ON COVERING POINTS WITH CONICS AND STRIPS IN THE PLANE

A Thesis

by

PRAVEEN TIWARI

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Jianer Chen
Committee Members,	Jennifer L. Welch
	Joseph M. Landsberg
Head of Department,	Duncan M. Walker

December 2012

Major Subject: Computer Science

ABSTRACT

Geometric covering problems have always been of focus in computer scientific research. The generic geometric covering problem asks to cover a set $\mathcal{S}$ of n objects with another set of objects whose cardinality is minimum, in a geometric setting. Many versions of geometric cover have been studied in detail, one of which is line cover: Given a set of points in the plane, find the minimum number of lines to cover them. In Euclidean space $\mathcal{R}^m$, this problem is known as Hyperplane Cover, where lines are replaced by affine hyperplanes bounded by dimension d. Line cover is $\mathbb{NP}$-hard, so is its hyperplane analogue. Our thesis focuses on few extensions of hyperplane cover and line cover.

One of the techniques used to study $\mathbb{NP}$-hard problems is Fixed Parameter Tractability (FPT), where, in addition to input size, a parameter k is provided for input instance. We ask to solve the problem with respect to k, such that the running time is a function in both n and k, strictly polynomial in n, while the exponential component is limited to k. In this thesis, we study FPT and parameterized complexity theory, the theory of classifying hard problems involving a parameter k.

We focus on two new geometric covering problems: covering a set of points in the plane with conics (conic cover) and covering a set of points with strips or fat lines of given width in the plane (fat line cover). A conic is a non-degenerate curve of degree two in the plane. A *fat line* is defined as a strip of finite width w. In this dissertation, we focus on the parameterized versions of these two problems, where, we are asked to cover the set of points with k conics or k fat lines. We use the existing techniques of FPT algorithms, kernelization and approximation algorithms to study these problems. We do a comprehensive study of these problems, starting with $\mathbb{NP}$-hardness results to studying their parameterized hardness in terms of parameter k.

We show that conic cover is fixed parameter tractable, and give an algorithm of running time $O^*\left((k/1.38)^{4k}\right)$, where, O^* implies that the running time is some polynomial in input size. Utilizing special properties of a parabola, we are able to achieve a faster algorithm and show a running time of $O^*\left((k/1.15)^{3k}\right)$.

For fat line cover, first we establish its $\mathbb{NP}$-hardness, then we explore algorithmic possibilities with respect to parameterized complexity theory. We show $W[1]$-hardness of fat line cover with respect to the number of fat lines, by showing a parameterized reduction from the problem of stabbing axis-parallel squares in the plane. A parameterized reduction is an algorithm which transforms an instance of one parameterized problem into an instance of another parameterized problem using a FPT-algorithm. In addition, we show that some restricted versions of fat line cover are also $W[1]$-hard. Further, in this thesis, we explore a restricted version of fat line cover, where the set of points are integer coordinates and allow only axis-parallel lines to cover them. We show that the problem is still $\mathbb{NP}$-hard. We also show that this version is fixed parameter tractable having a kernel size of $O\left(k^2\right)$ and give a FPT-algorithm with a running time of $O^*\left(3^k\right)$. Finally, we conclude our study on this problem by giving an approximation algorithm for this version having a constant approximation ratio 2.

To my parents, who have always been supportive of my endeavor.

ACKNOWLEDGMENTS

A word of thanks to my advisor Dr. Jianer Chen for being a constant source of great ideas as well as providing me with a positive attitude and motivation throughout the course of this research.

I would also like to thank Wenjun Li for his helpful insights and inputs which contributed to the culmination of this work.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Algorithmic research aims at finding logical and provable ways to solve some of the most difficult problems which exist today. Computer science and technology serves as a powerful tool in speeding up this quest. Using computational methods, we have been able to crack problems which seemed to be unsolvable few decades ago. A certain class of problems though, still challenge us to find ways to achieve a solution faster than trivial enumeration. In fact, the most important problem in computer science till date, $\mathbb{P} \stackrel{?}{=} \mathbb{NP}$ [1], is based on this notion: Is solving a problem as easy as checking a given solution? In other words, given a possible solution to a problem, we can always answer very quickly whether it is a valid solution or not. Is it possible to find that solution fast enough, without having to enumerate all the possible solutions? Intuitively, the answer seems to be negative, but no one has been able to formally prove either way. The theory of $\mathbb{NP}$-completeness, originated from the works of Cook [2], Karp [3] et al. has laid foundation for our understanding of these kind of problems and we can now classify these problems based on their hardness.

Once $\mathbb{NP}$-hardness of a given problem is established, we focus on finding alternate ways to work on it. Following are few of the common approaches:

- Find a solution which is close to optimal, but still useful and acceptable for practical purposes, called approximation algorithms,
- Explore the feasibility of a faster algorithm by introducing a natural parameter in the problem which is independent of the input size, called fixed parameter tractable (FPT) algorithms,
- Utilize a uniformly random input set to answer with a certainty the average

performance of a solution, called randomized algorithms,

- Learning from the input set and utilize these heuristics to solve the problem at hand. The solution here, might not be acceptable for general case.

Each of these approaches tries to establish some sort of guarantee on the efficiency of the result with respect to the trivial brute force method, either in terms of the solution itself or its running time. This dissertation primarily studies techniques of FPT-algorithms for solving problems.

A. Motivation

Curve fitting to scattered data is a well-known and quite significant problem in Computational Geometry. Here, one wishes to find methods of covering a set of experimental data points with a curve of best fit. This problem commonly occurs in any kind of prediction based models involving trends and patterns, machine learning, with applications in economics, bio-informatics, etc. Since we are heavily dependent on measured data, the complexity of the problem increases with the amount of data. Hence, it is natural to look for faster methods of achieving a close and approximate fit. An extension to curve fitting could be to fit the data with k curves or any other object in general, instead of just one. This direction is the focus of this research work.

Various forms of the problem of covering objects with objects in geometric settings have been discussed in literature. Each problem defines what covering means in different ways. For example, for closed connected objects, any object A within the region enclosed by another closed object B is said to be covered by B. Similarly, for geometric covering of points with lines, a point is said to be covered by a line if it lies on the line. Each such definition gives rise to a new version of geometric cover. These versions range from restricted models of well known $\mathbb{NP}$-hard problems

like Rectilinear Euclidean Traveling Salesman problem [4], Rectilinear Steiner Tree Problem [5], Facility Location [6], etc. to application specific problems in Robotics, Statistical Analysis [7], VLSI design, radiotherapy [8] etc. Langerman et al. [9] define an abstract model of covering objects applicable in any geometric setting in order to answer questions related to its hardness. In a series of publications [8], [10], [11], Megiddo et al. studied the complexity of different geometric cover problems, ranging from covering different shapes with respect to lines, to geometric location problems like p-center, p-median, facility location, etc.

Another aspect of geometric cover is of stabbing points with objects: given a set of objects in the plane, find minimum number of lines which intersect or 'stab' them in the plane [12], [13] (we discuss this problem in more detail in Chapter III). Shape fitting is another common geometric covering problem which occurs in computer vision, graphics and modeling algorithms. To achieve a faster approximation for these problems, the idea of core sets has been introduced [14]. Intuitively, we choose a smaller set Q of points in the input set P, and fit a shape S perfectly by using an inefficient algorithm. Then, we try to extend the shape to cover points from the rest of the set $P \setminus Q$ as well. The goal now is to find this smaller set Q, known as core set, so that S can be extended efficiently to fit all the points in P. This idea is similar to finding a FPT kernel (FPT kernel is discussed in detail in Chapter II) and has been utilized for approximating size of core sets [15].

Another related problem is to cover a set of points S with another known set S' of objects (lines). The concept of ϵ-nets is one way to approach these problems. Suppose there is a set of objects S. We wish to divide these objects into a set of groups, say A, based on certain properties. Naturally, there would be overlaps among groups. Then we ask, find the smallest subset of S such that each object is there in at least one group in A. Such sets are called ϵ-nets, and the problem is to bound the

size of ϵ-nets, where ϵ is the parameter expressed in terms of size of the input. An example could be a database table, where we wish to group all the rows by certain columns in a way that in the final query result, each row has some value in these selected columns. This problem is also related to shape fitting of objects to set of points. There have been attempts to establish bounds on the size of ϵ-nets [16], [17], and the latest result sets a lower bound of $\Omega\left((1/\epsilon)\left(\omega\left(1/\epsilon\right)\right)\right)$ [16], where $\omega\left(\right)$ is a slow function related to inverse of Ackermann function.

One of the basic geometric cover problems seeks to cover a set of points with lines in the plane, also known as LINE COVER (LC). More formally, given n points in $\mathbb{R}^2$, LC asks to find k lines required to cover these points. This problem, originally referred to as Point Covering [10], has been known to be $\mathbb{NP}$-hard for decades, [10]. LC has also been considered in higher dimensions, in the form of HYPERPLANE COVER (HC), to find k hyperplanes, bounded by dimension d, that cover a given set of n points in Euclidean Space.

HC has been studied in detail both with respect to approximation algorithms [18], as well as parameterized complexity, [9]. An exact algorithm with respect to number of lines as a parameter is known, [19], which also is the best known algorithm for LC. Our research considers new versions of line cover, which are its natural extensions and have potential practical applications. In this dissertation, we discuss the following two general problems:

1. Covering set of points in general position with conics in the plane (Conic Cover).

2. Covering set of points in the plane with strips of fixed width (Fat Line Cover).

With respect to parameterized complexity theory, no results are known for these problems. This gives us motivation to explore new horizons and achieve results that hopefully open a new line of research. Our focus is on answering questions with re-

spect to hardness of these problems (and also, some of their special cases) and explore algorithmic possibilities in terms of Parameterized Complexity Theory, specifically, FPT. In this section, we formally introduce all the problems discussed in this dissertation, and provide an outline of the organization of this thesis at the end.

B. Covering Points with Conics

Classically, a conic section (or just conic) is defined as the non-degenerate curve of intersection of a plane with a cone. By slicing the cone with plane in different ways, we can obtain one of the following three conics: the parabola, the ellipse and the hyperbola. A circle is generally treated as a special case of ellipse where the intersecting plane is perpendicular to the axis of cone. Formally, a conic is the locus of a point p which maintains a fixed ratio, known as eccentricity e, between its distance from some fixed point, known as focus f, and its distance from a fixed line D, known as the directrix [20]. The value of e determines the kind of conic section, viz. an ellipse ($0 < e < 1$), a parabola ($e = 1$) or a hyperbola ($e > 1$). Problems related to conic sections have been of importance since ancient times and studied in great detail by Euclid, Archimedes, Apollonius, Kepler, Pascal, Descartes, etc.

While HC generalizes LC to higher dimensional space, we wish to try to generalize LC with respect to higher degree curves instead of lines. In particular, given a set of points in the plane in general position, we ask to find k conics to cover them. A point is said to be covered by a conic if it lies on the curve.

Definition 1 (CONIC COVER)**.** *Given a set S of n points in the plane, find k conics required to cover them.*

This problem may be considered as an extension of curve fitting where the relation is not a linear fit. For some problems in the field of Machine Learning, Rec-

ommender Systems, Robotics, etc. we wish to fit the data with a relation of second degree. In other words, instead of a linear curve, we wish to fit data with a conic in the plane. While these practical problems require only single curve to fit the data, we seek the answer in terms of a parameter k. We use the necessary and sufficient conditions to represent a conic by an equation of second degree to extend the idea of solving HC, and achieve a parameterized algorithm of running time of $O^*\left((k/1.38)^{4k}\right)$.

Definition 2 (PARABOLA COVER)**.** *Given a set S of n points on a plane, find k parabolas required to cover them.*

A parabola has some special properties which, we show in Chapter IV, would help us obtain a faster algorithm, when compared with general conics, with respect to covering points in the plane. We give an FPT algorithm with a faster running time for Parabola Cover: $O^*\left((k/1.15)^{3k}\right)$.

C. Covering Points with Fat Lines

This problem arises from an amalgamation of linear regression, shape fitting, ϵ-nets and line cover. In practical experiments, especially those involving large sets of data, observations are often prone to errors. The errors might occur due to a limitation in precision or some deviation(s) from established relations. Discussed by Guibas et al. in [21], the concept of ϵ-geometry as an aspect of precision related geometry was introduced. Their work aims at providing faster ways of computation in geometric settings.

Suppose there are n points in an input set, $n-1$ of which are collinear. The remaining point, say p, does not lie exactly on the line through rest of $n-1$ points, but is only at a distance ϵ from that line. It is quite possible that this ϵ distance is due to precision, or lack thereof, in measurements, and p could actually have been

on the line of best fit. Now, for this hypothetical problem, the line cover solution of cardinality two is not something useful. A better solution, it could be argued, would allow p an ϵ perturbation so that it becomes collinear with rest of the points in the set. In general, we want to imagine a set of k lines which cover 'most' of the n points in input set S, and, are such that each of the rest of the points is at a distance no more than ϵ from at least one of these lines, and hence defined as covered by such a line. We can thus imagine a line which is no more one dimensional, instead, it has a finite width w (where $w \leq 2\epsilon$) such that these k lines cover all the points in S. We call such lines *fat lines*.

Definition 3 (FAT LINE)**.** *A fat line represented as $L : (l_1, l_2, w)$ is defined as a line or strip of finite width w in the plane. In other words, it is the region between two parallel lines in the plane which are at a distance w from each other.*

Definition 4 (FAT LINE COVER (FLC))**.** *Given a set S of n points in a plane, find k fat lines of width w to cover S.*

This thesis focuses on significance of this problem from the perspective of computational complexity. By showing a polynomial-time reduction from line cover, we show that FLC is indeed $\mathbb{NP}$-hard. By showing a parameterized reduction from stabbing circles in the plane, we establish $W[1]$-hardness of FLC with respect to the number of fat lines even when the fat lines are restricted to being axis parallel and points are arbitrary.

Next, we consider a restricted version of FLC and explore its hardness and possibility of a FPT algorithm, where we restrict the set of points to be integral coordinates, and fat lines to be only axis parallel and of fixed width.

Definition 5 (RECTILINEAR FAT LINE COVER WITH INTEGRAL COORDINATES (RFLC))**.** *Given a set S of n points in a plane such that the coordinates of*

each point are integers, find a minimum number of rectilinear fat lines to cover S, each fat line having a constant width $w = c\,(\geq 1)$.

The line cover version ($c = 0$) of this restricted form has been proven to be polynomial time solvable [8], but it is still an open question whether this restricted fat line cover is hard at all. We explore both the hardness, as well as possibility of efficient algorithms for this problem. We consider the case when $w = 1$, for which we prove that even this version is $\mathbb{NP}$-hard, give an FPT algorithm solvable in $O\left(n^2 \cdot 3^k\right)$ time with kernel size $O\left(k^2\right)$ and an approximation algorithm with constant ratio.

D. Organization

In this thesis, we do a systematic study of Conic Cover and FLC, and their special cases as defined above. The work is organized as follows: Chapter I introduces the problems covered in this work along with the motivation and background behind studying them. Chapter II reviews some fundamental concepts of theoretical computer science, specifically $\mathbb{NP}$-Completeness theory, Parameterized Complexity Theory and Approximation Algorithms. To lay a foundation for results in later chapters, we discuss some earlier results and key ideas in Chapter III. Chapter IV is devoted to finding FPT algorithms for general Conic Cover and Parabola Cover.

Chapter V deals with FLC. It first discusses a problem from Computational Geometry: of covering a set of points in the plane with one single fat line, with a hope to get a direction for solving FLC. Next, we establish NP-hardness of FLC. We do a survey of the existing results on geometric covering with respect to parameterized complexity and derive W[1]-hardness of FLC. With an aim to find a natural direction to work on for FLC, we explore the hardness, fixed parameter tractability, kernelization and approximability of RFLC. In the last chapter, we conclude our work

discussing possible directions to explore in future.

CHAPTER II

PRELIMINARIES

A. $\mathbb{NP}$-Completeness Theory

The theory of $\mathbb{NP}$-Completeness aims at differentiating the problems which seem easy to solve from the ones which have no known solution better than trivial enumeration. This classification is done in terms of running time involved in finding a solution. A problem Q is known as a decision problem if for each input instance $x \in Q$, one only needs to know whether x is a 'yes' or 'no' instance of Q. Q is said to belong to complexity class $\mathbb{P}$ if there exists a deterministic polynomial time algorithm to solve the problem. There are certain kind of problems for which no polynomial time solvable algorithm is known. If, given an instance x of a problem decision Q and a hint y, we can check using a polynomial time algorithm whether it is a 'yes' instance or a 'no' instance of the problem, we say the problem belongs to the complexity class $\mathbb{NP}$.

For example, the famous Clique problem asks to find the largest complete subgraph or clique in a given graph. The decision version of this problem asks to find a clique of size k in a graph. Given a solution to this problem, we can easily verify whether it is a complete graph of size k or not, in polynomial time. Hence, clique belongs to $\mathbb{NP}$. But, no one has been able to able to give a polynomial time algorithm to solve this problem.

However, the fact that there is no known polynomial time solvable algorithm for a given problem does not imply that it is hard. It is clear to see that some decision problems can be solved as well as checked for a valid instance in polynomial time. This implies that $\mathbb{P} \subseteq \mathbb{NP}$. For the rest of the problems, we introduce the concept of

hardness. It is based on the idea that each problem in a particular class is at least as hard as the hardest of them. Thus, to establish its hardness, we only need transform an instance of a known hard problem Q' into an instance of the given problem Q such that a solution to Q implies a solution to Q'. This idea is known as reduction. According to this method, given a problem Q, we say that Q is $\mathbb{NP}$-hard if every problem in $\mathbb{NP}$ can be reduced to Q in polynomial time. If Q belongs to $\mathbb{NP}$ as well, then Q is called $\mathbb{NP}$-Complete [3].

This key method of reduction has been used to establish hardness of a number of well-known problems. As a corollary, we can assert that efficiently solving one $\mathbb{NP}$-Complete problem implies efficient ways to solve all of $\mathbb{NP}$-Complete problems. Some of well known real life problems in computer science fall into this category of problems, e.g. Traveling Salesman, Job Scheduling, Knapsack, etc. What $\mathbb{NP}$-Completeness claims is that these problems are potentially quite difficult compared to other problems, and it quite possible that there may not exist faster methods, with running time polynomial in input size, to solve these problems. Even so, this does not imply that $\mathbb{NP}$-Complete problems can not be solved faster than trivial enumeration. In fact, various techniques have been developed specifically to address this. Our dissertation work focuses on two such techniques: Fixed Parameter Tractability with respect to Parameterized Complexity Theory and Approximation Algorithms.

B. Parameterized Complexity Theory

Various techniques have been developed to work on $\mathbb{NP}$-hard problems, one of these, parameterized complexity is a way to find exact algorithms for $\mathbb{NP}$-hard problems. Not all $\mathbb{NP}$-hard problems are difficult. Some of the problems or their special cases have structural properties which could be leveraged to achieve faster results than

trivial brute force. Parameterized complexity aims at generalization of the notion of polynomial time, where, one considers a parameter k while looking for a solution, in addition to input size n. k is not related to n and represents a structural property that also participates in running time of the solution. The Parameterized Complexity Theory [22] seeks to classify problems such that, given a problem and a parameter k, we can find an algorithm whose running time is polynomial in input size (the degree of polynomial could be large), while it may be exponential in k, exactly solves the problem.

Definition 6 (Parameterized Problem [23])**.** *A parameterized problem (also called parameterized language) is a subset* Π *of* $\Sigma^* \times N$ *where* Σ *is some alphabet. In the input instance* $(I; k) \in \Sigma^* \times N$ *of a parameterized problem, we call* I *as the main part of the input and* k *as the parameter of the input. We also agree that* $n = |(I; k)|$.

This notion of parameter k becomes clear when instead of asking for a rather unknown optimal value (minimum or maximum) in a solution, which could be very small or very large, we are more interested in finding how large or small the final answer could be. For example, a well known problem Vertex Cover asks: given a graph $G(V, E)$, find the minimum number of vertices to cover all the edges. But in practical problems, we are more interested in vertex covers of certain sizes. So, one might ask, given a graph $G(V, E)$, find a vertex cover of size k [24]. This problem is still $\mathbb{NP}$-hard, and thus, still interesting from theoretical point of view, but we have gained some additional information. These parameters may not be dependent on input size. So, instead of asking a solution to just be a function of the input size, our problem definition adds more dimension(s), which we call parameter(s), and thus, requires us to think in terms of the parameter(s) as well to arrive at the final solution. Of course, this idea would be useful only if the outcome is polynomial in input size,

and the parameter k is independent of input, because then we can choose k to be small and thus reduce the overall running time.

This field has steadily grown in the past couple of decades, and has made substantial contributions to many problems in both computer science theory and applied algorithms. As explained above, this field has strong connections with real life problems involving computing, heuristics, bioinformatics, etc. Various techniques for working on hard practical problems with respect to parameter k have been intensively studied and are available in literature [25], we review a few of them below.

Definition 7 (Fixed Parameter Tractable (FPT))**.** *A parameterized problem* Π *is called fixed parameter tractable if there exists an algorithm* A *and a function* $f : N \to N$ *such that* A *correctly solves* Π *in running time* $O\left(f\left(k\right) \cdot n^{c}\right)$*, where* c *is a constant independent of parameter* k*.* A *is then called a FPT-algorithm.*

What FPT-algorithm implies is a guarantee on the running time performance to be polynomial in input size by making sure that the exponential term is confined to a parameter.

Many known $\mathbb{NP}$-hard problems have turned out to be FPT, and techniques for FPT algorithms have emerged as a good set of tools to attack hard problems. Vertex Cover [24] has a best known running time of $O^{*}\left(1.2738^{k}\right)$. FPT algorithm for undirected Feedback Vertex Set [26] takes $O^{*}\left(3.83^{k}\right)$ time. Multicut problem can be solved exactly in $O^{*}\left(2^{O\left(k^{3}\right)}\right)$ [27]. Edge Bipartization has a FPT algorithm with $O^{*}\left(2^{k}\right)$. Some of these running times are practically achievable with the computation power today, while some are still practically infeasible; all are quite an improvement from brute force method, and interesting nonetheless. Various approaches have been proposed to achieve FPT-algorithms for parameterized problems, viz. kernelization, branch and bound, iterative compression, etc. We discuss kernelization below.

Definition 8 (Kernelization)**.** *[25] Let* Π *be a parameterized problem with inputs* (S, k)*, where* S *is the problem instance and* k *is the parameter. Kernelization means to replace the instance* (S, k) *by a reduced instance* (S', k') *such that* $k' \leq f(k)$*,* $|S'| \leq g(k)$*, where* f *and* g *are arbitrary functions depending only on* k *and* $(S, k) \in \Pi$ *if and only if* $(S', k') \in \Pi$*. The reduction from* (S, k) *to* (S', k') *must be commutable in polynomial time.*

Intuitively, kernelization seeks to separate the easy part of a problem, solvable in polynomial time, from the hard part called the kernel [28], by analyzing its structure. The kernel, even though hard to solve, has a size much smaller compared to the original input size, and thus, even though a brute force method to solve the kernel takes time exponential in k, the overall algorithm is fast. The function $g(k)$ is the size of the kernel. Kernelization is achieved through reduction rules. These rules shrink the problem instances into a kernel. The kernel can be solved using some exhaustive search technique. We can clearly see that there is a relation between FPT and kernelization.

Theorem 1. *[29] For every parameterized problem* (Q, k)*, the following are equivalent:*

- $(Q, k) \in FPT$*,*
- Q *is decidable, and* (Q, k) *has a kernelization.*

This implies that whenever there is a problem in FPT, it is guaranteed to exhibit a kernel, and vice-versa. Sometimes finding a kernel is easier. As a result, if we are able to find a kernelization algorithm for a problem, it implies that it is FPT. But, not all problems are FPT, or exhibit a kernel. This means we need a way to find out which problem is FPT and which is not. This idea is analogous to $\mathbb{NP}$-hardness

reduction. Each problem is at least as hard as other problem in FPT. So, if we can reduce an unknown problem P to a known FPT problem Q, then we can claim that Q also has an FPT algorithm (or a kernel).

Definition 9 (FPT-Reduction [23])**.** *A parameterized reduction from a parameterized problem Q to a parameterized problem Q' (denoted as $Q \leq_{FPT} Q'$) is an algorithm that, given an instance $(I;k)$ for Q computes an instance $(I';k')$ for Q' such that :*

1. *$(I;k) \in Q$ if and only if $(I';k') \in Q'$,*
2. *$k' = g(k)$ is a function depending only k,*
3. *the computation can be achieved in $O(f(k) \cdot n^c)$, where n is the input size, c is a constant independent of of both n and k, and f is any arbitrary function $f : N \to N$ depending only on k.*

Thus, to prove that a problem Q is fixed parameter tractable, we just need to show $Q \leq_{FPT} Q'$. This approach clearly indicates that there could be some problems not satisfying the above condition, and such problems do not belong in FPT. In fact, similar to polynomial time hierarchy $\mathbb{PH}$ [30], a classification of parameterized problems called W-hierarchy [23] has been defined in an attempt to organize them with respect to this hardness. The class $W[1]$ is seen as parameter analogue of the class $\mathbb{NP}$, and we have an equivalent concept of $W[1]$-hardness.

We say that parameterized problem Q is $W[1]$-hard if it can be proved that it does not exhibit FPT. In other words, if we can show that a known $W[1]$-hard problem Q' is fpt-reducible to Q, then Q is at least as hard as all the problems in $W[1]$, and is thus W[1]-hard. It is clear from the definition that Q is not fixed parameter tractable unless $W[1] = FPT$. Some known $W[1]$-hard problems are Independent Set [28], Clique [28],etc. In addition to $W[1]$, W-hierarchy of complexity classes [23] has

different levels $t = 1, 2, ...$, and it has been proved that

$$FPT \subseteq W[1] \subseteq W[2] \subseteq ... \subseteq W[t]$$

This also means that if a problem $Q \in W[t]$-hard is FPT, then $FPT = W[t]$.

C. Approximation Algorithms

Given an optimization problem (maximization or minimization) Q, an approximation scheme A is an algorithm which guarantees that the solution given by A is close to optimal solution (by a certain ratio r). We say that A has an approximation ratio r if, for any x belonging to an instance $I \in Q$, it produces a solution y_A in polynomial time such that the ratio of y_A to the optimal solution of x is bounded by r. Approximation algorithms have been found extremely helpful in solving numerous well-known problems like Euclidean Traveling Salesman [31], Job Scheduling [32], Bin Packing [1] [33], etc. This discipline has been a continued focus of research for decades, and still contributes to our understanding of $\mathbb{NP}$-hard problems.

One of the classes of approximations schemes is to guarantee a fixed ratio for the final solution. An optimization problem Q is said to be in APX if there is a polynomial time algorithm to solve Q with approximation ratio bounded by a constant. In other words, there is a guarantee that the problems in APX can be solved such that the solution is within a certain percentage of the optimal solution. Problems like Min Vertex Cover, Metric TSP, Max3SAT, etc. belong to APX. Other other hand, the general TSP is not in APX. Similar to W-hierarchy, we have further classification of these problems based on approximation schemes, interested readers are referred to [30] for a detailed study.

CHAPTER III

RELATED WORK

In this section, in order to build a background for our work, we do a survey of existing results for Line Cover, Hyperplane Cover and General Stabbing with respect to approximation algorithms and parameterized complexity. We also discuss some related problems which motivated our study on fat line cover. We use the idea for Hyperplane Cover to solve Conic Cover (Chapter IV) and General Stabbing to establish that Fat Line Cover is W[1]-hard (Chapter V).

A. Line Cover and Hyperplane Cover

As discussed in chapter I , Line Cover is a fundamental computational geometric problem which asks to cover a set of points in the plane with minimum number of lines. The parameterized version of Line Cover is defined as below:

Definition 10 (PARAMETERIZED LINE COVER [18])**.** *Given a set S of n points in the plane, and a parameter k, find k lines such that every point in S is covered by at least one of these lines.*

Line Cover can be directly related to a number of path finding problems, which are in direct relation with general Traveling Salesman Problem, with applications in VLSI design, minimum bends and rectilinear traveling salesman, etc. All these applications involve decision-making robots, and straight line motion is preferred to taking turns. As a result, this problem, and its variants, have re-gained focus in algorithmic research.

Line Cover was established to be $\mathbb{NP}$-hard in strong sense by Megiddo et al. [10]. Later, Hassin et al. [8] considered various bounded versions of general hitting

objects with straight lines, giving constant ratio approximation algorithms for some of them and also established that Line Cover is polynomial time solvable when lines are restricted to be axis-parallel. It was also established to be APX-hard by [34]. Later, Langerman et al. [9] introduce an abstract set covering problem which models many geometric and non-geometric covering problems. An extension of Line Cover in Euclidean Space R^d is defined:

Definition 11 (PARAMETERIZED HYPERPLANE COVER [9])**.** *Given a set S of n points in R^d, does there exist a set of k (affine) hyperplanes bounded by dimension d such that each point of S is contained in at least one of the hyperplanes?*

Since Line Cover is $\mathbb{NP}$-hard, Hyperplane Cover, by extension, is also $\mathbb{NP}$-hard. In addition, a FPT-algorithm with running time of $O\left(k^{dk+d} + kn\right)$ was given [9]. When the same algorithm is analyzed for Line Cover, we get a running time of $O^*\left(k^{2k+2}\right)$. Later, Grantson et al. [18] gave the running time of $O^*\left(k^{2k}/4.84k\right)$ by introducing new reduction rules for Line Cover and a kernel of size $O\left(k^2\right)$. In addition, Castro et al. [4] also worked on various versions of Line Cover, including rectilinear line cover. They have provided new reduction rules in addition to those given by Grantson et al. [18] for solving Line Cover, and experimentally concluded that although their reduction rules don't affect the asymptotic running time or the kernel size, their algorithms run faster in practice.

The best known algorithm for Hyperplane Cover was given by Wang et al. [19] with a running time of $O^*\left(k^{(d-1)k}/1.3^k\right)$ which, incidentally, also gives best asymptotic running time of $O^*\left(k^{2k}/1.35^k\right)$ for Line Cover. This result, which we formally give as a theorem below, is given as corollary to main result:

Theorem 2. *[19] For any integers $1 \leq d \leq m$, the d-hyperplane-cover problem in the Euclidean space R^m can be solved in time $O^*\left(\left(\left(\left(d+1\right)k\right)!\right) / \left(\left(\left(d+1\right)!\right)^k k!\right)\right) =$*

$O^*\left(k^{dk}/c_d^k\right)$, *where* $c_d = \sqrt{(}2\pi\left(d+1\right)/e) > 1.3$.

The d-hyperplane cover in the result above is different from the standard definition of hyperplane cover: the dimension d of the hyperplane is the dimension of subspace formed by the set of points represented by

$$H[B] - p_0 = \{p - p_0 \mid p \in H[B]\}$$

where, B is a set of $d+1$ points forming the hyperplane $H[B]$ in $\mathcal{R}^m$ and p_0 is some point in B. Hence, a d-hyperplane cover in $\mathcal{R}^d$ in [19] is the standard hyperplane cover in $\mathcal{R}^d$ with hyperplane bounded by dimension $d+1$.

To solve this version, consider this combinatorial problem [19]: find the number of different ways to put $n\,(= dk)$ objects (points) in k boxes (hyperplanes) where each box has a capacity (dimension) d. With available mathematical tools, and using combinatorial arguments, we can compute the number of ways to do the placement. The way to find this is as follows: for each item i in the input set S, put it in one of the boxes which is not full. If a hyperplane h is possible with items in some box b, add h in current solution, remove rest of the points from S covered by h. Now the problem is reduced to finding a placement for rest of the objects. We can solve it by recursively working on the reduced input set and updated box placement. In addition, if there is an empty box, we need to add i to that as well - the case where i was the first element in the hyperplane and no hyperplane has been formed, and no additional points are removed from S. This case also needs a separate recursion on the same set S and updated box placement.

Wang et al. [19] give a recurrence relation to find the number of ways to get the placement: $(dk)!/\left((d!)^k\, k!\right)$. This resulting value also gives us the maximum number of nodes we need to work on in the recurrence tree before we can claim

whether we have found a solution to $(d-1)$-hyperplane cover or no solution exists otherwise. As a result, the running time to exactly solve d-hyperplane cover is given as $O^*\left((d+1)\,k\right)!/\left(\left((d+1)!\right)^k\,k!\right)$. This result, when used for standard hyperplane cover in $\mathcal{R}^d$ gives a running time complexity of $(dk)!/\left(\left(d!\right)^k\,k!\right)$.

In chapter IV, we show how to build on the above idea to achieve a FPT-algorithm for conic cover.

B. General Stabbing Problem

In [12], Giannopoulos et al. consider the following generic geometric cover problem: Given a set $\mathcal{S}$ of n translates of an object in $\mathcal{R}^d$, find a set of k lines with the property that every object in $\mathcal{S}$ is "stabbed" (intersected) by at least one line. A translate of an object is defined as its copy being moved to a different location in space. The objects could be squares or circles in the plane, cubes in 3-dimension or unit balls in $\mathcal{R}^d$ being stabbed by lines.

Definition 12 (STABBING SQUARES WITH LINES [12])**.** *Given a set of n unit-squares in the plane, find k lines such that each square is stabbed by at least one of the lines.*

This problem is known to be $\mathbb{NP}$-hard [9]. By giving a parameterized reduction from a known $W[1]$-complete problem k-Clique [28], it is proved that stabbing axis parallel unit squares with lines is $W[1]$-hard with respect to k. We survey the following results that we use later to establish the hardness of Fat-Line Cover.

Theorem 3. *[12] Stabbing a set of axis-parallel unit squares in the plane with k axis-parallel lines is $W[1]$-hard with respect to k.*

By applying simple transformations, following results are further established:

Theorem 4. *[12] Stabbing a set of disjoint axis-parallel unit squares in the plane with k lines of arbitrary directions is $W[1]$-hard with respect to k.*

Theorem 5. *[12] Let O be a connected object in the plane.*
(i) If the stabbing lines are to be parallel to two different directions u, v that are part of the input, the problem of stabbing a set of disjoint translates of O with k lines is $W[1]$-hard with respect to k, unless O is contained in a line parallel to u or v.
(ii) The problem of stabbing a set of disjoint translates of O with k lines in arbitrary directions is $W[1]$-hard with respect to k.

From theorem 5, we can see that any general stabbing problem is $W[1]$-hard when the objects to be stabbed are similar to squares. In addition to these hardness results, some restricted versions of the problem are considered and possible FPT results are established:

Theorem 6. *[12] Stabbing a set of n axis-parallel disjoint unit squares with k axis-parallel lines is fixed-parameter tractable.*

Some aspects of General Stabbing Problem with respect to Rectangles have been studied by [35]. d-Dimensional Rectangle Stabbing, where, given a set of axis-parallel d-dimensional hyperrectangles, a set of axis-parallel $(d-1)$-dimensional hyperplanes and a positive integer k, the question is whether one can select at most k of the hyperplanes such that every hyperrectangle is intersected by at least one of these hyperplanes. [35] show, by giving a non-trivial parameterized reduction from a known $W[1]$-complete problem Multicolored Clique [36], that for $d \geq 3$ the problem is W[1]-hard with respect to the parameter k.

The above hardness results indicate that geometric cover problems are in general hard with respect to parameterized complexity. In addition, these results hint towards the hardness of fat line cover as well. Moreover, the problem of stabbing axis-parallel

unit squares in the plane with lines seems to be a good candidate to be used for parameterized reduction. We use these results as our foundation to establish that fat line cover is $W[1]$-hard.

CHAPTER IV

ON COVERING POINTS WITH CONICS

Conic Sections are found in a wide range of applications like computer vision [37] and signal processing [38], pattern recognition, artificial intelligence, chromosome analysis [39], computer aided design [40], etc. In this section, we first consider general conic cover, and then parabola cover separately. Algebraically, a conic is represented by a general equation of second degree in the plane, given by:

$$Ax^2 + 2Hxy + By^2 + Cx + Dy + E = 0 \tag{4.1}$$

The set of points satisfying this equation could possibly represent either a parabola, an ellipse, a hyperbola, or just a pair of straight lines [20]. To eliminate the case of pair of straight lines, we assume that the input points are given in general position in the plane, i.e. no three points are collinear.

Many artificial intelligence and machine learning problems involve second degree constraints in addition to regular variables. In certain applications, like chromosome analysis of human genome, chromosomes occur in pairs. Naturally, they are related and we can expect to have degree two constraints than linear. The problem then is to fit a conic to the given set of data. Conic cover asks to find k such conics to fit the data. In this section, we study conic cover and its special case, parabolic cover with respect to parameterized complexity.

A. Solving for Conic Cover Problem

As an extension of line-cover to higher degree curves, conic cover is also $\mathbb{NP}$-hard. Consequently, we wish to explore the possibility of faster algorithms using techniques developed in theoretical computer science. In particular, utilizing the ideas to solve

hyperplane cover we arrive at a FPT-algorithm for conic cover. We can clearly see from equation 4.1 that five points are enough to uniquely determine a conic section. This simple observation can be used to extend the algorithm for hyperplane cover for our purposes.

Proposition 1. *Given five points in general position in the plane, we can uniquely determine a conic passing through all these points.*

As explained in Chapter 3, Hyperplane Cover can be visualized as the problem of arranging n objects in k boxes with each box having a capacity d. Analogously, we can imagine conic cover as the problem of arranging n objects in k boxes such that each box has a capacity of at least 5 (since we need 5 points to uniquely determine a conic). The rough idea for the approach, then, is as follows:

Start with the set of points S. For every point p in S, add it to each box that has exactly 4 points. Next, find the conic section C passing through these 5 points using equation 4.1. Now, remove all the points in S that pass through C and get the reduced set S'. Do this for each box with 4 points and recursively work on the reduced set S'. In addition, if there is a box having at most 3 points, add p to that box, and branch on this new configuration recursively. We try this for every configuration possible, until we get a 'yes' for k and S becomes empty, or return 'no'. We are now ready to give an algorithm (ConicCover) which utilizes this approach. The following theorem analyzes ConicCover.

Theorem 7. *Let S be a set of n points in R^2, the Conic Cover Problem can be solved correctly in $O^*\left((k/1.38)^{4k}\right)$ time.*

Proof. First, we prove the correctness of ConicCover and then compute the running time. At any point of time, each bin contains at most five points. According to

Algorithm 1 ConicCover($S; n_0, n_1, n_2, n_3, n_4, n_5$)

Input: A set S of points in the plane, and a bin placement with configuration:$(n_0, n_1, n_2, n_3, n_4, n_5)$

Output: An extension of the given configuration to k parabolas corresponding to k bins that cover all points in S, if such an extension exists.

1: **if** $S = \emptyset$ and $n_5 = k$ **then**
2: Return the input configuration
3: **end if**
4: **if** $n_5 = k$ and $S \neq \emptyset$ **then**
5: return 'FAIL'
6: **end if**
7: Pick a point p from S
8: **for** each i, $0 \leq i \leq 4$ **do**
9: **if** $i = 4$ **then**
10: Add point p to each bin B with i points
11: Find conic C formed by the points in B, remove the set of points $s \in S$ from S that lie on C
12: Recursively call Conic Cover $(S \backslash s; n_0, n_1, n_2, n_3, n_4 - 1, n_5 + 1)$
13: **else if** $n_i \geq 0$ **then**
14: Add p to each bin with i points, and recursively branch on $S \backslash \{p\}$ calling Conic Cover $(S \backslash \{p\}; n_0, ..., n_i + 1, n_{i+1} + 1, ..., n_5)$
15: **end if**
16: **end for**

Proposition 1, to uniquely determine a conic, we need at least five points in general position in the plane. Lines 1-5 specify the exit conditions:

First, a configuration of $n_5 = k$ and $S = \emptyset$ (lines 1-3) implies that all the points have been arranged in k bins, with each bin representing a conic formed from 5 points, and corresponding points lying on this conic removed from S. In other words, all points have been covered by k conics. In this case, the answer is 'yes' and input configuration to this branch is returned.

If, otherwise, we have $S \neq \emptyset$ and $n_5 = k$ (lines 4-6), it means there are still some points in S not covered by k conics. This only means that the current configuration would require more than k conics to cover S. Hence the answer is 'no', and the recursive call returns 'FAIL'.

Now, we come to the main part of the algorithm (lines 7-16). For every point $p \in S$, the algorithm first checks for bins having 4 points (line 9). By adding p to

such a bin we can find a candidate conic C. The algorithm then removes the points $s \in S$ that lie on C and work on the resulting set (lines 11-12). Thus, any point is removed from S if only if it has been covered by one of the conics. Otherwise, if there are still some bins with at most three points($n_i \geq 0$ and $i \neq 4$), the algorithm adds p from S to each of these bins without removing any of them from S (lines 13-14). Inductively, if in any branch, the algorithm produces a solution, then it is a set of k parabolas covering all points in S.

The idea here is similar to the one used to solve hyperplane cover, except that in HC, any hyperplane bounded by dimension d is allowed. In other words, every time a bin has points $0 \leq i \leq d$, the algorithm finds a hyperplane formed by those points and adds it to the solution. In Conic Cover, a conic needs at least five points, and so unless a bin has five points, we can not determine a conic to cover the points. Rest of the idea remains the same. Thus, it is same as hyperplane cover for $d = 5$ when hyperplanes of exact dimension d are allowed. Using the same argument used for HC, we can claim that the algorithm finds a solution if there exists one.

Now, for computing the running time, the analysis is given as follows: Lines 1-3 are executed once every recursive call and lines 4-6 are executed once every recursive call, each of these contribute constant values to running time per recursion. Lines 8-16 do the main work, involving two recursions: at lines 12 and 14 respectively. At any recursive call to conic cover, the algorithm loops from $i = 0$ *to* 4, for each box it makes a decision whether to place a point p in it or not, and then makes the recursive call for each configuration. This part works the same as for HC, as explained above. The analysis for HC is done for worst case in [19], i.e., when the dimension is exactly d, and thus is applicable for conic cover as is. The running time for hyperplane cover, from theorem 2 is

$$O^*\left(\frac{(dk)!}{(d!)^k\, k!}\right)$$

With $d = 5$ for Conic Cover, the running time is

$$O^*\left(\frac{(5k)!}{(5!)^k\, k!}\right)$$

We know that Stirling's Approximation for

$$r! \approx \sqrt{2\pi r}\,(r/e)^r$$

thus, we have running time for conic cover as

$$= O^*\left(\frac{\sqrt{10\pi k}\,(5k/e)^{5k}}{\left(\sqrt{10\pi}\right)^k (5/e)^{5k}\,\sqrt{2\pi k}\,(k/e)^k}\right)$$

$$= O^*\left(\frac{k^{4k}}{\left(\sqrt{10\pi/e}\right)^k}\right)$$

$$= O^*\left((k/1.38)^{4k}\right)$$

□

B. Solving for Parabola Cover

The equation 4.1 also gives us a way to distinguish among different types of conics. For a parabola, $H^2 = AB$, for ellipse, $H^2 < AB$, and for hyperbola $H^2 > AB$. Using this condition, the equation for a parabola becomes

$$Ax^2 + 2\sqrt{(}AB)xy + By^2 + Cx + Dy + E = 0 \tag{4.2}$$

By using this condition along with equation 4.2, it is clear that with just 4 points, we can reduce the possible number of parabolas to 2(to take care of the introduction

of square root). Thus, we do not need 5 points like general conic anymore, 4 are enough(see Fig. 1 and Fig. 2). We show below how we can use this idea by applying to hyperplane-cover algorithm to achieve a better running time than conic-cover.

Proposition 2. *Given four points in general position in the plane, there exist at most two parabolas that pass through the points.*

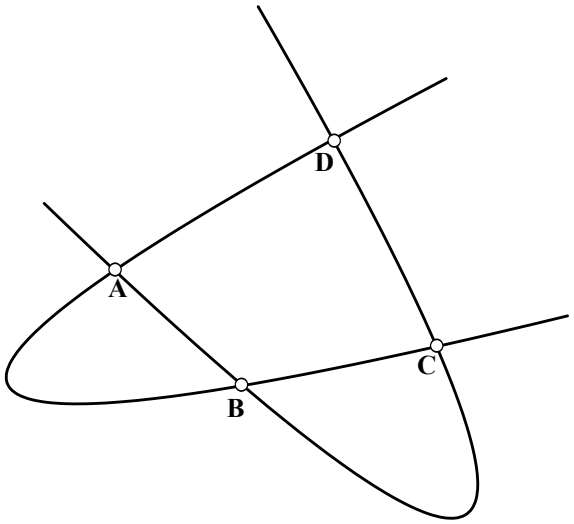

Fig. 1. Two parabolas construction from four points

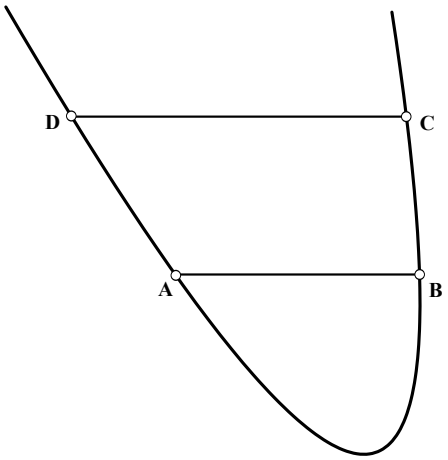

Fig. 2. One parabola construction from four points

The above proposition was first addressed by Newton in *Philosophiae naturalis principia mathematica*, 1687, as mentioned by Dörrie in [41] where a Euclidean construction was provided. Later, Lachance et al. [40] discuss this problem with respect to affine geometry and provide methods for distinguishing among no parabola, or only one or two parabola cases, and also how to construct them. If any three of the four points are collinear or the points form a parallelogram, no parabola is possible. If the points form a trapezoid shape, then only one parabola is possible. Otherwise,

there are two possibilities for parabola through these four points. The algorithm below gives these steps to distinguish among the cases and find the parabola(s), if possible.

Algorithm 2 CheckParabola(p_0, p_1, p_2, p_3)

Input: Four points in the plane
Output: Return parabolas through the points, if possible, return fail otherwise
1: **if** Three of the points $p_i : (i \in \{0, 1, 2, 3\})$ are collinear **then**
2: Return 'FAIL'
3: **else if** The four points form a parallelogram **then**
4: Return 'FAIL'
5: **else if** The four points form a trapezoid **then**
6: Find the parabola C through the four points
7: Return C
8: **else**
9: Find the two parabolas C_1, C_2 through the fours points
10: Return C_1, C_2
11: **end if**

Lemma 1. *CheckParabola correctly finds out whether, and how many, parabola are possible through given four points.*

Proof. The proof of correctness is relatively straight forward. Lines 1-4 decide the failure conditions: whenever any of three points are collinear, or the points form a parallelogram, no parabola is possible, and so algorithm returns failure. If points form a trapezoid (lines 5-7), then only one parabola is possible, and this case is handled. If the points do not satisfy any of the above cases, this implies that two possibilities are there, and hence, the algorithm finds and returns them. In particular, the algorithm handles all the four cases which are possible whenever there are four points in the plane. □

Now, for the running time analysis, lines 1-2 require checking for every three out of the four points, constant steps. Lines 3-4 again take constant steps to check for a parallelogram. Lines 5-7 check for trapezoid, and then find one parabola, the construction again takes constant computation (solving for an equation in four variables).

Lines 8-9 solve the equation 4.2 in four variables for two solutions. In conclusion, the total amount of time taken is constant or fixed for any input.

Algorithm 3 ParabolaCover$(S; n_0, n_1, n_2, n_3, n_4)$

Input: A set S of points in the plane, and a bin placement with configuration $(n_0; n_1; n_2; n_3; n_4)$

Output: An extension of the given configuration to k parabolas corresponding to k bins that cover all points in S if such an extension exists.

```
1: if S = ∅ and n_4 = k then
2:    Return the input configuration
3: end if
4: if n_4 = k and S ≠ ∅ then
5:    Return 'FAIL'
6: end if
7: Pick a point p from S
8: for Each i, 0 ≤ i ≤ 3 and each bin B containing i points do
9:    Add point p to B
10:   if i = 3 then
11:      result ← CheckParabola(p_0, p_1, p_2, p)
12:      if result ≠ 'FAIL' then
13:         if number of parabola returned = 2 then
14:            Find the points s_1 and s_2 that are covered by C_1 and C_2 respectively
15:         else
16:            Find the points s_1 that are covered by C and set s_2 = ∅
17:         end if
18:         if s_1 ≠ ∅ then
19:            Branch recursively on S\s_1 calling
               Parabola Cover (S\{s_1}; n_0, ..., n_3 − 1, n_4 + 1)
20:         end if
21:         if s_2 ≠ ∅ then
22:            Branch recursively on S\s_2 calling
               Parabola Cover (S\{s_2}; n_0, ..., n_3 − 1, n_4 + 1)
23:         end if
24:      end if
25:   else if n_i ≥ 0 then
26:      Add p to a bin, and recursively branch on S\{p} calling Parabola Cover
         (S\{p}; n_0, ..., n_i − 1, n_{i+1} + 1, ..., n_4)
27:   end if
28: end for
```

Coming back to solving Parabola Cover, the idea remains the same as for conic cover, except here, as soon as a point p is introduced into a bin having 3 points, we check whether we can find a parabola or a pair, using CheckParabola algorithm. If yes, then the algorithm removes all the points on the parabola(s), then recursively

work on the reduced set and updated configuration. If no parabola is possible with 4 points, the algorithm removes p from the bin, and considers the rest of the bins, working recursively. Using this idea, and the algorithm CheckParabola, we give the algorithm for Parabola Cover.

Theorem 8. *Let S be a set of n points in R^2, Parabola Cover Problem can be correctly solved in $O^*\left((k/1.15)^{3k}\right)$ time.*

Proof. The proof of correctness for ParabolaCover follows from Proposition 2, Theorem 7 and Lemma 1. Since using four points, we can reduce the number of candidate parabolas for four points to at most two, instead of five, we only need the bin capacity to be four. The additional conditions we have are because we can have no parabola or only one parabola through 4 points. In addition, if we have found the two possible parabolas from these four points, we need to branch for both the candidates. The algorithm takes care of this and thus correctly determines whether a parabola cover of size k exists or not. In any recursive call, CheckParabola always takes a constant amount of time, and so contributes negligibly to the running time.

Now, every branch involves additional branch for two candidate parabolas, it adds a factor $O\left(2^k\right)$ to the running time of every branch. Thus, the overall complexity of Parabola Cover algorithm (with d = 4) using the same argument as for conic cover, is

$$
\begin{aligned}
& O^*\left(\frac{(4k)!\cdot 2^k}{(4!)^k\, k!}\right) \\
= O^*\left(\frac{\sqrt{8\pi k}\,(4k/e)^{4k}\cdot 2^k}{\left(\sqrt{8\pi}\right)^k (4/e)^{4k}\,\sqrt{2\pi k}\,(k/e)^k}\right) \\
= O^*\left(\frac{k^{3k}\cdot 2^k}{\left(\sqrt{8\pi/e}\right)^k}\right)
\end{aligned}
$$

$$= O^* \left((k/1.15)^{3k} \right)$$

□

In this chapter, we studied conic cover, the problem of covering points with conics in the plane. We showed that conic cover is in FPT by giving a FPT-algorithm which utilizes the idea used to solve hyperplane cover. Since conic cover has not been explored with respect to parameterized complexity, the running time of FPT-algorithm for conic cover is best known and takes $O^* \left((k/1.38)^{4k} \right)$ in the worst case. We also studied parabola cover as a special case of conic cover, and were able to show that we can achieve a faster FPT-algorithm by utilizing a property from analytical geometry, thus arriving at a worst case asymptotic time complexity of $O^* \left((k/1.15)^{3k} \right)$, which, again is currently the best known.

CHAPTER V

FAT LINE COVER

We discussed in chapter I that fat line cover is an extension of line cover where lines are of a certain finite width w. First, we look at an extension of linear regression, where, we wish to find a bounding box of smallest width for a given set of points. In other words, find a way to determine a single fat line that covers a set of points. Next, we formally define this problem and device an algorithm to find one.

A. Closest-Band Problem

Closest-band originates from the idea of finding the diameter of a convex polygon (or a convex hull) of a set of points in the plane. Diameter of a convex polygon is a pair of vertices which are farthest from each other. A famous algorithm for finding diameter is using Rotating Calipers, introduced by Godfried [42]. In closest-band, we wish to find a point-edge pair which are farthest from each other.

Definition 13 (Closest-Band)**.** *The Closest-Band for a set of points S in a plane is defined as a pair of parallel lines that envelope all the points of S such that the distance between the lines is minimum.*

We use the following claims to obtain an algorithm to find closest-band.

Claim 1. *For two parallel lines l_1 and l_2 passing through any two points p_1 and p_2 respectively, in the plane, we can always rotate the lines in a direction such that the distance between them does not increase.*

Proof. Suppose there are two parallel lines l_1 and l_2 passing through any two points p_1 and p_2 respectively. Consider the segment $(\overline{p_1, p_2})$. The maximum distance between

l_1 and l_2 is achieved when the segment $(\overline{p_1, p_2})$ is perpendicular to both l_1 and l_2. In that case, the distance between the lines is same as the distance between the points p_1 and p_2. The minimum distance is achieved when the lines coincide with the segment $(\overline{p_1, p_2})$, the distance in this case is zero. Thus the distance between the lines l_1 and l_2 passing through two arbitrary points p_1 and p_2 varies between 0 and $|\,(\overline{p_1, p_2})\,|$.

Given two arbitrary points p_1 and p_2 on parallel lines l_1 and l_2 respectively, we can rotate the lines about the segment $(\overline{p_1, p_2})$ in clockwise as well as anti-clockwise direction. We consider these two cases and prove that the distance between the lines does not increase while rotating in one of the directions. Suppose that the line l_1 makes an angle α with the $(\overline{p_1, p_2})$ in anti-clockwise direction at p_1. Then the distance between the lines is given by $|\,(\overline{p_1, p_2})\,|.\sin(\alpha)$.

Case 1: Rotate l_1 and l_2 at an angle β anti-clockwise (see Fig. 3). The new angle

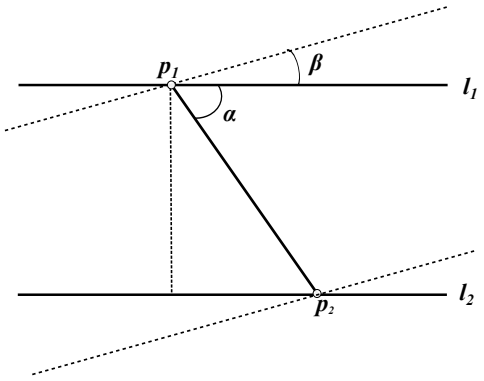

Fig. 3. Anti-clockwise rotation

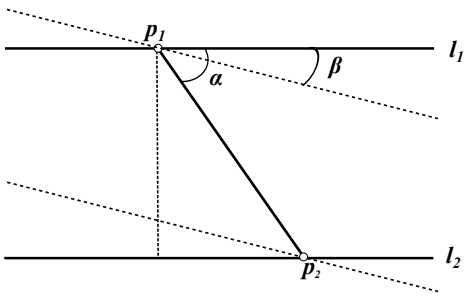

Fig. 4. Clockwise rotation

between l_1 and the segment $(\overline{p_1, p_2})$ at p_1 is $\alpha + \beta$ in anti-clockwise direction. The change in the distance between the lines l_1 and l_2 is given by:

$$\mathbb{D}_1 = |\,(\overline{p_1, p_2})\,|.(\sin(\alpha + \beta) - \sin(\alpha))$$

Case 2: Rotate l_1 and l_2 at an angle β clockwise (see Fig. 4). The new angle between

l_1 and the segment $(\overline{p_1, p_2})$ at p_1 is $\alpha - \beta$ in anti-clockwise direction. The change in distance is given by:

$$\mathbb{D}_2 = |\,(\overline{p_1, p_2})\,|.\,(\sin(\alpha - \beta) - \sin(\alpha))$$

Since *sine* of an angle is a strictly increasing function between 0° and 90° and strictly decreasing between 90° and 180°, $\mathbb{D}_1$ and $\mathbb{D}_2$ have opposite signs. This means, at least one of them is negative, implying that the distance strictly decreases in one direction (unless $\mathbb{D}_1 = \mathbb{D}_1 = 0$). □

Claim 2. *For any closest-band that covers a set of points S in a plane, at least one of the lines must be collinear with one of the edges of the convex hull of S.*

Proof. Suppose (l_1, l_2) are the lines of closest-band of S. It is clear to see that each of l_1 and l_2 have to lie outside of the boundary points of convex hull of S. Further, each of the lines have to pass through at least one point on the convex hull $CH(S)$. For, if the lines lie outside the convex hull, we can move the lines by some distance towards the closest boundary point of convex hull, thus reducing the distance between them.

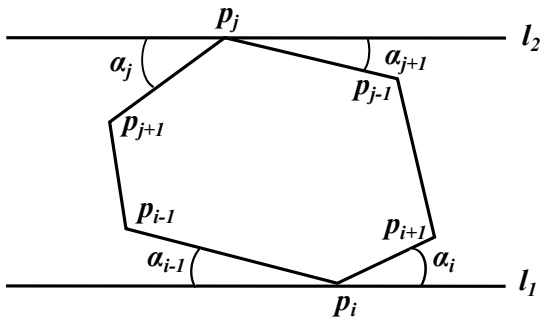

Fig. 5. The convex polygon P

Now, suppose that the closest-band (l_1, l_2) passes through points p_i and p_j on $CH(S)$ respectively and does not coincide with an edge through either of p_i and p_j. Consider the edges $\{p_{i-1}, p_i\}$, $\{p_i, p_{i+1}\}$, $\{p_{j-1}, p_j\}$ and $\{p_j, p_{j+1}\}$ on $CH(S)$ (see

Fig. 5). Suppose the edges make angles α_{i-1} and α_i with l_1 and α_{j-1} and α_j with l_2 respectively.

Since the points are on $CH(S)$, any other edge $\{p_k, p_{k+1}\}$ such that $k \in (i+1, j)$ would make an angle greater than α_i with l_1. Similarly, any other edge $\{p_{k-1}, p_k\}$ such that $k \in (j, i-1)$ would make an angle greater than α_{i-1} with l_1. Also, any other edge $\{p_k, p_{k+1}\}$ such that $k \in (j+1, i)$ would make an angle greater than α_j with l_2. Any other edge $\{p_{k-1}, p_k\}$ such that $k \in (i, j-1)$ would make an angle greater than α_{j-1} with l_2.

This implies that we can always rotate both l_1 and l_2 by an angle $\beta \in \{\alpha_i, \alpha_{i-1}, \alpha_j, \alpha_{j-1}\}$ without intersecting any other points in $CH(S)$. We can do this until one of the lines coincides with one of the edges $\{p_{i-1}, p_i\}$, $\{p_i, p_{i+1}\}$, $\{p_{j-1}, p_j\}$ and $\{p_j, p_{j+1}\}$. By lemma 1, we can always choose one direction β in which the distance between the lines decreases. This implies that l_1 and l_2 can not form a closest-band, contradicting our assumption. □

As a consequence of above claim 2, we break up our algorithm for finding closest-band into two parts: first we find all the vertex-edge antipodal pairs for the convex hull of points in S. Then, for each such pair, we find the closest one. The first part can be given as a separate algorithm, and is then called from the main algorithm for closest-band.

Algorithm 4 finds all the vertex-edge antipodal pairs for any convex polygon. For the running time analysis of *Vertex-Edge Antipodal Pairs*, given a polygon P, for each point $p \in P$, the algorithm finds the farthest edge from p and stores it in F. If there are m vertices of P, there are $m-1$ edges, and hence the algorithm takes a total of $O(m * (m-1)) = O(m^2)$ time.

We are now ready to give the algorithm for closest-band problem. The intuitive

Algorithm 4 Vertex-Edge Antipodal Pairs

Input: A convex polygon $P = \{p_1, p_2, ..., p_m\}$
Output: Pairs of point and edge that are farthest on P
1: $k \leftarrow 1$
2: $i \leftarrow 2$
3: **while** $k \leq m$ **do**
4: **while** p_i is not the farthest vertex from segment $\{p_{k-1}, p_k\}$ **do**
5: $i \leftarrow i + 1$
6: **end while**
7: $F \leftarrow [p_i, \{p_{k-1}, p_k\}]$
8: **end while**
9: Return F

idea is the following: For a given edge on convex hull of S, find the farthest point from it on $CH(S)$. From these vertex-edge antipodal pairs, find the one with the minimum distance between them. The closest-band would then be the line on which this edge lies and the line parallel to this edge passing through the other point of this pair.

Algorithm 5 Closest-Band

Input: A set S of n points on a plane
Output: A pair of parallel lines that envelope all the points of S such that the distance is shortest between them
1: Construct the convex hull $CH(S)$ of S
2: Call *Vertex-Edge Antipodal Pairs* on $CH(S)$
3: Scan each vertex-edge pair in F to find the pair with shortest distance, say $\{p, (\overline{p_j, p_{j+1}})\}$
4: Return the pair of lines through $(\overline{p_j, p_{j+1}})$ and the one parallel to it through p as the closest-band

Theorem 9. *The algorithm Closest-Band returns the pair of lines that are at shortest distance from each other and cover all the points of S in $O(n^2)$.*

Proof. The correctness of algorithm follows from Lemma 2. Now we do the running time analysis of *Closest-Band*. Given a set of points S in the plane, convex hull can be found in $O(n\ log\ n)$ time. As discussed earlier, *Vertex-Edge Antipodal Pairs* takes $O(m^2)$ time if there are m vertices in the convex polygon P. For the set S, the convex

hull can have all the points as its vertices, hence, $m = O(n)$, hence line 2 takes $O(n^2)$ time. Line 3 is a single scan of vertex-edge pairs in F, which could be of at most $m = O(n)$ size. Hence, line 3 represents a running time of $O(n)$. Overall, the time complexity of the algorithm *Closest-Band* is $O(n^2)$. □

The above polynomial time algorithm gives us motivation to study this problem further. In particular, a fat line of width w is the same as a closest-band of width w. We now wish to focus on fat line cover, and study its hardness. As we show below, fat line cover is, in fact, $\mathbb{NP}$-hard as well as $W[1]$-hard.

B. Hardness of Fat Line Cover

1. Fat Line Cover is NP-Hard

$\mathbb{NP}$ hardness of fat line cover is trivial to assert if we allow fat line to be of zero width, i.e., $(w \geq 0)$. In this case, line cover becomes a special case of fat line cover with $w = 0$. Instead, our problem restricts w to be strictly larger than zero. We prove below that this problem is $\mathbb{NP}$-hard. We establish the hardness of fat line cover by reducing it from line cover. Given an instance $X : (S, k)$ of line cover, we show that we can reduce it in polynomial time to an instance $Y : (S', k', w)$ of fat line cover. Given $X : (S, k)$, we construct $Y : (S', k', w)$ in the following way:

1. $S' = S$, i.e., using the same set of points.
2. $k' = k$.

We want to prove that there exists a fat line cover for Y if and only if there is a line cover for X. Let C be the solution for X. For each line $l \in C$, we construct a fat line $L : (l_1, l_2)$ whose width w is such that no new points are added to the fat line from S.

Lemma 2. *For any given line cover C for a set of points S, there exists a $w\,(> 0)$ such that if each line l in C is made w measure wide to make an equivalent fat line*

L, then L covers all the points covered by l, in addition, it does not cover any other point of S.

Proof. We consider the following problem: For any three points in S, find the smallest distance μ that one of the points need to be moved such that the points become collinear. The goal here is to do this for all such triplets in S. By choosing a value of w smaller than the smallest of μ found, we can ensure that each fat line covering points in S covers only collinear points in S, i.e. same as a normal line would. The value of w is obtained using a small procedure below:
We denote a point in S as $P : (P_x, P_y)$. For each triplet of points $\{A, B, C\} \in S$, if A, B, C are collinear points, $\mu_{ABC} = \infty$. Otherwise,

$$\mu_{ABC} = \mathcal{D}_{ABC} / \max\{\left|\left|\overline{AB}\right|\right|, \left|\left|\overline{BC}\right|\right|, \left|\left|\overline{CA}\right|\right|\}$$

where

$$\mathcal{D}_{ABC} = \frac{1}{2} \begin{vmatrix} A_x & A_y & 1 \\ B_x & B_y & 1 \\ C_x & C_y & 1 \end{vmatrix}$$

and

$$\left|\left|\overline{PQ}\right|\right| = \sqrt{(P_x - Q_x)^2 + (P_y - Q_y)^2}, P, Q \in (A, B, C)$$

$\mathcal{D}_{ABC}$ is the area of triangle formed by points A, B and C. Thus, the value of each μ_{ABC} is nothing but the smallest altitude of triangle ΔABC. We compute μ_{min} as the smallest of all μ_{ABC}:

$$\mu_{min} = \min_{A,B,C \in S} \{\mu_{ABC}\}$$

We now claim that choosing $w < \mu_{min}$ (say $0.9\mu_{min}$) for width of fat line L only contains the points covered by l, and no other point in S. Let h_a, h_b and h_c be the altitudes of ΔABC in the order $h_a < h_b < h_c$ for any three points $\{A, B, C\} \in S$ (see

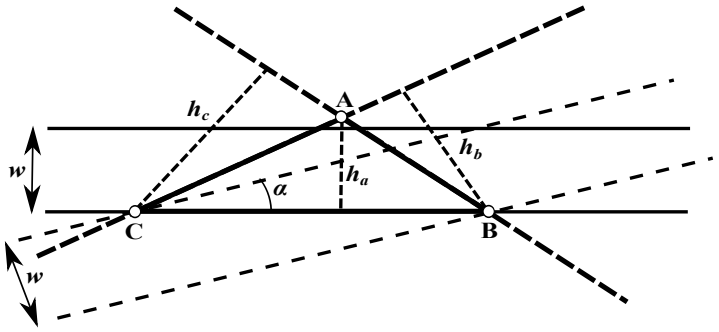

Fig. 6. Construction of a fat line

Fig. 6). Let $w = 0.9h_a$. Now, form a fat line L of width w for $\overline{BC}$. Let's rotate the fat line in counter-clockwise direction by an angle α at B such that, at this rotation, the point A comes inside the band. This implies that the whole triangle has to be inside rotated L, since B and C were already in L. Drop a perpendicular to side $\overline{AC}$ from B, let this value be h'. It is clear that $h' = h_c$. Since ΔABC is inside the band, $h_c < w$. But, we chose w such that it was less than h_c. This contradicts our assumption. Clearly, our claim holds true for any rotation of L in clockwise direction as well. As a consequence, we can assert that no orientation of L can cover the ΔABC. □

We utilize the above lemma to prove the following theorem:

Theorem 10. *FAT LINE COVER is* $\mathbb{NP}$*-Hard.*

Proof. The computation in lemma 2 can be trivially done in $O(n^3)$ time for triplets of points in S. As a consequence of lemma 2, to cover all the points in S, we need all the corresponding fat lines C' thus constructed from lines in C. This means that if there is a set of k lines that cover S, there exists a set of k fat lines that also covers S.

Now, we claim that for every fat line L in fat line cover of size k, with fat lines of width w as computed above, we can construct a corresponding line l which covers

all the points covered by L. For any fat line cover for S, with each fat line of width w as computed above, each fat line L is such that it only includes collinear points. We can replace each such fat line by a normal line l. Now, since only collinear points were included in each fat line, the normal lines thus constructed do not exclude any point in S. As a result, these lines cover all the points in S. In particular, if there is a set of fat lines of size k that covers S, there exist a set of k normal lines that covers S. □

2. Fat Line Cover is W[1]-Hard

As discussed in chapter III, the problem of stabbing of closed connected objects with lines in the plane, is $W[1]$-hard [12]. A closed connected object can be any shape like a square, circle, rectangle or any other polygon. To study parameterized hardness of fat line cover, we consider the case of general stabbing when the objects are circles of radius r:

Definition 14 (STABBING CIRCLES WITH LINES)**.** *Given a set S of n circles of radius r in the plane, find k lines in the plane such that each line stabs (intersects) at least one circle.*

The theorem 5 discussed in chapter III [12] established $W[1]$-hardness for any closed connected object in the plane. Thus, we can state that stabbing circles with lines in the plane is also $W[1]$-hard with respect to parameter k. Also, stabbing of closed connected objects is $W[1]$-hard even when the objects are disjoint or lines are restricted to be axis-parallel (theorems 4 and 5). As a consequence, with respect to circles of radius 1, we have the following results analogous to theorems 3, 4 and 5:

Corollary 1. *Stabbing circles with lines is $W[1]$-hard with respect to parameter k.*

Corollary 2. *Stabbing a set of disjoint circles with lines is $W[1]$-hard with respect to parameter k.*

Corollary 3. *Stabbing a set of disjoint circles with axis-parallel lines is $W[1]$-hard with respect to parameter k.*

Stabbing circles with lines can be seen as an alternate definition of covering points with fat lines. We show that fat line cover and even some of its restricted versions are $W[1]$-hard. To establish the results, we show a parameterized reduction from stabbing circles with lines to fat line cover to establish its $W[1]$-hardness.

The general idea for the construction is as follows: Given a set S of n circles, each of radius r, in the plane. For each circle $c \in S$, construct a point p. The coordinates of p are same as the coordinates of the center of c. Thus, we construct a set S' of n points in the plane. Let l be any line that covers a circle $c \in S$. For any circle that is covered by l, its center is at a distance at most r from l. The goal now is to cover S' with k fat lines where each fat line is of width $2r$. With this construction, we give the following Lemma and use it to give the main theorems of this section.

Lemma 3. *Stabbing a set S of circles in the plane has a solution of size k with respect to lines if and only if there is a fat line cover of size k for the corresponding points in S'.*

Proof. "⇒": We claim that there exists a corresponding fat line L such that the points it covers in S' are same as the centers of circles in S covered by l. For any line l that covers c, draw two lines l_1, l_2 parallel to and each at a distance r from l (see Fig. 7). The region on and between l_1, l_2 denoted as $L : (l_1, l_2)$ covers all the points that are centers of circles covered by l. Suppose this was not true and there is a circle $c' \in S$ not covered by l, but its center is in the region L. Since l divides L into two equal parts, this means that the center of c' would be at most r from l. But, in that

case, it would already be covered by l, which contradicts our assumption. This region L is actually a fat line of width $2r$. To cover the circles in S, k lines are needed. From this construction, for each such line, we draw a corresponding fat line to cover the points in S'. This means that if there exists a cover of size k that stabs the circles in S, there is a fat line cover of size k for points in S'.

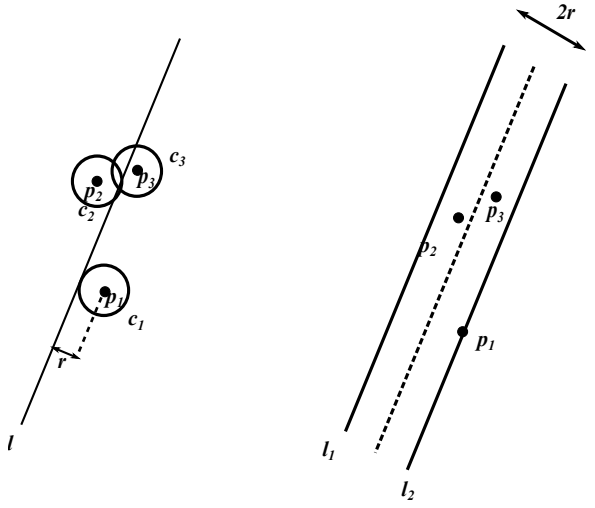

Fig. 7. Constructing fat line cover from stabbing circles with lines

"⇐": Now suppose there is a fat line cover of size k for the points in S'. Let $L : (l_1, l_2)$ be a fat line covering a point $p \in S'$. Clearly, p is at a distance at most r from a line l' that bisects L and is parallel to both l_1 and l_2. This implies there is a line that is at a distance no more than r from all the points covered by L. This line stabs all the circles of radius r centered at the points covered by L without covering any additional circles from S. Otherwise, if there was a circle c covered by l' but whose corresponding center p is not covered by L, then p would be at a distance more than r from the central line l'. But then, c could not be covered by l', a contradiction. Thus, for every fat line of width $2r$ that covers points in S', there is a corresponding line that covers exactly the corresponding circles of radius r in S. In particular, if

there exists a fat line cover of size k for S', there exists a set of k lines that stabs the corresponding set of circles S. □

We are now ready to give our main result in this section:

Theorem 11. *Fat Line Cover is $W[1]$-hard with respect to number of fat lines.*

Proof. The construction above, of a set of points S' corresponding to centers of circles in S, such that Lemma 3 holds true can be done in linear time with respect to size of input set S. In particular, Lemma 3 gives a parameterized reduction that results in a fat line cover of size k if and only if the solution to stabbing of circles is of size k. □

Analogous to above theorem, we can also give the following results:

Theorem 12. *Fat line cover is W[1]-hard even when fat lines are restricted to be axis-parallel.*

Proof. We use Theorem 2, the $W[1]$-hardness result of *stabbing axis-parallel squares with axis-parallel lines* given in chapter III. We show a reduction from the problem of stabbing axis-parallel squares with axis-parallel lines in the plane to fat line cover when fat lines are axis-parallel. The construction is similar to the one for Lemma 3: Given a set S of n axis-parallel squares in the plane, each of side $2r$. For each square $s \in S$, construct a point p whose coordinates are same as that of center of s (see Fig. 8). Thus, we constructed a set S' of n points in the plane.

Now, we claim that there exists a corresponding axis-parallel fat line L such that the points it covers in S' are same as the centers of the squares in S covered by l. As before, for each line l that covers s, draw two lines l_1 and l_2 parallel to and at a distance r from l. Since l is axis-parallel, l_1 and l_2 are also axis-parallel. Using the same argument as proof of Lemma 3, each point p corresponding to a square s

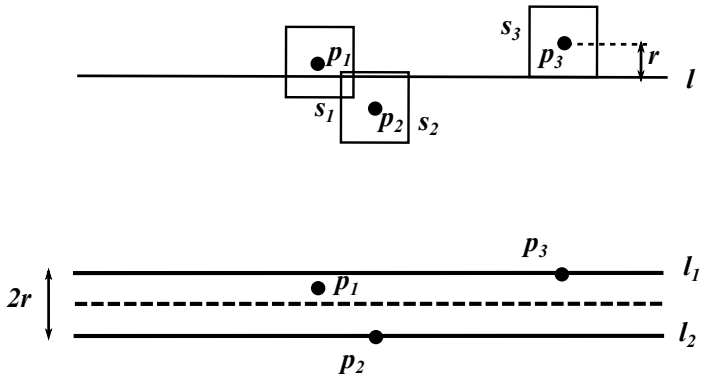

Fig. 8. Stabbing Squares with axis-parallel lines and fat line cover

covered by l, is also within r distance from either l_1 or l_2, and hence covered by the fat line. In general, if there exists a set of axis-parallel lines of size k that stabs the set S of axis-parallel squares, there exists a set of k axis-parallel fat lines that covers the corresponding points p in S'.

We also claim that there exists a corresponding axis-parallel line l such that the squares it covers in S are the same as the points in S' corresponding to centers of these squares covered by axis-parallel fat line L. The same argument holds here: for each fat line $L\,(l_1, l_2, 2r)$, draw a line l' corresponding to the bisector of boundary lines l_1 and l_2 of L. Since any point p is at a distance at most r from l_1 or l_2, the corresponding lines l' is also at most a distance r from the corresponding centers of the squares in S. As a result, for each point p which is covered by fat line L, each square s corresponding to p is covered by l. In particular, if there exists an axis-parallel fat line cover of size k for S', there exists a set of k axis-parallel lines which stabs the set of axis-parallel squares S. Thus, stabbing a set S of axis-parallel squares in the plane has a solution of size k if and only if there is a fat line cover of size k, with respect to axis-parallel fat lines, for the points in S'. This construction can be done in linear time with respect to size of S; as a result, the above parameterized reduction

establishes the proof. □

Theorem 13. *Given a set S of points in the plane such that distance between each pair of points in S is more than $2r$, then fat line cover is W[1]-hard with respect to fat lines of width $2r$.*

Proof. We show that Lemma 3 is enough to establish this result. We show a parameterized reduction from the problem of stabbing a set of disjoint circles in the plane, which, by Corollary 2, is $W[1]$-hard.

Let there be two circles c_1 and c_2 in S. According to our problem, c_1 and c_2 are disjoint. As a result, their centers, say p_1, p_2, would be at a distance d such that $d > 2r$. Therefore, for any line l that stabs both c_1 and c_2, we can draw two lines l_1 and l_2 parallel to l, each at a distance r from each other. As a result, the corresponding centers of c_1 and c_2, viz. p_1 and p_2, would still lie in the region between l_1 and l_2. But l_1 and l_2 and the region in between together represent a fat line L of width $2r$. Hence, we can cover p_1 and p_2 by the same fat line.

Now, given a fat line L covering two points p_1 and p_2, which are at a distance $d > 2r$, we can construct two corresponding circles c_1 and c_2 such that they are disjoint from each other, each of radius r. We draw a line l such that it bisects the region covered by L, and is at a distance r from each of the boundaries, viz. l_1 and l_2, of L. It is clear to see that l covers both c_1 and c_2.

In particular, we can do this construction such that for each of the k lines stabbing a set S of disjoint circles in the plane, there exists a fat line cover of size k with respect to points corresponding to centers of circles in S. Hence, Lemma 3 holds and this completes the proof. □

Theorem 14. *Given a set S of points in the plane such that distance between each pair of points in S is more than $2r$, then fat line cover is W[1]-hard with respect to*

axis-parallel fat lines of width $2r$.

Proof. This theorem is an extension of Theorem 2, and we show a parameterized reduction from the problem of stabbing disjoint squares in the plane where, in addition, the stabbing lines are axis-parallel. This problem is also $W[1]$-hard by Theorem 2. Suppose each square in S is of width $2r$.

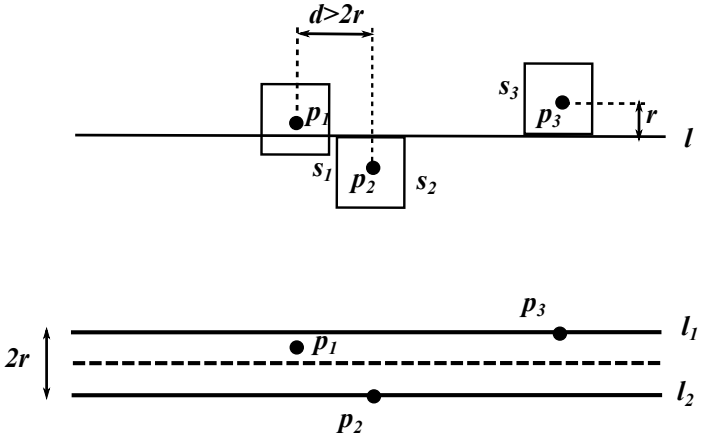

Fig. 9. Stabbing disjoint squares with axis-parallel lines and fat line cover

Since squares are disjoint, observe that the distance between centers of any two squares is more than $2r$ (see Fig. 9). Hence, there is only one direction (either vertical or horizontal) in which more than one line can stab any two squares. As a result, for any two squares s_1 and s_2 stabbed by a line l in horizontal direction, we claim that any fat line drawn which covers the corresponding centers p_1 and p_2 is also horizontal. Suppose this was not true, and there is a vertical fat line $L\,(l_1, l_2)$, which covers both p_1 and p_2. This means that each of the points is at most r distance from the boundary lines l_1 and l_2 of L. Observe that distance between p_1 and p_2 is $d > 2r$. This implies that width of L is more than $2r$. This is a contradiction since only fat lines of width $2r$ are acceptable. Consequently, we can say that for any horizontal line l which stabs disjoint squares in S, there exists a horizontal fat line L which covers all the points

corresponding to centers of squares covered by l.

Symmetrically, the above argument holds true if the stabbing line was vertical. In general, if the solution for stabbing disjoint squares with axis-parallel lines has a solution of size k with respect to number of lines, fat line cover has a solution of size k with respect to fat lines, where each fat line is of width $2r$ and distance between any pair of points is more than $2r$. □

After studying results by Giannopoulos et al. in Chapter III (theorems 2, 3, 4 and 5) [12], [13], it is quite clear that general versions of most of the geometric covering problems are $W[1]$-hard. In fact, the only general problem that is known to be in FPT is line cover and its higher dimension equivalent, hyperplane cover. However, most of the other problems have been established to be intractable with respect to parameter k.

Even when we consider the class of problems of covering points with objects instead of stabbing objects with lines, like covering points with disks, squares, triangles, etc. it has been established that few of them are $W[1]$-hard by Marx [43], [44]. It is quite discouraging to discover that despite having enough literature and an extensive amount of results available in computational geometry, we are unable to deploy them in any way to achieve parameterized solutions for hard problems in that domain. Even so, there still exist restricted versions of these problems, which are practical, for which FPT results have been given (e.g. theorem 6 [12], Dom et al. [35]). Motivated by these results, we conclude our discussion on fat line cover by constructing a restricted version, and get some positive results in the next section.

C. Rectilinear Fat Line Cover with Integral Coordinates (RFLC)

With the general fat line cover being intractable, we wish to explore special versions of the problem which are quite practical, and hence, still interesting. In this section, we add a further restriction to this problem: the given set of points have integral coordinates and the fat lines are axis-parallel. We contribute results with respect to hardness, FPT and approximation algorithms. The line cover version in this restricted form has been proven to be polynomial-time solvable [8], but it is still an open question whether this restricted fat line cover is hard at all.

1. RFLC is $\mathbb{NP}$-hard

We establish $\mathbb{NP}$-hardness of RFLC by showing a reduction from the following problem:

Definition 15 (Minimum hitting of horizontal unit segments [8])**.** *Given are n pairs $(a_i, b_i)\,(i = 1, ..., n)$ of integers and an integer k. Recognize whether there exist in the plane k straight lines parallel to the axes with the property that each line segment $[(a_i, b_i), (a_i + 1, b_i)]$ is hit by at least one of the lines.*

The above problem has been proven to be $\mathbb{NP}$-hard [8]. It is easy to see that this problem is quite similar to RFLC. Without loss of generality, we give a polynomial time reduction to the case of RFLC where width of each fat line is exactly 1. Let $s : (a, b)$ denote a horizontal segment of unit length with coordinates $\{(a, b) : (a + 1, b)\}$. Following is the outline of construction we use for $\mathbb{NP}$-hardness reduction. For each segment $s \in S$ construct a point $p : (a', b')$. Put $a' = a$. Choosing b' needs additional care. If the y-coordinate of s differs from the y-coordinate of another segment s' by exactly 1, then relocate p such that b' differs from y-coordinate of any other point by more than 1 (say 2). Otherwise, $b' = b$ (see Fig. 10). Thus, we have a set S' of n

points. Clearly, all points p constructed thus have integral coordinates.

Lemma 4. *Minimum hitting of horizontal unit segments can be reduced to RFLC in polynomial time such that it has a solution of size k if and only if RFLC has a solution of size k.*

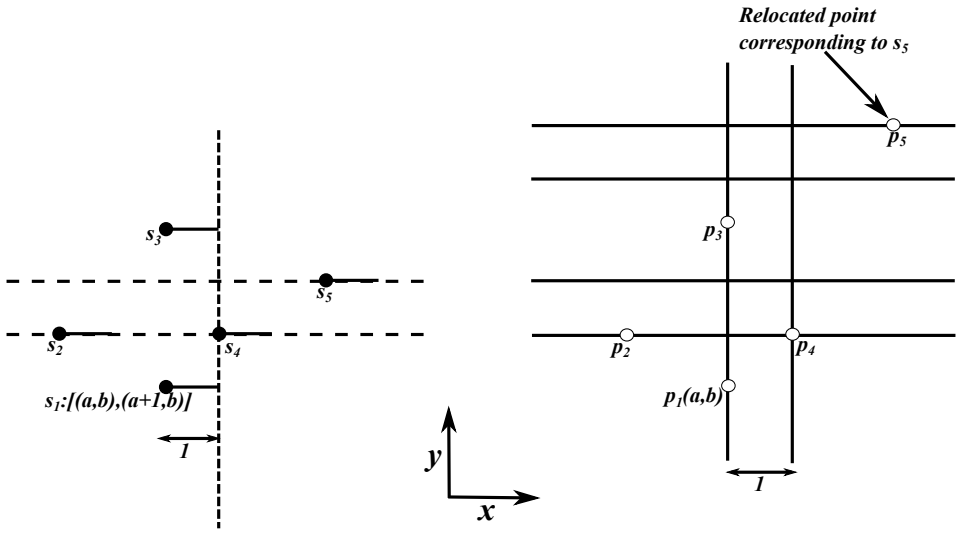

Fig. 10. Reduction from Hitting Unit Segments (left) to RFLC (right)

Proof. "⇒": For each line l that covers a segment s in the hitting set of S, if it is a vertical line, it can be replaced by a fat line $L : (l_1, l_2)$ such that l_1 is the left boundary and l_2 is the right boundary, that covers the same number of points in S'. If l passes through the right end point of s, then the corresponding point p lies on l_1. Similarly, if l passes through left end point of s, then p lies on l_2. Also, since the abscissa of any segment s is same as that of its corresponding point p, L does not cover any additional points in S'. Additionally, no new horizontal lines are required to cover these points.

If, on the other hand, l is horizontal, then our construction that all the points in S' have y-coordinates that differ by 2 unless they share the same y-coordinates in

S, ensures that any horizontal fat line that covers p only covers the corresponding segment s that lie on the same line without covering any new points from S'.

"$\Leftarrow$": Now, suppose there is a fat line cover of size k for the points in S'. For any fat line $L : (l_1, l_2)$. If L is vertical such that l_1 is the left boundary and l_2 is the right boundary of L, then, for any point $p\,(a, b)$ on l_1, draw a line through segment s such that it passes through $(a + 1, b)$, that is, the right end point of s. Similarly, if p is on l_2, draw a line through s such that it passes through (a, b), that is, the left end point of s. By doing this, we make sure that all the points covered by L cover all the corresponding segments by a single vertical line. Also, the corresponding vertical line does not cover any other segments of S. If this was not true, then there is a segment whose corresponding point lies neither on l_1 nor l_2. But, according to our construction, for each segment $s \in S$, the abscissa of corresponding point $p \in S'$ is same as the left end point of s. That means p should lie on one of l_1 or l_2.

Now, if L is horizontal, then draw a line through the corresponding segments of points covered by L. Since all the points are either on same y-coordinate or their y-coordinates differ by 2, each horizontal line can not have points on both the boundaries. This implies that there is only one horizontal line corresponding to each horizontal fat line. Thus, any line corresponding to a horizontal fat line covers the same number of segments in S as the points in S'. □

We use the above Lemma to establish $\mathbb{NP}$-hardness of RFLC.

Theorem 15. *Rectilinear Fat Line Cover with Integral Coordinates is* $\mathbb{NP}$*-hard.*

Proof. In the construction above, for each segment s, to construct a corresponding point p, we need to check whether it shares the same y-coordinate with another segment in S. This can be done in $O\,(n^2)$ worst case time. Thus, combining with Lemma 4, we have a polynomial time reduction from a known $\mathbb{NP}$-hard problem

minimum hitting of horizontal unit segments to RFLC. As a consequence, RFLC is $\mathbb{NP}$-hard. □

2. Solving for RFLC

Using the ideas from line cover and other similar problems [13], we can get fairly straight forward FPT and Kernelization algorithms to solve RFLC. We consider the case where each fat line is of fixed width 1. For kernelization, we give the following reduction rule.

Reduction Rule 1. *If there is an axis-parallel line (not a fat line) that covers at least $2k+1$ points in S, remove the extra points on the line from S.*

Claim 3. *Reduction Rule 1 is safe.*

Proof. Let there be a line l that contains at least $2k+1$ points. Let $s(l)$ denote the points covered by l. Without loss of generality, let l be a vertical line. Clearly, l would be part of some fat line from the solution, formed either with the line to its left, or to its right. For, if l was not part of some fat line from the solution, then, to cover $s(l)$, we would need at least $k+1$ horizontal fat lines to cover them with each covering at most two points. The same is true for any such horizontal line as well. So, if such a line is going to be part of the solution, we don't need more than $2k+1$ points to identify such a line, so, we can remove the rest of the points on that line from S. □

Based on above reduction rule, we can conclude a problem kernel of size $O(4k^2)$ for RFLC with fat lines of width 1.

Theorem 16. *After repeatedly applying reduction rule 1, a RFLC problem instance, where fat lines are of width 1, (S,k) is reduced to (S',k) where S' is $O(4k^2)$ points.*

Proof. It is clear to see that after repeatedly applying reduction rule 1, there is no line having more than $2k+1$ points in S. This means that if there exist a solution for RFLC of k fat lines, there would be at most $2k$ lines corresponding to them. Since each such line covers at most $2k+1$ points, we are left with no more than $4k^2+2k$ $= O(k^2)$ points left in the problem instance. □

Algorithm 6 Kernel-RFLC(S, k)

Input: An instance of RFLC (S, k)
Output: A solution to RFLC, if it exists
1: **if** $S = \emptyset$ **then**
2: Return "Accept"
3: **else if** $k = 0$ **then**
4: Return
5: **end if**
6: **while** there exists a line l such that $|s(l)| \geq 2k+1$ **do**
7: Remove extra points in l from S
8: **end while**
9: Enumerate-Solve(S, k) by brute force

The kernelization algorithm *Kernel-RFLC* uses the reduction rule to solve RFLC. It looks for lines that contain at least $2k+1$ points and removes rest of the points on each such line, from S. After exhaustively applying this reduction rule, lines with at most $2k+1$ points would remain, and the *Enumerate-Solve* function solves this instance by brute force. For running time analysis of *Kernel-RFLC*, we observe that when we apply reduction rule, since lines are only axis-parallel, we have only $2n$ lines to check from. This can be done in $O(n^2)$. Now, we have only $O(4k^2+2)$ points left. To find out k fat lines where each point has possible 4 candidate fat lines, it takes $O^*\left((4k^2+2)^{4k}\right)$ time.

The existence of a problem kernel with respect to parameter k implies the problem is in FPT [25]. Thus, we wish to explore the possibility an exact algorithm with respect to parameter k using another technique. To achieve this, a trivial idea is the following: For each point in the plane, there are four possible fat lines, two vertical

and two horizontal, that can cover it (see Fig. 11). The approach then is to pick a point, draw these four possible fat lines, and branch recursively by including one of these fat lines at a time and removing the points covered. This gives a running time of $O^*\left(4^k\right)$. However, using the following observation, we can reduce the four possible fat lines to three.

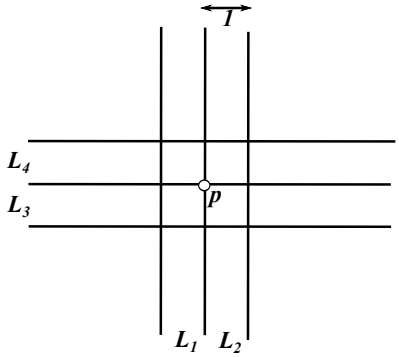

Fig. 11. Candidate rectilinear fat lines for a point p

Consider the convex hull C of the set of points S. Find the smallest axis aligned rectangle that encloses C. Clearly, at least two of the edges cover points from S. Pick a point p on the farthest edge of this rectangle. Without loss of generality, suppose this farthest edge is vertical and p is the right most point. Now, consider the four possible fat lines (two horizontal L_1, L_2, two vertical L_3, L_4), through p.

We claim that there is at least one fat line $L_i : (l_1, l_2)$ such that all points covered by L_i either lie on l_1 or l_2. And therefore, we can eliminate this fat line from consideration leaving only three possible fat lines that can cover this point p: one vertical, and two horizontal. This holds true even when the farthest edge is horizontal. If, in case the point lies on the corner of the rectangle, we can pick any of the case, horizontal or vertical, and continue. The algorithm *Exact-RFLC* uses this idea to achieve a FPT-algorithm for RFLC for fat lines of width 1, with respect to parameter k.

Algorithm 7 Exact-RFLC(S, C, k)

Input: An instance of RFLC: set of points S, RFLC C, parameter k
Output: A RFLC of at most k fat lines, if it exists, $\emptyset$ otherwise

if $S \neq \emptyset$ and $k = 0$ **then**
  Return $\emptyset$
else if $S = \emptyset$ **then**
  Return C
end if
Find a point $p \in S$ such that p's rectilinear distance is largest from origin
if p is the top most (or bottom most) point on convex hull of S **then**
  Draw a horizontal fat line L_1 through p
  Draw vertical fat lines L_2 and L_3 through p
  $C_1 =$ Exact-RFLC$(S \setminus s(L_1), C \cup \{L_1\}, k-1)$
  $C_2 =$ Exact-RFLC$(S \setminus s(L_2), C \cup \{L_2\}, k-1)$
  $C_3 =$ Exact-RFLC$(S \setminus s(L_3), C \cup \{L_3\}, k-1)$
  if $C_1 = \emptyset$ and $C_2 = \emptyset$ and $C_3 = \emptyset$ **then**
    Return $\emptyset$
  else
    Return $min\{C_i : i = 1, 2, 3 \mid C_i \neq \emptyset\}$
  end if
else if p is the right most(or left most) point on convex hull of S **then**
  Draw a vertical fat line L_1 through p
  Draw horizontal fat lines L_2 and L_3 through p
  $C_1 =$ Exact-RFLC$(S \setminus s(L_1), C \cup \{L_1\}, k-1)$
  $C_2 =$ Exact-RFLC$(S \setminus s(L_2), C \cup \{L_2\}, k-1)$
  $C_3 =$ Exact-RFLC$(S \setminus s(L_3), C \cup \{L_3\}, k-1)$
  if $C_1 = \emptyset$ and $C_2 = \emptyset$ and $C_3 = \emptyset$ **then**
    Return $\emptyset$
  else
    Return $min\{C_i : i = 1, 2, 3 \mid C_i \neq \emptyset\}$
  end if
end if

Theorem 17. *If there exists a Rectilinear Fat Line Cover of size k, for a set S of Integral Coordinates, such that each fat line is of width 1, then Exact-RFLC solves it correctly in $O^*\left(3^k\right)$ time.*

Proof. First, consider the proof of correctness of the algorithm. During any recursive call, if the set S is non-empty but $k = 0$, this means there are more than k fat lines needed to cover S along that path. The algorithm return $\emptyset$ in this case (lines 1-2). If, on the other hand, S becomes empty, that means the path covered all the points in S using no more than k fat lines. The algorithm returns fat line cover C in that case (lines 3-4). Now we prove that the approach used in the algorithm (lines 6-26) does indeed consider all possible candidate fat lines. Consider any point $p \in S$. Clearly, there are four possible fat lines that can cover p. Suppose these four lines are $\{L_i : i = 1, 2, 3, 4\}$, with L_1, L_2 vertical and L_3, L_4 horizontal, such that p lies on right boundary of L_1, left boundary of L_2, upper boundary of L_3 and lower boundary of L_4 (see Fig. 12).

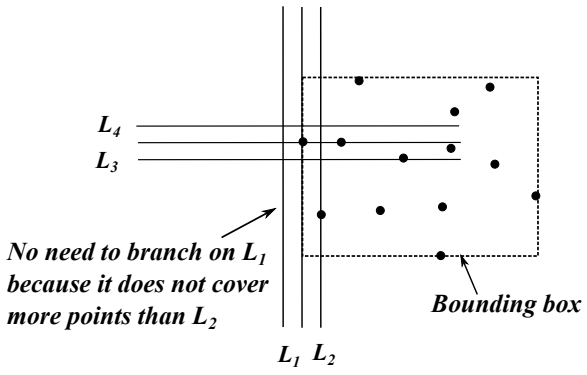

Fig. 12. FPT for RFLC

If it is the left most point, we can ignore L_1 since L_2 contains all the points covered by L_1 including p. If, this was not true, then there would be a point on left boundary of L_1, which implies that p is not the farthest rectilinear point, a contradiction of our

assumption. Now suppose L_1 is still part of the solution. This is only possible when there were points on the right boundary of L_1 which have been covered by L_2 in some other branch of recursion and have been removed from S. In that case, we can still ignore L_1 since it has already been considered in some previous branch. Similarly, if p is right most point we can ignore L_2, if p is top most we ignore L_4, if p is bottom most we ignore L_3.

Thus, in any call to Exact-RFLC(S, C, k), three branches for candidate fat lines suffice to be considered for covering p. Inductively, to cover all the points in S, we need to consider three possible fat lines in each branch for the farthest point in S. Lines 6-26 consider these cases, and branch depending upon whether p is horizontally or vertically farthest, and returns either empty set in case of failure, or the smallest of the non-empty cover sets, if multiple sets of size at most k are possible.

For the computation of running time, since each branch includes at least one fat line in the solution, and we want the answer for k, each branch of Exact-RFLC can go at most depth k, where we would be able to answer 'yes' or 'no' depending upon whether we have any points left to cover or not. Determining the farthest point p can be done in polynomial time by first sorting the points according to their x and y coordinates separately, and then determining the farthest from origin. Rest of the steps in each branch are linear. So, each branch takes overall polynomial steps. Now, for every point, we have three possibilities of fat lines, as described above, and a branching depth of at most k, it implies that this algorithm takes $O\left(n^2 \cdot 3^k\right)$ running time. □

Next, we explore the possibility of having an approximation algorithm for RFLC. The motivation comes from the positive results with respect to approximation for hitting objects with lines. The problem of hitting segments in the plane (horizontal

or vertical or both) and hitting objects with lines with finite number of slopes have constant ratio polynomial time approximation algorithms with constant ratio [8]. To study RFLC in this direction, we consider the following restricted version of line cover:

Definition 16 (RECTILINEAR LINE COVER)**.** *Given a set S of n points in the plane, find minimum number of axis-parallel lines to cover them.*

This problem is solvable in polynomial time, which we state as proposition below:

Proposition 3. *[8] The rectilinear line cover problem with integral coordinates in the plane can be solved in polynomial time.*

The idea to solve rectilinear line cover is as follows: Each point $p \in S$ has only two possible candidate lines, one vertical and one horizontal. Enumerate all these possible lines in C (at most $2n$ of them). Create a graph $G : (V, E)$ where each vertex in V corresponds to a line in C and there is an edge between two vertices of V if and only if the lines corresponding to these vertices intersect in a point from the input set. Since two parallel lines do not intersect, G is a bipartite graph, and so, the rectilinear fat line cover for points in S corresponds to vertex cover of size k for G, which is polynomial time solvable (König's theorem [45]).

Since rectilinear line cover is polynomial time solvable, we can utilize the algorithm to solve it to approximate RFLC. The intuition is that each line which covers a point $p \in S$, can form the boundary of at most two fat lines. As a result, for each line l in line cover of size k, at most $2k$ candidate lines are possible. Thus, a RFLC, with fat lines of width 1, which covers the set of points S is at most twice the size of rectilinear line cover of points S. The algorithm Approx-RFLC utilizes this idea to solve RFLC with an approximation ratio 2.

Algorithm 8 Approx-RFLC($\mathcal{S}, \mathcal{C}, k$)

Input: An instance of RFLC: set of points $\mathcal{S}$, RFLC $\mathcal{C}$, parameter k
Output: A RFLC of at most $2k$ fat lines, each of width 1
1: Solve the set $\mathcal{S}$ for rectilinear line cover, let the set be $\mathcal{L}$
2: **for** Each line $l \in \mathcal{L}$ **do**
3: Draw two lines l_1, l_2 parallel to l, each of width 1
4: Add fat lines $L_1 : (l_1, l)$ and $L_2 : (l, l_2)$ to the solution set $\mathcal{F}$
5: **end for**
6: Return $\mathcal{F}$

Theorem 18. *The rectilinear fat line cover with integral coordinates has a polynomial time approximation algorithm of approximation ratio 2.*

Proof. We claim that optimal solution for line cover is at most twice the optimal solution of RFLC. Let C be the optimal solution for RFLC covering input set S with fat lines, and there is a fat line $L : (l_1, l_2) \in C$. l_1 and l_2 are two parallel lines, both either vertical or horizontal, and cover some points in S. Thus, each of l_1 and l_2 is a candidate line to rectilinear line cover of S. Now, to cover all the points in S with lines, in the worst case, both l_1 and l_2 would be needed. Therefore, each fat line corresponds to two lines that cover points in S.

On the other hand, suppose C' is a solution to rectilinear line cover. Each line $l \in C'$ is either vertical or horizontal. Also, each such line l contributes itself as a boundary to two possible fat lines. In worst case, both fat lines would be needed in optimal RFLC. Thus, the optimal solution to rectilinear line cover is at most twice the size of optimal solution of RFLC. Since rectilinear line cover is polynomial time solvable, RFLC has a PTAS with approximation ratio 2. □

The running time analysis of *Approx-RFLC* is done as follows: Line 1 computes rectilinear line cover for points in S. The best known running time for solving minimum vertex cover on bipartite graphs is $O(V^2\ log\ V + VE)$. For each point $p \in S$, there are two candidate rectilinear lines, thus $V = 2n$, where n is the number of points

in S. Now, since the lines are rectilinear, only two lines can intersect at a point p, hence, there are at most n edges in graph $G : (V, E)$. Hence, rectilinear line cover takes $O\left(4n^2\ log\left(2n\right) + 2n^2\right) = O\left(n^2\ log\ n\right)$ time. Now, L is of size $O\left(n\right)$, hence, lines 2-5 take $O\left(n\right)$ time. Thus, the total running time of *Approx-RFLC* is $O\left(n^2\ log\ n\right)$.

In this chapter, we did a systematic study of fat line cover, the problem of covering a set of points in the plane with k fat lines, each of width w. With an aim at achieving a FPT-algorithm, we considered closest-band problem which asks to find a fat line of minimum width to cover all the points in the input. We gave a polynomial time algorithm with worst case complexity of $O\left(n^2\ log\ n\right)$ to solve closest-band. Fat line cover, on the other hand, turned out to be much harder. We showed that it is $\mathbb{NP}$-hard as well as $W[1]$-hard with respect to number of fat lines k. In addition, our study also revealed that when fat lines are restricted to be axis-parallel, or the points are not allowed to be close enough, the problem still remain $W[1]$-hard.

We also considered RFLC, a very restricted form of fat line cover, showed that it is still $\mathbb{NP}$-hard. With respect to parameterized complexity we arrived at positive results for this problem when fat lines are of width $w = 1$. We demonstrated that RFLC exhibits a quadratic kernel and the kernelization algorithm to solve RFLC takes $O^*\left(\left(4k^2 + 2\right)^{4k}\right)$ running time. A FPT-algorithm of running time $O^*\left(3^k\right)$ was also proposed in this chapter. We concluded our study by giving a polynomial time algorithm with an approximation ratio 2, and a running time of $O\left(n^2\ log\ n\right)$ in the worst case.

CHAPTER VI

CONCLUSIONS AND FUTURE WORK

A. Conclusions

In this work, we studied two geometric cover problems: conic cover and fat line cover, with respect to computational complexity. Both of these variations open up new area of algorithmic research. Based on the algorithm for hyperplane cover, we were able to show that both conic cover and parabola cover exhibit fixed-parameter tractability. In particular, we demonstrated a FPT-algorithm for conic cover with respect to parameter k. Improving upon the same idea, we successfully gave a FPT-algorithm for parabola cover with a better worst case running time than general conic cover.

For fat line cover, we started by exploring the idea of closest-band in a hope to extend it to find k fat lines. Instead, we arrived at its $W[1]$-hardness with respect to the parameter k. This answers in negative the question of a FPT-algorithm for fat line cover. In addition, we showed that some restricted versions of fat line cover are also $W[1]$-hard. These results are analogous to those given for general stabbing problem. The inability to have a FPT-algorithm for general fat line cover supports the notion that, in general, geometric cover problems are hard with respect to parameter k.

Realizing that some restricted versions of geometric covering problems have known FPT-algorithms, e.g., line cover, stabbing axis-parallel disjoint unit squares with axis-parallel lines, etc., we studied rectilinear fat line cover (RFLC). We proved that it is $\mathbb{NP}$-hard, and also in FPT when fat lines are of width 1. This also implied a kernelization algorithm. In addition, we gave a polynomial time approximation algorithm with a constant ratio. This gives a logical end to our research on fat line

cover.

B. Future Work

Although we provided FPT-algorithms for conic cover and parabola cover, we have not shown that the running time results are tight. A possible goal for future could be to study the tightness of these results. In addition, we would like this idea to be studied - akin to hyperplane cover as an extension of line cover - with respect to any general second degree curves in higher dimensional space. Our results are purely theoretical and apply a standard model to all kinds of problems.

We could explore covering of points with respect to higher degree curves in the plane, like cubics, quadric surfaces, etc. which are fairly common in the fields of AI, machine learning, etc. In geometry, we have results which can transform a problem in higher degree and in lower dimension space to one in lower degree and higher dimensional space. It would be interesting to use this approach to see if we can get a FPT-algorithm for general curves in higher degree in space in any dimension. Studying these problems further with a goal of utilizing similar geometric insights and arrive at novel FPT-algorithms would be an interesting area to work on.

The results for RFLC given in this work restrict fat lines to be of width exactly 1. We hope that in future, results pertaining to fat lines of any width c would be worked on. None of the results for RFLC have been proved to be tight, and so, there is still scope of further research in this respect. Can the running time of FPT-algorithm for RFLC be improved, or the kernel size? Similarly, the approximation algorithm has a ratio 2, can it be reduced further? Does it exhibit a PTAS or FPTAS? These questions are still open and we hope that in future, these would be interesting problems to work on. With regards to general geometric cover problems, we would

like to see a categorization of these problems being explored in future, which could indicate which problems are hard with respect to parameterized complexity theory.

Another possible direction is to consider different ground fields: can we use properties from precision based arithmetic, rational numbers, real numbers, complex numbers, finite fields, etc. to get a better algorithm? For each of the different field types, a different version of the problem, and hence, a possibility of a different algorithm could be explored.

REFERENCES

[1] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman & Co., New York, 1979.

[2] Stephen A. Cook, "The complexity of theorem-proving procedures," in *Proceedings of the third annual ACM symposium on Theory of computing*. 1971, STOC '71, pp. 151–158, ACM.

[3] Richard M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, R. Miller and J. Thatcher, Eds., pp. 85–103. Plenum Press, 1972.

[4] Vladimir Estivill-Castro, Apichat Heednacram, and Francis Suraweera, "The rectilinear k-bends TSP," in *Proceedings of the 16th annual international conference on Computing and combinatorics*. 2010, COCOON'10, pp. 264–277, Springer-Verlag.

[5] Michael R. Garey and David S. Johnson, "The Rectilinear Steiner Tree Problem in NP Complete," *SIAM Journal of Applied Mathematics*, vol. 32, pp. 826–834, 1977.

[6] Nimrod Megiddo, "Linear-Time Algorithms for Linear Programming in R^3 and Related Problems," *SIAM J. Comput.*, vol. 12, no. 4, pp. 759–776, 1983.

[7] Kasturi Varadarajan, Srinivasan Venkatesh, Yinyu Ye, and Jiawei Zhang, "Approximating the Radii of Point Sets," *SIAM J. Comput.*, vol. 36, no. 6, pp. 1764–1776, Feb. 2007.

[8] Refael Hassin and Nimrod Megiddo, "Approximation algorithms for hitting objects with straight lines," *Discrete Appl. Math.*, vol. 30, no. 1, pp. 29–42, Jan. 1991.

[9] Stefan Langerman and Pat Morin, "Covering Things with Things," in *Proceedings of the 10th Annual European Symposium on Algorithms*. 2002, ESA '02, pp. 662–673, Springer-Verlag.

[10] Nimrod Megiddo and Arie Tamir, "On the complexity of locating linear facilities in the plane," *Oper. Res. Lett.*, vol. 1, no. 5, pp. 194–197, Nov. 1982.

[11] Arie Tamir, "New results on the complexity of p-center problems," *SIAM J. Comput*, vol. 12, pp. 751–758, 1983.

[12] Panos Giannopoulos, Christian Knauer, Günter Rote, and Daniel Werner, "Fixed-parameter tractability and lower bounds for stabbing problems," *CoRR*, vol. abs/0906.3896, 2009.

[13] Panos Giannopoulos, Christian Knauer, and Sue Whitesides, "Parameterized complexity of geometric problems," *Comput. J.*, vol. 51, no. 3, pp. 372–384, 2008.

[14] Sariel Har-Peled, "No, Coreset, No Cry," in *FSTTCS 2004: Foundations of Software Technology and Theoretical Computer Science*, Kamal Lodaya and Meena Mahajan, Eds., vol. 3328 of *Lecture Notes in Computer Science*, pp. 324–335. Springer Berlin / Heidelberg, 2005.

[15] Mihai Bādoiu, Sariel Har-Peled, and Piotr Indyk, "Approximate clustering via core-sets," in *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*. 2002, STOC '02, pp. 250–257, ACM.

[16] Noga Alon, "A Non-linear Lower Bound for Planar Epsilon-Nets," in *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. 2010, FOCS '10, pp. 341–346, IEEE Computer Society.

[17] János Pach and Gábor Tardos, "Tight lower bounds for the size of epsilon-nets," in *Proceedings of the 27th annual ACM symposium on Computational geometry*. 2011, SoCG '11, pp. 458–463, ACM.

[18] Magdalene Grantson and Christos Levcopoulos, "Covering a set of points with a minimum number of lines," in *Proceedings of the 6th Italian conference on Algorithms and Complexity*. 2006, CIAC'06, pp. 6–17, Springer-Verlag.

[19] Jianxin Wang, Wenjun Li, and Jianer Chen, "A parameterized algorithm for the hyperplane-cover problem," *Theor. Comput. Sci.*, vol. 411, no. 44-46, pp. 4005–4009, 2010.

[20] S.L. Loney, *The Elements of Coordinate Geometry*, Macmillan and Co., New York, 1896.

[21] David Salesin, Jorge Stolfi, and Leonidas Guibas, "Epsilon geometry: building robust algorithms from imprecise computations," in *Proceedings of the fifth annual symposium on Computational geometry*. 1989, SCG '89, pp. 208–217, ACM.

[22] Rodney G. Downey, Michael R. Fellows, and Michael A. Langston, "The Computer Journal Special Issue on Parameterized Complexity: Foreword by the Guest Editors," *Comput. J.*, vol. 51, no. 1, pp. 1–6, 2008.

[23] Rodney G. Downey and Dimitrios M. Thilikos, "Confronting Intractability via Parameters," *CoRR*, vol. abs/1106.3161, 2011.

[24] Jianer Chen, Iyad A. Kanj, and Ge Xia, "Improved parameterized upper bounds for vertex cover," in *Proceedings of the 31st international conference on Mathematical Foundations of Computer Science*. 2006, MFCS'06, pp. 238–249, Springer-Verlag.

[25] R. Niedermeier, *Invitation to Fixed-Parameter Algorithms*, Oxford Lecture Series in Mathematics And Its Applications. OUP Oxford, New York, 2006.

[26] Yixin Cao, Jianer Chen, and Yang Liu, "On feedback vertex set new measure and new structures," in *Proceedings of the 12th Scandinavian conference on Algorithm Theory*. 2010, SWAT'10, pp. 93–104, Springer-Verlag.

[27] Rajesh Chitnis, MohammadTaghi Hajiaghayi, and Dániel Marx, "Fixed-parameter tractability of directed multiway cut parameterized by the size of the cutset," in *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*. 2012, SODA '12, pp. 1713–1725, SIAM.

[28] Rodney G. Downey and Michael R. Fellows, *Parameterized Complexity*, Springer-Verlag, New York, 1999, 530 pp.

[29] J. Flum, *Parameterized Complexity Theory*, Texts in Theoretical Computer Science. Springer, Berlin, 2006.

[30] C. M. Papadimitriou, *Computational Complexity*, Addison-Wesley, Reading, Massachusetts, 1994.

[31] Sanjeev Arora, "Polynomial time approximation schemes for Euclidean TSP and other geometric problems," in *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*. 1996, FOCS '96, pp. 2–, IEEE Computer Society.

[32] Edward G. Coffman Jr. and Ronald L. Graham, "Optimal Scheduling for Two-Processor Systems," *Acta Inf.*, vol. 1, pp. 200–213, 1972.

[33] V. Vazirani, *Approximation Algorithms*, Springer-Verlag New York, Inc., 2001.

[34] Vullikanti S. Anil Kumar, Sunil Arya, and Hariharan Ramesh, "Hardness of Set Cover with Intersection 1," in *Proceedings of the 27th International Colloquium on Automata, Languages and Programming*. 2000, ICALP '00, pp. 624–635, Springer-Verlag.

[35] Michael Dom and Somnath Sikdar, "The parameterized complexity of the rectangle stabbing problem and its variants," in *In Proc. 2nd FAW, volume 5059 of LNCS*. 2008, pp. 288–299, Springer.

[36] Michael R. Fellows, Danny Hermelin, Frances Rosamond, and Stéphane Vialette, "On the parameterized complexity of multiple-interval graph problems," *Theor. Comput. Sci.*, vol. 410, no. 1, pp. 53–61, Jan. 2009.

[37] Fred L. Bookstein, "Fitting conic sections to scattered data," *Computer Graphics and Image Processing*, vol. 9, no. 1, pp. 56 – 71, 1979.

[38] Peipei Li, Caiming Zhang, Kunpeng Wang, and Xuemei Li, "A new method of fitting conic to scattered data," in *Image and Signal Processing (CISP), 2010 3rd International Congress on*, Oct. 2010, vol. 6, pp. 2722 –2726.

[39] Keith Alan Paton, "Conic sections in chromosome analysis," in *Proceedings of the 1st international joint conference on Artificial intelligence*. 1969, IJCAI'69, pp. 105–105, Morgan Kaufmann Publishers Inc.

[40] Michael A. Lachance and Arthur J. Schwartz, "Four point parabolic interpolation," *Comput. Aided Geom. Des.*, vol. 8, no. 2, pp. 143–149, May 1991.

[41] H. Dörrie, *100 Great Problems of Elementary Mathematics*, Dover Books on Mathematics. Dover Publications, New York, 1965.

[42] Godfried Toussaint, "Solving Geometric Problems with the Rotating Calipers," *In Proc. IEEE MELECON 83*, pp. 10–02, 1983.

[43] Dániel Marx, "Efficient approximation schemes for geometric problems?," in *Proceedings of the 13th annual European conference on Algorithms*, Berlin, Heidelberg, 2005, ESA'05, pp. 448–459, Springer-Verlag.

[44] Dániel Marx, "Parameterized complexity of independence and domination on geometric graphs," in *Proceedings of the Second international conference on Parameterized and Exact Computation*. 2006, IWPEC'06, pp. 154–165, Springer-Verlag.

[45] Dénes König, "Gráfok és mátrixok," *Matematikai és Fizikai Lapok*, vol. 38, pp. 116–119, 1931.