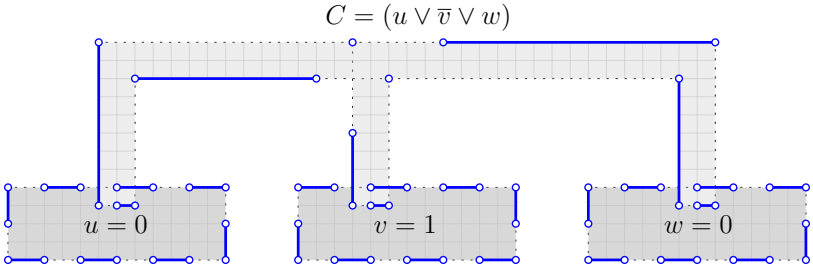



Figure 4.6: If $u = 0$, $v = 1$, and $w = 0$, then no variable satisfies $C$ and there does not exist any perfect strong matching on the blue points.

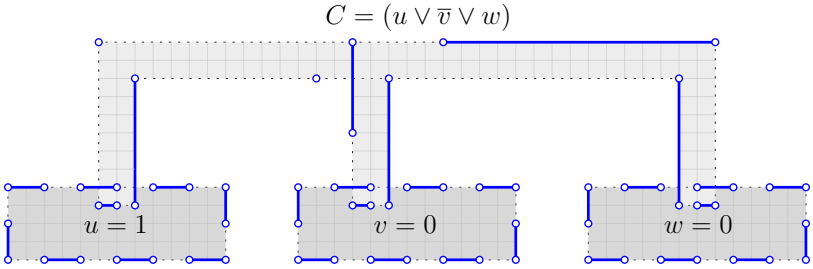



Figure 4.7: If $u = 1$, $v = 0$, and $w = 1$, then two variables satisfies $C$ and there does not exist any perfect strong matching on the blue points.

no two points share the same $x$- or $y$-coordinate, and second show that two points of $S$ can be matched in the perturbed point-set if and only if they can be matched in the original one.

Alliez et al. [14] proposed the transformation that replaces each point $p = (x, y)$ by the point $\lambda(p) = ((1+\varepsilon)x + \varepsilon^2 y, \varepsilon^3 x + y)$ for some small enough $\varepsilon > 0$, with the aim of removing the degeneracies in a point set for computing the Delaunay triangulation under the $L_\infty$ metric. Although this transformation can be used for our purpose, by using the fact that the points in the proof of Theorem 4.1 belong to a grid $[0..N]^2$, where $N$ is polynomially-bounded, we use the simpler transformation $\lambda(p) = ((1+\varepsilon)x + \varepsilon y, \varepsilon x + (1+\varepsilon)y)$ for $\varepsilon = 1/(2N+1)$, which is linear in $\varepsilon$. Both transformations change the relative positions of the initial points in the manner shown in Figure 4.8. Some useful properties of the transformation we use, stated in the next lemma, were not stated by Alliez et al. [14].

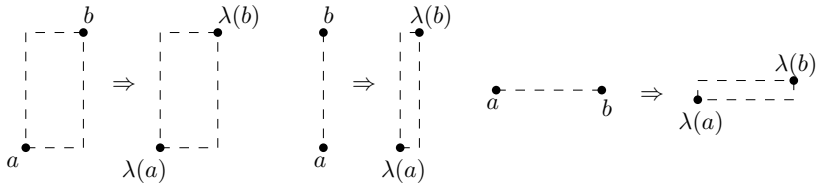



Figure 4.8: Perturbation of the point-set to put $S$ in general position.

**Lemma 4.2.** *Let $N$ be a natural number and $P \subseteq [0..N]^2$. The function $\lambda : P \to \mathbb{Q}^2$ such that*

$$\lambda(p) = \left( x(p) + \frac{x(p)+y(p)}{2N+1}, y(p) + \frac{x(p)+y(p)}{2N+1} \right)$$

*satisfies the next properties:*

(a) *$\lambda$ is injective and the point-set $\lambda(P) = \{\lambda(p) : p \in P\}$ is in general position.*

(b) *For every two distinct points $a, b \in P$ such that $x(a) = x(b)$ or $y(a) = y(b)$, we have that $D(a,b) \cap P = \{a, b\}$ if and only if $D(\lambda(a), \lambda(b)) \cap \lambda(P) = \{\lambda(a), \lambda(b)\}$.*

(c) *For every three distinct points $a, b, c \in P$ such that $x(a) \neq x(b)$ and $y(a) \neq y(b)$, we have that $c$ belongs to the interior of $D(a,b)$ if and only if $\lambda(c)$ belongs to the interior of $D(\lambda(a), \lambda(b))$.*

*Proof.* Properties (a-c) are a consequence of $0 \leq \frac{x(p)+y(p)}{2N+1} \leq \frac{2N}{2N+1} < 1$. □

**Theorem 4.3.** *The* MonoPRM *problem remains* NP*-complete on point-sets in general position.*

*Proof.* Let $S$ be the colored point-set generated in the reduction of the proof of Theorem 4.1. Let $N$ be a polynomially-bounded natural number such that $S \subset [0..N]^2$, and let $S' = \lambda(S)$, where $\lambda$ is the function of Lemma 4.2. Consider the next observations:

(a) If $a, b \in S$ are red points that can be matched in $S$ because $x(a) = x(b)$ and $y(b) = y(a) - 1$, then $\lambda(a)$ and $\lambda(b)$ can also be matched in $S'$ (Property (b) of Lemma 4.2).

(b) If $a, b \in S$ are blue points that can be matched in $S$, then we have that either $x(a) = x(b)$ or $y(a) = y(b)$, which implies that $\lambda(a)$ and $\lambda(b)$ can also be matched in $S'$ by Property (b) of Lemma 4.2.

(c) If $a, b \in S$ are blue points that cannot be matched in $S$ because $D(a,b)$ is a segment containing a point $c \in S$ in its interior, then neither $\lambda(a)$ and $\lambda(b)$ can be matched in $S'$ (Property (b) of Lemma 4.2).

(d) If $a, b \in S$ are blue points that cannot be matched in $S$ because $D(a,b)$ is a box containing a point $c \in S$ in the interior, then neither $\lambda(a)$ and $\lambda(b)$ can be matched in $S'$ since the box $D(\lambda(a), \lambda(b))$ contains $\lambda(c)$ (Property (c) of Lemma 4.2).

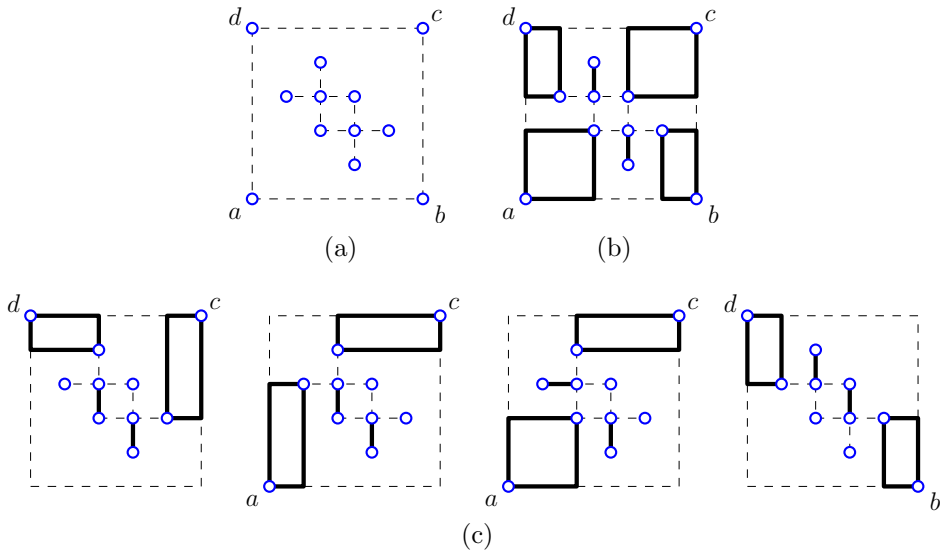



Figure 4.9: (a) The point-set $M_1 \cup M_2$. (b) A perfect strong matching of $M_1 \cup M_2$. (c) If exactly two points among $a, b, c, d$ are removed, then the remaining points do not have any perfect strong matching.

The above observations imply that there exists a perfect strong rectangle matching in $S$ if and only if it exists in $S'$. The result thus follows since $S'$ is in general position by Property (a) of Lemma 4.2. □

Combining the construction of Theorem 4.1 with the perturbation of Lemma 4.2, we can prove that the MonoPRM problem is also NP-complete when all points have the same color, and that the BicPRM problem is also NP-complete.

**Lemma 4.4.** *Let* $M_1 = \{(0,0),(5,0),(5,5),(0,5)\}$ *and* $M_2 = \{(1,3),(2,2),(2,3),(2,4),(3,1),(3,2),(3,3),(4,2)\}$ *be two point-sets. The point-set* $M_1 \cup M_2$ *has a perfect strong matching with rectangles, and for every proper subset* $M_1' \subset M_1$ *the point-set* $M_1' \cup M_2$ *does not have any perfect strong matching with rectangles.*

*Proof.* The proof is straightforward (see Figure 4.9a, Figure 4.9b, and Figure 4.9c). □

**Theorem 4.5.** *The MonoPRM problem remains* NP*-complete even if all elements of* $S$ *have the same color.*

*Proof.* Let $R$ and $B$ be the sets of red and blue points, respectively, in the proof of Theorem 4.1. The red points in that proof were used to block the forbidden matching rectangles. In the uncolored setting, we need another strategy to block such rectangles: we use scaled versions of the set $M_1 \cup M_2$ of Lemma 4.4. Let $Q$ be a set of (artificial) *green* points constructed so that, for every two points $p, q \in B$, we have that $D(p,q)$ contains elements of $B \cup Q$ in its interior if and only if $D(p,q)$ is a forbidden matching rectangle in $R \cup B$. In other words, $p, q$ can be matched in $R \cup B$ if and only if they can be matched in $B \cup Q$. The

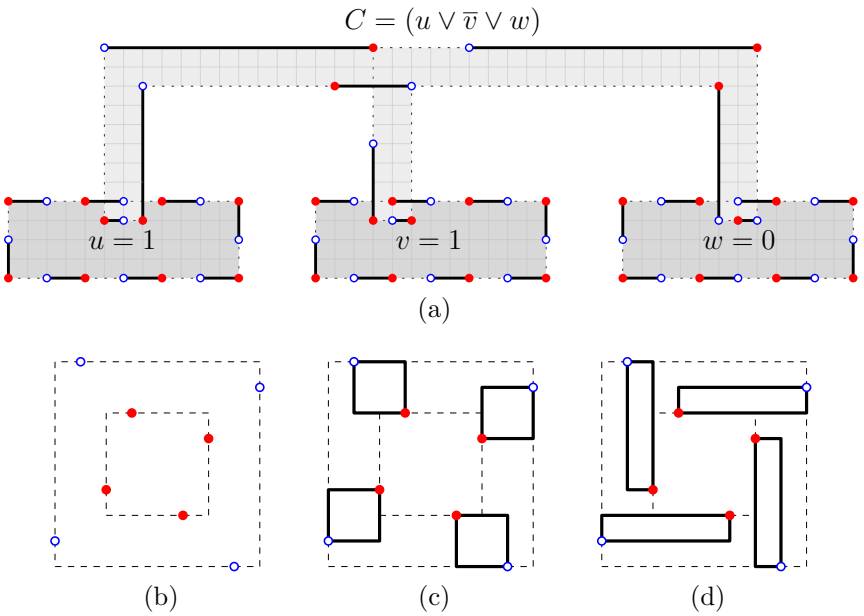



Figure 4.10: Proof of Theorem 4.6. (a) Changing the colors of the blue points in the gadgets of the proof of Theorem 4.1. (b) The eight points (close enough one another) that replace each green point. (c) One of the only two ways to match the points corresponding to a green point in order to obtain a perfect matching. (d) The other way.

point-set $\mathcal{S}_1 = B \cup Q$ belongs to a grid of the form $[0..N]^2$, where $N$ is polynomially-bounded in $|R| + |B|$, and is not in general position. Let $\mathcal{S}_2 = \lambda(\mathcal{S}_1)$, where $\lambda$ is the function of the Lemma 4.2. We now replace each green point $g$ of $Q$ by a translated and properly stretched copy $S_g$ of the set $M_1 \cup M_2$ of Lemma 4.4, with all elements colored blue (see Figure 4.9a). Let $\mathcal{S} = \lambda(B) \cup (\bigcup_{g \in Q} S_g)$. Putting the elements of $S_g$ close enough one another for every $g$, we can guarantee by Lemma 4.4 that, if we want to obtain a perfect strong matching in $\int$, then we must have a perfect strong matching in each $S_g$ in particular (see Figure 4.9b). Therefore, the set $S_g$ acts as the green point $g$ blocking the forbidden matching rectangles in $B$. The construction of $\mathcal{S}$ starts from the planar 3-SAT formula $\varphi$ of the proof of Theorem 4.1, and using all the above arguments, we can claim that there exists a perfect strong matching in $\mathcal{S}$ if and only if the formula $\varphi$ is satisfiable. Hence, the MonoPRM problem with input points of the same color is NP-complete since there exists a polynomial-time reduction from the Planar 1-in-3 SAT problem. □

**Theorem 4.6.** *The BicPRM problem is* NP*-complete, even if the points in the set $S$ are in general position.*

*Proof.* Let $R$ and $B$ be the sets of red and blue points, respectively, in the proof of Theorem 4.1. As in the proof of Theorem 4.5, we need to a new strategy to fulfill the role of the points in $R$ (i.e., block the forbidden matching rectangles). This time we use a scaled and colored versions of the set $M_1 \cup M_2$ of Lemma 4.4. But first, we must deal with the matching rectangles:

change, as illustrated in Figure 4.10a, the color of some elements of $B$ to red to obtain a colored point-set $\mathcal{S}_0$ such that, for every segment matching two blue points in $R \cup B$, the color of exactly one of the matched points is changed to red. This can be done because the number of points in each clause and variable gadget is even. For every point $p \in B$, let $p'$ denote the corresponding point in $\mathcal{S}_0$, and vice versa. Let $Q$ be a set of (artificial) green points constructed so that for every two distinct points $p', q' \in \mathcal{S}_0$ we have that $D(p', q')$ contains elements of $\mathcal{S}_0 \cup Q$ in its interior if and only if $D(p, q)$ is a forbidden matching rectangle in $R \cup B$. The point-set $\mathcal{S}_1 = S_0 \cup Q$, which is not in is not in general position, belongs to the grid of the form $[0..N]^2$, where $N$ is polynomially-bounded. Let $\mathcal{S}_2 = \lambda(\mathcal{S}_1)$, where $\lambda$ is the function of the Lemma 4.2. We now replace each green point $g$ of $Q$ by the set $S_g$ of eight red and blue points in general position (see Figure 4.10b). Let $\mathcal{S} = \lambda(\mathcal{S}_0) \cup (\bigcup_{g \in Q} S_g)$. Putting the elements of $S_g$ close enough one another for every $g$, we can guarantee that $\mathcal{S}$ is also in general position and that for every $g$ the points of $S_g$ appear together in both the left-to-right and the top-down order of $S$. This last condition ensures that if we want to obtain a perfect strong matching in $S$ then we must have a perfect strong matching for each $S_g$ in particular (see Figure 4.10c and Figure 4.10d) because for all $g$ every red point of $S_g$ cannot be matched with any blue point not in $S_g$. Therefore, the set $S_g$ acts as the green point $g$ blocking the forbidden matching rectangles in $S_0$. The BicPRM problem is thus NP-complete, even on points in general position. □

## 4.3 Approximation algorithms

In the previous section we showed that both the MonoMRM and the BicMRM problems are NP-hard, like the more general MISR problem. However, we show that their optimal solutions are easier to approximate than that of the MISR problem: while no constant-approximation algorithm is known to this date for the MISR problem, we present in this section two 4-approximation algorithms for the MonoMRM and the BicMRM problems, respectively. First, we start by considering a special case of these problems that can be solved optimally in polynomial time. We then show this result to obtain the mentioned approximation algorithms for the MonoMRM and the BicMRM problems.

### 4.3.1 A special case solvable in polynomial time

Given a point-set $P$ in the plane, we say that $\mathcal{H}$ is a *set of rectangles on* $P$ if every element of $\mathcal{H}$ is of the form $D(a, b)$, where $a, b \in P$ and $D(a, b)$ contains exactly the points $a$ and $b$ of $P$. As mentioned in Section 2.2.3, in such a set of rectangles only four types of intersections can occur: piercing, corner, point and side (see Figure 4.11 for an illustration, and refer to Definition 2.7 for a detailed definition). We say that the set $\mathcal{H}$ is *corner-complete* if for every pair of elements $D(a, b)$ and $D(a', b')$ of $\mathcal{H}$ that have a corner intersection, the two rectangles $D(a, b')$ and $D(a', b)$ also belong to $\mathcal{H}$. Let $G_{p,c}(\mathcal{H})$ denote the spanning subgraph of $G(\mathcal{H})$ with edge set the edges that correspond to the piercing and the corner intersections.

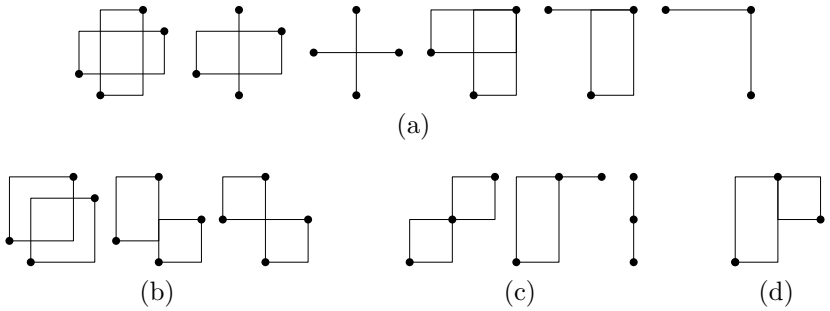



Figure 4.11: The four types of intersection: (a) piercing; (b) corner; (c) point; and (d) side.

**Lemma 4.7.** *Let $P$ be a point-set and $\mathcal{H}$ be any corner-complete set of rectangles on $P$. A maximum independent set of $G_{p,c}(\mathcal{H})$ can be found in polynomial time.*

*Proof.* Given finite point-sets $X$ and $Y$ in the plane, Soto and Telha [124] showed how to find in polynomial time a maximum independent set and a minimum hitting set of a corner-complete set $\mathcal{R}(X,Y)$ of rectangles on $X \cup Y$, where each rectangle has an element of $X$ as bottom-left corner, and an element of $Y$ as top-right corner. A *hitting set* is a finite point-set $H$ such that each rectangle of $\mathcal{R}(X,Y)$ contains at least one of the points of $H$. They noted that the rectangles of $\mathcal{R}(X,Y)$ have only two types of intersections, piercing and corner. Their overall algorithm and arguments are the following ones (see Section 4 of [124]):

1. Sort the rectangles $\mathcal{R}(X,Y)$ in *right-top* order: the rectangle $D(a,b)$ is before the rectangle $D(a',b')$ if and only if $x(a) < x(a')$, or $x(a) = x(a')$ and $y(b) < y(b')$.

2. Construct a subset $\mathcal{K} \subseteq \mathcal{R}(X,Y)$ by processing the rectangles of $\mathcal{R}(X,Y)$ in right-top order and adding only those that keep $\mathcal{K}$ free of corner intersections.

3. Using that only piercing intersections are possible in $\mathcal{K}$, compute in polynomial time a maximum independent set $\mathcal{R}_0$ and a minimum hitting set $H^\star$ for $\mathcal{K}$, which always satisfy $|\mathcal{R}_0| = |H^\star|$ since $G(\mathcal{K})$ is perfect.

4. Prove that $H^\star$ is also a hitting set of $\mathcal{R}(X,Y)$ [124, see Lemma 1], which implies that $\mathcal{R}_0$ is a maximum independent set of $\mathcal{R}(X,Y)$.

We extend steps (1-3) to find a maximum independent set of $G_{p,c}(\mathcal{H})$. For this, first partition $\mathcal{H}$ into the following three sets: $\mathcal{H}_0$, $\mathcal{H}_1$, and $\mathcal{H}_2$. The set $\mathcal{H}_0$ contains all the segments; $\mathcal{H}_1$ contains the boxes $D(a,b)$ such that $y(a) < y(b)$; and $\mathcal{H}_2$ contains the boxes $D(a,b)$ such that $y(a) > y(b)$. By definition, for every box $D(a,b)$ we have $x(a) < x(b)$. Sort the boxes of $\mathcal{H}_1$ by using the right-top order of Step 1. Then, one can construct the subset $\mathcal{K}_1 \subseteq \mathcal{H}_1$, having no corner intersection, by processing the boxes of $\mathcal{H}_1$ in such an order, and adding to $\mathcal{K}_1$ only those that keep $\mathcal{K}_1$ free of corner intersections. Similarly and using symmetry, one can construct the subset $\mathcal{K}_2 \subseteq \mathcal{H}_2$, having no corner intersection, by processing the boxes of $\mathcal{H}_2$ in the following order: the box $D(a,b)$ is before the box $D(a',b')$ if and only if $x(a) < x(a')$, or $x(a) = x(a')$ and $y(b) > y(b')$. By construction, and the fact that a box of

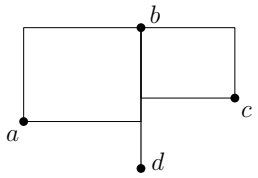



Figure 4.12: The set $\mathcal{K} = \{D(a,b), D(b,c), D(d,b)\}$ is a corner-complete set of rectangles on $\{a,b,c,d\}$, and free of corner intersections. The maximum independent set of $G_{p,c}(\mathcal{K})$ is the set $\{D(a,b), D(b,c)\}$, whereas the minimum hitting set has cardinality one.

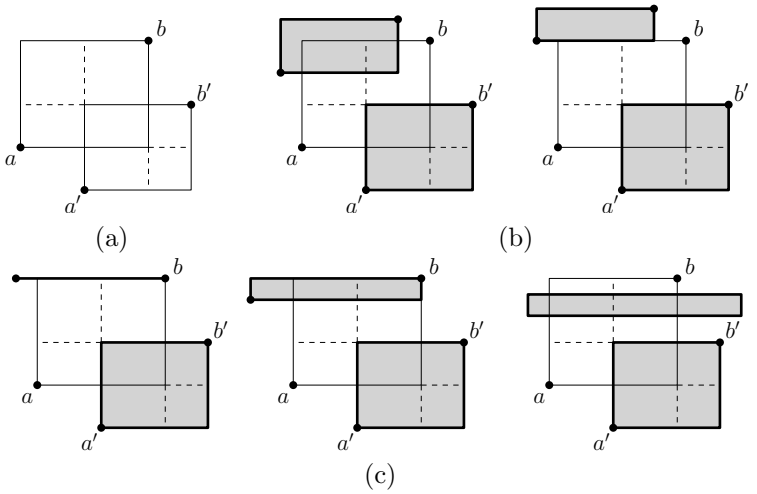



Figure 4.13: (a,b,c) Cases in the proof of Lemma 4.7.

$\mathcal{H}_1$ and a box of $\mathcal{H}_2$ cannot realize a corner intersection, and a segment of $\mathcal{H}_0$ cannot realize corner intersections with any other rectangle of $\mathcal{H}$, the set $\mathcal{K} = \mathcal{H}_0 \cup \mathcal{K}_1 \cup \mathcal{K}_2 \subseteq \mathcal{H}$ is free of corner intersections. A maximum independent set of $G_{p,c}(\mathcal{K})$ can be found in polynomial time as done in Step 3 since $\mathcal{K}$ is free of corner intersections.

To show that a maximum independent set of $G_{p,c}(\mathcal{K})$ is indeed a maximum independent set of $G_{p,c}(\mathcal{H})$, the arguments given in Step 4 cannot directly be applied. A reason for this is the following: Pairs of rectangles in $\mathcal{H}$ having a point or a side intersection can be hit by the same point, whereas they can appear in $\mathcal{K}$ without being adjacent in $G_{p,c}(\mathcal{K})$ (see Figure 4.12 for an example). Then, it cannot be guaranteed that the cardinality of a maximum independent set of $G_{p,c}(\mathcal{K})$ equals the cardinality of a minimum hitting set of $\mathcal{K}$. We then adapt the arguments of Step 4 to work directly on independent sets of rectangles, not using hitting sets.

We will prove that any box $B \in I$ that is not in $\mathcal{K}_1 \cup \mathcal{K}_2$ can be *replaced* by some suitable rectangle $R$, obtaining an independent set of $G_{p,c}(\mathcal{K})$ with cardinality $|I|$. Suppose that $B$ belongs to $\mathcal{H}_1$ (the case where $B$ belongs to $\mathcal{H}_2$ is analogous). Box $B$ is replaced by a rectangle $R$ that satisfies either of the following two conditions:

(i) $R$ is in $\mathcal{H}_0 \cup \mathcal{K}_1$.

(ii) $R$ is not in $\mathcal{H}_0 \cup \mathcal{K}_1$, but appears after $B$ in the right-top order defined for $\mathcal{H}_1$.

Hence, by applying this replacement procedure iteratively for boxes of $I$ that are not in $\mathcal{K}_1 \cup \mathcal{K}_2$, we will obtain at the end an independent set of size $|I|$ consisting only of elements from $\mathcal{H}_0 \cup \mathcal{K}_1 \cup \mathcal{K}_2$. The replacement can be done as follows. Assume that a box $B = D(a', b') \in \mathcal{H}_1 \setminus \mathcal{K}_1$ belongs to $I$. Then, by construction of $\mathcal{K}_1$, there exists a box $D(a, b) \in \mathcal{K}_1$ with $x(a) < x(a')$ that has a corner intersection with $D(a', b')$, and it can be assumed that $D(a, b)$ is the first box in the order of $\mathcal{H}_1$ that satisfies this condition (see Figure 4.13a). Since $\mathcal{H}$ is corner-complete, and the election of $D(a, b)$, the rectangle $D(a, b')$ belongs to $\mathcal{H}_0 \cup \mathcal{K}_1$. The rectangle $D(a', b)$ either belongs to $\mathcal{H}_0 \cup \mathcal{K}_1$, or does not belong to $\mathcal{H}_0 \cup \mathcal{K}_1$ (i.e. belongs to $\mathcal{H}_1 \setminus \mathcal{K}_1$ ) and appears after $D(a', b')$ in the order of $\mathcal{H}_1$. Now, we only need to prove that $D(a', b')$ can be replaced in $I$ by either $D(a, b')$ or $D(a', b)$. In any case, one of conditions (i) and (ii) is ensured. That is, we need to prove that

$$(I \setminus \{D(a', b')\}) \cup \{D(a', b)\} \quad \text{or} \quad (I \setminus \{D(a', b')\}) \cup \{D(a, b')\}$$

is also an independent set of $G_{p,c}(\mathcal{H})$. Indeed, if $(I \setminus \{D(a', b')\}) \cup \{D(a', b)\}$ is an independent set of $G_{p,c}(\mathcal{H})$, then we are done. Otherwise, at least one of the next two cases is satisfied: (1) there is a rectangle of $I \setminus \{D(a', b')\}$ that has a corner intersection with both $D(a, b)$ and $D(a', b)$ (see Figure 4.13b); or (2) there is a rectangle (either a box or a segment) of $I \setminus \{D(a', b')\}$ that has a piercing intersection with both $D(a, b)$ and $D(a', b)$ (see Figure 4.13c). In both cases $D(a, b')$ is independent in $G_{p,c}(\mathcal{H})$ from any rectangle in $I \setminus \{D(a', b')\}$. Hence, $(I \setminus \{D(a', b')\}) \cup \{D(a, b')\}$ is an independent set of $G_{p,c}(\mathcal{H})$. This completes the proof. □

### 4.3.2 Approximation algorithms

Let $S = R \cup B$ be a two-colored point-set in the plane. Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be the next two subsets of rectangles of $\mathcal{R}(S)$ (see Figure 4.14):

- $\mathcal{R}_1$ contains the blue rectangles $D(a, b) \in \mathcal{R}(S)$ such that $y(a) \leq y(b)$ (i.e. the rectangles with bottom-left corner a blue point), and the red rectangles $D(a', b') \in \mathcal{R}(S)$ such that $y(a') \geq y(b')$ (i.e. the rectangles with bottom-right corner a red point).
- $\mathcal{R}_2$ contains the blue rectangles $D(a, b) \in \mathcal{R}(S)$ such that $y(a) \geq y(b)$ (i.e. the rectangles with bottom-right corner a blue point), and the red rectangles $D(a', b') \in \mathcal{R}(S)$ such that $y(a') \leq y(b')$ (i.e. the rectangles with bottom-left corner a red point).

The following lemma will be used in the proof of Lemma 4.9.

**Lemma 4.8.** *Let $H_1$ and $H_2$ be independent sets of $G_{p,c}(\mathcal{R}_1)$ and $G_{p,c}(\mathcal{R}_2)$, respectively. The graphs $G(H_1)$ and $G(H_2)$ are acyclic.*

*Proof.* We prove the lemma for $G(H_1)$. The proof for $G(H_2)$ is analogous. Note that in $H_1$ every blue rectangle is independent from every red rectangle, and rectangles of the same color can have point intersections only. For every $k \geq 1$, every single path of length $k$ in $G(H_1)$

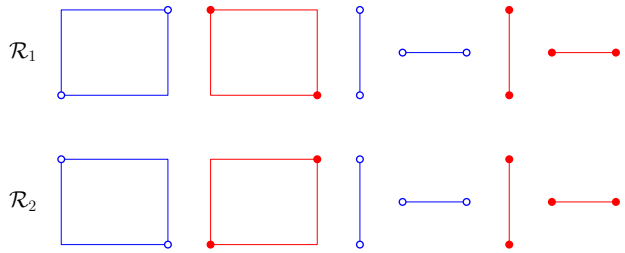



Figure 4.14: The types of rectangles in the subsets $\mathcal{R}_1$ and $\mathcal{R}_2$.

is a sequence $\langle D(a_0, a_1), D(a_1, a_2), D(a_2, a_3), \ldots, D(a_k, a_{k+1})\rangle$ of rectangles of $H_1$, with point intersections between consecutive rectangles, such that: $x(a_0) \leq x(a_1) \leq \cdots \leq x(a_{k+1})$, and $y(a_0) \leq y(a_1) \leq \cdots \leq y(a_{k+1})$ if the rectangles $D(a_0, a_1), \ldots, D(a_k, a_{k+1})$ are all blue or $y(a_0) \geq y(a_1) \geq \cdots \geq y(a_{k+1})$ if they are red. Under these monotone properties, the graph $G(H_1)$ cannot have any cycle. □

**Lemma 4.9.** *For $\mathcal{R} \in \{\mathcal{R}_1, \mathcal{R}_2\}$, there exists a polynomial-time (1/2)-approximation algorithm for the maximum independent set of $G(\mathcal{R})$.*

*Proof.* Consider the set $\mathcal{R}_1$, the arguments for the set $\mathcal{R}_2$ are analogous. Let $\mathsf{OPT}_1$ denote the size of a maximum independent set in $\mathcal{R}_1$. Observe that a blue and a red rectangle in $\mathcal{R}_1$ can have only a piercing intersection, that two rectangles of the same color cannot have a side intersection, and that $\mathcal{R}_1$ is a corner-complete set of rectangles on $S$. Let $H_1$ denote a maximum independent set of $G_{p,c}(\mathcal{R}_1)$, which can be found in polynomial time by Lemma 4.7. The graph $G(H_1)$ is acyclic (Lemma 4.8) and thus 2-colorable. Such a 2-coloring of $G(H_1)$ can be found in polynomial time and gives an independent set $I_1$ of $H_1$ with at least $|H_1|/2$ rectangles, which is an independent set in $\mathcal{R}_1$ as well. The set $I_1$ is the approximation and satisfies $\mathsf{OPT}_1 \leq |H_1| \leq 2|I_1|$. The result thus follows. □

**Theorem 4.10.** *There exists a polynomial-time 4-approximation algorithm for the* MonoMRM *problem.*

*Proof.* Let $\mathsf{OPT}$ denote the size of a maximum independent set in $\mathcal{R}(S)$, and $\mathsf{OPT}_1$ and $\mathsf{OPT}_2$ denote the sizes of the maximum independent sets in $\mathcal{R}_1$ and $\mathcal{R}_2$, respectively. Let $I_1$ be a (1/2)-approximation for the maximum independent set in $\mathcal{R}_1$ and $I_2$ be a (1/2)-approximation for the maximum independent set in $\mathcal{R}_2$ (Lemma 4.9). The approximation for the MonoMRM problem is to return the set with maximum cardinality between $I_1$ and $I_2$. Since $\mathsf{OPT} \leq \mathsf{OPT}_1 + \mathsf{OPT}_2 \leq 2|I_1| + 2|I_2| \leq 4\max\{|I_1|, |I_2|\}$, the result follows. □

Consider now the BicMRM problem and the set $\overline{\mathcal{R}}(S)$. Let $\overline{\mathcal{R}}_1$, $\overline{\mathcal{R}}_2$, $\overline{\mathcal{R}}_3$, and $\overline{\mathcal{R}}_4$ be the next four subsets of rectangles of $\overline{\mathcal{R}}(S)$:

- $\overline{\mathcal{R}}_1$ contains the rectangles with a blue point in the bottom-left corner.
- $\overline{\mathcal{R}}_2$ contains the rectangles with a red point in the bottom-left corner.

- $\overline{\mathcal{R}}_3$ contains the rectangles with a blue point in the bottom-right corner.
- $\overline{\mathcal{R}}_4$ contains the rectangles with a red point in the bottom-right corner.

Each of the above four subsets is a corner-complete set of rectangles on $S$, where every two rectangles have either a corner or a piercing intersection. Then, the maximum independent set in each subset can be found in polynomial time (Lemma 4.7). These observations imply the next result:

**Theorem 4.11.** *There exists a polynomial-time 4-approximation algorithm for the BicMRM problem.*

In the next section we summarize the results described in this chapter and present some future directions of research on the MonoMRM and BicMRM problems.

## 4.4 Discussion

We have proved that finding a maximum strong matching of a two-colored point set, with either rectangles containing points from the same color or rectangles containing points of different colors, is NP-hard, and provided a polynomial-time 4-approximation algorithm for each case. Our results show that, although the MonoMRM and BicMRM problems are NP-hard as the more general MISR problem, their solutions can be approximated in polynomial time by better factors than those of the MISR problem.

**Extension to Higher Dimensions.** The hardness proofs that we described in Section 4.2 were presented in two dimensions. However, they can be extended easily to the higher dimensional versions of the problem, for any dimension $d$: simply create a set of points having the first two coordinates as the points of the constructions in Section 4.2, and give spurious values to the remaining $d-2$ dimensions (for instance, set them all to zero). However, there is no clear way to extend to higher dimensions the approximation algorithms described in Section 4.3. The main obstacle in such direction is that our algorithms heavily rely on the fact that only four types of intersections can occur in a set of axis-aligned rectangles, and on their well-behaved structures. However, as the dimension increases, the number of possible types of intersections increases exponentially with the dimension, and therefore also the number of cases that must be considered.

**Better Approximations.** It is still an open problem to find a better $\mathcal{O}(1)$-approximation algorithm, or a PTAS, for the MonoMRM and BicMRM problems, or to reasonably argue that no such algorithm exist. There are special cases of instances of the MonoMRM problem that can be solved optimally, or approximated by factor better than the 4-approximation described here. For instance, for the case in which all the points have the same color, Bereg et al. [32] described a 3/2-approximation algorithm running in polynomial time. It is open whether there are measures of the input instances, which arise naturally in practice, gradually

separating the ones that are easy to solve or approximate from those that are hard. For instance, one could consider the ratio between the amounts of red and blue points, and study whether one can gradually obtain approximations in the range $[3/2, 4]$ as the ratio changes from 1/2 to 1. Finally, finding a $\mathcal{O}(1)$-approximation algorithm for the general Maximum Independent Set of Rectangles problem remains an even more interesting open question.

# Chapter 5

# Computing Minimum Coverage Kernels

In this chapter, we consider the Minimum Coverage Kernel problem: given a set $\mathcal{B}$ of $d$-dimensional boxes, find a subset of $\mathcal{B}$ of minimum size covering the same region as $\mathcal{B}$. This problem is NP-hard, but as for many NP-hard problems on graphs, the problem becomes solvable in polynomial time under restrictions on the graph induced by $\mathcal{B}$. We consider various classes of graphs, show that Minimum Coverage Kernel remains NP-hard even for severely restricted instances, and provide two polynomial time approximation algorithms for this problem.

## 5.1 Introduction

In Section 2.4 we introduced the Minimum Coverage Kernel problem, and explored the relation between this problem and the Box Cover and the Orthogonal Polygon Covering problems. First, let us remember the definition of the problem:

> Pb Minimum Coverage Kernel problem: Given a set $\mathcal{B}$ of $n$ boxes in $\mathbb{R}^d$, compute a minimum-size subset $C$ of $\mathcal{B}$ covering the same region as $\mathcal{B}$.

The fact that the Minimum Coverage Kernel problem is intermediate between the Box Cover and the Orthogonal Polygon Covering problems (as seen in Section 2.4.1), yields that Minimum Coverage Kernel problem is NP-hard, and that polynomial-time approximation algorithms for the Box Cover problem can also be used for the Minimum Coverage Kernel problem. However, one could expect better polynomial-time approximations for the Minimum Coverage Kernel problem given the gap in the known approximabilities of the Box Cover and the Orthogonal Polygon Covering problems ($\mathcal{O}(\log \mathtt{OPT})$ for Box Cover vs. $\mathcal{O}(\sqrt{\log n})$ for Orthogonal Polygon Covering). We study the exact computation and approximability of Minimum Coverage Kernel in various restricted settings.

In Section 5.2, we study the computational complexity of the Minimum Coverage Kernel problem under three restrictions of the intersection graph, commonly considered for other problems [13]: planarity of the graph, bounded clique-number, and bounded vertex-degree. We show that the problem remains NP-hard even when the intersection graph of the boxes has clique-number at most 4, and the maximum vertex-degree is at most 8 (Theorem 5.1). For the Box Cover problem we show that it remains NP-hard even under the severely restricted setting where the intersection graph of the boxes is planar, its clique-number is at most 2 (i.e., the graph is triangle-free), the maximum vertex-degree is at most 3, and every point is contained in at most two boxes (Theorem 5.2).

We complement these hardness results with two approximation algorithms for the Minimum Coverage Kernel problem running in polynomial time. We describe a $\mathcal{O}(\log n)$-approximation algorithm which runs in time within $\mathcal{O}(\mathtt{OPT} \cdot n^{\frac{d+1}{2}} \log^2 n)$ (Theorem 5.7); and a randomized algorithm computing a $\mathcal{O}(\log \mathtt{OPT})$-approximation in expected time within $\mathcal{O}(\mathtt{OPT} \cdot n^{\frac{d+1}{2}} \log^2 n)$, with high probability (at least $1 - \frac{1}{n^{\Omega(1)}}$) (Theorem 5.8). Our main contribution in this matter is not the existence of polynomial time approximation algorithms (which can be inferred from results on Box Cover, see Section 2.4.2 for details), but a new data structure (Theorem 5.6) which allows to significantly improve the running time of finding those approximations (when compared to the approximation algorithms for Box Cover). This is relevant in applications where a minimum coverage kernel needs to be computed repeatedly [10, 56, 95, 118].

## 5.2 Hardness under Restricted Settings

We prove that the Minimum Coverage Kernel problem remains NP-hard for restricted classes of the intersection graph of the input set of boxes. We consider three main restrictions: when the graph is planar, when the size of its largest clique (namely the clique-number of the graph) is bounded by a constant, and when the degree of a vertex with maximum degree (namely the vertex-degree of the graph) is bounded by a constant.

### 5.2.1 Hardness of Minimum Coverage Kernel

Consider the $k$-Coverage Kernel problem: given a set $\mathcal{B}$ of $n$ boxes, find whether there are $k$ boxes in $\mathcal{B}$ covering the same region as the entire set. Proving that $k$-Coverage Kernel is NP-complete under restricted settings yields the NP-hardness of Minimum Coverage Kernel under the same conditions. To prove that $k$-Coverage Kernel is NP-hard under restricted settings, we reduce instances of the Planar 3-SAT problem (a classical NP-complete problem [110]) to restricted instances of the $k$-Coverage Kernel problem:

> Pb Planar 3-SAT problem: Given a Boolean formula in 3-CNF whose incidence graph[1]is planar, find whether there is an assignment which satisfies the formula.

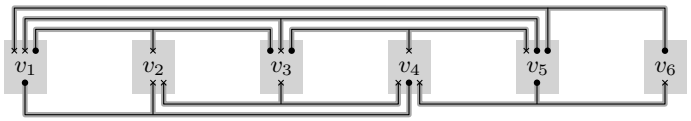



Figure 5.1: Planar embedding of the formula $\varphi = (v_1 \vee \overline{v_2} \vee v_3) \wedge (v_3 \vee \overline{v_4} \vee \overline{v_5}) \wedge (\overline{v_1} \vee \overline{v_3} \vee v_5) \wedge (v_1 \vee \overline{v_2} \vee v_4) \wedge (\overline{v_2} \vee \overline{v_3} \vee \overline{v_4}) \wedge (\overline{v_4} \vee v_5 \vee \overline{v_6}) \wedge (\overline{v_1} \vee v_5 \vee v_6)$. The crosses and dots at the end of the clause legs indicate that the connected variable appears in the clause negated or not, respectively.

As mentioned already in Section 4.2, the (planar) incidence graph of any planar 3-SAT formula $\varphi$ can be represented in the plane, as illustrated in Figure 5.1, where all variables lie on a horizontal line, and all clauses are represented by *non-intersecting* three-legged combs [90]. Again, we refer to such a representation of $\varphi$ as the *planar embedding* of $\varphi$. Based on this planar embedding, we prove the results in Theorem 5.1. Although our arguments are for two dimensions, their extension to higher dimensions is trivial.

**Theorem 5.1.** *Let $\mathcal{B}$ be a set of $n$ boxes in the plane and let $G$ be the intersection graph of $\mathcal{B}$. Solving the $k$-*Coverage Kernel *problem on $\mathcal{B}$ is NP-Complete even if $G$ has clique-number at most 4, and vertex-degree at most 8.*

*Proof.* Given any set $\mathcal{B}$ of $n$ boxes in $\mathbb{R}^d$, and any subset $\mathcal{K}$ of $\mathcal{B}$, certifying that $\mathcal{K}$ covers the same region as $\mathcal{B}$ can be done in time within $\mathcal{O}(n^{d/2})$ using Chan's algorithm [45] for computing the Klee's measure of $\mathcal{B}$ (see Section 2.3.2). Therefore, the $k$-Coverage Kernel problem is in NP. To prove that it is NP-complete, given a planar 3-SAT formula $\varphi$ with $n$ variables and $m$ clauses, we construct a set $\mathcal{B}$ of $\mathcal{O}(n+m)$ boxes with a coverage kernel of size $31m + 3n$ if and only if there is an assignment of the variables satisfying $\varphi$. We use the planar embedding of $\varphi$ as a starting point, and replace the components corresponding to variables and clauses, respectively, by gadgets composed of several boxes. We show that this construction can be obtained in polynomial time, and thus any polynomial time solution for the $k$-Coverage Kernel problem yields a polynomial time solution for the Planar 3-SAT problem. We replace the components in that embedding corresponding to variables and clauses, respectively, by gadgets composed of several boxes, adding a total number of boxes polynomial in the number of variable and clauses.

**Variable gadgets.** Let $v$ be a variable of a planar 3-SAT formula $\varphi$, and let $c_v$ be the number of clauses of $\varphi$ in which $v$ appears. The gadget for $v$ is composed of $4c_v + 2$ rectangles colored either red or blue (see Figure 5.3 for an illustration): $4c_v$ *horizontal* rectangles (of $3 \times 1$ units of size), separated into two "rows" with $2c_v$ rectangles each, and two *vertical* rectangles (of $1 \times 4$ units of size) connecting the rows. The rectangles in each row are enumerated from left to right, starting by one. The $i$-th rectangle of the $j$-th row is defined by the product of intervals $[4(i-1), 4i-1] \times [4(j-1), 4(j-1)+1]$, for all $i = [1..2c_v]$ and $j = 1, 2$. The

[1]The *incidence graph* of a 3-SAT formula is a bipartite graph with a vertex for each variable and each clause, and an edge between a variable vertex and a clause vertex for each occurrence of a variable in a clause.

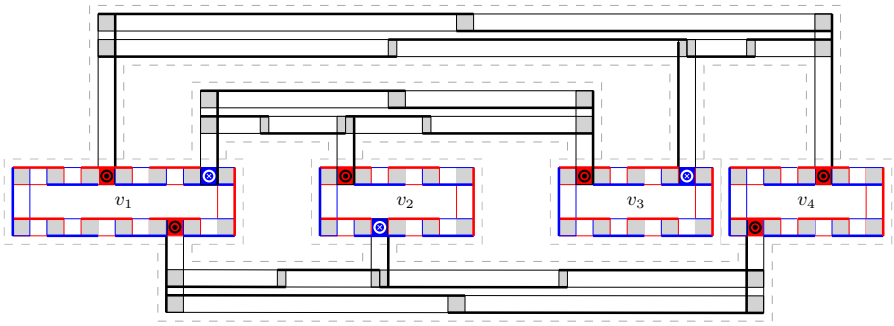



Figure 5.2: Variable and clause gadgets for $\varphi = (\overline{v_1} \vee v_2 \vee v_3) \wedge (v_1 \vee \overline{v_2} \vee v_4) \wedge (v_1 \vee \overline{v_3} \vee v_4)$. The bold lines highlight one side of each rectangle in the instance, while the dashed lines delimit the regions of the variable and clause components in the planar embedding of $\varphi$. Finding a minimum subset of rectangles covering the non-white regions yields an answer for the satisfiability of $\varphi$.

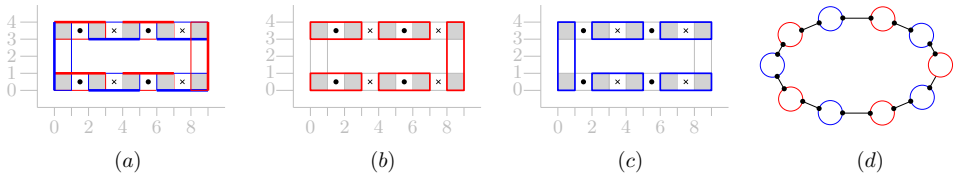



Figure 5.3: *a*) Gadget for a variable appearing in two clauses. A clause gadget connects with the variable in the regions marked with a dot or a cross, depending on the sign of the variable in the clause. The optimal way to cover all the redundant regions (the gray regions) is to choose either all the red rectangles (as in *b*) or all the blue rectangles (as in *c*). *d*) The intersection graph of the variable gadget.

gadget occupies a rectangular region of $(4c_v + 1) \times 4$ units. Although the gadget is defined with respect to the origin of coordinates, it is later translated to the region corresponding to $v$ in the planar embedding of $\varphi$, which we assume without loss of generality to be large enough to fit the gadget. Every horizontal rectangle is colored red if its numbering is odd, and blue otherwise. Besides, the vertical leftmost (resp. rightmost) rectangle is colored blue (resp. red). As we will see later, these colors are useful when connecting a clause gadget with its variables.

Observe that: (*i.*) every red (resp. blue) rectangle intersects exactly two others, both blue (resp. red), sharing with each a squared region of $1 \times 1$ units (which we call *redundant regions*); (*ii.*) the optimal way to cover the redundant regions is by choosing either all the $2c_v + 1$ red rectangles or all the $2c_v + 1$ blue rectangles (see Figure 5.3 for an example).

**Clause gadgets.** Let $C$ be a clause with variables $u$, $v$, and $w$, appearing in this order from left to right in the embedding of $\varphi$. Assume, without loss of generality, that the component for $C$ in the embedding is above the variables. We create a gadget for $C$ composed of 9 black

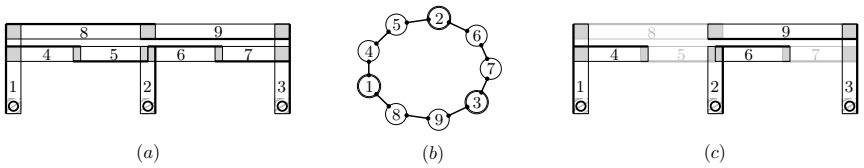
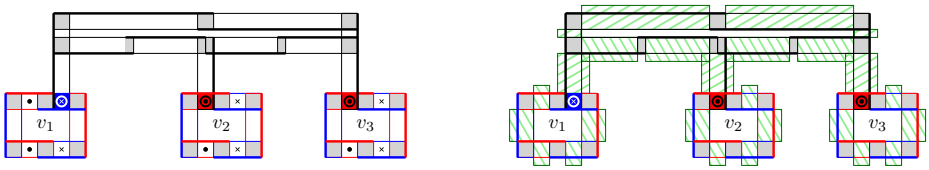



Figure 5.4: *a*) A clause gadget with nine black rectangles: three vertical legs (1,2, and 3) and six horizontal rectangles (4-9). We call the regions where they meet *redundant regions*. The striped regions at the bottom of each leg connect with the variables. *b*) The intersection graph of the gadget. Any minimum cover of the edges (redundant regions) requires 5 vertices (rectangles). *c*) Any cover of the redundant regions which includes the three legs has 6 or more rectangles.

rectangles, located and enumerated as in Figure 5.4.*a*. The vertical rectangles numbered 1, 2 and 3 correspond to the legs of $C$ in the embedding, and connect with the gadgets of $u, v$, and $w$, respectively. The remaining six horizontal rectangles connect the three legs between them. The vertical rectangles have one unit of width and their height is given by the height of the respective legs in the embedding of $C$. Similarly, the horizontal rectangles have one unit of height and their width is given by the separation between the legs in the embedding of $C$ (see Figure 5.2 for an example of how these rectangles are extended or stretched as needed). Note that: (*i*.) every rectangle in the gadget intersects exactly two others (again, we call *redundant regions* the regions where they meet); (*ii*.) any minimum cover of the redundant regions (edges in Figure 5.4.*b*) has five rectangles, one of which must be a leg; and (*iii*.) any cover of the redundant regions which includes the three legs must have at least six rectangles (e.g., see Figure 5.4.*c*).

**Connecting the gadgets.** Let $v$ be a variable of a formula $\varphi$ and $c_v$ be the number of clauses in which $v$ occurs. The legs of the $c_v$ clause gadgets are connected with the gadget for $v$, from left to right, in the same order they appear in the embedding of $\varphi$. Let $C$ be the gadget for a clause containing $v$ whose component in the embedding of $\varphi$ is above (resp. below) that for $v$. $C$ connects with the gadget for $v$ in one of the rectangles in the upper (resp. lower) row, sharing a region of $1 \times 1$ units with one of the red (resp. blue) rectangles if the variable appears positive (resp. negative) in the clause (see Figure 5.5.*a*). We call this region where the variable and clause gadgets meet as *connection region*, and its color is given by the color of the respective rectangle in the variable gadget. Note that a variable gadget has enough connection regions for all the $c_v$ clauses in which it appears, because each row of the gadget has $c_v$ rectangles of each color.

**Completing the instance.** Each rectangle in a variable or clause gadget, as described, has a region that no other rectangle covers (i.e., of depth 1). Thus, the coverage kernel of the instances described up to here is trivial: all the rectangles. To avoid this, we cover all the regions of depth 1 with *green* rectangles (as illustrated in Figure 5.5.*b*) which are forced to



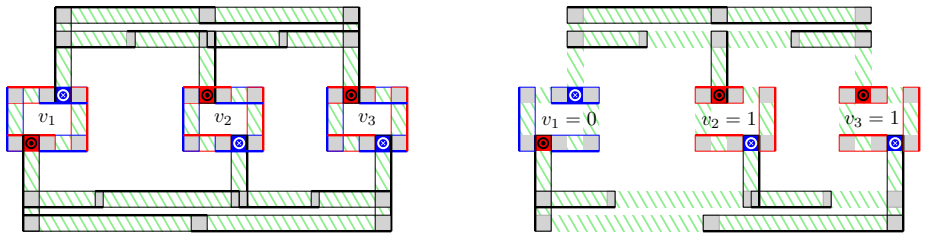

$(a)$ $(b)$

Figure 5.5: An instance for $(\overline{v_1} \vee v_2 \vee v_3)$. *a)* The clause gadget connects with a variable gadget in one of its red or blue connection regions depending on the sign of the variable in the clause. *b)* To complete the instance, green boxes are added to cover all the regions with depth 1. The green rectangles are forced to be in any kernel by making them large enough so that each covers an exclusive region.

be in any coverage kernel[2]. For every clause gadget we add 11 such green rectangles, and for each variable gadget for a variable $v$ occurring in $c_v$ clauses we add $3c_v + 2$ green rectangles.

Let $\varphi$ be a formula with $n$ variables and $m$ clauses. The instance of $k$-Coverage Kernel that we create for $\varphi$ has a total of $41m + 4n$ rectangles: $(i.)$ each clause gadget has 9 rectangles for the comb, and 11 green rectangles, for a total of $20m$ rectangles over all the clauses; $(ii.)$ a gadget for a variable $v$ has $4c_v + 2$ red and blue rectangles, and we add a green rectangle for each of those that does not connect to a clause gadget ($3c_v + 2$ per variable), thus adding a total of $7c_v + 4$ rectangles by gadget; and $(iii.)$ over all variables, we add a total of $\sum_{i=1}^{n} (7c_{v_i} + 4) = 7(3m) + 4n = 21m + 4n$ rectangles[3].

**Intuition: from minimum kernels to boolean values.** Consider a gadget for a variable $v$. Any minimum coverage kernel of the gadget is composed of all its green rectangles together with either all its blue or all its red rectangles. Thus, the minimum number of rectangles needed to cover all the variable gadgets is fixed, and known. If all the red rectangles are present in the kernel, we consider that $v = 1$, otherwise if all the blue rectangles are present, we consider that $v = 0$ (see Figure 5.6 for an example). In the same way that choosing a




$(a)$ $(b)$

Figure 5.6: *a)* An instance for $\varphi = (\overline{v_1} \vee v_2 \vee v_3) \wedge (v_1 \vee \overline{v_2} \vee \overline{v_3})$ (the area covered by green rectangles is highlighted with green falling lines). *b)* A cover of the gadgets corresponding to the assignment $v_1 = 0, v_2 = 1, v_3 = 1$ that does not satisfy $\varphi$.

[2]For simplicity, these green rectangles were omitted in Figure 5.2 and Figure 5.6.
[3]Note that $\sum_{i=1}^{n} c_{v_i} = 3m$ since exactly 3 variables occurs in each clause.

value for $v$ may affect the output of a clause $C$ in which $v$ occurs, choosing a color to cover the gadget for $v$ may affect the number of rectangles required to cover the gadget for $C$. For instance, consider that the gadget for $v$ is covered with blue rectangles (i.e., $v = 0$), and that $v$ occurs unnegated in $C$ (see the gadgets for $v_1$ and the second clause in Figure 5.6). The respective leg of the gadget for $C$ meets $v$ in one of its red rectangles. Since red rectangles were not selected to cover the variable gadget, that leg is forced to cover the connection region shared with the variable gadget, and thus is forced to be in any kernel of the gadget for $C$. This corresponds to the fact that the literal of $v$ in $C$ evaluates to 0. If the same happens for the three variables in $C$ (i.e., $C$ is not satisfied by the assignment), then to cover its gadget at least six of the black rectangles will be required (see the gadget for the second clause in Figure 5.6). However, if at least one of its legs can be disposed of (i.e., at least one of the literals evaluates to 1), then the clause gadget can be covered with five of its black rectangles (see the gadget for the first clause in Figure 5.6). The minimum number of rectangles needed to cover all the variable gadgets is fixed and known: all the green rectangles, and one half of the red/blue rectangles of each variable. Therefore, it suffices to show that there is an assignment satisfying a 3-SAT formula if and only if every clause gadget can be covered by five of its black rectangles (plus all its green rectangles).

**Reduction.** We prove the theorem in two steps. First, we show that such an instance has a coverage kernel of size $31m + 3n$ if and only $\varphi$ is satisfiable. Therefore, answering $k$-Coverage-Kernel over this instance with $k = 31m+3n$ yields an answer for the Planar 3-SAT problem on $\varphi$. Finally, we will show that the instance described matches all the restrictions in Theorem 5.1, under minor variations.

($\Rightarrow$) Let $\varphi$ be a 3-CNF formula with $n$ variables $v_1, \ldots, v_n$ and $m$ clauses $C_1, \ldots, C_m$; let $\mathcal{B}$ be a set of boxes created as described above for $\varphi$, and let $\{v_1 = \alpha_1, \ldots, v_n = \alpha_n\}$ be an assignment which satisfies $\varphi$. We create a coverage kernel $\mathcal{K}$ of $\mathcal{B}$ as follows:

- For each variable gadget for $v_i \in \{v_1, \ldots, v_n\}$ such that $\alpha_i = 0$ (resp. $\alpha_i = 1$), add to $\mathcal{K}$ all but its red (resp. blue) rectangles, thus covering the entire gadget minus its red (resp. blue) connection regions. This uncovered regions, which must connect with clauses in which the literal of $v_i$ evaluates to 0, will be covered later with the legs of the clause gadgets. Over all variables, we add to $\mathcal{K}$ a total of $\sum_{i=1}^{n} ((7-2)c_{v_i} + (4-1)) = 5(3m) + 3n = 15m + 3n$ rectangles.

- For each clause $C_i \in \{C_1, \ldots, C_n\}$, add to $\mathcal{K}$ all its green rectangles and the legs that connect with connection regions of the variable gadgets left uncovered in the previous step. Note that at least one of the legs of $C_i$ is not added to $\mathcal{K}$ since at least one of the literals in the clause evaluates to 1, and the connection region corresponding to that literal is already covered by the variable gadget. Thus, the redundant regions of the gadget for $C_i$ can be covered with five of its black rectangles (including the legs already added). So, finally add to $\mathcal{K}$ black rectangles from the clause for $C_i$ as needed (up to a

total of five), until all the redundant regions are covered (and with the green rectangles, the entire gadget). Over all clauses, we add a total of $(11+5)m = 16m$ rectangles.

By construction, $\mathcal{K}$ is a coverage kernel of $\mathcal{B}$: it covers completely every variable and clause gadget. Moreover, the size of $\mathcal{K}$ is $(15m+3n)+16m = 31m+3n$.

($\Leftarrow$) Let $\varphi$ be 3-CNF formula with $n$ variables $v_1, \ldots, v_n$ and $m$ clauses $C_1, \ldots, C_m$, let $\mathcal{B}$ be a set of boxes created as described above for $\varphi$, and let $\mathcal{K}$ be a coverage kernel of $\mathcal{B}$ whose size is $(15m+3n)+16m = 31m+3n$. Any coverage kernel of $\mathcal{B}$ must include all its green rectangles. Furthermore, to cover any clause gadget at least five of its black rectangles are required, and to cover any variable rectangle, at least half of its red/blue rectangles are required. Thus, any coverage kernel of $\mathcal{B}$ must have at least $(11+5)m = 16m$ of the clause rectangles, and at least $\sum_{i=1}^{n}((3+4/2)c_{v_i}+(3)) = 5(3m)+3n = 15m+3n$ of the variable rectangles are required. Hence, $\mathcal{K}$ must be a coverage kernel of minimum size.

Since the redundant regions of any two gadgets are independent, $\mathcal{K}$ must cover each gadget optimally (in a local sense). Given that the intersection graph of the red/blue rectangles of a variable gadget is a ring (see Figure 5.3.d for an illustration), the only way to cover a variable gadget optimally is by choosing either all its blue or all its red rectangles (together with the green rectangles). Hence, the way in which every variable gadget is covered is consistent with an assignment for its variable as described before in the intuition. Moreover, the assignment induced by $\mathcal{K}$ must satisfy $\varphi$: in each clause gadget, at least one of the legs was discarded (to cover the gadget with 5 rectangles), and at least the literal in the clause corresponding to that leg evaluates to 1.

**Meeting the restrictions.** Now we prove that the instance of Minimum Coverage Kernel generated for the reduction meets the restrictions of the theorem. First, we show the bounded clique-number and vertex-degree properties for the intersection graph of a clause gadget and its three respective variable gadgets. In Figure 5.7 we illustrate the intersection graph for the clause $\varphi = (\overline{v_1} \vee v_2 \vee v_3)$. The sign of the variables in the clause does not change the maximum vertex degree or the clique-number of the graph, so the figure is general enough for our purpose. Since we consider the rectangles composing the instance to be closed rectangles, if two rectangles containing at least one point in common (in their interior or boundary), their respective vertices in the intersection graph are adjacent. Note that in Figure 5.7 the vertices with highest degree are the ones corresponding to legs of the clause gadget (rectangles 1, 2, and 3). There are 4-cliques in the graph, for instance the right lower corner of the green rectangle denoted $h$ is covered also by rectangles $1, 4$ and $d$, and hence their respective vertices form a clique. However, since there is no point that is covered by five rectangles at the same time, there are no 5-cliques in the graph. Finally, note that, since the clause gadgets are located according to the planar embedding of the formula, they are pairwise independent.[4] Thus, the bounds on the clique-number and vertex-degree of the

[4]Two clause gadgets are pairwise independent if the rectangles composing them are pairwise independent, as well as the rectangles where they connect with their respective variables gadgets.

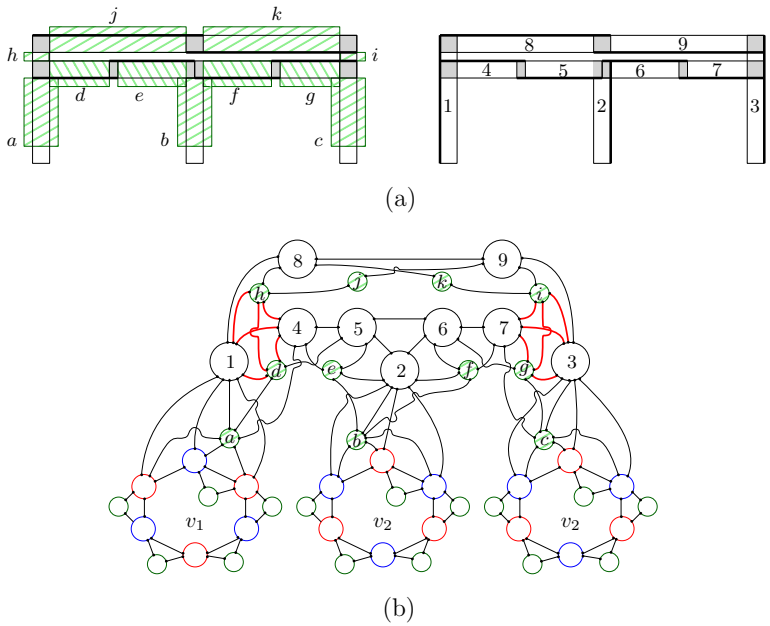



Figure 5.7: The intersection graph of the instance of the Minimum Coverage Kernel problem corresponding to $\varphi = (\overline{v_1} \vee v_2 \vee v_3)$ (see the complete instance in Figure 5.5.b). *a*) Numbering of the rectangles of the clause gadget. *b*) The intersection graph of the instance: the vertices corresponding to the legs have degree 8, which is the highest; and the red fat edges highlight two 4-cliques.

intersection graph of any clause gadget extend also to the intersection graph of an entire general instance. □

### 5.2.2 Extension to Box Cover

Since the Minimum Coverage Kernel problem is a special case of the Box Cover problem, the result of Theorem 5.1 also applies to the Box Cover problem. However, in Theorem 5.2 we show that this problem remains hard under even more restricted settings.

**Theorem 5.2.** *Let $P$, $\mathcal{B}$ be a set of $m$ points and $n$ boxes in the plane, respectively, and let $G$ be the intersection graph of $\mathcal{B}$. Solving the* Box Cover *problem on $P$ and $\mathcal{B}$ is* NP*-complete even if every point in $P$ is covered by at most two boxes of $\mathcal{B}$, and $G$ is planar, has clique-number at most 2, and vertex-degree at most 4.*

*Proof.* We use the same reduction from Planar 3-SAT, but with three main variations in the gadgets: we drop the green rectangles of both the variable and clause gadgets, add points within the redundant and connection region of both variable and clause gadgets, and separate the rectangles numbered 5 and 6 of each clause gadget so they do not intersect (see Figure 5.8 for an example). Since the interior of every connection or redundant region is covered by at most two of the rectangles in the gadgets, every point of the instance we create

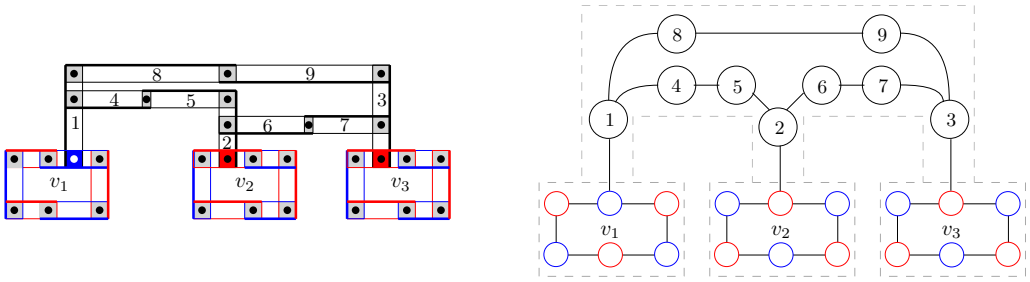



Figure 5.8: *a*) The instance of the Box Cover problem corresponding to $\varphi = (\overline{v_1} \vee v_2 \vee v_3)$. *b*) The intersection graph of the instance: the vertices corresponding to the legs have degree 3, which is the highest; there are no 3-cliques in the graph; and the graph is planar and can be drawn within the planar embedding of $\varphi$ (highlighted with dashed lines).

is contained in at most two boxes. In Figure 5.8.b we illustrate the intersection graph for the clause $\varphi = (\overline{v_1} \vee v_2 \vee v_3)$. Since the sign of the variables in the clause does not change the maximum vertex degree or the clique-number of the graph, or its planarity, the properties we mention next are also true for any clause. Note that three is the maximum vertex-degree of the intersection graph, and that there are no 3-cliques. Also note that the intersection graph can be drawn within the planar embedding of $\varphi$ so that no two edges cross, and hence the graph is planar. Again, due to the pairwise independence of the clause gadgets, these properties extend to the entire intersection graph of a general instance. □

In the next section, we complement these hardness results with two approximation algorithms for the Minimum Coverage Kernel problem.

## 5.3 Efficient approximation of Minimum Coverage Kernels

Let $\mathcal{B}$ be a set of $n$ boxes in $\mathbb{R}^d$, and let $\mathcal{D}(\mathcal{B})$ be a coverage discretization of $\mathcal{B}$ (as defined in Section 2.4.1). A *weight index* for $\mathcal{D}(\mathcal{B})$ is a data structure which can perform the following operations:

- *Initialization:* Assign an initial unitary weight to every point in $\mathcal{D}(\mathcal{B})$;
- *Query:* Given a box $b \in \mathcal{B}$, find the total weight of the points in $b$.
- *Update:* Given a box $b \in \mathcal{B}$, multiply the weights of all the points within $b$ by a given value $\alpha \geq 0$;

We assume that the weights are small enough so that arithmetic operations over the weights can be performed in constant time. There is a trivial implementation of a weight index

with initialization and update time within $\mathcal{O}(n^d)$, and with constant query time. In this section we describe an efficient implementation of a weight index, and combine this data structure with two existing approximation algorithms for the Box Cover problem [101, 39] and obtain improved approximation algorithms (in the running time sense) for the Minimum Coverage Kernel problem.

### 5.3.1 An Efficient Weight Index for a Set of Boxes

We describe a weight index for $\mathcal{D}(\mathcal{B})$ which can be initialized in time within $\mathcal{O}(n^{\frac{d+1}{2}})$, and with query and update time within $\mathcal{O}(n^{\frac{d-1}{2}} \log n)$. Let us consider first the case of a set $I$ of $n$ intervals.

**A weight index for a set of intervals.** A trivial weight index which explicitly saves the weights of each point in $\mathcal{D}(I)$ can be initialized in time within $\mathcal{O}(n \log n)$, has linear update time, and constant query time. We show that by sacrificing query time (by a factor within $\mathcal{O}(\log n)$) one can improve the update time to within $\mathcal{O}(\log n)$. The main idea is to maintain the weights of each point of $\mathcal{D}(I)$ indirectly using a tree.

Consider a balanced binary tree whose leaves are in one-to-one correspondence with the values in $\mathcal{D}(I)$ (from left to right in a non-decreasing order). Let $p_v$ denote the point corresponding to a leaf node $v$ of the tree. In order to represent the weights of the points in $\mathcal{D}(I)$, we store a value $\mu(v)$ at each node $v$ of the tree subject to the following invariant: for each leaf $v$, the weight of the point $p_v$ equals the product of the values $\mu(u)$ of all the ancestors $u$ of $v$ (including $v$ itself). The $\mu$ values allow to increase the weights of many points with only a few changes. For instance, if we want to double the weights of all the points we simply multiply by 2 the value $\mu(r)$ of the root $r$ of the tree. Besides the $\mu$ values, to allow efficient query time we also store at each node $v$ three values $\texttt{min}(v), \texttt{max}(v), \omega(v)$: the values $\texttt{min}(v)$ and $\texttt{max}(v)$ are the minimum and maximum $p_u$, respectively, such that $u$ is a leaf of the tree rooted at $v$; the value $\omega(v)$ is the sum of the weights of all $p_u$ such that $u$ is a leaf of the tree rooted at $v$.

Initially, all the $\mu$ values are set to one. Besides, for every leaf $l$ of the tree $\omega(l)$ is set to one, while $\texttt{min}(l)$ and $\texttt{max}(l)$ are set to $p_l$. The $\texttt{min}$, $\texttt{max}$ and $\omega$ values of every internal node $v$ with children $l, r$, are initialized in a bottom-up fashion as follows: $\texttt{min}(v) = \min\{\texttt{min}(l), \texttt{min}(r)\}$; $\texttt{max}(v) = \max\{\texttt{max}(l), \texttt{max}(r)\}$; $\omega(v) = \mu(v) \cdot (\omega(l) + \omega(r))$. It is simple to verify that after this initialization, the tree meets all the invariants mentioned above. We show in Theorem 5.3 that this tree can be used as a weight index for $\mathcal{D}(I)$.

**Theorem 5.3.** *Let $I$ be a set of $n$ intervals in $\mathbb{R}$. There exists a weight index for $\mathcal{D}(I)$ which can be initialized in time within $\mathcal{O}(n \log n)$, and supports queries and updates in time within $\mathcal{O}(\log n)$.*

*Proof.* Since intervals have linear union complexity, $\mathcal{D}(I)$ has within $\mathcal{O}(n)$ points, and it can be computed in linear time after sorting, for a total time within $\mathcal{O}(n \log n)$. We store the points in the tree described above. Its initialization can be done in linear time since the tree

has within $\mathcal{O}(n)$ nodes, and when implemented in a bottom-up fashion, the initialization of the $\mu, \omega, \texttt{min}$, and $\texttt{max}$ values, respectively, cost constant time per node.

To analyze the query time, let $\mathsf{totalWeight}(a, b, t)$ denote the procedure which finds the total weight of the points corresponding to leaves of the tree rooted at $t$ that are in the interval $[a, b]$. This procedure can be implemented as follows:

1. if $[a, b]$ is disjoint to $[\texttt{min}(t), \texttt{max}(t)]$ return 0;
2. if $[a, b]$ completely contains $[\texttt{min}(t), \texttt{max}(t)]$ return $\omega(r)$;
3. if both conditions fail (leaves must meet either 1. or 2.), let $l, r$ be the left and right child of $t$, respectively;
4. if $a > \texttt{max}(l)$ return $\mu(t) \cdot \mathsf{totalWeight}(a, b, r)$;
5. if $b < \texttt{min}(r)$ return $\mu(t) \cdot \mathsf{totalWeight}(a, b, l)$;
6. otherwise return $\mu(t)(\mathsf{totalWeight}(a, \infty, l) + \mathsf{totalWeight}(-\infty, b, r))$.

Due to the invariants to which the $\texttt{min}$ and $\texttt{max}$ values are subjected, every leaf $l$ of $t$ corresponding to a point in $[a, b]$ has an ancestor (including $l$ itself) which is visited during the call to $\mathsf{totalWeight}$ and which meets the condition in step 2. For this, and because of the invariants to which the $\omega$ and $\mu$ values are subjected, the procedure $\mathsf{totalWeight}$ is correct. Note that the number of nodes visited is at most 4 times the height $h$ of the tree: when both children need to be visited, one of the endpoints of the interval to query is replaced by $\pm\infty$, which ensures that in subsequent calls at least one of the children is completely covered by the query interval. Since $h \in \mathcal{O}(\log n)$, and the operations at each node consume constant time, the running time of $\mathsf{totalWeight}$ is within $\mathcal{O}(\log n)$.

Similarly, to analyze the update time, let $\mathsf{updateWeights}(a, b, t, \alpha)$ denote the procedure which multiplies by a value $\alpha$ the weights of the points in the interval $[a, b]$ stored in leaves descending from $t$. This can be implemented as follows:

1. if $[a, b]$ is disjoint to $[\texttt{min}(t), max(t)]$, finish;
2. if $[a, b]$ completely contains $[\texttt{min}(t), \texttt{max}(t)]$ set $\mu(r) = \alpha \cdot \mu(r)$, set $\omega(r) = \alpha \cdot \omega(r)$, and finish;
3. if both conditions fail, let $l, r$ be the left and right child of $t$, respectively;
4. if $a > \texttt{max}(l)$, call $\mathsf{updateWeights}(a, b, r, \alpha)$;
5. else if $b < \texttt{min}(r)$, call $\mathsf{updateWeights}(a, b, l, \alpha)$;
6. otherwise, call $\mathsf{updateWeights}(a, \infty, l, \alpha)$, and $\mathsf{updateWeights}(-\infty, b, r, \alpha)$;
7. finally, after the recursive calls set $\omega(t) = \mu(t) \cdot (\omega(l) + \omega(r))$, and finish.

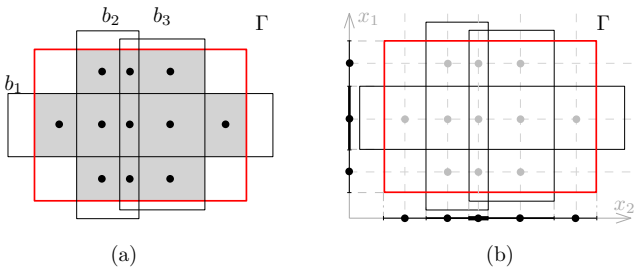



Figure 5.9: (a) An illustration in the plane of three boxes equivalent to slabs when restricted to the box $\Gamma$. The dots correspond to a set of points which discretize the (grayed) region within $\Gamma$ covered by the slabs.

Note that, for every point $p_v$ in $[a, b]$ corresponding to a leaf $v$ descending from $t$, the $\mu$ value of exactly one of the ancestors of $u$ changes (by a factor of $\alpha$): at least one changes because of the invariants to which the `min` and `max` values are subjected (as analyzed for totalWeight); and no more than one can change because once $\mu$ is assigned for the first time to some ancestor $u$ of $v$, the procedure finishes leaving the descendants of $v$ untouched. The analysis of the running time is analogous to that of totalWeight, and thus within $\mathcal{O}(\log n)$. □

The weight index for set of intervals described in Theorem 5.3 plays an important role in obtaining an index for a higher dimensional set of boxes. In a step towards such a data structure, we first describe how to use one-dimensional indexes to obtain indexes for another special case of sets of boxes, this time in high dimension.

**A weight index for a set of slabs.** A box $b$ is said to be a slab within another box $\Gamma$ if $b$ covers completely $\Gamma$ in all but one dimension (see Figure 5.9.a for an illustration). Let $\mathcal{B}$ be a set of $n$ $d$-dimensional boxes that are slabs within another box $d$-dimensional box $\Gamma$. Let $\mathcal{B}|_\Gamma$ denote the set $\{b \cap \Gamma \mid b \in \mathcal{B}\}$ of the boxes of $\mathcal{B}$ restricted to $\Gamma$. We describe a weight index for $\mathcal{D}(\mathcal{B}|_\Gamma)$ with initialization time within $\mathcal{O}(n \log n)$, and with update and query time within $\mathcal{O}(\log n)$.

For all $i = [1..d]$, let $\mathcal{B}_i$ be the subset of slabs that are orthogonal to the $i$-th dimension, and let $I_i$ be the set of intervals resulting from projecting $\Gamma$ and each rectangle in $\mathcal{B}|_\Gamma$ to the $i$-th dimension (see Figure 5.9.b for an illustration). The key to obtain an efficient weight index for a set of slabs is the fact that weight indexes for $\mathcal{D}(I_1), \ldots \mathcal{D}(I_d)$ can be combined without much extra computational effort into a weight index for $\mathcal{D}(\mathcal{B}|_\Gamma)$. Let $p$ be a point of $\mathcal{D}(\mathcal{B}|_\Gamma)$ and let $x_i(p)$ denote the value of the $i$-th coordinate of $p$. Observe that for all $i \in [1..d]$, $x_i(p) \in \mathcal{D}(I_i)$ (see Figure 5.9.b for an illustration). This allows the representation of the weight of each point $p \in \mathcal{D}(\mathcal{B})$ by means of the weights of $x_i(p)$ for all $i \in [1..d]$. We do this by maintaining the following *weight invariant*: the weight of a point $p \in \mathcal{D}(\mathcal{B}|_\Gamma)$ is equal to $\prod_{i=1}^{d}$ (weight of $x_i(p)$ in $\mathcal{D}(I_i)$).

**Lemma 5.4.** *Let $\mathcal{B}$ be a set of $n$ $d$-dimensional boxes that are equivalent to slabs when restricted to another $d$-dimensional box $\Gamma$. There exists a weight index for $\mathcal{D}(\mathcal{B})$ which can be initialized in time within $\mathcal{O}(n \log n)$, supporting queries and updates in time within $\mathcal{O}(\log n)$.*

*Proof.* Let $\mathcal{B}_i$ be the subset of $\mathcal{B}|_\Gamma$ orthogonal to the $i$-th dimension, and let $I_i$ be the set of intervals resulting from projecting $\Gamma$ and each rectangle in $\mathcal{B}_i$ to the $i$-th dimension. Initialize a weight index for $\mathcal{D}(I_i)$ as in Theorem 5.3, for all $i \in [1..d]$. Since the weights of all the points in the one-dimensional indexes are initialized to one, the weight of every point in $\mathcal{D}(\mathcal{B}|_\Gamma)$ is also initialized to one, according to the weight invariant. This initialization can be done in total time within $\mathcal{O}(n \log n)$.

Let $b \in \mathcal{B}|_\Gamma$ be a box which covers $\Gamma$ in every dimension except for the $i$-th one, for some $i \in [1..d]$ (i.e., $b \in B_i$), and let $P_i$ be the subset of $\mathcal{D}(I_i)$ contained within the projection of $b$ to the $i$-th dimension. The set of points of $\mathcal{D}(\mathcal{B}|_\Gamma)$ that are within $b$ can be generated by the expression $\{(a_1, \ldots, a_d) \mid a_1 \in \mathcal{D}(I_1) \wedge \ldots \wedge a_{i-1} \in \mathcal{D}(I_{i-1}) \wedge a_i \in P_i \wedge a_{i+1} \in \mathcal{D}(I_{i+1}) \wedge \ldots \wedge a_d \in \mathcal{D}(I_d)\}$. Therefore, the total weight of the points within $b$ is given by the total weight of the points $x_i(p)$ for all $p \in P_i$ multiplied by the total weight of the points in $\mathcal{D}(I_j)$, for all $j = [1..d]$ distinct from $i$.

To query the total weight of the points of $\mathcal{D}(\mathcal{B}|_\Gamma)$ within a box $b \in B_i$ we query the weight index of $\mathcal{D}(I_i)$ to find the total weight of the points in the projection of $b$ to the $i$-th dimension (in time within $\mathcal{O}(\log n)$), then query the remaining $d-1$ indexes to find the total weight store in the index (stored at the $\omega$ value of the root of the trees), and return the product of those $d$ values. Clearly the running time is within $\mathcal{O}(\log n)$.

The update is similar: to multiply by a value $\alpha$ the weight of all the points of $\mathcal{D}(\mathcal{B}|_\Gamma)$ within a box $b \in B_i$ we simply update the weight index of $\mathcal{D}(I_i)$ multiplying by $\alpha$ all the weights of the points within the projection of $b$ to the $i$-th dimension, and leave the other $d-1$ weight indexes untouched. The running time of this operation is also within $\mathcal{O}(n \log n)$, and the invariant remains valid after the update. □

Lemma 5.4 shows that there are weight indexes for a set of slabs $\mathcal{B}$ within another box $\Gamma$ that significantly improve the approach of explicitly constructing $\mathcal{D}(\mathcal{B}|_\Gamma)$, an improvement that grows exponentially with the dimension $d$. We take advantage of this to describe a similar improvement for the general case.

**A Weight Index for The General Case.** We now show how to maintain a weight index of a general set $\mathcal{B}$ of $d$-dimensional boxes. The main idea is to partition the space into cells such that, within each cell $c$, any box $b \in \mathcal{B}$ either completely contains $c$ or is equivalent to a *slab*. Then, we use weight indexes for slabs (as described in Lemma 5.4) to index the weights within each of the cells. This approach was first introduced by Overmars and Yap [114] in order to compute the volume of the region covered by a set boxes (see Section 2.3.1 for a detailed description), and similar variants were used since then to compute other measures [24, 45, 127]. The following lemma summarizes the key properties of the partition we use:

**Lemma 5.5** (Lemma 4.2 of Overmars and Yap [114])**.** *Let $\mathcal{B}$ be a set of $n$ boxes in $d$-dimensional space. There is a binary partition tree for storing any subset of $\mathcal{B}$ such that*

- *It can be computed in time within $\mathcal{O}(n^{d/2})$, and it has $\mathcal{O}(n^{\frac{d}{2}})$ nodes;*
- *Each box is stored in $\mathcal{O}(n^{\frac{d-1}{2}})$ leafs;*

- *The boxes stored in a leaf are slabs within the cell corresponding to the node;*
- *Each leaf stores $\mathcal{O}(\sqrt{n})$ boxes.*

Consider the tree of Lemma 2.1. Analogously to the case of intervals, we augment this tree with information to efficiently support the operations of a weight index. At every node $v$ we store two values $\mu(v), \omega(v)$: the first allows to multiply all the weights of the points of $\mathcal{D}(\mathcal{B})$ that are maintained in leaves descending from $v$ (allowing to support updates efficiently); while $\omega(v)$ stores the total weight of these points (allowing to support queries efficiently). To ensure that all/only the nodes that intersect a box $b$ are visited during a query or update operation, we store at each node the boundaries of the cell corresponding to that node. Furthermore, at every leaf node $l$ we implicitly represent the points of $\mathcal{D}(\mathcal{B})$ that are inside the cell corresponding to the node $l$ using a weight index for slabs.

To initialize this data structure, all the $\mu$ values are set to one. Then the weight index within each leaf cell are initialized. Finally, the $\omega$ values of every node $v$ with children $l, r$, are initialized in a bottom-up fashion setting $\omega(v) = \mu(v) \cdot (\omega(l) + \omega(r))$. We show in Theorem 5.6 how to implement the weight index operations over this tree and we analyze its running times.

**Theorem 5.6.** *Let $\mathcal{B}$ be a set of $n$ $d$-dimensional boxes. There is a weight index for $\mathcal{D}(\mathcal{B})$ which can be initialized in time within $\mathcal{O}(n^{\frac{d+1}{2}})$, supporting queries and updates in time within $\mathcal{O}(n^{\frac{d-1}{2}} \log n)$.*

*Proof.* The initialization of the index, when implemented as described before, runs in constant time for each internal node of the tree, and in time within $\mathcal{O}(\sqrt{n} \log n)$ for each leaf (due to Lemma 5.4, and to the last item of Lemma 2.1). Since the tree has $\mathcal{O}(n^{\frac{d}{2}})$ nodes, the total running time of the initialization is within $\mathcal{O}(n^{\frac{d+1}{2}} \log n)$.

Since the implementations of the query and update operations are analogous to those for the intervals weight index (see the proof of Theorem 5.3), we omit the details of their correctness. While performing a query/update operation at must $\mathcal{O}(n^{\frac{d-1}{2}})$ leaves are visited (due to the third item of Lemma 2.1), and since the height of the tree is within $\mathcal{O}(\log n)$, at most $\mathcal{O}(n^{\frac{d-1}{2}} \log n)$ internal nodes are visited in total. Hence, the cost of a query/update operation within each leaf is within $\mathcal{O}(\log n)$ (by Lemma 5.4), and is constant in each internal node. Thus, the total running of a query/update operation is within $\mathcal{O}(n^{\frac{d-1}{2}} \log n)$. □

### 5.3.2 Approximation algorithms for Minimum Coverage Kernel.

Approximating the Minimum Coverage Kernel of a set $\mathcal{B}$ of boxes via approximation algorithms for the Box Cover problem requires that $\mathcal{D}(\mathcal{B})$ is explicitly constructed. However, the weight index described in the proof of Theorem 5.6 can be used to significantly improve the running time of these algorithms. We describe below two examples.

The first algorithm we consider is the greedy $\mathcal{O}(\log n)$-approximation algorithm by Lovász [101]. Such a greedy strategy applies naturally to the Minimum Coverage Kernel problem:

iteratively pick the box which covers the most yet uncovered points of $\mathcal{D}(\mathcal{B})$, until there are no points of $\mathcal{D}(\mathcal{B})$ left to cover. We described this approach in details in Section 2.4.2. To avoid the explicit construction of $\mathcal{D}(\mathcal{B})$, three operations must be simulated: (*i.*) find how many uncovered points are within a given a box $b \in \mathcal{B}$; (*ii.*) delete the points that are covered by a box $b \in \mathcal{B}$; and (*iii.*) find whether a subset $\mathcal{B}'$ of $\mathcal{B}$ covers all the points of $\mathcal{D}(\mathcal{B})$.

For the first two we use the weight index described in the proof of Theorem 5.6: to delete the points within a given box $b \in \mathcal{B}$ we simply multiply the weights of all the points of $\mathcal{D}(\mathcal{B})$ within $b$ by $\alpha = 0$; and finding the number of uncovered points within a box $b$ is equivalent to finding the total weight of the points of $\mathcal{D}(\mathcal{B})$ within $b$. For the last of the three operations we use the following observation:

**Observation 5.1.** *Let $\mathcal{B}$ be a set of $d$-dimensional boxes, and let $\mathcal{B}'$ be a subset of $\mathcal{B}$. The volume of the region covered by $\mathcal{B}'$ equals that of $\mathcal{B}$ if and only if $\mathcal{B}'$ and $\mathcal{B}$ cover the exact same region.*

Let $\texttt{OPT}$ denote the size of a minimum coverage kernel of $\mathcal{B}$, and let $N$ denote the size of $\mathcal{D}(\mathcal{B})$ ($N \in \mathcal{O}(n^d)$). The greedy algorithm described by Lovász [101], when run over the sets $\mathcal{B}$ and $\mathcal{D}(\mathcal{B})$, works in $\mathcal{O}(\texttt{OPT} \log N)$ steps; and at each stage a box is added to the solution. The size of the output is within $\mathcal{O}(\texttt{OPT} \log N) \subseteq \mathcal{O}(\texttt{OPT} \log n)$. This algorithm can be modified to achieve the following running time, while achieving the same approximation ratio:

**Theorem 5.7.** *Let $\mathcal{B}$ be a set of $n$ boxes in $\mathbb{R}^d$ with a minimum coverage kernel of size $\texttt{OPT}$. Then, a coverage kernel of $\mathcal{B}$ of size within $\mathcal{O}(\texttt{OPT} \log n)$ can be computed in time within $\mathcal{O}(\texttt{OPT} \cdot n^{\frac{d+1}{2}} \log^2 n)$.*

*Proof.* We initialize a weight index as in Theorem 5.6, which can be done in time within $\mathcal{O}(n^{\frac{d+1}{2}})$, and compute the volume of the region covered by $\mathcal{B}$, which can be done in time within $\mathcal{O}(n^{d/2})$ [45]. Let $C$ be an empty set. At each stage of the algorithm, for every box $b \in \mathcal{B} \setminus C$ we compute the total weight of the points inside $b$ (which can be done in time within $\mathcal{O}(n^{\frac{d-1}{2}} \log n)$ using the weight index). We add to $C$ the box with the highest total weight, and update the weights of all the points within this box to zero (by multiplying their weights by $\alpha = 0$) in time within $n^{\frac{d-1}{2}} \log n$. If the volume of the region covered by $C$ (which can be computed in $\mathcal{O}(n^{d/2})$-time [45]) is the same as that of $\mathcal{B}$, then we stop and return $C$ as the approximated solution. The total running time of each stage is within $\mathcal{O}(n^{\frac{d+1}{2}} \log n)$. This, along with the fact that the number of stages is within $\mathcal{O}(\texttt{OPT} \log n)$, yields the result of the theorem. □

Now, we show how to improve the running time of Brönnimann and Goodrich's $\mathcal{O}(\log \texttt{OPT})$-approximation algorithm [39] via a weight index. We provided a detailed description of their algorithm in Section 2.4.2. Below, we review their main idea. Let $w : \mathcal{B} \to \mathbb{R}$ be a weight function for the elements of $\mathcal{B}$, let $\texttt{OPT}$ denote the size of an optimal Box Cover of the set system $(\mathcal{D}(\mathcal{B}), \mathcal{B})$, and let $k$ be an integer value such that $k/2 < \texttt{OPT} < k$. Initialize the weight of each $b \in \mathcal{B}$ to 1, and repeat the following *weight-doubling step* until every element

$p \in \mathcal{D}(\mathcal{B})$ is $\frac{1}{2k}$-heavy (see Definitions 2.10-2.12): find an $\frac{1}{2k}$-light point $p$ and double the weights of all the elements of $\mathcal{R}$ containing $p$. When there are no $\frac{1}{2k}$-light points, return a $\frac{1}{2k}$-net $C$ with respect to the final weights as the approximated solution. Since each point in $\mathcal{D}(\mathcal{B})$ is $\frac{1}{2k}$-heavy, $C$ covers all the points of $\mathcal{D}(\mathcal{B})$. Hence, if a $\frac{1}{2k}$-net of size $\mathcal{O}(kg(k))$ can be computed efficiently, this algorithm computes a solution of size $\mathcal{O}(kg(k))$. Brönnimann and Goodrich [39] showed that for a given $k$, if more than $\mu_k = 4k\log(n/k)$ weight-doubling steps are performed, then $\texttt{OPT} > 2k$, so that the correct $k$ can be computed by via doubling search. For more details, please refer to Section 2.4.2.

Note that in Brönnimann and Goodrich's algorithm [39] the weights are assigned to the boxes instead of the points of $\mathcal{D}(\mathcal{B})$. However, these weights define implicitly a weight function $w'$ over $\mathcal{D}(\mathcal{B})$: the weight $w'(p)$ of a point $p \in \mathcal{D}(\mathcal{B})$ is given by $w'(p) = \sum_{\{b \in \mathcal{B} | b \text{ contains } p\}} w(b)$. Note that a point $p \in \mathcal{D}(\mathcal{B})$ is $\frac{1}{2k}$-light with respect to $w$ if and only if $w'(p) < \frac{1}{2k} w(\mathcal{B})$. We keep track of $w'$ instead of $w$ to allow an efficient implementation of the weight-doubling steps, and the process of finding $\varepsilon$-light points. We simulate the operations over $w'$ using again a weight index, but this time with a minor variation to that of Theorem 5.6: in every node of the space partition tree, besides the $\omega, \mu$ values, we also store the minimum weight of the points within the cell corresponding to the node. During the initialization and update operations of the weight index this value can be maintained as follows: for a node $v$ with children $l, r$, the minimum weight $\texttt{min}_\omega(v)$ of a point in the cell of $v$ can be computed as $\texttt{min}_\omega(v) = \omega(v) \cdot \min\{\texttt{min}_\omega(l), \texttt{min}_\omega(r)\}$. This value allows to efficiently detect whether there are $\frac{1}{2k}$-light points, and to find one in the case of existence by tracing down, in the partition tree, the path from which that value comes.

To compute a $\frac{1}{2k}$-net, we choose a sample of $\mathcal{B}$ by performing at least $(16dk \log 16dk)$ random independent draws from $\mathcal{B}$. We then check whether it is effectively a $\frac{1}{2k}$-net, and if not, we repeat the process, up to a maximum of $\mathcal{O}(\log n)$ times. Haussler and Welzl [75] showed that such a sample is a $\frac{1}{2k}$-net with probability at least $1/2$. Thus, the expected number of samples needed to obtain a $\frac{1}{2k}$-net is constant, and since we repeat the process up to $\mathcal{O}(\log n)$ times, the probability of effectively finding one is at least $1 - \frac{1}{n^{\Omega(1)}}$. We analyze the running time of this approach in the following theorem.

**Theorem 5.8.** *Let $\mathcal{B}$ be a set of $n$ boxes in $\mathbb{R}^d$ with a minimum coverage kernel of size $\texttt{OPT}$. A coverage kernel of $\mathcal{B}$ of size within $\mathcal{O}(\texttt{OPT} \log \texttt{OPT})$ can be computed in $\mathcal{O}(\texttt{OPT} n^{\frac{d+1}{2}} \log^2 n)$-expected time, with probability within $1 - \frac{1}{n^{\Omega(1)}}$.*

*Proof.* The algorithm performs several stages guessing the value of $k$, until $k/2 < \texttt{OPT} < k$. Within each stage we initialize a weight index in time within $\mathcal{O}(n^{\frac{d+1}{2}})$. Finding whether there is a $\frac{1}{2k}$-light point can be done in constant time: the root of the partition tree stores both $w(\mathcal{D}(\mathcal{B}))$ and the minimum weight of any point in the $\omega$ and $\texttt{min}_\omega$ values, respectively. For every light point, the weight-doubling steps consume time within $\mathcal{O}\left(n \times \left(n^{\frac{d-1}{2}} \log n\right)\right) \subseteq \mathcal{O}(n^{\frac{d+1}{2}} \log n)$ (by Theorem 5.6). Since at each stage at most $4k \log(n/k)$ weight-doubling steps are performed, the total running time of each stage is within $\mathcal{O}(kn^{\frac{d+1}{2}} \log n \log \frac{n}{k}) \subseteq \mathcal{O}(kn^{\frac{d+1}{2}} \log^2 n)$. Given that $k$ increases geometrically while guessing its right value, and since the running time of each stage is a polynomial function, the sum of the running times of all the stages is

asymptotically dominated by that of the last stage, for which we have that $k \leq \texttt{OPT} \leq 2k$. Thus the result of the theorem follows. □

Compared to the algorithm of Theorem 5.7, this last approach obtains a better approximation factor on instances with small Coverage Kernels ($\mathcal{O}(\log n)$ vs. $\mathcal{O}(\log \texttt{OPT})$), but the improvement comes at a cost not only in the running time (worst case vs. expected), but in the probability of finding such a good approximation.

## 5.4 Discussion

Whether it is possible to close the gap between the factors of approximation of the Box Cover and the Orthogonal Polygon Covering problems has been a long standing open question [93]. The Minimum Coverage Kernel problem, intermediate between those two, has the potential of yielding answers in that direction, and has natural applications of its own [56, 95, 118]. Trying to understand the differences in hardness between these problems, we studied different restricted settings. We showed that while Minimum Coverage Kernel remains NP-hard under severely restricted settings, the same can be said for the Box Cover problem under even more extreme settings; and showed that while the Box Cover and Minimum Coverage Kernel can be approximated by at least the same factors, the running time of obtaining some of those approximations can be significantly improved for the Minimum Coverage Kernel problem.

**Tighter bounds.** We showed that the Minimum Coverage Kernel problem remains NP-hard even if the intersection graph of the boxes has clique-number bounded by a constant $c \geq 4$, or has vertex-degree bounded by a constant $d \geq 8$. However, it is open whether these bounds are tight: for the special cases where the intersection graph has clique-number bounded by $c \geq 3$, or vertex-degree $d \geq 7$, it remains open whether the problem is NP-hard or not. For the Box Cover problem we showed it is NP-hard even if the intersection graph of the boxes has clique-number bounded by a constant $c \geq 2$, or has bound vertex-degree $d \geq 4$. In this case, the bound for the clique-number is trivially tight: if the intersection graph has clique-number $c = 1$, then no two boxes intersect, and the solution to the problem is trivial. However, It is open whether vertex-degree bound is tight.

**Improvements in small dimensions.** In two and three dimensions, the Box Cover problem can be approximated up to a factor within $\mathcal{O}(\log \log \texttt{OPT})$ [18]. We do not know whether the running time of this algorithm can be also improved for the case of Minimum Coverage Kernel via a weight index. We omit this analysis since the approach described in Section 5.3 is relevant when the dimension of the boxes is high (while still constant), as in various applications [56, 95, 118] of the Minimum Coverage Kernel problem.

# Chapter 6

# Conclusions and Further Research

This Ph.D. thesis deals with some open questions on three groups of problems, and study their computational complexity. We provide partial answers to these open questions, and give new elements which extend the state of the art on their solutions. We summarize the results achieved for each question introduced in Chapter 1, and describe new research directions opened by this work.

## Depth Distribution: between Klee's Measure and Maximum Depth

In Chapter 3, we explored the relation between the Klee's Measure and the Maximum Depth problems. Motivated by the striking similarities between the known algorithms and the lower bound arguments, respectively, for these two problems we considered whether there was a way to formalize this relation (see Question 1 in page 3). We showed that this close relation was a hint of a hidden natural problem: the computation of the Depth Distribution of a set $\mathcal{B}$ of boxes, whose computation generalizes that of the Klee's measure, and the maximum depth, and of other central problems with no apparent relation to orthogonal boxes (such as the Integer Multiplication problem). We described how to compute the Depth Distribution in time within $\mathcal{O}(n^{\frac{d+1}{2}} \log n)$, using space within $\mathcal{O}(n \log n)$ (Theorem 3.3). Unfortunately, this running time is worse than that of computing only the Klee's measure or the maximum depth by a factor within $\mathcal{O}(\sqrt{n})$.

Studying whether the gap between the computational complexities of these problems was ineluctable, we found some insights on the relation between the Klee's Measure and Maximum Depth problems with the Matrix Multiplication and Polynomial Multiplication problems (see Question 3 in page 4). We argued the unlikeliness of computing the depth distribution of a set of boxes in the same running time as computing its Klee's measure or maximum depth, by showing that the Matrix Multiplication problem is a special case of the Depth Distribution problem. Computing the depth distribution of a set of boxes in the same time as its Klee's measure, for instance, would imply breakthrough results for a long-standing open question on the Matrix Multiplication problem (Theorem 3.11). This relation between the Depth Distribution and the Matrix Multiplication problems shows that the relation between the Klee's Measure, the

Maximum Depth and the Matrix Multiplication problems may be closer than expected: the k-Clique problem, a special case of both Klee's Measure and Maximum Depth problems, is by itself a special case of Matrix Multiplication [44], while the Matrix Multiplication problem is a special case of the Depth Distribution problem, a slight generalization of both the Klee's Measure and Maximum Depth problems.

In view of the many special cases of the instances of the Klee's Measure and the Maximum Depth problems which can be solved considerably faster than the general case, we also studied whether there were measures gradually separating the easy instances from the hard ones, and whether there were algorithms whose running times were sensitive to these measures. We described three measures of difficulty of the input instances of Depth Distribution, Klee's Measure and Maximum Depth, respectively, and introduced sensitive algorithms for each one (see Question 2 in page 3).

## Matching colored points with independent axis-aligned rectangles

In Chapter 4, we studied the computational complexity of exact and approximate solutions for the problems of finding maximum monochromatic and bichromatic matchings of points with rectangles (MonoMRM and BicMRM for short, respectively). Both problems are special cases of the Maximum Independent Set of Rectangles (MISR) problem, a classical NP-hard problem. The first question we considered was whether these problems remained NP-hard as well (see Question 4 in page 5). We showed that, both the MonoMRM and the BicMRM problems are NP-hard (in Theorems 4.1 and 4.6, respectively).

Given these two hardness results, the subsequent natural question was whether there are any constant-factor approximation algorithm or a PTAS for these two problems (see Question 5 in page 5). We showed that their optimal solutions accept better approximations than those known to this day for the MISR problem: while no constant-approximation algorithm is known for the MISR problem, we presented two 4-approximation algorithms for the MonoMRM and the BicMRM problems, respectively. These approximation algorithms also yield a 4-approximation algorithm for the problem of finding a maximum strong rectangle matching of points of the same color, studied by Bereg et al. [32]. However, the approximation ratio is smaller than the 3/2-factor described by Bereg et al. [32]. The existence of a PTAS for any of the MonoMRM or the BicMRM problem remains open.

Trying to understand when do these problems become *easy*, we studied their computational complexity under different restricted settings (see Question 6 in page 6). We provided results on this matter in two directions: we showed that these problems remain hard under restricted settings, but introduced a restricted class of instances which can be solved optimally in polynomial time (and which is key to approximate the solutions of these problems in polynomial time). More precisely, on one hand we showed that the BicMRM problem remains NP-hard, even if the points are in general position, and that the MonoMRM problem keeps being NP-hard even if the matching rectangles are restricted to axis-aligned segments, or the points are required to be in general position, or all the points have the same color (solving an open question of Bereg et al. [32]). On the other hand, we showed that if the set of rectangles induced by the input set of points is corner-complete, optimal monochromatic/bichromatic

matchings can be computed in polynomial time, which generalizes a result by Soto and Telha [124]. This result is a key component of the approximation algorithms that we introduced for the MonoMRM and the BicMRM problems.

## Computing Minimum Coverage Kernels

Finally, in Chapter 5, we studied the computational complexity of the Minimum Coverage Kernel problem, intermediate between the Box Cover and the Orthogonal Polygon Covering problems. Trying to understand the differences in hardness between these problems (see Question 7 in page 7) we studied various restricted settings. We showed that Minimum Coverage Kernel remains NP-hard under various restrictions over the intersection graph of the input set of boxes, and showed that the Box Cover problem remains NP-hard under even more extreme settings.

With respect to the approximation of these problems (see Question 8 in page 7) we showed that while the Box Cover and Minimum Coverage Kernel can be approximated within at least the same factors, the running time required to obtain some of those approximations can be significantly improved for the Minimum Coverage Kernel problem. However, it is still unknown whether for the Minimum Coverage Kernel problem, one can obtain approximation algorithms with an approximation factor closer to the $\mathcal{O}(\sqrt{\log n})$-factor achieved for the Orthogonal Polygon Covering problem.

## Future Directions of Research

Together with the research questions that were only partially answered here, this work also opens various research directions, we mention below some of them.

We argued that in two dimensions computing the depth distribution might be computationally more expensive than computing the Klee's measure or the maximum depth. For this, we introduced reductions from the Matrix Multiplication and the Integer Multiplication problems to the Depth Distribution problem. However, it is unclear whether those arguments can be extended to higher dimensions. It is open whether for special cases of the boxes (such as squares or orthants) one can obtain improved algorithms for computing the depth distribution.

We showed that the MonoMRM and BicMRM problems are NP-hard, and introduced 4-approximation polynomial-time algorithms for them. However, it is unknown whether better $O(1)$-approximation algorithm can be obtained, or whether one can argue on the unlikeliness of finding a PTAS for any of these two problems. The approximation algorithms that we introduced work only for the two dimensional case of the MonoMRM and BicMRM problems, and it is open whether these can be extended for higher dimensions. Another question that remains open is whether there are approximation algorithms take advantage of the unbalance in the color of the input points.

We showed that the Minimum Coverage Kernel problem remains NP-hard even if the intersection graph of the boxes has clique-number bounded by a constant $c \geq 4$, or has vertex-degree bounded by a constant $d \geq 8$. It is open whether these bounds are tight. For the Box Cover problem we showed it is NP-hard even if the intersection graph of the boxes has clique-number bounded by a constant $c \geq 2$, or has vertex-degree bounded by a constant $d \geq 4$. In this case, we know that the bound for the clique number is trivially tight, but it is open whether this is also the case for vertex-degree bound. Another open direction of research is whether the theoretical improvements that we achieved for the running times of approximation algorithms for the Minimum Coverage Kernel problem get reflected into practice. However, an important barrier in this direction is that there seems to be no good test data sets for this problem.

# Bibliography


[1] M. Abo Khamis, H. Q. Ngo, C. Ré, and A. Rudra. "Joins via Geometric Resolutions: Worst-case and Beyond". In: *Proceedings of the 34th ACM Symposium on Principles of Database Systems (PODS), Melbourne, Victoria, Australia, May 31 - June 4, 2015*. Ed. by T. Milo and D. Calvanese. ACM, 2015, pp. 213–228.

[2] B. M. Ábrego, E. M. Arkin, S. Fernández-Merchant, F. Hurtado, M. Kano, J. S. Mitchell, and J. Urrutia. "Matching Points with Squares". In: *Discrete & Computational Geometry (DCG)* 41.1 (2009), pp. 77–95.

[3] A. Adamaszek and A. Wiese. "Approximation Schemes for Maximum Weight Independent Set of Rectangles". In: *Proceedings of the 2013 IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS)*. 2013, pp. 400–409.

[4] P. Afshani. "Fast Computation of Output-Sensitive Maxima in a Word RAM". In: *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), Portland, Oregon, USA, January 5-7, 2014*. Ed. by C. Chekuri. SIAM, 2014, pp. 1414–1423.

[5] P. Afshani, L. Arge, and K. G. Larsen. "Higher-dimensional orthogonal range reporting and rectangle stabbing in the pointer machine model". In: *Symposuim on Computational Geometry (SoCG), Chapel Hill, NC, USA, June 17-20, 2012*. Ed. by T. K. Dey and S. Whitesides. ACM, 2012, pp. 323–332.

[6] P. Afshani, J. Barbay, and T. M. Chan. "Instance-Optimal Geometric Algorithms". In: *Journal of the ACM (JACM)* 64.1 (Mar. 2017), 3:1–3:38.

[7] P. K. Agarwal and N. H. Mustafa. "Independent set of intersection graphs of convex objects in 2D". In: *Computational Geometry (CG)* 34.2 (2006), pp. 83 –95.

[8] P. K. Agarwal, M. J. van Kreveld, and S. Suri. "Label placement by maximum independent set in rectangles". In: *Computational Geometry (CG)* 11.3-4 (1998), pp. 209–218.

[9] P. K. Agarwal. "An improved algorithm for computing the volume of the union of cubes". In: *Proceedings of the 26th ACM Symposium on Computational Geometry (SoCG), Snowbird, Utah, USA*. Ed. by D. G. Kirkpatrick and J. S. B. Mitchell. ACM, 2010, pp. 230–239.

[10] P. K. Agarwal and J. Pan. "Near-Linear Algorithms for Geometric Hitting Sets and Set Covers". In: *Proceedings of the 30th Annual Symposium on Computational Geometry (SoCG)*. New York, NY, USA: ACM, 2014, 271:271–271:279.

[11] H. Ahn and Y. Okamoto. "Adaptive Algorithms for Planar Convex Hull Problems". In: *Frontiers in Algorithmics, 4th International Workshop (FAW), Wuhan, China, August 11-13, 2010. Proceedings*. Ed. by D. Lee, D. Z. Chen, and S. Ying. Vol. 6213. Lecture Notes in Computer Science. Springer, 2010, pp. 316–326.

[12] H.-K. Ahn, S. W. Bae, E. D. Demaine, M. L. Demaine, S.-S. Kim, M. Korman, I. Reinbacher, and W. Son. "Covering points by disjoint boxes with outliers". In: *Computational Geometry (CG)* 44.3 (2011), pp. 178 –190.

[13] V. E. Alekseev, R. Boliac, D. V. Korobitsyn, and V. V. Lozin. "NP-hard graph problems and boundary classes of graphs". In: *Theoretical Computer Science (TCS)* 389.1-2 (2007), pp. 219–236.

[14] P. Alliez, O. Devillers, and J. Snoeyink. *Removing Degeneracies by Perturbing the Problem or the World*. Tech. rep. 3316. INRIA, 1997.

[15] N. Alon, R. Yuster, and U. Zwick. "Finding and Counting Given Length Cycles". In: *Algorithmica* 17.3 (1997), pp. 209–223.

[16] G. Aloupis et al. "Non-crossing matchings of points with geometric objects". In: *Computational Geometry (CG)* 46.1 (2013), pp. 78–92.

[17] D. Applegate, G. Călinescu, D. S. Johnson, H. J. Karloff, K. Ligett, and J. Wang. "Compressing rectilinear pictures and minimizing access control lists". In: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), New Orleans, Louisiana, USA, January 7-9, 2007*. 2007, pp. 1066–1075.

[18] B. Aronov, E. Ezra, and M. Sharir. "Small-Size $\varepsilon$-Nets for Axis-Parallel Rectangles and Boxes". In: *SIAM Journal on Computing (SICOMP)* 39.7 (2010), pp. 3248–3282.

[19] J. Babu, A. Biniaz, A. Maheshwari, and M. H. M. Smid. "Fixed-orientation equilateral triangle matching of point sets". In: *Theoretical Computer Science (TCS)* 555 (2014), pp. 55–70.

[20] N. Bansal and K. Pruhs. "The Geometry of Scheduling". In: *SIAM Journal on Computing (SICOMP)* 43.5 (2014), pp. 1684–1698.

[21] J. Barbay. "From Time to Space: Fast Algorithms That Yield Small and Fast Data Structures". In: *Space-Efficient Data Structures, Streams, and Algorithms - Papers in Honor of J. Ian Munro on the Occasion of His 66th Birthday (IANFEST)*. Ed. by A. Brodnik, A. Lpez-Ortiz, V. Raman, and A. Viola. Vol. 8066. Lecture Notes in Computer Science. Springer, 2013, pp. 97–111.

[22] J. Barbay and C. Kenyon. "Adaptive intersection and t-threshold problems". In: *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), San Francisco, CA, USA, January 6-8, 2002*. Ed. by D. Eppstein. ACM/SIAM, 2002, pp. 390–399.

[23] J. Barbay, P. Pérez-Lantero, and J. Rojas-Ledesma. "Computing Coverage Kernels Under Restricted Settings". In: *Computing and Combinatorics - 24th International Conference, (COCOON) 2018, Qing Dao, China, July 2-4, 2018, Proceedings*. Ed. by L. Wang and D. Zhu. Vol. 10976. Lecture Notes in Computer Science. Springer, 2018, pp. 180–191.

[24] J. Barbay, P. Pérez-Lantero, and J. Rojas-Ledesma. "Depth Distribution in High Dimensions". In: *23rd International Conference on Computing and Combinatorics (COCOON), Hong Kong, China, August 3-5, 2017, Proceedings*. Ed. by Y. Cao and J. Chen. Vol. 10392. Lecture Notes in Computer Science. Springer, 2017, pp. 38–49.

[25] J. Barbay, A. López-Ortiz, and T. Lu. "Faster Adaptive Set Intersections for Text Searching". In: *Experimental Algorithms, 5th International Workshop (WEA), Cala Galdana, Menorca, Spain, May 24-27, 2006, Proceedings*. Ed. by C. Àlvarez and M. J. Serna. Vol. 4007. Lecture Notes in Computer Science. Springer, 2006, pp. 146–157.

[26] J. Barbay, T. M. Chan, G. Navarro, and P. Pérez-Lantero. "Maximum-weight planar boxes in $O(n^2)$ time (and better)". In: *Information Processing Letters (IPL)* 114.8 (2014), pp. 437–445.

[27] R. Beigel. "Unbounded Searching Algorithms". In: *SIAM Journal on Computing (SICOMP)* 19.3 (1990), pp. 522–537.

[28] M. Ben-Or. "Lower Bounds for Algebraic Computation Trees (Preliminary Report)". In: *Proceedings of the 15th Annual ACM Symposium on Theory of Computing (STOC), 25-27 April, 1983, Boston, Massachusetts, USA*. Ed. by D. S. Johnson et al. ACM, 1983, pp. 80–86.

[29] J. L. Bentley. *Algorithms for Klee's rectangle problems*. Tech. rep. Computer Science Department, Carnegie Mellon University, 1977.

[30] J. L. Bentley and D. Wood. "An Optimal Worst Case Algorithm for Reporting Intersections of Rectangles". In: *IEEE Transactions on Computers (TRANCOMP)* 29.7 (1980), pp. 571–577.

[31] J. L. Bentley and A. C. Yao. "An Almost Optimal Algorithm for Unbounded Searching". In: *Information Processing Letters (IPL)* 5.3 (1976), pp. 82–87.

[32] S. Bereg, N. Mutsanas, and A. Wolff. "Matching points with rectangles and squares". In: *Computational Geometry (CG)* 42.2 (2009), pp. 93–108.

[33] M. d. Berg, O. Cheong, M. v. Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications (CGTA)*. 3rd ed. Springer-Verlag TELOS, 2008.

[34] P. Berman, B. DasGupta, S. Muthukrishnan, and S. Ramaswami. "Improved approximation algorithms for rectangle tiling and packing". In: *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms (SODA), January 7-9, 2001, Washington, DC, USA*. Ed. by S. R. Kosaraju. ACM/SIAM, 2001, pp. 427–436.

[35] N. Beume, C. M. Fonseca, M. López-Ibáñez, L. Paquete, and J. Vahrenhold. "On the Complexity of Computing the Hypervolume Indicator". In: *IEEE Transactions on Evolutionary Computation (TRANSEC)* 13.5 (2009), pp. 1075–1082.

[36] J. Boissonnat, M. Sharir, B. Tagansky, and M. Yvinec. "Voronoi Diagrams in Higher Dimensions under Certain Polyhedral Distance Functions". In: *Discrete & Computational Geometry (DCG)* 19.4 (1998), pp. 485–519.

[37] K. Bringmann. "An improved algorithm for Klee's measure problem on fat boxes". In: *Computational Geometry, Theory and Applications (CGTA)* 45.5-6 (2012), pp. 225–233.

[38] K. Bringmann. "Bringing Order to Special Cases of Klee's Measure Problem". In: *Mathematical Foundations of Computer Science 2013 - 38th International Symposium (MFCS), Klosterneuburg, Austria, August 26-30, 2013. Proceedings*. Ed. by K Chatterjee and J. Sgall. Vol. 8087. Lecture Notes in Computer Science (LNCS). Springer, 2013, pp. 207–218.

[39] H. Brönnimann and M. T. Goodrich. "Almost Optimal Set Covers in Finite VC-Dimension". In: *Discrete & Computational Geometry (DCG)* 14.4 (1995), pp. 463–479.

[40] L. E. Caraballo, C. Ochoa, P. Pérez-Lantero, and J. Rojas-Ledesma. "Matching colored points with rectangles". In: *Journal of Combinatorial Optimization (JOCO)* 33.2 (2017), pp. 403–421.

[41] P. Chalermsook. "Coloring and Maximum Independent Set of Rectangles". In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Vol. 6845. LNCS. Springer Berlin Heidelberg, 2011, pp. 123–134.

[42] P. Chalermsook and J. Chuzhoy. "Maximum independent set of rectangles". In: *Proc. of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Philadelphia, USA, 2009, pp. 892–901.

[43] T. M. Chan. "A note on maximum independent sets in rectangle intersection graphs". In: *Information Processing Letters (IPL)* 89.1 (2004), pp. 19 –23.

[44] T. M. Chan. "A (slightly) faster algorithm for Klee's Measure Problem". In: *Proceedings of the 24th ACM Symposium on Computational Geometry (SoCG), College Park, MD, USA, June 9-11, 2008*. Ed. by M. Teillaud. ACM, 2008, pp. 94–100.

[45] T. M. Chan. "Klee's Measure Problem Made Easy". In: *54th Annual IEEE Symposium on Foundations of Computer Science (FOCS), Berkeley, CA, USA, 26-29 October, 2013*. IEEE Computer Society, 2013, pp. 410–419.

[46] T. M. Chan. "Polynomial-time approximation schemes for packing and piercing fat objects". In: *Journal of Algorithms (JALG)* 46.2 (2003), pp. 178–189.

[47] T. M. Chan and S. Har-Peled. "Approximation Algorithms for Maximum Independent Set of Pseudo-Disks". In: *Discrete & Computational Geometry (DCG)* 48.2 (2012), pp. 373–392.

[48] T. M. Chan and N. Hu. "Geometric Red-Blue Set Cover for Unit Squares and Related Problems". In: *Proceedings of the 25th Canadian Conference on Computational Geometry, CCCG 2013, Waterloo, Ontario, Canada, August 8-10, 2013*. 2013.

[49] T. M. Chan, K. G. Larsen, and M. Patrascu. "Orthogonal range searching on the RAM, revisited". In: *Proceedings of the 27th ACM Symposium on Computational Geometry (SoCG), Paris, France, June 13-15, 2011*. Ed. by F. Hurtado and M. J. van Kreveld. ACM, 2011, pp. 1–10.

[50] K. L. Clarkson and K. R. Varadarajan. "Improved Approximation Algorithms for Geometric Set Cover". In: *Discrete & Computational Geometry (DCG)* 37.1 (2007), pp. 43–58.

[51] C. R. Cook and D. J. Kim. "Best Sorting Algorithm for Nearly Sorted Lists". In: *Communications of the ACM (CACM)* 23.11 (1980), pp. 620–624.

[52] D. Coppersmith and S. Winograd. "Matrix Multiplication via Arithmetic Progressions". In: *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC), 1987, New York, New York, USA*. Ed. by A. V. Aho. ACM, 1987, pp. 1–6.

[53] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms (3. ed.)* MIT Press, 2009.

[54] J. R. Correa, L. Feuilloley, P. Pérez-Lantero, and J. A. Soto. "Independent and Hitting Sets of Rectangles Intersecting a Diagonal Line: Algorithms and Complexity". In: *Discrete & Computational Geometry (DCG)* 53.2 (2015), pp. 344–365.

[55] J. C. Culberson and R. A. Reckhow. "Covering Polygons Is Hard". In: *Journal of Algorithms (JALG)* 17.1 (1994), pp. 2–44.

[56] J. Daly, A. X. Liu, and E. Torng. "A Difference Resolution Approach to Compressing Access Control Lists". In: *IEEE/ACM Transactions on Networking (TON)* 24.1 (2016), pp. 610–623.

[57] F. d'Amore, V. H. Nguyen, T. Roos, and P. Widmayer. "On Optimal Cuts of Hyper-rectangles". In: *Computing* 55.3 (1995), pp. 191–206.

[58] E. D. Demaine, A. López-Ortiz, and J. I. Munro. "Adaptive set intersections, unions, and differences". In: *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), San Francisco, CA, USA, January 9-11, 2000.* Ed. by D. B. Shmoys. ACM/SIAM, 2000, pp. 743–752.

[59] R. G. Downey and M. R. Fellows. "Fixed-Parameter Tractability and Completeness I: Basic Results". In: *SIAM Journal on Computing (SICOMP)* 24.4 (1995), pp. 873–921.

[60] A. Dumitrescu and R. Kaye. "Matching colored points in the plane: Some new results". In: *Computational Geometry (CG)* 19.1 (2001), pp. 69 –85.

[61] A Dumitrescu and W. L. Steiger. "On a matching problem in the plane". In: *Discrete Mathematics (DM)* 211 (2000), pp. 183–195.

[62] H. Edelsbrunner. “A new approach to rectangle intersections part I”. In: 13.3-4 (1983), pp. 209–219.

[63] T. Erlebach, K. Jansen, and E. Seidel. “Polynomial-Time Approximation Schemes for Geometric Intersection Graphs”. In: *SIAM Journal on Computing (SICOMP)* 34.6 (2005), pp. 1302–1323.

[64] V. Estivill-Castro and D. Wood. “A Survey of Adaptive Sorting Algorithms”. In: *ACM Computing Surveys (ACMCS)* 24.4 (1992), pp. 441–476.

[65] U. Feige. “A Threshold of ln $n$ for Approximating Set Cover”. In: *Journal of the ACM (JACM)* 45.4 (1998), pp. 634–652.

[66] S. P. Fekete, K. Huang, J. S. B. Mitchell, O. Parekh, and C. A. Phillips. “Geometric Hitting Set for Segments of Few Orientations”. In: *Approximation and Online Algorithms - 13th International Workshop, WAOA 2015, Patras, Greece, September 17-18, 2015. Revised Selected Papers*. Ed. by L. Sanità and M. Skutella. Vol. 9499. Lecture Notes in Computer Science. Springer, 2015, pp. 145–157.

[67] R. J. Fowler, M. Paterson, and S. L. Tanimoto. “Optimal Packing and Covering in the Plane are NP-Complete”. In: *Information Processing Letters (IPL)* 12.3 (1981), pp. 133–137.

[68] M. L. Fredman and B. W. Weide. “On the Complexity of Computing the Measure of $\cup_1^n[a_i, b_i]$”. In: *Communications of the ACM (CACM)* 21.7 (1978), pp. 540–544.

[69] M. Fürer. “Faster Integer Multiplication”. In: *SIAM Journal on Computing (SICOMP)* 39.3 (2009), pp. 979–1005.

[70] F. L. Gall. “Powers of tensors and fast matrix multiplication”. In: *International Symposium on Symbolic and Algebraic Computation (ISSAC), Kobe, Japan, July 23-25, 2014*. Ed. by K. Nabeshima, K. Nagasaka, F. Winkler, and Á. Szántó. ACM, 2014, pp. 296–303.

[71] M. C. Golumbic. “{CHAPTER} 8 - Interval Graphs”. In: *Algorithmic Graph Theory and Perfect Graphs*. Ed. by M. C. Golumbic. Academic Press, 1980, pp. 171 –202.

[72] M. Grötschel, L. Lovász, and A. Schrijver. “Polynomial Algorithms for Perfect Graphs”. In: *Topics on Perfect Graphs*. Vol. 88. 1984, pp. 325 –356.

[73] D. Harvey, J. van der Hoeven, and G. Lecerf. “Even faster integer multiplication”. In: *Journal of Complexity (JOC)* 36 (2016), pp. 1–30.

[74] R. Hassin and N. Megiddo. “Approximation algorithms for hitting objects with straight lines”. In: *Discrete Applied Mathematics (DAM)* 30.1 (1991), pp. 29–42.

[75] D. Haussler and E. Welzl. “epsilon-Nets and Simplex Range Queries”. In: *Discrete & Computational Geometry (DCG)* 2 (1987), pp. 127–151.

[76] D. S. Hochbaum and W. Maass. “Approximation Schemes for Covering and Packing Problems in Image Processing and VLSI”. In: *Journal of the ACM (JACM)* 32.1 (1985), pp. 130–136.

[77] H. Imai and T. Asano. "Finding the Connected Components and a Maximum Clique of an Intersection Graph of Rectangles in the Plane". In: *Journal of Algorithms (JALG)* 4.4 (1983), pp. 310–323.

[78] D. S. Johnson. "Approximation Algorithms for Combinatorial Problems". In: *Journal of Computer and System Sciences (JCSS)* 9.3 (1974), pp. 256–278.

[79] A. Joshi and N. S. Narayanaswamy. "Approximation Algorithms for Hitting Triangle-Free Sets of Line Segments". In: *Algorithm Theory - 14th Scandinavian Symposium and Workshops (SWAT), Copenhagen, Denmark, July 2-4, 2014. Proceedings*. Ed. by R. Ravi and I. L. Gørtz. Vol. 8503. Lecture Notes in Computer Science. Springer, 2014, pp. 357–367.

[80] H. Kaplan, N. Rubin, M. Sharir, and E. Verbin. "Counting colors in boxes". In: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), New Orleans, Louisiana, USA, January 7-9, 2007*. 2007, pp. 785–794.

[81] A. Karatsuba and Y. Ofman. "Multiplication of Multidigit Numbers on Automata". In: *Soviet Physics-Doklady* 7 (1963), pp. 595–596.

[82] R. M. Karp. "Reducibility Among Combinatorial Problems". In: *Proceedings of a symposium on the Complexity of Computer Computations (COCO), held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York.* Ed. by R. E. Miller and J. W. Thatcher. The IBM Research Symposia Series. Plenum Press, New York, 1972, pp. 85–103.

[83] M. J. Katz, J. S. B. Mitchell, and Y. Nir. "Orthogonal segment stabbing". In: *Computational Geometry (CG)* 30.2 (2005), pp. 197–205.

[84] S. Khanna, S. Muthukrishnan, and M. Paterson. "On Approximating Rectangle Tiling and Packing". In: *Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms (SODA)*. Vol. 95. SIAM. 1998, p. 384.

[85] D. G. Kirkpatrick and R. Seidel. "Output-size sensitive algorithms for finding maximal vectors". In: *Proceedings of the First Annual Symposium on Computational Geometry (SoCG), Baltimore, Maryland, USA, June 5-7, 1985*. Ed. by J. O'Rourke. ACM, 1985, pp. 89–96.

[86] D. G. Kirkpatrick and R. Seidel. "The Ultimate Planar Convex Hull Algorithm?" In: *SIAM Journal on Computing (SICOMP)* 15.1 (1986), pp. 287–299.

[87] V. Klee. "Can the Measure of $\cup_1^n[a_i, b_i]$ be Computed in Less Than $O(n \log n)$ Steps?" In: 84.4 (1977), pp. 284–285.

[88] J. M. Kleinberg and É. Tardos. *Algorithm design*. Addison-Wesley, 2006.

[89] D. E. Knuth. *The Art of Computer Programming, Volume III: Sorting and Searching, 2nd Edition*. Addison Wesley Longman Publishing Co., Inc., 1998.

[90] D. E. Knuth and A. Raghunathan. "The Problem of Compatible Representatives". In: *SIAM Journal on Discrete Mathematics (JDM)* 5.3 (1992), pp. 422–427.

[91] D. E. Knuth. *The Art of Computer Programming, Volume II: Seminumerical Algorithms, 3rd Edition*. Addison-Wesley, 1998.

[92] J. Kratochvíl and J. Nešetřil. "Independent set and clique problems in intersection-defined classes of graphs". In: *Commentationes Mathematicae Universitatis Carolinae (CMUC)* 31.1 (1990), pp. 85–93.

[93] V. S. A. Kumar and H. Ramesh. "Covering Rectilinear Polygons with Axis-Parallel Rectangles". In: *SIAM Journal on Computing (SICOMP)* 32.6 (2003), pp. 1509–1541.

[94] V. S. A. Kumar, S. Arya, and H. Ramesh. "Hardness of Set Cover with Intersection 1". In: *27th International Colloquium on Automata, Languages and Programming (ICALP), Geneva, Switzerland, July 9-15, 2000, Proceedings*. Ed. by U. Montanari, J. D. P. Rolim, and E. Welzl. Vol. 1853. Lecture Notes in Computer Science. Springer, 2000, pp. 624–635.

[95] L. V. S. Lakshmanan, R. T. Ng, C. X. Wang, X. Zhou, and T. Johnson. "The Generalized MDL Approach for Summarization". In: *Proceedings of 28th International Conference on Very Large Data Bases (VLDB), August 20-23, 2002, Hong Kong, China*. Morgan Kaufmann, 2002, pp. 766–777.

[96] D. T. Lee and C. K. Wong. "Finding Intersection of Rectangles by Range Search". In: *Journal of Algorithms (JALG)* 2.4 (1981), pp. 337–347.

[97] C. Levcopoulos and J. Gudmundsson. "Approximation Algorithms for Covering Polygons with Squares and Similar Problems". In: *Randomization and Approximation Techniques in Computer Science, International Workshop, RANDOM'97, Bolognna, Italy, July 11-12. 1997, Proceedings*. Ed. by J. D. P. Rolim. Vol. 1269. Lecture Notes in Computer Science. Springer, 1997, pp. 27–41.

[98] C. Levcopoulos and O. Petersson. "Adaptive Heapsort". In: *Journal of Algorithms (JALG)* 14.3 (1993), pp. 395–413.

[99] L. Lewin-Eytan, J. Naor, and A. Orda. "Admission Control in Networks with Advance Reservations". In: *Algorithmica* 40.4 (2004), pp. 293–304.

[100] D. R. Lick and A. T. White. "k-Degenerate graphs". In: *Canadian Journal of Mathematics (CJM)* 22 (1970), pp. 1082–1096.

[101] L. Lovász. "On the ratio of optimal integral and fractional covers". In: *Discrete Mathematics (DM)* 13.4 (1975), pp. 383–390.

[102] A. Lubiw. "A weighted min-max relation for intervals". In: *Journal of Combinatorial Theory (JCT), Series B* 53.2 (1991), pp. 151–172.

[103] C. Lund and M. Yannakakis. "On the Hardness of Approximating Minimization Problems". In: *Journal of the ACM (JACM)* 41.5 (1994), pp. 960–981.

[104] H. Mannila. "Measures of Presortedness and Optimal Sorting Algorithms". In: vol. 34. 4. 1985, pp. 318–325.

[105] J. Matoušek. “Cutting Hyperplane Arrangements”. In: *Discrete & Computational Geometry (DCG)* 6 (1991), pp. 385–406.

[106] J. Matoušek. “Reporting Points in Halfspaces”. In: *Computational Geometry (CG)* 2 (1992), pp. 169–186.

[107] D. W. Matula and L. L. Beck. “Smallest-last Ordering and Clustering and Graph Coloring Algorithms”. In: *Journal of the ACM (JACM)* 30.3 (July 1983), pp. 417–427.

[108] A. Moffat and O. Petersson. “A Framework for Adaptive Sorting”. In: *Discrete Applied Mathematics (DAM)* 59.2 (1995), pp. 153–179.

[109] A. Moffat, O. Petersson, and N. C. Wormald. “Sorting and/by Merging Finger Trees”. In: *Third International Symposium on Algorithms and Computation, (ISAAC), Nagoya, Japan, December 16-18, 1992, Proceedings*. Ed. by T. Ibaraki, Y. Inagaki, K. Iwama, T. Nishizeki, and M. Yamashita. Vol. 650. Lecture Notes in Computer Science. Springer, 1992, pp. 499–508.

[110] W. Mulzer and G. Rote. “Minimum-weight triangulation is NP-hard”. In: *Journal of the ACM (JACM)* 55.2 (2008).

[111] N. H. Mustafa and S. Ray. “Improved Results on Geometric Hitting Set Problems”. In: *Discrete & Computational Geometry (DCG)* 44.4 (2010), pp. 883–895.

[112] V. H. Nguyen and P. Widmayer. “Binary Space Partitions for Sets of Hyperrectangles”. In: *Algorithms, Concurrency and Knowledge: 1995 Asian Computing Science Conference (ACSC), Pathumthani, Tailand, December 11-13, 1995, Proceedings*. 1995, pp. 59–72.

[113] F. Nielsen. “Fast stabbing of boxes in high dimensions”. In: *Theoretical Computer Science (TCS)* 246.1-2 (2000), pp. 53–72.

[114] M. H. Overmars and C. Yap. “New Upper Bounds in Klee’s Measure Problem”. In: *SIAM Journal on Computing (SICOMP)* 20.6 (1991), pp. 1034–1045.

[115] D. Papadias, Y. Tao, G. Fu, and B. Seeger. “An Optimal and Progressive Algorithm for Skyline Queries”. In: *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, San Diego, California, USA, June 9-12, 2003*. Ed. by A. Y. Halevy, Z. G. Ives, and A. Doan. ACM, 2003, pp. 467–478.

[116] C. H. Papadimitriou. *Computational complexity*. Academic Internet Publ., 2007.

[117] F. P. Preparata and M. I. Shamos. *Computational Geometry - An Introduction*. Texts and Monographs in Computer Science. Springer, 1985.

[118] K. Q. Pu and A. O. Mendelzon. “Concise descriptions of subsets of structured sets”. In: *ACM Transactions on Database Systems (TODS)* 30.1 (2005), pp. 211–248.

[119] N. Rajgopal, P. Ashok, S. Govindarajan, A. Khopkar, and N. Misra. “Hitting and Piercing Rectangles Induced by a Point Set”. In: *Computing and Combinatorics (CC)*. Vol. 7936. LNCS. 2013, pp. 221–232.

[120] C. S. Rim and K. Nakajima. "On rectangle intersection and overlap graphs". In: *IEEE Transactions on Circuits and Systems* 42.9 (1995), pp. 549–553.

[121] L. Roditty and V. V. Williams. "Subquadratic time approximation algorithms for the girth". In: *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), Kyoto, Japan, January 17-19, 2012*. Ed. by Y. Rabani. SIAM, 2012, pp. 833–845.

[122] A. Schönhage and V. Strassen. "Schnelle Multiplikation großer Zahlen". In: *Computing* 7.3-4 (1971), pp. 281–292.

[123] R. Seidel. "Constructing Higher-Dimensional Convex Hulls at Logarithmic Cost per Face". In: *Proceedings of the 18th Annual ACM Symposium on Theory of Computing (STOC), May 28-30, 1986, Berkeley, California, USA*. Ed. by J. Hartmanis. ACM, 1986, pp. 404–413.

[124] J. A. Soto and C. Telha. "Jump Number of Two-Directional Orthogonal Ray Graphs". In: *Integer Programming and Combinatorial Optimization (IPCO)*. Vol. 6655. LNCS. Springer Berlin Heidelberg, 2011, pp. 389–403.

[125] V. Strassen. "Gaussian Elimination is Not Optimal". In: *Numerische Mathematik* 13.4 (Aug. 1969), pp. 354–356.

[126] H. Yildiz and S. Suri. "On Klee's measure problem for grounded boxes". In: *Symposuim on Computational Geometry (SoCG), Chapel Hill, NC, USA, June 17-20, 2012*. Ed. by T. K. Dey and S. Whitesides. ACM, 2012, pp. 111–120.

[127] H. Yildiz, J. Hershberger, and S. Suri. "A Discrete and Dynamic Version of Klee's Measure Problem". In: *Proceedings of the 23rd Annual Canadian Conference on Computational Geometry (CCCG), Toronto, Ontario, Canada, August 10-12, 2011*. 2011.

[128] H. Yildiz, L. Foschini, J. Hershberger, and S. Suri. "The Union of Probabilistic Boxes: Maintaining the Volume". In: *19th Annual European Symposium on Algorithms (ESA), Saarbrücken, Germany, September 5-9, 2011. Proceedings*. Ed. by C. Demetrescu and M. M. Halldrsson. Vol. 6942. Lecture Notes in Computer Science. Springer, 2011, pp. 591–602.

[129] H. Yu. "An Improved Combinatorial Algorithm for Boolean Matrix Multiplication". In: *42nd International Colloquium on Automata, Languages, and Programming (ICALP), Proceedings, Part I, Kyoto, Japan*. Ed. by M. M. Halldórsson, K. Iwama, N. Kobayashi, and B. Speckmann. Vol. 9134. Lecture Notes in Computer Science. Springer, 2015, pp. 1094–1105.

[130] D. Zuckerman. "Linear Degree Extractors and the Inapproximability of Max Clique and Chromatic Number". In: *Theory of Computing (TOC)* 3.1 (2007), pp. 103–128.