Optimization of Geometric Measures of Sets of Moving Objects

by

Ikaro Ruan Penha Costa

A thesis submitted to
The Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements
of the degree of

Master of Science

Department of Computer Science
The University of Manitoba
Winnipeg, Manitoba, Canada
June 2024

Thesis advisor

Stephane Durocher and Prosenjit Bose

Author

Ikaro Ruan Penha Costa

Optimization of Geometric Measures of Sets of Moving Objects

Abstract

Given a set S of objects, each moving with linear motion in $\mathbb{R}^d$, consider the diameter $\mathsf{D}(S,t)$ of S at time t. In this thesis we explore optimization of extent and proximity measures of S. For instance, one possibility is to identify minimum diameter $\mathsf{D}(S,t)$ of S over the domain of time t. $\mathsf{D}(S,t)$ is an example of a measure of extent of S. On the basis of this model, other geometric measures could also be explored to be optimized for sets of objects in motion. Let n be the cardinality of S and let $\mathsf{M}(S,t)$ be a geometric measure of extent or proximity at time t. Given an integer k, select a subset $Q \subseteq S$ such that $|Q| = k$ and Q has extreme measure $\mathsf{M}(Q,t)$ over all possible subsets Q of cardinality k. The present thesis focuses on minimizing and maximizing $\mathsf{M}(Q,t)$, in one and two dimensions ($d = 1$ or $d = 2$), for which the measure corresponds to set diameter, set width, minimum axis-aligned bounding box, and minimum enclosing disk. For each measure, exact polynomial-time algorithms are proposed for selecting an optimal subset of S and finding the time t of which the subset optimizes the measure.

Keywords: Moving Objects, Extent Measure, Optimization, Polynomial Motion, Linear Movement.

Contents

List of Figures

Acknowledgments

I primarily thank God for all the blessings and opportunities He has given me. I would like to express my gratitude to my family for all the support, especially my mother, Christiana, and my grandparents, Terezinha, Cícera, and Costa Neto. I also thank my aunt, Ivina, for being present in my life.

I would also like to thank my supervisors Steph Durocher and Prosenjit Bose for all the guidance, advice, support, and patience. I also want to express my gratitude to Franklin Bristow for all the lessons and for teaching me how to be a better instructor. I thank Robert Guderian for his encouragement. I also thank the University of Manitoba and the Department of Computer Science for their support.

In terms of advice and guidance, I also express immense gratitude to Juliana Ribeiro Alexandre de Sousa. A special thanks goes to my friends Jéssica Sayuri Tahara and Paloma Morais for believing in me more than I do. Of equal importance, I am grateful for all the support from my friends Avleen Kaur, Hermie Monterde, and Brock Klippenstein.

Last but not least, I would like to thank myself for all my dedication and not giving up on graduate studies. After all of my previous experiences as a graduate student, resilience is an important trait for which I am thankful.

This thesis is dedicated to my mother, Christiana, and my grandparents: Terezinha Penha, Vicente Penha (in memorian), Cícera Costa, and Costa Neto.

Chapter 1

Introduction

Algorithms for extent and proximity measures have been widely studied in the field of computational geometry. For sets of static objects, algorithmic solutions for problems involving motion in geometry date back to the end of the 1990's by Guibas [1998]. More recently, optimization of minimum spanning trees of linearly moving objects has been studied by Akitaya et al. [2021]. Inspired by this model, the optimization of other geometric measures on sets of moving objects is proposed. Specifically, the measures are: diameter, width, minimum axis-aligned bounding box, and minimum enclosing disk of a set of moving objects.

Given a set of points $P \subset \mathbb{R}^d$, the maximum distance between any two points in P is the *diameter* of P. The smallest axis-parallel d-dimensional box enclosing P is called the *minimum axis-aligned bounding box* of P. The smallest d-dimensional ball containing P is called the *minimum enclosing ball* of P. Furthermore, the width of P is the minimum distance between any pair of parallel, but oppositely oriented, half-spaces each containing P.

The aforementioned measures are closely related to other problems in Computational Geometry, such as triangulations, tessellations, and robot motion planning, as seen in de Berg et al. [2008]. That is, diameter, width, bounding boxes, and minimum enclosing circles can be used to define the region of space to be triangulated or even used to define the complexity of robot motion as shown by Elbanhawi and Simic [2014]. For example, the longest side of the minimum bounding box for a set of moving agents is used as a complexity measure of its movement as explained by Cicerone et al. [2021]. These topics have been extensively studied in the literature, as presented by de Berg et al. [2008] and Devadoss and O'Rourke [2011].

Furthermore, extent and proximity measures also have their importance in the field of Geometric Processing for parameterization schemes and shape correspondence, as seen in Lévy et al. [2002] and Liu et al. [2008]. In parameterization, the goal is to provide a continuous map that transforms any surface to a plane. In order to reduce distortion of the parameterization, different target planes are used. Lévy et al. [2002] shows that, for points p and q, the points that realize the diameter of the surface, the plane defined by p and q will produce reduced distortion when used as target for the parameterization.

Kinetic geometric problems were first examined by the computational geometry community near the end of the 1990's by Basch et al. [1997]. A kinetic problem is one whose set S is composed of objects moving in Euclidean space with respect to a continuous time interval. The main idea is to maintain a set of properties as the objects move continuously (as opposed to a sequence of discrete changes). One well developed approach is a framework called a Kinetic Data Structure (KDS), as explained by

Guibas [2018]. The objective is to build geometric structures to compute the measures continuously over a set of discrete events, where the geometric measures and properties are updated discretely as time passes. The KDS consists of a set of measurable properties and certificates that validate the existence of these properties. The KDS maintains these properties and updates certificates as each certificate becomes invalid throughout the whole motion of the objects in S.

Work done by Gupta et al. [1996] and Chan [2004] exemplifies that computing optimization problems over moving points via an off-line algorithm can be more efficient than using kinetic data structure framework. When the given measure and its optimization can be computed off-line, as opposed to being maintained continuously throughout the motion (e.g., finding the minimum bounding box, meanwhile the on-line algorithm computes how the bounding box change over time).

Chan [2004] solves the minimum diameter of sets of points moving in linear trajectories with constant velocity in $\mathcal{O}(n \log n)$ time. More recently, Akitaya et al. [2021] propose off-line solutions to optimizations of minimum moving spanning trees. Let S be a set of moving points and let $\mathcal{T}(S)$ be the set of all possible spanning trees obtained from points in S. The problem proposed by Akitaya et al. [2021] is to find the minimum moving spanning tree $J \in \mathcal{T}(S)$ which minimizes one of two different measures. Let d denote the Euclidean distance between two points. The considered measures for $J \in \mathcal{T}(S)$ at time t are:

(M1) $b_J(t) = \sup_{pq \in J} \mathrm{d}(p, q)$;

(M2) $w_J(t) = \sum_{pq \in J} \mathrm{d}(p, q)$.

For minimizing $b_J(t)$ over $J \in \mathcal{T}(S)$ and $t \in [0,1]$, Akitaya et al. [2021] propose an exact $\mathcal{O}(n^2)$-time algorithm. For the minimization of $w_J(t)$, Akitaya et al. [2021] present both an $\mathcal{O}(n^2)$-time 2-approximation algorithm and an $\mathcal{O}(n \log n)$-time $(2+\varepsilon)$-approximation algorithm for any $\varepsilon > 0$. By comparison, the KDS framework proposed by Rahmati and Zarei [2012] for maintaining the kinetic MST processes $\mathcal{O}(n^4)$ events and each event takes $\mathcal{O}(\log^2 n)$ time to revalidate.

The main caveat to comparing the time complexity of on-line and off-line algorithms is that on-line algorithms often assume that the input arrives progressively over time, whereas off-line algorithms need the whole input up front. The goal of a KDS is to process or assert properties and metrics as objects continuously change positions. For example, a Kinetic Tournament can be used to maintain a certain measure during the entire motion using a KDS. The on-line algorithm needs to process the entire motion of the elements in S. In contrast, an off-line algorithm may ignore parts of the motion that does not include local extrema.

Consequently, inspired by the computation model defined in Akitaya et al. [2021], this thesis seeks to optimize geometric measures on sets and subsets of moving objects by exploring the polynomial motion of each object via an *off-line* algorithm. Let $\mathsf{M}(Q_k, t)$ denote a generic extent or proximity measure a subset of moving objects $Q_k \subset S$ of fixed size, $|Q_k| = k \leq |S|$, at time t. The goal is to find:

$$\min_{Q_k \subset S} \min_{t \in [0,1]} \mathsf{M}(Q_k, t)$$

This problem is called the Min-Min optimization for the measure M.

The variants of the type Max-Min, Min-Max, and Max-Max optimizations are also

examined in this thesis. Different examples of the measure M examined include set diameter, set width, minimum axis-aligned bounding box, and minimum enclosing disk. Exact algorithms are given for sets of objects with linear motion in one and two dimensions.

The time complexity for the proposed algorithms is summarized in Tables 1.1 and 1.2. The chapters are organized as follows:

- Chapter 2 discusses all the necessary background and algorithms for optimizing each measure. Then, the problems examined in this thesis are defined formally, and previous related work is reviewed.

- Chapter 3 examines the optimization of diameter for one-dimensional subsets of linearly moving objects. Set width, minimum axis-aligned bounding box, and minimum enclosing disk are shown to be equivalent to diameter on the one-dimensional Euclidean space.

- Chapter 4 provides polynomial-time solutions for each measure and each maximization and minimization variant considered in this thesis when the objects are moving on the two-dimensional Euclidean plane.

- Chapter 5 revisits the problem definition and the proposed algorithmic solutions. Directions for future work are also discussed.

One-Dimensional Problems	
	Time Complexity
Diameter	
Min-min	$\mathcal{O}(n^2)$
Max-min	$\mathcal{O}(n \log n)$
Min-max	$\mathcal{O}(n^2)$
Max-max	$\mathcal{O}(n)$

Table 1.1: Summary of one results for each measure and each optimization variant of geometric measures of sets of linearly moving objects.

Two-Dimensional Problems	
	Time Complexity
Diameter	
Min-min	$\mathcal{O}(n^3)$
Max-min	$\mathcal{O}(n \log n)$
Min-max	$\mathcal{O}(n^2)$
Max-max	$\mathcal{O}(n)$
Width	
Min-min	$\mathcal{O}(n^4)$
Max-min	$\mathcal{O}(n^4)$
Min-max	$\mathcal{O}(n^4)$
Max-max	$\mathcal{O}(n^4)$
Axis-aligned Bounding Box: Area	
Min-min	$\mathcal{O}(n^5 \log n)$
Max-min	$\mathcal{O}(n^5 \log n)$
Min-max	$\mathcal{O}(n^5 \log n)$
Max-max	$\mathcal{O}(n^5 \log n)$
Axis-aligned Bounding Box: Perimeter	
Min-min	$\mathcal{O}(n^5 \log n)$
Max-min	$\mathcal{O}(n^5 \log n)$
Min-max	$\mathcal{O}(n^2)$
Max-max	$\mathcal{O}(n^2)$
Minimum Enclosing Disc Area	
Min-min	$\mathcal{O}(n^5)$
Max-min	$\mathcal{O}(n^5)$
Min-max	$\mathcal{O}(n^5)$
Max-max	$\mathcal{O}(n^5)$

Table 1.2: Summary two-dimensional results for each measure and each optimization variant of geometric measures of sets of linearly moving objects.

Chapter 2

Background, Problem Definition, and Related Work

The extent and proximity measures defined in this chapter are in d-dimensional Euclidean space, $\mathbb{R}^d$. The problems discussed in this thesis focus on the dimensions $d = 1$ and $d = 2$. Section 2.1 will define the necessary theory and measures in order to elucidate the problems defined in Section 2.2. The previous related work is discussed in Section 2.3.

2.1 Background and Definitions

2.1.1 Set Diameter

Denote by P a set of n points in $\mathbb{R}^d$. Let $\mathrm{d}(p, q)$ denote the Euclidean distance between $p, q \in \mathbb{R}^d$. The diameter $\mathsf{D}(P)$ is defined as follows.

Definition 1. *The diameter of P, denoted $\mathsf{D}(P)$, is the maximum Euclidean distance between two points of the set P. For $p, q \in P$:*

$$\mathsf{D}(P) = \max_{p,q \in P} \mathrm{d}(p, q)$$

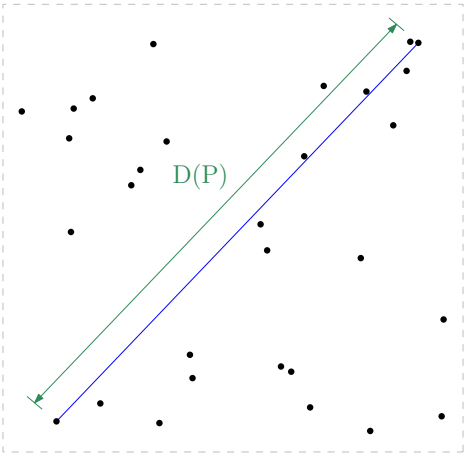


Figure 2.1: Example of set of points in two dimensions and its diameter.

In one dimension, all the points of P lie on the real line. To compute the diameter when $d = 1$, traverse the set of points and maintain the minimum and maximum of the coordinate values. Then, the difference between the maximum and minimum coordinates will be the diameter of P.

In higher dimension, an algorithm based on a convex hull construction has its own importance in the future discussion. As presented in Toussaint [1983, 2014], the pair of antipodal points that form the diameter of P are also elements of the convex hull of P. The convex hull corresponds to the smallest convex set containing P. From this, using the strategy of rotating calipers, Preparata and Shamos [2012] show that the diameter of P for $d = 2$ can be computed in $\mathcal{O}(n \log n)$ time by modifying the convex hull algorithm. Figure 2.1 shows an example of the diameter of a set of points in the

plane, $d = 2$. The diameter probem is harder for $d \geq 3$. When $d = 3$, Ramos [2000] develops a deterministic optimal $\mathcal{O}(n \log n)$-time diameter solution. When $d > 3$, the known deterministic and exact algorithms run in $\mathcal{O}(n^2)$ time as seen in Har-Peled [2001]. An approximate solution is proposed by Chan [2002] and runs in $\mathcal{O}(n + 1/\varepsilon^{d-1})$ time in $\mathbb{R}^d$ for any $\varepsilon > 0$.

2.1.2 Set Width

For a set of points $P \subset \mathbb{R}^d$, the width $\mathsf{W}(P)$ consists of the minimum distance between any two parallel hyperplanes such that the region between the hyperplanes contains P. In one dimension, the width represents the minimum distance between two points $u, v \in P$ so that all other points in P are in between u and v. The width $\mathsf{W}(P)$ is realized by the distance of extreme points. That is, when $d = 1$ the width of P is equivalent to the diameter $\mathsf{D}(P)$. When $d = 2$, the hyperplanes are two parallel lines, not necessarily axis-parallel, such that all points from P are inside the closed region between the lines. The minimum distance between any two such lines defines the width $\mathsf{W}(P)$.

This measure can also be defined in terms of half-spaces. In a general dimension d, let H and H' be two parallel oppositely oriented half-space boundaries. Denote by $\mathrm{d}(H, H')$ the distance between the two parallel half-spaces. In this case, we can characterize the width of a set of points as the minimum distance between H and H' such that the intersection of the half-planes contains P. Definition 2 formalizes the idea. Figure 2.2 shows an illustration of the two-dimensional width.

Definition 2. *The width* $\mathsf{W}(P)$ *of the set of points* P *is the smallest distance between*

the boundary of two parallel oppositely oriented half-spaces H and H' whose intersection contains P. That is:

$$\mathsf{W}(P) = \min_{\substack{H,H' \subset \mathbb{R}^d \\ P \subset (H \cap H')}} \mathrm{d}(H, H')$$

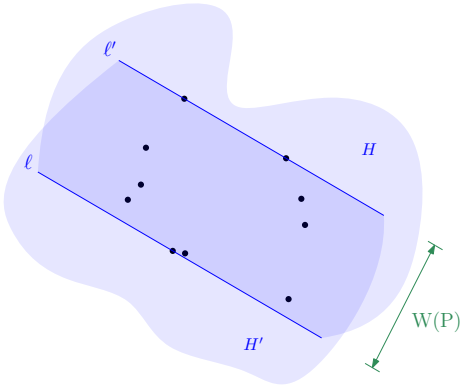


Figure 2.2: Example of width of a set of points in two dimensions based on the half-spaces definition.

When $d = 2$, a brute-force algorithm to compute the width of P consists of using a property shown by Preparata and Shamos [2012], where the width is realized by exactly three points of P. That is, two points define the boundary of H and the other point will give the direction and magnitude of the translation to form H'. Then, among all combinations of three points selected from P and all possible embeddings of H and H' from the three given points, find the one that gives the minimum $\mathrm{d}(H, H')$.

Since there are $\mathcal{O}(n^3)$ possible combinations of three points $\{u, v, w\} \subset P$, two points of $\{u, v, w\}$ can define H and the third point will give the translation of H to form H'. Let $\vec{h}$ denote the shortest vector of oriented distance from H to H', now let $\vec{h}(p)$ be the smallest oriented distance from the boundary of H and a point $p \in P$. If $\vec{h}(p)$ and $\vec{h}$ have same sign (i.e. same relative orientation from the boundary of H)

and $|\vec{h}(p)| \leq |\vec{h}|$, then p is contained in the intersection of H and H'. For each $p \in P$, it takes constant time to determine if the point p is inside $H \cap H'$. Consequently, it takes $\mathcal{O}(n)$ time to determine if $H \cap H'$ encloses P.

Hence, a solution to the width of P can derived from maintaining the combination $\{u, v, w\}$ whose distance $\mathrm{d}(H, H')$ is minimal and $H \cap H'$ encloses P. Since checking if $H \cap H'$ encloses P takes $\mathcal{O}(n)$ time, this algorithm takes $\mathcal{O}(n^4)$ time in total. Although width in two dimensions can be computed in $\mathcal{O}(n \log n)$ time, as shown by Preparata and Shamos [1985], this brute-force algorithm gives the necessary background for the width optimization discussed in Chapter 4.

2.1.3 Minimum Axis-Aligned Bounding Boxes

A *box* B can be defined as a quadrilateral whose boundary faces (edges) are pairwise parallel or pairwise normal. The minimum axis-aligned bounding box corresponds to the box B with minimum measure M such that P is enclosed by B, i.e., $P \subset B$. For example, M can defined as perimeter, area, volume, or even hypervolume of B. Definition 3 formalizes the minimum axis-aligned bounding box definition.

Definition 3. *The minimum bounding box is the box B which is axis-parallel, encloses all points of P, and minimizes a measure M.*

Similarly to computing the diameter in one dimension, finding an axis-aligned minimum bounding box takes $\mathcal{O}(n)$ time in each dimension, resulting in $\mathcal{O}(nd)$ total time. For $d = 1$, observe that a box B is reduced to a line segment connecting the extreme points of P. That is, Definition 3 can also be reduced to computing the diameter of P in one dimension.

For the two-dimensional case, denote the area and the perimeter of B, respectively, by $\mathsf{A}(B)$ and $\mathsf{P}(B)$. Since the area and perimeter functions are expressed in terms of two independent variables, the length and height of the box, optimizing the area and perimeter can be considered as different objective functions. In this case, when $d = 2$, the minimum bounding box optimization can have the following objective functions:

(3A) Area: $\mathsf{A}_B(P) = \min_{P \subset B} \mathsf{A}(B)$

(3B) Perimeter: $\mathsf{P}_B(P) = \min_{P \subset B} \mathsf{P}(B)$

Suppose that one searches for a subset of P of fixed size that optimizes a measure M. Different subsets of fixed cardinality k of P can be the solution for optimizing different measures M. In two dimensions, Figure 2.3 shows that optimizing a bounding box of k out of the n points of P have different results for area and perimeter. In this case, when $k = 4$ the blue rectangle has optimal area while the green rectangle has optimal area.

Considering the case of $k = n$ and d dimensions, an optimal bounding box can be found by maintaining the points with minimum and maximum coordinates in each axis of the d dimensions, resulting in $\mathcal{O}(nd)$ time. Although this is an efficient algorithm, Chapter 4 refers to a brute-force method for $d = 2$ which will be described as follows.

In two dimensions, a brute-force method to compute the minimum bounding box is to find the minimum among all possible quadrilaterals formed by points of P such that the quadrilateral encloses k elements of P. Since a quadrilateral can be formed by two, three, or four points, there are $\mathcal{O}(n^4)$ possible quadrilaterals formed from points of P, given that P has n points. Given a quadrilateral $\mathcal{L}$, it takes constant

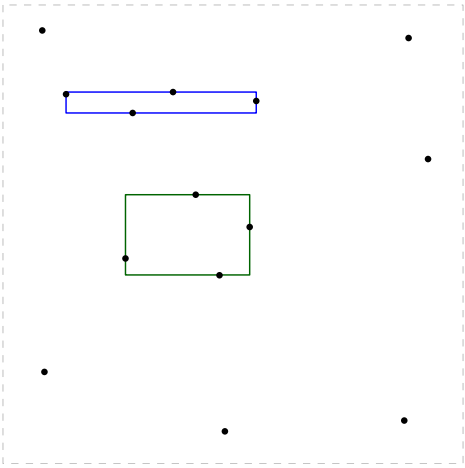


Figure 2.3: Example of minimum axis-aligned bounding box when selecting $k = 4$ points of P, minimized by area in blue and minimized by perimeter in green. The blue rectangle has area 1.47 and perimeter 8.16 while the green rectangle has area 3.72 and perimeter 7.90.

time to determine if $p \in P$ is inside $\mathcal{L}$. Hence, for each combination of two, three, and four points, it takes $\mathcal{O}(n)$ time to determine if k points of P are enclosed by the corresponding quadrilateral $\mathcal{L}$. Given that computing area or perimeter takes constant time, the brute-force solution takes $\mathcal{O}(n^5)$ time.

2.1.4 Minimum Enclosing Ball

Let d_m denote the m-Minkowski distance, also known as Chebyshev distance. That is, given $m \in \mathbb{R}^+$, for $u, v \in \mathbb{R}^d$, $\mathrm{d}_m(u, v) = \left(\sum_{i=1}^{d} |u_i - v_i|^m\right)^{1/m}$. A *ball* can be defined similarly to the *box* by replacing its distance metric for the 2-Minkowski distance, i.e., the Euclidean distance. For a center $c \in \mathbb{R}^d$ and a radius $r \in \mathbb{R}$, the ball is the region composed by $C_c(r) = \{p \in \mathbb{R}^d \mid \mathrm{d}_m(p, c) \leq r\}$. An enclosing ball is represented by a ball $C_c(r)$ where all points of P are in the closed interior of $C_c(r)$.

Definition 4. *The minimum enclosing ball is the ball $C_c(r)$ with minimum radius r*

so that $C_c(r)$ encloses P.

When $d = 1$, a ball is a closed interval. The minimum enclosing ball in one dimension is reduced to computing the one-dimensional diameter of P. In two dimensions, the ball $C_c(r)$ becomes a disk. Since both the area and the circumference of the disk are dependent only on a single variable, the radius, optimizing area or circumference of the enclosing disk would be analogous to optimizing the radius r. Hence, let $\mathsf{A}(C_c(r))$ denote the area of disk $C_c(r)$, Definition 4 can be focused on optimizing the disk area:

$$\min_{P \subset C_c(r)} \mathsf{A}(C_c(r))$$

and Figure 2.4 gives an illustration for the two-dimensional minimum enclosing disk.

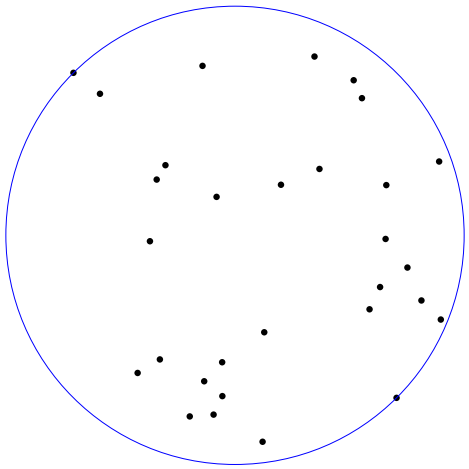


Figure 2.4: Example of minimum enclosing disk from same set of points in Figure 2.3.

A well-known method for obtaining minimum enclosing disks is via a randomized algorithm. Welzl [2005] proposed an exact randomized incremental algorithm. This approach starts with an empty disk and a random center chosen from the points of

P, then recursively adds the points of P one at a time. The stopping condition is an observation of the points composing the enclosing disk: let $u \in P$ be the initial center, if $C_c(r)$ contains all points of $P - \{u\}$, then $C_c(r)$ is the minimum enclosing disk for P; otherwise, u must be on the boundary of the resulting minimum enclosing disk. This approach takes expected $\mathcal{O}(n)$ time. Megiddo [1983] proposes a deterministic algorithm which takes $\mathcal{O}(n)$ time to find the minimum enclosing disk in two dimensions.

2.1.5 Arrangements

Two-dimensional arrangements in geometry refer to a collection of either lines or curves inducing a space subdivision and the related problems of intersections, zones between lines or curves, and envelopes containing the upper or lower elements, for example, as discussed by Halperin and Sharir [2017]. Let L denote a set of lines or x-monotone Jordan curves that induce a space subdivision, denote by $\mathcal{A}(L)$ the *arrangement* of L.

The collection L can be composed of any of the following: circles, parabolas, or curves, in general. Since $\mathcal{A}(L)$ is a space subdivision, the arrangement has an associated graph embedded in the plane, with a vertex at each point of intersection, and an edge corresponding to each segment of line or curve. The topology of the arrangement will be relevant for the solutions presented in Chapters 3 and 4. Faces, which are the regions enclosed by edges, and their complexities are also of importance for arrangements; further discussion can be found in de Berg et al. [2008]. An illustration for arrangements of lines and parabolas is provided in Figure 2.5.

Let $\mathcal{A} = \{f_1, \ldots, f_n\}$ be an arrangement of n hyperplanes such that each f_i is a

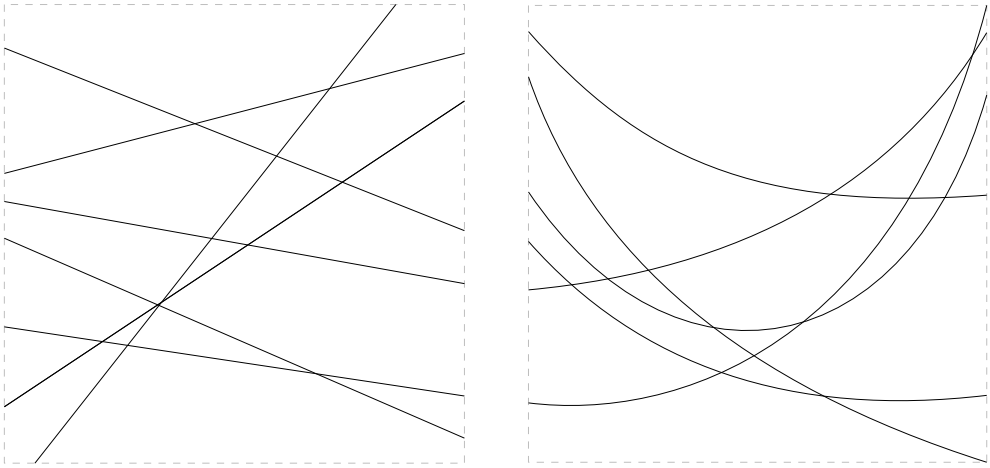


Figure 2.5: Examples of arrangements of lines (left) and parabolas (right).

real-valued, continuous, defined function representing the elements of the arrangement, where each pair of functions intersects at most in s points. The lower envelope $g_{\mathcal{A}}$ of $\mathcal{A}$ is defined as

$$g_{\mathcal{A}}(x) = \min_{1 \leq i \leq n} f_i(x)$$

and, analogously, the upper envelope $f_{\mathcal{A}}$ of $\mathcal{A}$ can be written as $f_{\mathcal{A}}(x) = \max_{1 \leq i \leq n} f_i(x)$. Figure 2.6 depicts the lower and upper envelopes of the arrangements presented in Figure 2.5.

For two positive integers n and s, the Davenport-Schinzel sequence for the pair (n, s), denoted $DS(n, s)$-sequence, is formalized in Definition 5 based on the work presented in Sharir and Agarwal [1995].

Definition 5. *A sequence of m integers $(u_m) = (u_1, \ldots, u_m)$ is a (n, s) Davenport-Schinzel sequence, denoted $DS(n, s)$-sequence, if (u_m) satisfies the following:*

(5A) $1 \leq u_i \leq n$ *for each* $i \leq m$*;*

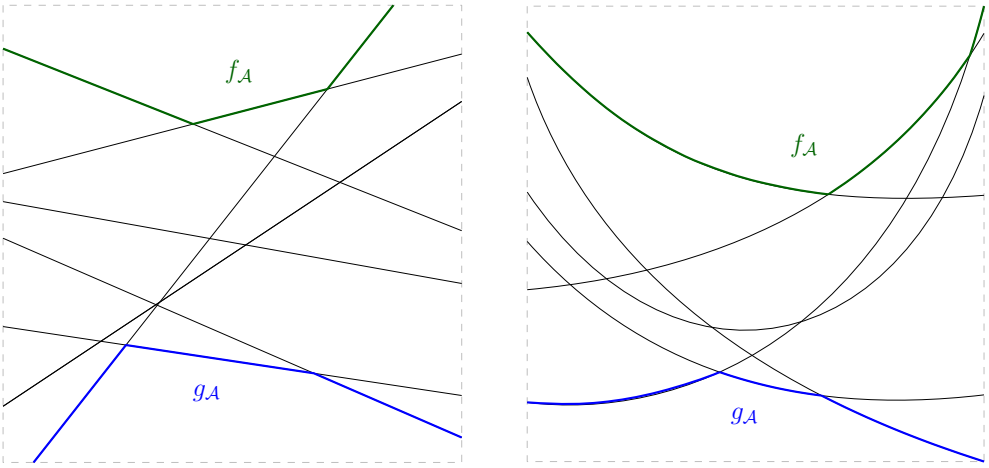


Figure 2.6: The lower and upper envelopes of the arrangements from Figure 2.5.

(5B) $u_i \neq u_{i+1}$ *for each* $i < m$*;*

(5C) there are no $s+2$ *indices* $1 \leq i_1 < i_2 < \cdots < i_{s+2} \leq m$ *such that:*

$$u_{i_1} = u_{i_3} = u_{i_5} = \cdots = a$$

$$u_{i_2} = u_{i_4} = u_{i_6} = \cdots = b$$

and $a \neq b$*.*

The integer s is the order of (u_m), while n is the number of symbols that make up the sequence (u_m). Denote by $|(u_m)|$ the length of the sequence, which is equal to m. An important measure for levels and envelopes of arrangements is the maximum length of a possible $DS(n, s)$-sequence. This is denoted by $\lambda_s(n)$ and defined as:

$$\lambda_n(s) = \{|(u_m)| \mid (u_m) \text{ is a } DS(n, s)\text{-sequence}\}$$

The computation of lower (upper) envelopes proposed by Sharir and Agarwal [1995]

consists of a divide-and-conquer algorithm. In the divide phase, the arrangement $\mathcal{A} = \{f_1, \ldots, f_n\}$ is split as evenly as possible into two subsets $\mathcal{A}_1$ and $\mathcal{A}_2$ and their lower envelopes $g_{\mathcal{A}_1}$ and $g_{\mathcal{A}_2}$ computed. It is important to observe that when an arrangement has a unique function, this function is its lower envelope. The conquer phase consists of determining the most lower (upper) between the envelopes of $\mathcal{A}_1$ and $\mathcal{A}_2$. Theorem 1 states the running time of finding envelopes of an arrangement. Also, from Sharir and Agarwal [1995], the maximum length of a possible Davenport-Schinzel sequence associated to $\mathcal{A}$ is $\lambda_s(n) = \mathcal{O}(n)$ for $s = 2$ and $s = 3$ (where s is defined below in Theorem 1).

Theorem 1 (Sharir and Agarwal [1995])**.** *Given an arrangement $\mathcal{A}$ of n continuous and totally defined functions such that s is the maximum number of intersection between pairs of functions of elements in $\mathcal{A}$, the lower envelope of $\mathcal{A}$ can be computed in $\mathcal{O}(\lambda_s(n) \log n)$ time, where $\lambda_s(n)$ is the maximum length of a possible Davenport-Schinzel sequence associated to $\mathcal{A}$.*

It is vital for the subsequent discussion to formalize the idea of levels in arrangements. As exposed by Sharir and Agarwal [1995], for a point $p \in \mathcal{A}(L) = \{f_1, \ldots, f_n\}$ the level of p is the number of curves below p including p's curve. That is, consider a vertical ray r oriented downwards from p; the level of p is the number of intersections of r and the elements of $\mathcal{A}(L)$. Consider the edge $e = \{p_i, p_j\}$ in the arrangement $\mathcal{A}(L)$, if the points connecting p_i to p_j are of level k, then e is said to have level k. For some $k \leq n$, the k-level of $\mathcal{A}$ is formalized in Definition 6. Figure 2.7 shows the 3-level of the arrangements in Figure 2.5.

Definition 6. *The k-level of $\mathcal{A}$ is the set of edges in $\mathcal{A}$ whose level is k.*

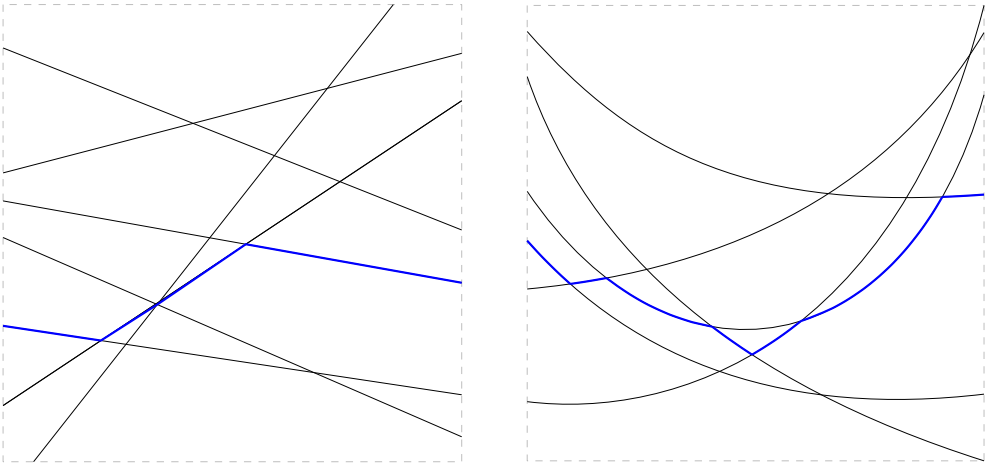


Figure 2.7: Highlighted in blue is the 3-level for the arrangements originally presented in Figure 2.5.

The algorithms and combinatorial properties of k-levels have been widely discussed in the Computational Geometry literature. More recently, there have been developments in k-levels for pseudocircles arrangements, as in Chiu et al. [2019], and also for pseudoplanes, done by Har-Peled et al. [2017] and Sharir and Ziv [2021]. Dey [1997, 1998] proves an upper bound of $\mathcal{O}(nk^{1/3})$ on the number of edges in a k-level of a set of n lines while Tóth [2000] proves a lower bound of $ne^{\Omega(\sqrt{\log k})}$, which are currently the best bounds known at the time this thesis is being written.

In the case of general lines and curves, the three-article series by Chan [2003, 2005, 2008] give multiple combinatorial properties of k-levels. A version of a line sweeping algorithm for constructing a k-level is proposed by Edelsbrunner and Welzl [1986], along with the data structure developed by Chan [1999], the algorithm for line arrangements takes $\mathcal{O}((n+f)\log n)$ time, where f is the k-level complexity (bound on the number of vertices on the k-level). Cole et al. [1984] shows that k-level of Jordan curve arrangements in the plane can be found in $\mathcal{O}(n\log n + f\log^2 k)$ time. For the

specific case of parabolas, using the work of Tamaki and Tokuyama [1995] on cutting parabolas into segments, Chan [2008] shows that the k-level of parabola arrangements has $\mathcal{O}(n^{3/2} \log n)$ edges. In the case of parabolas and parabolas segments in $\mathbb{R}^2$, their arrangements can be constructed in $\mathcal{O}(n^2)$ as stated by Halperin and Sharir [2017].

Finding lower or upper envelopes and k-levels are among the techniques used to propose solutions for the optimization problems discussed in the next two chapters, Chapter 3 and 4.

2.2 Problem Definition

In the case of geometric problems involving motion, the literature uses the terminology of *moving points* to determine a set of objects moving on a certain in space. In this thesis, the term *moving points* will be replaced by *moving objects*, considering that points will have fixed position in the space, while a moving object changes its position through time. The position of a moving object is represented by a continuous function $p : [0, 1] \to \mathbb{R}^d$. To bound the number of combinatorial events that must be considered, the functions are restricted to be polynomials.

A set $S = \{p_1(t), \ldots, p_n(t)\}$ of moving objects is composed of continuous and polynomial functions $p_i : [0, 1] \to \mathbb{R}^d$. For an object $p_i \in S$ and a time t, the function $p_i(t)$ will return the coordinates in the Euclidean space of p_i at the specified time t. Sets of moving objects can also have extent and proximity measures calculated for each time $t \in [0, 1]$. In this thesis, the cases of $d = 1$ and $d = 2$ are examined.

The notion of points in general position is important to avoid degeneracies in $\mathbb{R}^d$. Before defining general position, it is important to introduce the definition of a

d-dimensional flat. A set A is said to be affine if for any pair of points u and v in A, the line $\overline{uv}$ passing through u and v lies in the set A. In geometry, a *d-dimensional flat* refers to a subset of an affine set in which the subset itself is also an affine set in $\mathbb{R}^d$, i.e., let $A \subset \mathbb{R}^d$ be an affine set, if a subset $B \subset A$ is also affine, then B is said to be a dimensional flat of A. For example, lines are two-dimensional flats of the Euclidean plane $\mathbb{R}^2$.

A set of points P in $\mathbb{R}^d$ is said to be in general position if no s points of P lie on a $(s-2)$-dimensional flat and no $d+2$ points lie on a d-sphere. That is, when $d = 2$, P is in general position if no three points of P are collinear and no four points of P are cocircular. In the case of a set S of moving objects in $\mathbb{R}^d$, consider the arrangement of line segments or curves $\mathcal{A}(S)$. Then, if the points of intersection in $\mathcal{A}(S)$ are in general position and no three elements in $\mathcal{A}(S)$ intersect in a common point, S is said to be in general position.

Definition 7. *The set of moving objects S is said to be in general position if the set containting the points of intersections in the corresponding arrangement $\mathcal{A}(S)$ is in general position.*

A natural and valid question is whether this optimization of geometric measures of moving spanning trees has been extended to other metrics. Specifically, the main question is whether optimizing extent and proximity measure on sets and subsets of moving objects has been considered in any previous research. Common examples of geometric measures include diameter, width, area and perimeter of smallest axis-parallel enclosing box, and area of the minimum enclosing disk. Any of those can be selected as a measure of a given set of geometric objects, e.g., of a a set of points

in $\mathbb{R}^d$. Throughout this thesis, the term *measure* is used to refer to diameter, width, area and perimeter of the smallest axis-aligned enclosing box, and area of the smallest enclosing disk. Each of these measures is formally defined in Section 2.1.

Chan [2004] devised an efficient algorithm to find the optimal diameter of a set of moving objects. The work proposed by Akitaya et al. [2021] consider optimization of Moving Spanning Tree measures over the time domain. Another possibility of optimization is to consider selecting subsets of S of smaller fixed cardinality k such that a measure M is extremal. That is, let $\mathcal{Q}_k$ denote the collection of subsets of S with fixed cardinality k, find $Q \in \mathcal{Q}_k$ such that measure M is extremal among the elements of $\mathcal{Q}_k$. Then, consider combining the two optimizations and finding $Q \in \mathcal{Q}_k$ such that M is extremal over all elements in $\mathcal{Q}_k$ and also over the time domain $[0, 1]$. To the author's best knowledge, the questions when $k < n$ have not been answered before in the literature. For example, a solution of questions of the type: "at which time t during the motion, any three objects u, v, and w selected from the n objects in S, are u, v, and w closest in terms of set diameter?"

Hence, given a measure $\mathsf{M}(S, t)$ calculated over the set of moving objects $S \subset \mathbb{R}^d \times [0, 1]$ and time $t \in [0, 1]$, this measure may be optimized over the time domain and also over all the possible subsets $Q \subset S$ of size $|Q| = k \leq n$. That is, each of the measures previously defined can be optimized along the motion of the objects in S. Problem 1 summarizes the optimizations to be solved in the present thesis.

Problem 1. *The following are the optimization of measure* M *solved in this thesis:*

(1A) Min-min: $\min_{Q \in \mathcal{Q}_k} \min_{t \in [0,1]} \mathsf{M}(Q,t)$

(1B) Max-min: $\max_{Q \in \mathcal{Q}_k} \min_{t \in [0,1]} \mathsf{M}(Q,t)$

(1C) Min-max: $\min_{Q \in \mathcal{Q}_k} \max_{t \in [0,1]} \mathsf{M}(Q,t)$

(1D) Max-max: $\max_{Q \in \mathcal{Q}_k} \max_{t \in [0,1]} \mathsf{M}(Q,t)$

For each $p_i(t) \in S$, the function $p_i(t)$ is restricted to be a polynomial of degree one. When $d = 1$, the diameter is equivalent to set width, minimum bounding box, and minimum enclosing circle. Hence, a single algorithm suffices for the optimization of each of the four measures. The algorithm is presented in Chapter 3.

When $d = 2$, each measure has its own significance and importance as previously discussed. Then, Chapter 4 proposes solutions for each variation of the optimization in Problem 1 where $\mathsf{M}(Q,t)$ is one of set diameter, set width, minimum axis-aligned bounding box, or minimum enclosing disk.

2.3 Previous Work on Optimization Problems for Sets of Moving Objects

An important initial discussion is to distinguish off-line and on-line algorithms. An algorithm is said to be off-line when it has complete input available at the beginning of execution. Meanwhile, on-line algorithms can make decisions with incomplete data or produce results as data is fed into the algorithm, as explained by Cormen et al. [2022].

A Kinetic Data Structure (KDS) is a framework for indexing moving objects that computes a set of certificate validations and remediations as each object moves. A KDS allows object trajectories to be modified during exection; hence, a KDS is

considered an on-line algorithm.

The problem of kinetic width and kinetic bounding box in two dimensions is discussed by Guibas et al. [2001] and Meulemans et al. [2019], with applications in robot motion panning. The two-dimensional smallest enclosing disk has been discussed in Agarwal et al. [2001]. Many other kinetic geometric problems have been discussed and solved in a kinetic fashion as shown by Guibas [1998]. All these solutions use a kinetic data structure framework to solve the problem continuously, that is, they perform an on-line solution.

Since the problems from Guibas [1998] also assume the movement of the points can be approximated by polynomial functions, it is possible to develop an off-line solution of optimization of geometric measures over the time domain using the optimal solution achieved throughout the motion, as computed using a KDS. Doing so tends to require more time than would be required an off-line algorithm, since a KDS computes and maintains a given measure throughout the entire motion, as opposed to focusing only on the extreme values.

Gupta et al. [1996] proposed an $\mathcal{O}(n \log^3 n)$-time solution to find the minimum diameter of sets that contain n objects in motion. One year later, Clarkson [1997] proposes a randomized solution to the minimum diameter of n objects in motion which runs in $\mathcal{O}(n \log n)$ expected time. Then, Chan [2004] shows that the minimum diameter of objects in motion can be found deterministically in $\mathcal{O}(n \log n)$ time. Banik et al. [2014] gives combinatorial properties of the minimum enclosing of a set of n points when one is moving with constant velocity. More recently, the work proposed by Akitaya et al. [2021] is based on an off-line solution of the minimum moving spanning

tree for objects in motion.

The work done by Rahmati and Zarei [2012] uses a kinetic data structure framework to solve kinetic minimum spanning tree problems. In the performance analysis section from Rahmati and Zarei [2012], it is mentioned that the framework needs to compute $\mathcal{O}(n^4)$ events and each event takes $\mathcal{O}(\log^2 n)$ time to be certified. Hence, finding a minimization of all possible spanning trees along the whole motion using the framework would have an asymptotic bound estimated of $\mathcal{O}(n^4 \log^2 n)$ time.

In contrast, Akitaya et al. [2021] define how to build an algorithm which takes $\mathcal{O}(n \log n)$ time for a $(2+\epsilon)-$approximation algorithm to solve the minimization of the minimum spanning tree in motion. Although, the on-line algorithm has to process all certificates through the whole motion, this exemplifies a remarkable gain in time complexity for the optimization of proximity measures. Based on this model, Wachholz and Suri [2023] proves that the problems of MST, Graph Matching, and Travelling Salesman Person are NP-Hard for moving objects, even in one-dimensional Euclidean space and linear unit speed motion.

Therefore, a course of research would be to find optimization of geometric measures for points in movement S, and also subsets $Q \subset S$, based on an off-line approach rather than building a kinetic data structure framework for the whole motion of the elements in S. Moreover, the variations of maximization and minimization over all possible subsets Q of S and over the time domain are also considered.

Chapter 3

One-Dimensional Problems

As discussed in Chapter 2, diameter, width, minimum bounding box, and minimum enclosing disk are equivalent in the one-dimensional case. Let S denote a set of n objects moving with constant velocity. For each $i = 1, \ldots, n$, the element $p_i(t) = a_i t + b_i$ of S denotes the linear (constant-velocity) motion of p_i in $\mathbb{R}$. Each line can be plotted over the duration of the motion $[0, 1]$, resulting in an arrangement of lines in the plane. Figure 3.1 shows an example of objects moving on the line and its corresponding line arrangement, where the x-axis denotes the time interval [0, 1].

The distance between two points in S at time t can be interpreted as the vertical distance between the corresponding lines at the x-coordinate $x = t$ in the arrangement of lines $\mathcal{A}(S) = \{p_1(t), p_2(t), \ldots, p_n(t)\}$. That is, for two lines $p_i, p_j \in \mathcal{A}(S)$, we can observe that $p_i(t) - p_j(t)$ represents the vertical line segments between p_i and p_j at time t. In this case, the minimum width of the set S is realized by the minimum difference between the lower and upper envelopes of $\mathcal{A}(S)$.

Let $\ell_S(t)$ be the minimum vertical line segment stabbing all lines of $\mathcal{A}(S)$ at time t.

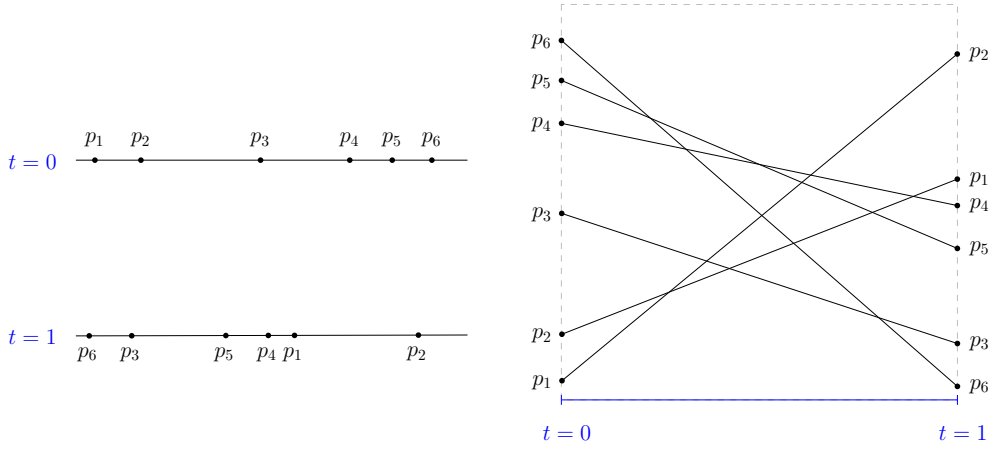


Figure 3.1: Example of a set S of objects moving on the line and its corresponding line arrangement on the right.

Alternatively, the diameter optimization of S can be defined as finding the minimum vertical stabbing line segment ℓ_S over the time domain. The geometric interpretation of vertical stabbing line segment $\ell_S(t)$ as the diameter of S at time t can be used to compute the Min-Min, Max-Min, Min-Max, and Max-Max optimizations of the diameter in one dimension.

3.1 Min-Min

Recall that $\mathcal{Q}_k$ denotes the subsets of S with fixed cardinality k. This section focuses on the one-dimensional case of the optimization $\min_{Q \in \mathcal{Q}_k} \min_{t \in [0,1]} \mathsf{D}(Q, t)$. For the case of $k = n$, Chan [2004] gives an efficient deterministic solution in $\mathcal{O}(n \log n)$ time. The case for an arbitrary cardinality k such that $k < n$ is discussed as follows.

Let $T \subseteq \mathbb{R}^2$ be the set of points containing the positions of the objects in S at times $t = 0$ and $t = 1$, i.e., T contains $p_i(0)$ and $p_i(1)$ for each $p_i \in S$. Also, let T contain the intersection points between any pair of lines in $\mathcal{A}(S)$. An important initial observation

is that the one-dimensional diameter has local extrema at the times of occurrence of the elements in T, as seen in Lemma 2. Moreover, if the Min-Min solution is realized by a vertical stabbing line segment $\ell_S = \overline{pq}$ in $\mathcal{A}(S)$ at time $t \in (0, 1)$, one of its endpoints is an intersection point of $\mathcal{A}(S)$, i.e., either p or q is an intersection in $\mathcal{A}(S)$. That is, the optimization occurs at the boundary of the time interval or at the discrete events realized by the intersection points of $\mathcal{A}(S)$.

Denote by $f_S(t)$ and $g_S(t)$ the lower and upper envelope of $\mathcal{A}(S)$, respectively, at time t. In case the lower envelope of $\mathcal{A}(S)$ over $[0, 1]$ consists of a single line, and similarly for the upper envelope of $\mathcal{A}(S)$, the minimum vertical stabbing line segment $\ell_S(t)$ has its extrema at $t = 0$ or $t = 1$. Let $|\ell_S(t)|$ denote the length of the vertical stabbing line segment ℓ_S at time t. Observe that $|\ell_S(t)|$ is the subtraction of two elements from S, which means that $|\ell_S(t)|$ is a linear and continuous function whose local extrema occur at the boundary of the time domain. Otherwise, the local extrema of the minimum vertical stabbing line segment can occur at the same time as any intersection of $\mathcal{A}(S)$. Lemmas 1 and 2 formalize the idea.

Lemma 1. *In the one-dimensional case, if the lower and upper envelopes of $\mathcal{A}(S)$ are formed each by a single line over $[0, 1]$, then the minimum vertical stabbing line $\ell_S(t)$ of $\mathcal{A}(S)$ is realized at either the initial or final time, $t = 0$ or $t = 1$.*

Lemma 2. *In the one-dimensional case, the extreme values of the diameter of a set of objects S moving in a linear trajectory and with constant velocity will happen at the boundary of the time domain or at the time of intersections of the arrangement $\mathcal{A}(S)$.*

Proof. Let the lower and upper envelopes of $\mathcal{A}(S)$, respectively, be f_S and g_S. By definition, both f_S and g_S are piecewise-linear continuous functions such that f_S

is a convex function and g_S is a concave function. Observe that $-g_S$ is a convex function and the vertical stabbing line segments ℓ_S between envelopes of S is in the form $\ell_S(t) = f_S(t) - g_S(t)$. Hence, $\ell_S(t)$ is a convex piecewise-linear function and its maximum happens at the boundary of time. The intersection points of f_S and g_S are points such that the derivatives $\frac{\mathrm{d}f_S}{\mathrm{d}t}$ and $\frac{\mathrm{d}g_S}{\mathrm{d}t}$ are undefined, i.e., these intersections are local extrema of ℓ_S. Therefore, the local extrema of ℓ_S occur at the boundary of the time domain or at the time of intersections in $\mathcal{A}(S)$. □

Recall that Min-Min optimization is written in the form $\min_{Q \in \mathcal{Q}_k} \min_{t \in [0,1]} \mathsf{D}(Q, t)$. Also, from the geometric interpretation of diameter as vertical stabbing line segment, that is, $\ell_S(t) = \mathsf{D}(S, t)$, the Min-Min optimization can be alternatively defined as finding the minimum over all minimum vertical stabbing line segments formed from sets in $\mathcal{Q}_k$. That is, for $Q \in \mathcal{Q}_k$, let ℓ_Q denote the minimum vertical stabbing line segment of $\mathcal{A}(Q)$ over $t \in [0, 1]$, the Min-Min problem is equivalent to computing the length $|\ell_{min}|$ where:

$$|\ell_{min}| = \min_{Q \in \mathcal{Q}_k} |\ell_Q|$$

For the case of $k = 1$, the collection $\mathcal{Q}_1$ is composed of singletons. Since there are no intersections when a single object is in movement, $\mathsf{D}(Q, t) = 0$ for any $Q \in \mathcal{Q}_1$ and $t \in [0, 1]$. For $k = 2$, there are two points in movement in each element of $\mathcal{Q}_2$, then two cases are possible for each $Q \in \mathcal{Q}_2$, the lines in $\mathcal{A}(Q)$ will intersect or not. In case the lines of $\mathcal{A}(Q)$ intersect at time t_k then we can observe that $\mathsf{D}(Q, t_k) = 0$ is the minimum width attained by the points in Q. Now, consider that the lines in $\mathcal{A}(Q)$ do not intersect. From Lemma 1, the minimum diameter of Q over time occurs at the

boundary of the time domain, that is, $\min\{\mathsf{D}(Q,0), \mathsf{D}(Q,1)\}$.

When $3 \leq k \leq n$, the solution can also be divided into two significant cases: $k = n$ and $3 \leq k < n$. When $k = n$, computing the minimum distance between the upper and lower envelopes of $\mathcal{A}(S)$ is a solution to $\min_{t\in[0,1]} \mathsf{D}(S,t)$. As explained by de Berg et al. [2008] finding lower or upper envelopes can be done in $\mathcal{O}(n \log n)$ time, this approach will also take $\mathcal{O}(n \log n)$ time.

In the case of $3 \leq k < n$, a strategy can be defined from Lemma 2. For a fixed k and for each of $Q \in \mathcal{Q}_k$, one of the endpoints of ℓ_Q is an intersection point q of two lines in $\mathcal{A}(S)$, the arrangement of all lines composed of elements of S. Since $Q \subseteq S$, we have that $\mathcal{A}(Q) \subseteq \mathcal{A}(S)$ meaning that the intersection q in $\mathcal{A}(Q)$ is also an intersection in $\mathcal{A}(S)$. Since q is an intersection points of two lines in $\mathcal{A}(S)$, then a local extrema of diameter, geometrically given as the minimum vertical stabbing line segment ℓ_Q, is either the line formed by q and intersects $k-2$ lines vertically above q in $\mathcal{A}(S)$ or intersects $k-2$ lines vertically below q. See Figure 3.2 for an example of $k = 4$.

Let T contain all intersections of $\mathcal{A}(S)$. Include $p_i(0)$ and $p_i(1)$ in T for each $p_i \in S$. Then, for each $q \in T$, compute $|\ell_q^+|$ the length of the vertical line segment starting at q and intersecting $k-2$ lines above q. Analogously, compute $|\ell_q^-|$ the length of the line segments starting at q and intersecting $k-2$ lines below q. From all the lengths computed, the solution is given by: $\min_{q\in T} \min\{|\ell_q^+|, |\ell_q^-|\}$. This procedure is formalized in Algorithms 3.1 and 3.2 and an illustration is given in Figure 3.2.

Line 1 in Algorithm 3.1 refers to the construction of the line arrangement $\mathcal{A}(S)$. de Berg et al. [2008] explains that an arrangement of n lines can be constructed

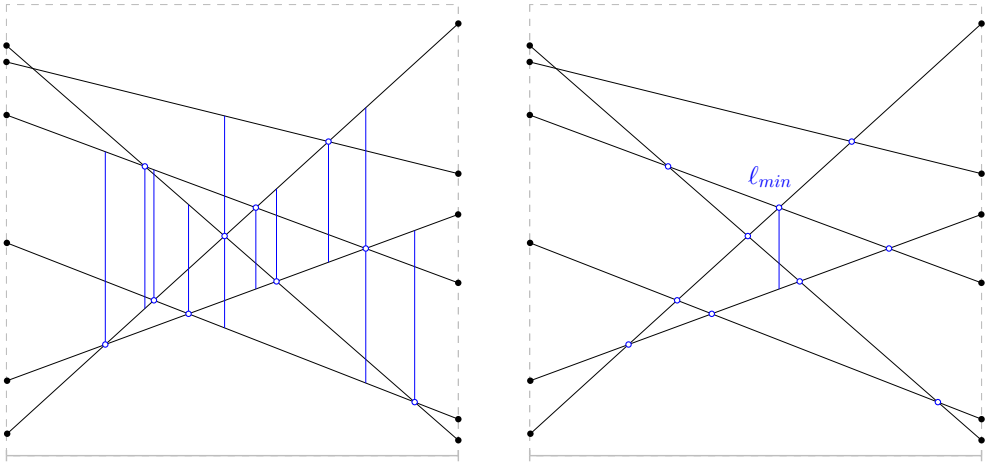


Figure 3.2: From a set S of moving objects on the left, the minimum vertical stabbing line $\ell_{\min}$ for $k = 4$ is displayed on the right.

MIN-MIN-1D(S, k)

1 Compute the line arrangement $\mathcal{A}(S)$
  // T has pairs (q, t) of intersection point q at time t.
2 Let T contain the intersections in $\mathcal{A}(S)$ and time of occurrence, i.e., (p, t).
3 Add $(p_i(0), 0)$ and $(p_i(1), 1)$ to T for each $p_i \in S$
4 $|\ell_{\min}| = \infty$
5 **for** each $(q, t) \in T$
6 $\ell_q =$ MIN-VERTICAL-LINE-ENDPOINT$(\mathcal{A}(S), q, k, 2)$
7 $|\ell_{\min}| = \min\{|\ell_{\min}|, |\ell_q|\}$
8 **return** $|\ell_{\min}|$

Algorithm 3.1: Solution to Min-Min optimization for Diameter in one dimension.

incrementally in $\mathcal{O}(n^2)$ time via a doubly linked list of edges. In general, Mulmuley [1993] proposes a randomized incremental algorithm for constructing arrangements of n Jordan curves in $\mathcal{O}(n \log n + s)$ time, where s is the number of intersections of the arrangement. In the case of arrangement of lines, $s = \mathcal{O}(n^2)$, so the randomized algorithm would also construct the arrangement in $\mathcal{O}(n^2)$ time.

Theorem 2 (Mulmuley [1993])**.** *Given a set L of n Jordan curves, the curve arrange-*

MIN-VERTICAL-LINE-ENDPOINT($\mathcal{A}(S), q, k, j$)
// Integer j is 1 is q has no intersection at time t, or 2 otherwise.
1 ℓ_q^+ = vertical line starting at q intersecting $k - j$ lines of $\mathcal{A}(S)$ above q.
2 ℓ_q^- = vertical line starting at q intersecting $k - j$ lines of $\mathcal{A}(S)$ below q.
3 **return** $\min\{|\ell_q^+|, |\ell_q^-|\}$

Algorithm 3.2: Find minimum vertical segment stabbing k lines of the given line arrangement at time t.

ment $\mathcal{A}(L)$ can be constructed in $\mathcal{O}(n \log n + s)$ expected time and $\mathcal{O}(n + s)$ expected space via a randomized incremental algorithm, where s is the number of intersections in $\mathcal{A}(L)$.

Using the line sweep algorithm, the intersections computed in line 2 of Algorithm 3.1 can be computed in $\mathcal{O}(n^2)$ time as proposed by de Berg et al. [2008]. Given that L is composed of lines, there are $\mathcal{O}(n^2)$ intersections in $\mathcal{A}(L)$. Since the line sweep algorithm maintains the order of the objects with respect to their position on the line, at each intersection the order of the objects can be updated in $\mathcal{O}(1)$ time (swap the intersecting objects at time t on the list). Hence, finding the k-th distant object from q can be done in constant time from the ordered list of objects. Observe that Algorithm 3.2 can be done in constant time, as the arrangement maintains the lines ordered along the vertical axis. Therefore, the loop in lines 4 to 7 will run in $\mathcal{O}(n^2)$ time and Algorithm 3.1 takes $\mathcal{O}(n^2)$ time.

Theorem 3. *Let S be the set of n objects moving with constant velocity on the real line. Also, let $\mathcal{Q}_k$ denote the collection of subsets of S with fixed cardinality k. The Min-Min optimization of the one-dimensional diameter, i.e., $\min_{Q \in \mathcal{Q}_k} \min_{t \in [0,1]} \mathsf{D}(Q, t)$, can be solved in $\mathcal{O}(n^2)$ time.*

3.2 Max-Min

An algorithm is presented for Max-Min optimization of diameter by applying another observation of the distance between the upper and lower envelopes of the line arrangement associated to the motion of the objects in S. For $Q \in \mathcal{Q}_k$, recall that ℓ_Q is the vertical line segment from the upper to the lower envelope of $\mathcal{A}(Q)$, and $|\ell_Q|$ is the length of the corresponding segment. The Problem (1B), which is the Max-Min optimization for diameter can also have the geometric interpretation of finding the maximum length over all the minimum vertical stabbing line segments of the sets in the collection $\mathcal{Q}_k$, which is $|\ell_{max}| = \max_{Q \in \mathcal{Q}_k} |\ell_Q|$.

For any $Q \in \mathcal{Q}_k$, $\mathcal{A}(Q) \subseteq \mathcal{A}(S)$, then $\mathcal{A}(Q)$ is a subset of the lines in $\mathcal{A}(S)$, or $\mathcal{A}(Q)$ can be formed by removing lines from $\mathcal{A}(S)$. Let ℓ_S be the minimum vertical stabbing line segment of $\mathcal{A}(S)$. Since $\mathcal{A}(Q)$ for $Q \in \mathcal{Q}_k$ can be formed by removing lines from $\mathcal{A}(S)$, then the length $|\ell_Q|$ is either the same as $|\ell_S|$ or $|\ell_Q|$ is smaller than $|\ell_S|$, i.e., $|\ell_Q| \leq |\ell_S|$. The idea is formalized in Observation 1 and its proof comes from the intuitive idea discussed above.

Observation 1. *Let $\mathcal{A}$ be an arrangement of n lines in the plane. Let ℓ denote the corresponding minimum vertical stabbing line segment of $\mathcal{A}$. If $\mathcal{A}'$ is a line arrangement such that $\mathcal{A}' \subseteq \mathcal{A}$, then the minimum vertical stabbing line segment ℓ' of $\mathcal{A}'$ is the same as ℓ or ℓ' is smaller than ℓ in length.*

Theorem 4. *The minimum vertical stabbing line segment ℓ_S of $\mathcal{A}(S)$, from the set S, is the maximum of all minimum vertical stabbing line segments of $\mathcal{A}(Q)$ for all $Q \in \mathcal{Q}_k$ and all integers k such that $3 \leq k \leq n$. That is, $\max_{Q \in \mathcal{Q}_k} \min_{t \in [0,1]} \mathsf{D}(Q,t)$ is equivalent to*

$\min_{t\in[0,1]} \mathsf{D}(S,t)$.

Proof. Suppose that the diameter minimization $\min_{t\in[0,1]} \mathsf{D}(S,t)$ is realized by the elements $p, q \in S$ at time t'. For $Q \in \mathcal{Q}_k$, the subset Q can be formed by removing $n-k$ elements from S. Hence, from Observation 1, the minimum diameter of Q is the same or smaller than the minimum diameter of S. That is:

$$\min_{t\in[0,1]} \mathsf{D}(Q,t) \leq \min_{t\in[0,1]} \mathsf{D}(S,t) = \mathrm{d}(p,q,t')$$

In this case, the maximum of $\min_{t\in[0,1]} \mathsf{D}(Q,t)$ for any $Q \in \mathcal{Q}_k$ is realized by the points p and q at time t'. For every k such that $3 \leq k \leq n$, there is at least one $Q' \in \mathcal{Q}_k$ such that $\{p,q\} \subset Q'$. Therefore:

$$\max_{Q\in\mathcal{Q}_k} \min_{t\in[0,1]} \mathsf{D}(Q,t) = \mathsf{D}(Q',t') = \mathrm{d}(p,q,t') = \min_{t\in[0,1]} \mathsf{D}(S,t)$$

□

In the one-dimensional case, the lines of the arrangement $\mathcal{A}(S)$ can intersect at one point. As in the proof of Theorem 4, suppose the minimization of diameter is realized by $p, q \in S$. From Lemma 2, if the minimization of diameter occurs at the interior of the time domain $[0,1]$, then either p or q intersect with another element of S. The minimum diameter of a set of moving objects in one dimension may be realized by three points, as showed in Figure 3.3. Hence, the theorem above is only valid for $k \geq 3$.

For the special case of $k = 2$, the diameter $\mathsf{D}(Q,t)$ for any $Q \in \mathcal{Q}_2$ will be defined as the distance between two elements in S. That is, for $p, q \in Q$, $\mathsf{D}(Q,t) = |p(t) - q(t)|$

which is an absolute value function composed with a polynomial of degree one. In this case, if $\mathsf{D}(Q,t)$ is minimized in the interior of time domain $[0,1]$, then $\mathsf{D}(Q,t) = 0$. Since Max-Min searches for the maximum among the minimum diameters, for $k = 2$ the solution occurs at either $t = 0$ or $t = 1$. That is, the Max-Min diameter optimization is equivalent to $\max_{Q \in \mathcal{Q}_2} \min\{\mathsf{D}(Q,0), \mathsf{D}(Q,1)\}$.

Another observation is that $\max_{Q \in \mathcal{Q}_2} \mathsf{D}(Q,0)$ is equivalent to $\max_{p,q \in S} \mathrm{d}(p,q,0)$, meaning that it is also equivalent to $\mathsf{D}(S,0)$ definition. The equivalence is also true for $\max_{Q \in \mathcal{Q}_2} \mathsf{D}(Q,1)$. Therefore, for $k = 2$, $\max_{Q \in \mathcal{Q}_2} \min\{\mathsf{D}(Q,0), \mathsf{D}(Q,1)\}$ is equivalent to $\min\{\mathsf{D}(S,0), \mathsf{D}(S,1)\}$.

When $3 \leq k \leq n$, the Max-Min problem for diameter is equivalent to finding the minimal vertical stabbing line segment of $\mathcal{A}(S)$, that is:

$$\max_{Q \in \mathcal{Q}_k} \min_{t \in [0,1]} \mathsf{D}(Q,t) = \min_{t \in [0,1]} \mathsf{D}(S,t) = |\ell_S|$$

for every fixed k such that $3 \leq k \leq n$. For $3 \leq k \leq n$, $\min_{t \in [0,1]} \mathsf{D}(S,t)$ can be efficiently solved in $\mathcal{O}(n \log n)$ time via the diameter minimization proposed by Chan [2004]. However, the slower solution presented in Algorithm 3.3 is important for further discussion of two-dimensional problems in Chapter 4.

When $k = 2$, Algorithm 3.3 will execute lines 4 to 6. Since lines 4 and 5 take linear time, then the algorithm will take $\mathcal{O}(n)$ time when $k = 2$. Observe that the line arrangement can be constructed in $\mathcal{O}(n^2)$ time. Also, the envelopes of $\mathcal{A}(S)$ can be computed in $\mathcal{O}(n \log n)$ time. When $3 \leq k \leq n$, the algorithm will take $\mathcal{O}(n^2)$ time. Hence, Algorithm 3.3 can solve the Max-Min optimization in total $\mathcal{O}(n^2)$ time. On the other hand, replacing lines 8 and 9 by the solution of minimum diameter proposed

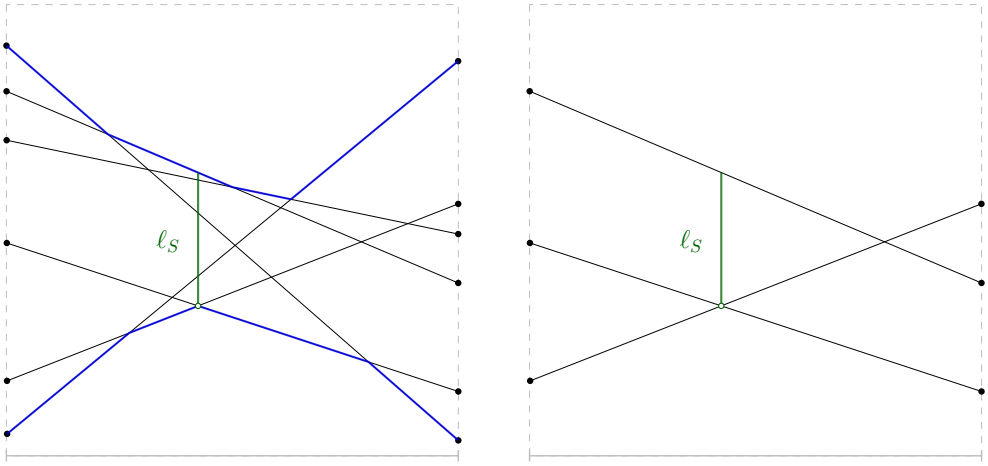


Figure 3.3: Solution of Max-Min from minimum vertical stabbing line segment ℓ_S of S for $3 \leq k \leq n$ and three lines forming ℓ_S.

MAX-MIN-1D-SLOWER(S, k)

```
1  if k ≤ 1
2      return 0
3  if k = 2
4      D_0 = diameter of S at time t = 0
5      D_1 = diameter of S at time t = 1
6      return min{D_0, D_1}
7  else // In the case of 3 ≤ k ≤ n
8      Form the line arrangement A(S) from S
9      |ℓ_S| = Minimum distance between upper and lower envelopes of A(S)
10     return |ℓ_S|
```

Algorithm 3.3: Solution for Max-Min optimization for diameter in one dimension.

by Chan [2004] results in a $\mathcal{O}(n \log n)$-time solution.

Theorem 5. *Given the set S of n moving objects with constant velocity on the real line and $\mathcal{Q}_k$ the collection of subsets of S with fixed cardinality k, the Max-Min optimization of the one-dimensional diameter $\max_{Q \in \mathcal{Q}_k} \min_{t \in [0,1]} \mathsf{D}(Q, t)$ can be solved in $\mathcal{O}(n \log n)$ time.*

3.3 Min-Max and Max-Max

In the solution of Min-Max and Max-Max optimizations for one-dimensional diameter, a common step between the two solutions is to find the maximum of the objective function over the time interval, which is $\max_{t\in[0,1]} \mathsf{D}(Q,t)$. Solving $\max_{t\in[0,1]} \mathsf{D}(Q,t)$ for any $Q \in \mathcal{Q}_k$ and any $k \leq n$ is equivalent to finding two objects whose Euclidean distance is maximum over the time domain. Using the fact that the Euclidean distance is a convex function, the maximum of $\mathsf{D}(Q,t)$ over time will be realized at the boundary of the time domain $[0,1]$. Hence, the solution of such maximization will be in the form:

$$\max_{t\in[0,1]} \mathsf{D}(Q,t) = \max\{\mathsf{D}(Q,0), \mathsf{D}(Q,1)\}$$

Observation 2. *Since the diameter of a set of points on the real line is equivalent to the maximum pairwise distance and the Euclidean distance is a convex function, the following is true.*

(i) The diameter version of Problem (1B), Min-Max, is equivalent to:

$$\min_{Q\in\mathcal{Q}_k} \max\{\mathsf{D}(Q,0), \mathsf{D}(Q,1)\}$$

(ii) The diameter version of Problem (1D), Max-Max, is equivalent to:

$$\max_{Q\in\mathcal{Q}_k} \max\{\mathsf{D}(Q,0), \mathsf{D}(Q,1)\}$$

In this case, the Min-Max diameter optimization can be solved with a similar

idea to the approach done for the Min-Min optimization but now restricted to the initial ($t = 0$) and final time ($t = 1$) only. That is, find the minimum over the vertical stabbing line segments of $\mathcal{A}(S)$ that intersects the k lines at time $t = 0$ and time $t = 1$. Denote by $S(t)$ the array of points that represents the position of objects in S at time t. Since finding the k-th largest element in an array takes $\mathcal{O}(n)$ time using the selection algorithm from Cormen et al. [2022], it is possible to find the k-th most distant object from an object $p \in S$.

The Max-Max problem, based on Observation (ii), can also be interpreted as the maximum between the diameter of $S(0)$ and $S(1)$ since Theorem 8 states that maximization over subsets $Q \in \mathcal{Q}_k$ is realized by the set S. Hence, this solution can be formalized in Algorithm 3.5. Figure 3.4 shows examples of minimum vertical stabbing line segments that are solutions of Min-Max and Max-Max diameter optimization in the one-dimensional case and $k = 4$.

MIN-MAX-1D(S, k)

```
    // Array storing length of vertical stabbing segment at initial and final time.
1   R[0] = ∞
2   R[1] = ∞
3   for t = 0, 1
4       for each p_i ∈ S
5           p_j = k-th distant object from p_i using selection algorithm
6           |ℓ_p| = p_i(t) − p_{i+k−1}(t)
7           R[t] = min{R[t], |ℓ_p|}
8   return max{R[0], R[1]}
```

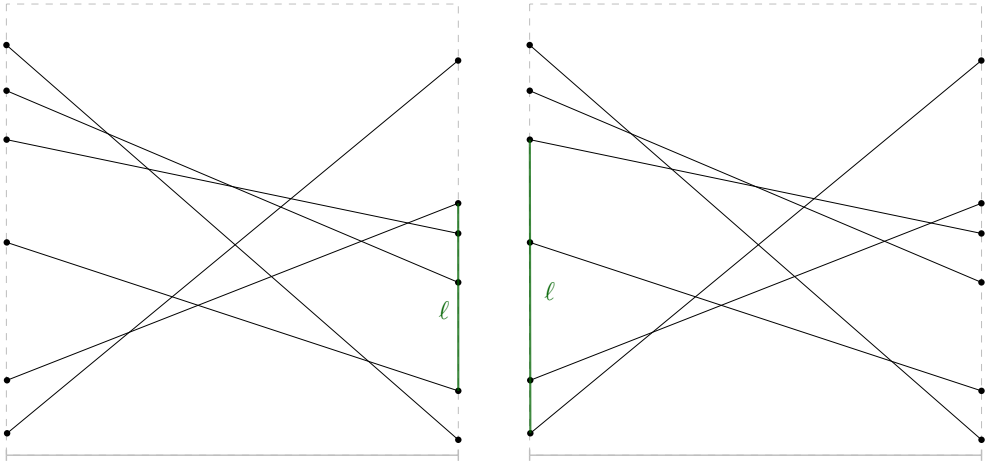
Algorithm 3.4: Solution for Min-Max one dimensional diameter optimization.

Both solutions of Min-Max and Max-Max occur when $t = 0$ or $t = 1$. Min-Max optimization uses the selection algorithm to find the k-th most distant object. Hence, Min-Max in the form of Algorithm 3.4 takes $\mathcal{O}(n^2)$ time. Meanwhile, since computing

MAX-MAX-1D(S, k)
1 $\mathsf{D}_0 =$ diameter of set $S(0)$
2 $\mathsf{D}_1 =$ diameter of set $S(1)$
3 **return** $\max\{\mathsf{D}_0, \mathsf{D}_1\}$

Algorithm 3.5: Max-Max optimization solution for diameter in one dimension.

the diameter of points in one dimension takes $\mathcal{O}(n)$ time, the Max-Max solution in Algorithm 3.5 takes $\mathcal{O}(n)$ time.



(a) Min-Max Solution. (b) Max-Max Solution.

Figure 3.4: Solutions of Min-Max and Max-Max diameter optimization in one dimension for $k = 4$.

Theorem 6. *Let S the set of n objects moving with constant velocity on the real line. Let $\mathcal{Q}_k$ be the collection of subsets of S with fixed cardinality k. The Min-Max and Max-Max optimization of the one-dimensional diameter, respectively,*

$$\min_{Q\in\mathcal{Q}_k} \max_{t\in[0,1]} \mathsf{D}(Q,t) \qquad \textit{and} \qquad \max_{Q\in\mathcal{Q}_k} \max_{t\in[0,1]} \mathsf{D}(Q,t),$$

can be solved in $\mathcal{O}(n^2)$ time and $\mathcal{O}(n)$ time.

Chapter 4

Two-Dimensional Problems

Let S be a set of n constant-velocity objects moving in rectilinear trajectory and let S be in general position in $\mathbb{R}^2$, as defined in Chapter 2. The following sections will present polynomial-time solutions for each of the Min-Min, Max-Min, Min-Max, and Max-Max optimizations of each measure defined in Chapter 2; namely, diameter, width, axis-aligned bounding box area and perimeter, and minimum enclosing disk area.

4.1 Diameter

4.1.1 Min-Min

For an object $p(t) \in S$, $p(t)$ is parameterized in the form of $p(t) = (p_x(t), p_y(t))$ so that $p_x(t)$ and $p_y(t)$ are polynomials of degree one. Let S_x be the projection of S on the x-axis. That is, $S_x = \{p_x(t) \mid p(t) \in S\}$. Also, define S_y to be the projection of S onto the y-axis, i.e., $S_y = \{p_y(t) \mid p(t) \in S\}$. Then, both S_x and S_y are sets of linear

functions that represent the movement of S on each of the x and y-axes.

Given two objects $p(t), q(t) \in S$, the two-dimensional diameter $\mathsf{D}(S,t)$ at time t is defined as

$$\mathsf{D}(S,t) = \max_{p(t),q(t)\in S} \sqrt{(p_x(t) - q_x(t))^2 + (p_y(t) - q_y(t))^2}$$

Similarly, $\mathsf{D}^2(S,t) = [\mathsf{D}(S,t)]^2 = \max_{p(t),q(t)\in S} (p_x(t) - q_x(t))^2 + (p_y(t) - q_y(t))^2$.

Recall that $\mathcal{Q}_k$ denotes the collection of all subsets of S of fixed cardinality k. For $k = 1$, since $\mathcal{Q}_1$ is a set of singletons, the diameter is $D(Q,t) = 0$ for each $Q \in \mathcal{Q}_1$. In the case of $k = 2$, each $\mathsf{D}(Q,t)$ will be the Euclidean distance at time t between two objects in S. Observe that the arrangement of each squared diameter curves $\mathsf{D}^2(Q,t)$ will be composed of parabolas since $(p_x(t) - q_x(t))^2$ and $(p_y(t) - q_y(t))^2$ are polynomials in t of degree two. From Lemma 3, the extreme values of $\mathsf{D}^2(Q,t)$ occur at the same time t as $\mathsf{D}(Q,t)$. The specific case of $k = n$ is efficiently solved by Chan [2004] via a deterministic $\mathcal{O}(n \log n)$-time algorithm. Then, Min-Min diameter optimization could be found via a brute-force solution by computing the minimum over all the minima of $\mathsf{D}^2(Q,t)$ for each $Q \in \mathcal{Q}_k$ and fixed $k < n$. That is, finding the minimum diameter of moving objects takes $\mathcal{O}(n \log n)$ time from the Algorithm in Chan [2004]. This must be repeated for each of the $\binom{n}{k}$ subsets $Q \subset S$ of cardinality k. Hence, this approach would result in $\mathcal{O}\left(\binom{n}{k} n \log n\right)$ time. A better solution will be derived as follows.

Lemma 3. *The local extrema of* $\mathsf{D}(S,t)$ *occur at the same time* t *as the local extrema of squared diameter* $\mathsf{D}^2(S,t)$.

Proof. Observe the following derivative of $\mathsf{D}(S,t)$, using the Chain Rule, with respect to time:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathsf{D}(S,t) = \frac{1}{2\mathsf{D}(S,t)}\left(\frac{\mathrm{d}}{\mathrm{d}t}\mathsf{D}^2(S,t)\right) \tag{4.1}$$

Given general position of S, $\mathsf{D}(S,t) \neq 0$ for any $t \in [0,1]$. Then, $\frac{\mathrm{d}}{\mathrm{d}t}\mathsf{D}(S,t)$ is zero if and only if $\frac{\mathrm{d}}{\mathrm{d}t}\mathsf{D}^2(S,t)$ is also zero. □

From Lemma 3, the squared diameter $\mathsf{D}^2(S,t)$ can be used to solve Min-Min optimization. As discussed previously, for any $p, q \in S$ with linear motion over t, the squared Euclidean distance $\mathrm{d}^2(p,q,t)$ is quadratic with respect to t. For a given $p(t) \in S$, let D_p be the set of parabolas that define the distance from p to any other object in S at time t, $D_p = \{\mathrm{d}^2(p,q,t) \mid q \in S \setminus \{p\}\}$. Let $\mathcal{A}(D_p)$ be the arrangement of the parabolas in D_p.

Observe that the k-th most distant point from p at time t can be given as the k-level of the arrangement $\mathcal{A}(D_p)$. Also, the k-level of $\mathcal{A}(D_p)$ identifies the k closest moving objects to p throughout the motion, for a total of $k+1$ moving objects on or below the k-level in $\mathcal{A}_{D_p}$. Then, a feasible solution could be derived from the $(k-1)$-level of each $\mathcal{A}(D_p)$ for each $p \in S$. The first step is to construct the arrangement $\mathcal{A}(D_p)$ and the corresponding $(k-1)$-level of $\mathcal{A}(D_p)$. Let ℓ_p be the vertical line segment connecting the minimum point of the $(k-1)$-level of the $\mathcal{A}(D_p)$ and the x-axis, which corresponds to the parabola $\mathrm{d}^2(p,p,t)$. Hence, maintaining the minimum among the lengths $|\ell_p|$ for each $p \in S$ solves the optimization problem at hand. Algorithm 4.1 formalizes the idea.

As discussed in Chapter 2, building $\mathcal{A}(D_p)$ takes $\mathcal{O}(n^2)$ time. Computing the k-level of an arrangement takes $\mathcal{O}((n+f)\log n)$ where f is the k-level complexity.

```
MIN-MIN-2DIAMETER(S, k)
1  |ℓ_min| = ∞
2  for each p in S
3       Construct the arrangement A(D_p).
4       Compute the (k − 1)-level of A(D_p).
5       |ℓ_p| = length of minimum vertical lines from minimum point
        on the k-level of A(D_p) and the x-axis.
6       |ℓ_min| = min{|ℓ_p|, |ℓ_min|}
7  return (|ℓ|)^{1/2} // Since D_p is the squared distance between points.
```

Algorithm 4.1: Min-Min optimization solution for diameter in two dimension.

Since Chan [2008] shows that the k-level of a parabolic arrangement has complexity $\mathcal{O}(n^{3/2}\log n)$, a valid upper bound is that it takes $\mathcal{O}(n^{3/2}\log^2 n)$ time to compute the k-level of $\mathcal{A}(D_p)$. Then Algorithm 4.1 takes $\mathcal{O}(n^3)$ time since n arrangements of $n-1$ parabolas are constructed.

Theorem 7. *Denote by S the set of n objects moving with constant velocity in $\mathbb{R}^2$, and let $\mathcal{Q}_k$ denote the collection of subsets of S with fixed cardinality k. The Min-Min optimization of diameter, i.e., $\min_{Q\in\mathcal{Q}_k}\min_{t\in[0,1]}\mathsf{D}(Q,t)$, can be solved in $\mathcal{O}(n^3)$ time.*

4.1.2 Max-Min

Similarly to the one-dimensional case, the solution of $\max_{Q\in\mathcal{Q}_k}\min_{t\in[0,1]}\mathsf{D}(Q,t)$ is equivalent to $\min_{t\in[0,1]}\mathsf{D}(S,t)$. An important observation is that for $Q\in\mathcal{Q}_k$, the minimum diameter of Q is the same or smaller than the minimum diameter of S. Since the optimization searches for the maximum among the minimum diameters, $\min_{t\in[0,1]}\mathsf{D}(S,t)$ will solve Max-Min diameter optimization.

A possible construction of the subsets $Q\in\mathcal{Q}_k$ is by the removal of $n-k$ elements from S. As points are removed from S to construct Q, from definition, the diameter

of Q for any $Q \in \mathcal{Q}_k$ satisfies the following:

$$\min_{t\in[0,1]} \mathsf{D}(Q,t) \leq \min_{t\in[0,1]} \mathsf{D}(S,t)$$

Hence, the maximum of $\min_{t\in[0,1]} \mathsf{D}(Q,t)$ is attained when $\min_{t\in[0,1]} \mathsf{D}(Q,t) = \min_{t\in[0,1]} \mathsf{D}(S,t)$, i.e., either when $Q = S$ or when Q contains the pair of points that realize the diameter of S. Let p and q be the points that realize the minimum diameter of S over the time domain. If Q contains the points p and q, then the maximum over minimum diameters is attained by p and q. This idea is also true for a measure $\mathsf{M}(Q,t)$ as shown in Theorem 8, where M can be the diameter or the area of the minimum enclosing circle of Q.

Theorem 8. *Given $Q \in \mathcal{Q}_k$, for a fixed $k \leq n$, a subset of S. Let M be the diameter of Q or the area of the minimum enclosing disk of Q. If $\min_{t\in[0,1]} \mathsf{M}(S,t)$ is realized by the elements in $S_\mathsf{M} \subseteq S$ and if there is at least one $Q \in \mathcal{Q}_k$ such that $S_\mathsf{M} \subseteq Q$ then:*

$$\max_{Q\in\mathcal{Q}_k} \min_{t\in[0,1]} \mathsf{M}(Q,t) = \min_{t\in[0,1]} \mathsf{M}(S,t)$$

Proof. Since the minimization of measure M on S over the time domain is realized by the elements in S_M at time t', then $\min_{t\in[0,1]} \mathsf{M}(S,t) = \mathsf{M}(S_\mathsf{M}, t')$. Suppose $Q' \in \mathcal{Q}_k$ is such that $S_\mathsf{M} \subseteq Q'$ and Q' are formed by removing elements from $S \setminus S_\mathsf{M}$, then the minimization of the measure is still realized by S_M. From the construction of any $Q \in \mathcal{Q}_k$ by removing $n-k$ elements of S, it is true that:

$$\min_{t\in[0,1]} \mathsf{M}(Q,t) \leq \min_{t\in[0,1]} \mathsf{M}(S,t) = \mathsf{M}(S_\mathsf{M}, t')$$

Since $S_{\mathsf{M}} \subseteq Q'$, we obtain that the maximum of $\min_{t\in[0,1]} \mathsf{M}(Q,t)$ over $Q \in \mathcal{Q}_k$, from the equation above, is attained at $t = t'$ and $\mathsf{M}(Q',t') = \mathsf{M}(S_{\mathsf{M}},t')$.

$$\max_{Q\in\mathcal{Q}_k} \min_{t\in[0,1]} \mathsf{M}(Q,t) = \mathsf{M}(Q',t') = \min_{t\in[0,1]} \mathsf{M}(S,t)$$

□

Given the assumption of general position, no two objects can be located at a common point in $\mathbb{R}^2$. Hence, the two-dimensional diameter of moving objects in S is realized by exactly 2 objects. In this case, Theorem 8 is valid for any k such that $2 \leq k \leq n$ for diameter when $d = 2$.

Then, the Max-Min solution will be given by $\min_{t\in[0,1]}, \mathsf{D}(S,t)$. Since the minimization of diameter considers all objects in S, $k = n$, a solution can be given from the algorithm proposed by Chan [2004]. Hence, this solution will take $\mathcal{O}(n \log n)$ time.

Theorem 9. *Let S be the set of n constant-velocity moving objects in the Euclidean plane, $\mathbb{R}^2$, and let $\mathcal{Q}_k$ be the collection of subsets of S with cardinality k. The Max-Min optimization of diameter, in the form of $\max_{Q\in\mathcal{Q}_k} \min_{t\in[0,1]} \mathsf{D}(Q,t)$, can be solved in $\mathcal{O}(n \log n)$ time.*

4.1.3 Min-Max and Max-Max

The diameter, by definition, is the maximization of the Euclidean distance between two points. When two objects are moving with constant velocity, the distance between them is a convex function over time. The diameter of a set of n moving objects, which is the maximum of $\binom{n}{2}$ convex functions, is also a convex function. The maximum

value of a convex function defined on a closed interval is attained at the boundary of the domain. The optimization $\max_{t\in[0,1]} \mathsf{D}(S,t)$ is equivalent to $\max\{\mathsf{D}(S,0), \mathsf{D}(S,1)\}$

Min-Max

The Min-Max optimization is realized at the boundary of the time domain. Then, Min-Max is equivalent to:

$$\min_{Q\in\mathcal{Q}_k} \max_{t\in[0,1]} \mathsf{D}(Q,t) = \min_{Q\in\mathcal{Q}_k} \max\{\mathsf{D}(Q,0), \mathsf{D}(Q,1)\}$$

A solution can also be derived from the set of parabolas $D_p = \{\mathrm{d}^2(p,q,t) \mid q \in S\}$ for each $p \in S$. Let $R_0(p)$ and $R_1(p)$ be the array consisting of the parabolas of D_p evaluated, respectively, at times $t = 0$ and $t = 1$. The k-th largest element of each array is necessary to represent the k-th most distant point from p at each corresponding time. Then, sort each array and maintain the maximum among the k-th elements of $R_0(p)$ and $R_1(p)$, and let this result be $r(p)$. Then, for each $p \in S$ the Min-Max optimization solution consists of the minimum among the results of $r(p)$.

```
MIN-MAX-2DIAMETER(S, k)
 1  D = ∞
 2  for each p ∈ S
 3       Construct set of parabolas D_p
 4       R_0(p) is array of parabolas in D_p evaluated at t = 0
 5       R_1(p) is array of parabolas in D_p evaluated at t = 1
 6       r_0 = k-th element of R_0(p) using selection algorithm
 7       r_1 = k-th element of R_1(p) using selection algorithm
 8       r(p) = max{r_0, r_1}
 9       D = min{D, r(p)}
10  return D
```

Algorithm 4.2: Min-Max optimization solution for diameter in two dimension.

Observe that the Algorithm 4.2 is a $\mathcal{O}(n^2)$-time solution. That is, the loop in lines 2 to 9 is repeated for $|S| = n$ times. Since the highest cost of time in the operations from line 3 to 9 consists of constructing the set of parabolas D_p and obtaining the k-th most distant element from p. Since finding the k-th largest element in unordered array using the selection algorithm in section 9.3 of Cormen et al. [2022] takes linear time, and constructing D_p also takes linear time, the total time complexity is in the order of $\mathcal{O}(n^2)$.

Theorem 10. *Let S be the set of n objects moving with constant velocity in $\mathbb{R}^2$. Also, let $\mathcal{Q}_k$ be the collection of subsets of S with fixed cardinality k. The Min-Max optimization of diameter, i.e., $\min_{Q \in \mathcal{Q}_k} \max_{t \in [0,1]} \mathsf{D}(Q,t)$, can be solved in $\mathcal{O}(n^2)$ time.*

Max-Max

From the idea presented in the Max-Min diameter solution and in Theorem 8, which states the maximum of diameter over sets $Q \in \mathcal{Q}_k$ will also be given as the diameter of S, the Max-Max optimization can be written as:

$$\max_{Q \in \mathcal{Q}_k} \max_{t \in [0,1]} \mathsf{D}(Q,t) = \max\{\mathsf{D}(S,0), \mathsf{D}(S,1)\}$$

As discussed, the Max-Max optimization solution can be reduced to solving the maximum between $\mathsf{D}(S,0)$ and $\mathsf{D}(S,1)$. Then, it is possible to use an algorithm for the diameter of points in the plane. Then, the solution also takes $\mathcal{O}(n)$ time. This idea is summarized in Algorithm 4.3.

Theorem 11. *Let S denote the set of n constant-velocity moving objects in $\mathbb{R}^2$, and let $\mathcal{Q}_k$ be the collection of subsets of S with cardinality k. The Max-Max optimization*

MAX-MAX-2DIAMETER(S, k)
1 $D_0 =$ diameter of $S(0)$.
2 $D_1 =$ diameter of $S(1)$.
3 **return** $\max\{D_0, D_1\}$

Algorithm 4.3: Max-Max optimization solution for diameter in two dimension.

of diameter, in the form of $\max_{Q \in \mathcal{Q}_k} \max_{t \in [0,1]} \mathsf{D}(Q, t)$*, can be solved in* $\mathcal{O}(n)$ *time.*

4.2 Width

4.2.1 Min-Min

An important fact for computing the width of a set P of points is that, in two dimensions, the width is realized by three points of P, say u, v, and w. As discussed in Chapter 2, the width of P is the distance from a line ℓ formed by two of the points, say u and v, and the third point w, which gives the translation magnitude and direction from ℓ to form a second parallel line ℓ'. From Definition 2, P must be contained in the intersection of the oppositely oriented half-planes H, defined by ℓ, and H', formed by ℓ'. Given the importance on the proposed algorithm to solve the Min-Min optimization of width, a brief discussion will be made on how to obtain the point-to-line distance formula.

Three points uniquely define a triangle. Given u, v and w as above, let Δuvw be a triangle whose base is b, the length of the segment $\overline{uv}$, and height is h, the minimum distance from w to the line derived from the segment $\overline{uv}$. The area of the triangle is $A_\Delta = \dfrac{bh}{2}$ which can be rearranged to compute the height $h = \dfrac{2A_\Delta}{b}$. For the triangle Δuvw, the oriented area, which is represented by the area vector, can be calculated as

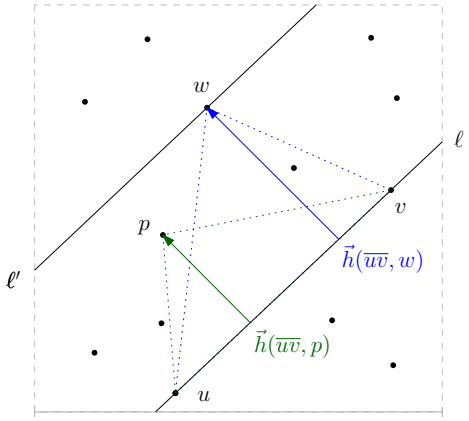


Figure 4.1: Example on how to determine if point p is enclosed by intersection of half-planes $H \cap H'$, where H is determined by the line $\overline{uv}$.

follows.

$$2\vec{A}(u,v,w) = \det \begin{bmatrix} \hat{i} & \hat{j} & \hat{k} \\ w_x - u_x & w_y - u_y & 0 \\ v_x - u_x & v_y - u_y & 0 \end{bmatrix} = \hat{k} \det \begin{bmatrix} w_x - u_x & w_y - u_y \\ v_x - u_x & v_y - u_y \end{bmatrix} \tag{4.2}$$

Let $A(u,v,w)$ denote the z-coordinate of the vector $\vec{A}(u,v,w)$, i.e., $A(u,v,w) = \vec{A}(u,v,w) \cdot \hat{k}$. Also, let $h(\overline{uv}, w)$ denote the signed magnitude of the height of Δuvw. The formula $h(\overline{uv}, w) = \dfrac{2A(u,v,w)}{b}$, which defines the minimum distance from the line $\overline{uv}$ to the point w, can be used to obtain the points of P that are inside the intersection of half-planes $H \cap H'$. That is, for any other point $p \in P$, if $h(\overline{uv}, p)$ and $h(\overline{uv}, w)$ have the same sign, both p and w are on the same side relative to the line $\overline{uv}$. If $h(\overline{uv}, p) < h(\overline{uv}, w)$, the point p is closer to the line $\overline{uv}$ compared to the distance of w to $\overline{uv}$, which means that p is in $H \cap H'$. Figure 4.1 illustrates the idea.

Consider now the set S of n objects moving with constant velocity. Recall that

$\mathcal{Q}_k$ is the collection of all subsets of S with fixed cardinality k. For $Q \in \mathcal{Q}_k$, the width of Q at time t is denoted by $\mathsf{W}(Q,t)$. A possible approach to solving Min-Min optimization of the width, written in the form $\min_{Q\in\mathcal{Q}_k} \min_{t\in[0,1]} \mathsf{W}(Q,t)$, can be defined based on the oriented-area formula. Considering the motion of objects in S, the oriented height formula of the triangle Δuvw, for $u, v, w, \in Q$, with the line ℓ defined from the segment $\overline{uv}$, based on Equation 4.2 the oriented height $h(\overline{uv}, w, t)$ can be written as:

$$\begin{aligned} h(\overline{uv}, w, t) &= \frac{(w_x(t) - u_x(t))(v_y(t) - u_y(t)) - (v_x(t) - u_x(t))(w_y(t) - u_y(t))}{\mathrm{d}(u, v, t)} \\ &= \frac{2A(u, v, w, t)}{b(t)} \end{aligned} \tag{4.3}$$

where $\mathrm{d}(u, v, t)$ corresponds to the length of the base of Δuwv at time t, also denoted $b(t)$, whose explicit formula is the Euclidean distance between u and v at time t.

Given a pair of points u and v defining the line ℓ, a possible solution is to build an arrangement of curves for each $p \in P$ in which the width is represented by the oriented height $h(\overline{uv}, p, t)$. Each curve in the arrangement would be expressed in the form of Equation 4.3. It would be an arrangement of a rational function whose numerator is a degree-two polynomial, while the denominator is a composition of a radical function and a degree-two polynomial.

Observe that for a fixed line $\overline{uv}$, the denominator of Equation 4.3 will be the same for any $w \in S$ forming the triangle Δuvw. For a fixed pair of objects u and v forming the line $\overline{uv}$, using the arrangement of oriented height constructed from the set $\{h(\overline{uv}, p, t) \mid p \in S\}$ to compute the width $\mathsf{W}(S,t)$ would be similar to using the arrangement constructed from the set of oriented area $\{A(u, v, p, t) \mid p \in S\}$ scaled by

the factor $\frac{1}{d(u,v,t)}$. The equations of $A(u,v,w,t)$ are degree-two polynomials which have a more simple format than the rational function composed of radicals in the denominator as in the format of $h(\overline{uv},w,t)$.

For each pair $\{u,v\}$ of elements in S, denote by $\mathcal{A}_h(\overline{uv},S)$ the arrangement of the curves whose expressions are in the form of Equation 4.3, oriented height $h(\overline{uv},w,t)$. Let $\mathcal{A}_A(\overline{uv},S)$ be the arrangement of curves expressed by the numerator of Equation 4.3, i.e., arrangement of curves of the oriented area $A(u,v,w,t)$. Lemma 4 shows that intersections between curves of both arrangements happen at the same time.

Lemma 4. *The intersections between the curves of $\mathcal{A}_h(\overline{uv},S)$ occur at the same time as the intersection between the corresponding curves in $\mathcal{A}_A(\overline{uv},S)$.*

Proof. For $p,q \in S$, given two curves $h(\overline{uv},p,t)$ and $h(\overline{uv},q,t)$ from the arrangement $\mathcal{A}_h(\overline{uv},S)$, suppose that these curves intersect at time t'. That is, their subtraction is equal to zero in the form:

$$h(\overline{uv},p,t') - h(\overline{uv},q,t') = \frac{2A(u,v,p,t') - 2A(u,v,q,t')}{b(t')} = 0$$

$$\Leftrightarrow A(u,v,p,t') - A(u,v,q,t') = 0$$

which corresponds to the intersection of curves $A(u,v,p,t)$ and $A(u,v,q,t)$ from the arrangement $\mathcal{A}_A(\overline{uv},P)$. □

Given the arrangement of the oriented area $\mathcal{A}_A(\overline{uv},S)$, let $A(u,v,v,t)$ or $A(u,v,u,t)$ be the x-axis for $x=t$. In case $A(u,v,p,t)$ is above the x-axis, i.e., $A(u,v,p,t) > 0$,

the object p is on one relative side of $\overline{uv}$, while if $A(u, v, p, t)$ is below the x-axis, p is on the opposite relative side of $\overline{uv}$. The Min-Min solution then searches for the k closest points to the segment $\overline{uv}$ in the time domain. In this case, a solution to the problem has to look for the k-th closest point to $\overline{uv}$ on both relative sides of the line ℓ defined from the segment $\overline{uv}$. Equivalently, in $\mathcal{A}_A(\overline{uv}, S)$ the solution searches for the k-th closest parabola above and below the x-axis.

Denote by $L_p(\overline{uv})$ the set of parabola segments above the x-axis formed from the oriented areas of triangles formed from the line $\overline{uv}$ and objects from S. Also, denote $L_n(\overline{uv})$ the parabola segments below the x-axis composed from the parabolas $-A(u, v, p, t)$ for $p \in S$. Then, construct the arrangements of parabola segments $\mathcal{A}(L_p(\overline{uv}))$ and $\mathcal{A}(L_n(\overline{uv}))$. Figure 4.2 illustrates an example of arrangements $\mathcal{A}(L_p(\overline{uv}))$ and $\mathcal{A}(L_n(\overline{uv}))$.

Given that $A(u, v, v, t) = 0$ and $A(u, v, u, t) = 0$ for every $t \in [0, 1]$, u and v are not in $\mathcal{A}(L_p(\overline{uv}))$ and $\mathcal{A}(L_n(\overline{uv}))$. Then, the k-th most distant point from the line $\overline{uv}$ corresponds to the $(k-2)$-level of $\mathcal{A}(L_p(\overline{uv}))$ and $\mathcal{A}(L_p(\overline{uv}))$. For every pair $\{u, v\}$ in $\mathcal{Q}_2$, the minimum among all minimum points of the $(k-2)$-levels of the arrangements will give the solution to the Min-Min width optimization. This idea is summarized in Algorithm 4.4.

Observe that $\mathcal{Q}_2$ has cardinality in the order of $\mathcal{O}(n^2)$, the number of pairs obtained from S. Lines 4 and 5 in Algorithm 4.4 takes $\mathcal{O}(n)$ time. Meanwhile, the construction of the parabolic arrangement takes $\mathcal{O}(n^2)$ time. As observed by Chan [2003], an arrangement of n parabola segments can be partitioned into $\mathcal{O}(n^{5/3})$ pseudo-segments. Hence, the $(k-2)$-level of $\mathcal{A}(L_p(\overline{uv}))$ and $\mathcal{A}(L_n(\overline{uv}))$ can be computed in $\mathcal{O}(n^{5/3} \log n)$

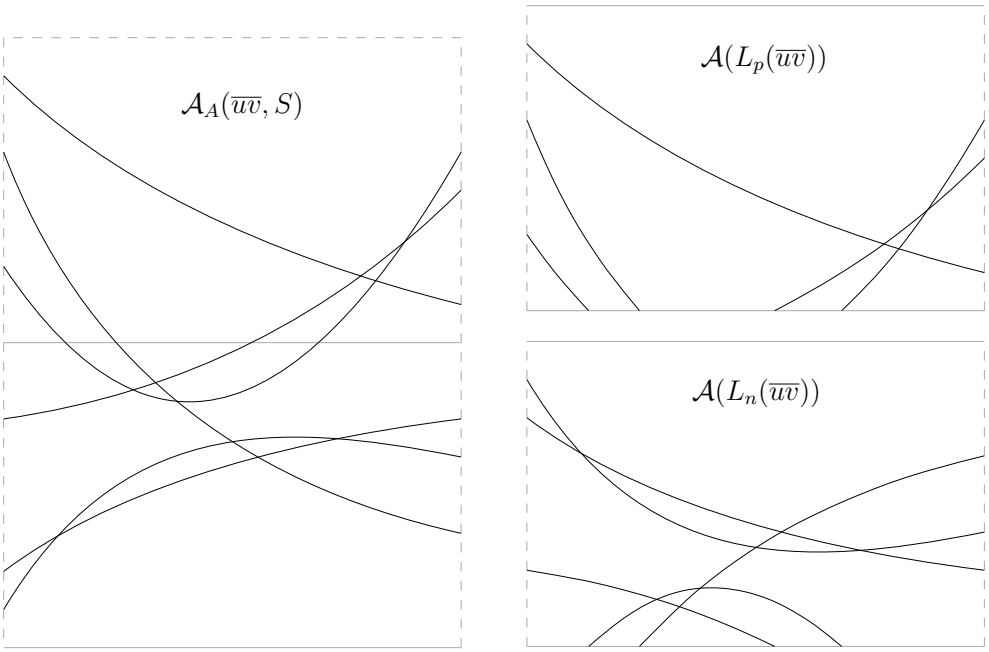


Figure 4.2: Example of parabolic arrangement $\mathcal{A}(\overline{uv}, S)$ and its division into parabolas segments above and below the x-axis, respectively $\mathcal{A}(L_p(\overline{uv}))$ and $\mathcal{A}(L_n(\overline{uv}))$

MIN-MIN-WIDTH(S, k)

```
1  W = ∞
2  Q_2 = unordered pairs of elements of S.
3  for each pair {u, v} in Q_2
4       L_p = set of parabola segments of oriented area above the x-axis
5       L_n = set of parabola segments of oriented below the x-axis
6       Compute the arrangement A(L_p(uv))
7       Compute the arrangement A(L_n(uv))
8       (w_p, t_p) = pair of point w_p and time t_p which realize minimum
        distance from (k − 2)-level of A(L_p(uv)) to the x-axis.
9       (w_n, t_n) = pair of point w_n and time t_n which realize minimum
        distance from (k − 2)-level of A(L_n(uv)) to the x-axis.
        // From the expression defined in Equation 4.3
10      W = min{W, (w_p(t_p)/ d(u, v, w_p, t_p)), (w_n(t_n)/ d(u, v, w_n, t_n))}
11 return W
```

Algorithm 4.4: Min-Min optimization solution for width in two dimension.

time. In this case, since the loop in the algorithm is repeated $\mathcal{O}(n^2)$ time, the total time complexity of Algorithm 4.4 is $\mathcal{O}(n^4)$ time.

Theorem 12. *Consider S as the set of n moving objects with constant velocity on the Euclidean plane, $\mathbb{R}^2$, and let $\mathcal{Q}_k$ be the collection of subsets of S with cardinality k. The Min-Min optimization of width, i.e., $\min_{Q \in \mathcal{Q}_k} \min_{t \in [0,1]} \mathsf{W}(Q,t)$ can be solved in $\mathcal{O}(n^4)$ time.*

4.2.2 Max-Min

As discussed in the previous section, the width of a set of points is realized by exactly three points. Denote $S_{\mathsf{W}} \subseteq S$ the subset of moving objects that realizes the minimum width of S at time t. Then, $|S_{\mathsf{W}}| = 3$. Now, consider the subsets $Q \in \mathcal{Q}_k$. If $k = 1$ or $k = 2$, the width is zero. Theorem 8 cannot be applied to width and a counter-example is illustrated in Figure 4.3. Observe in Figure 4.3 that for any $Q \in \mathcal{Q}_3$, the width $\mathsf{W}(Q,t)$ will be smaller than $\mathsf{W}(S,t)$ for a certain time t where $\mathsf{W}(S,t)$ is minimized.

In order to solve the Max-Min width optimization for $Q \in \mathcal{Q}_k$ in the time domain, the Min-Min solution can be modified to maintain the maximum width over each subset $Q \in \mathcal{Q}_k$. Algorithm 4.4 only needs a modification in line 10 to maintain the maximum instead of the minimum. Then, this will solve the Max-Min width optimization. In this case, the modification will not impact the final time complexity of the algorithm. Then, Max-Min solution takes $\mathcal{O}(n^4)$ time.

Theorem 13. *Let S be the set of n moving objects with constant velocity on the plane, $\mathbb{R}^2$, and let $\mathcal{Q}_k$ be the collection composed of all subsets of S with cardinality k. The*

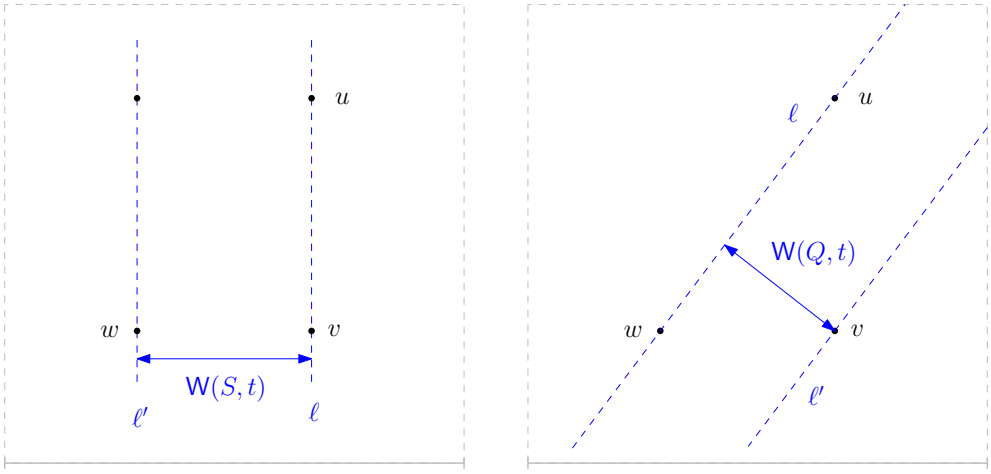


Figure 4.3: Example that Min-Max width solution for $Q \in \mathcal{Q}_k$ for $k = 3$ is not the same as the minimum of $\mathsf{W}(S, t)$.

Max-Min optimization of width, in the form $\max_{Q \in \mathcal{Q}_k} \min_{t \in [0,1]} \mathsf{W}(Q, t)$*, can be solved in* $\mathcal{O}(n^4)$ *time.*

4.2.3 Min-Max and Max-Max

To develop a solution for Min-Max and Max-Max optimization for the width, it is important to first check if the width with respect to time is a convex function. In the affirmative case, the maximization of the width over time is shown to be attained at the boundary of the time domain. For $u, v \in S$ and any point $p \in S$, there are no guarantees that the parabolas formed from oriented area $A(u, v, p, t)$ are convex.

Figure 4.4 shows an example where the parabola $A(u, v, p, t)$ is concave. In this figure, the image on the left illustrates initial and final position of each object, which induces relatively small width at times $t = 0$ and $t = 1$. On the right, the objects are at an intermediary time ($0 < t < 1$), the distance between p and the segment $\overline{uv}$ is larger when compared at times $t = 0$ and $t = 1$. Therefore, the parabola of the

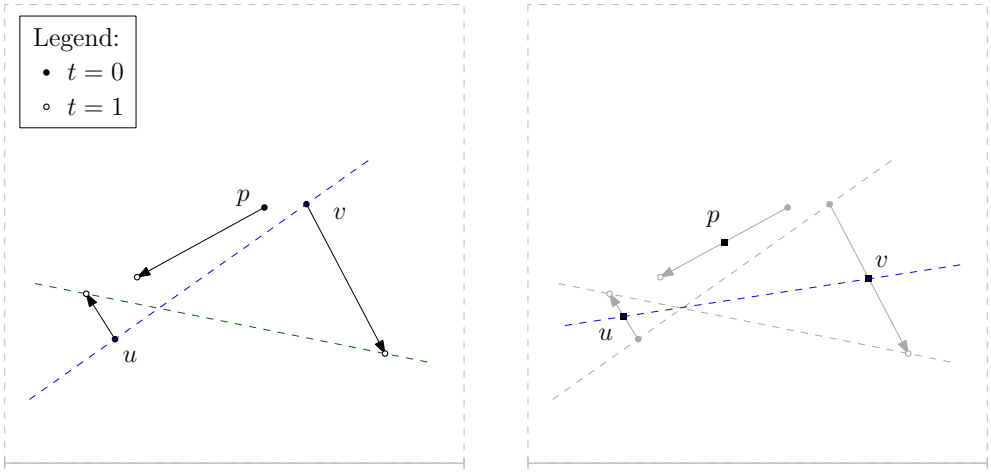


Figure 4.4: Representation of the linear constant-velocity motion of three objects u, v, and w such that the oriented-area of the triangle Δuvw is larger than times $t = 0$ and $t = 1$. That is, the parabola $A(u, v, p, t)$ is concave.

triangle oriented area $A(u, v, p, t)$ is a concave function.

In this case, the maximization of the width over time can also occur in the interior of the time domain $[0, 1]$. Hence, Algorithm 4.4 can also be modified solve the Min-Max and Max-Max optimization. The common step in the two problems is to find $\max_{t\in[0,1]} \mathsf{W}(Q, t)$ for each $Q \in \mathcal{Q}_k$. This step can be achieved by finding the maximum of the $(k-2)$-level on lines 8 and 9 of Algorithm 4.4.

For the solution of Max-Max, the line 10 of Algorithm 4.4 is modified to maintain the maximum of the width computed for different subsets of S. In case of the Min-Max solution, the line 10 remains maintaining the minimum of the width for each subset. These modifications are not going to affect the final time complexity of the Algorithm. Hence, both Max-Max and Min-Max solutions take $\mathcal{O}(n^4)$ time.

Theorem 14. *Let S be the set of n moving objects with constant velocity in the Eucledian plane, $\mathbb{R}^2$. Also, let $\mathcal{Q}_k$ be the collection of all subsets of S with cardinality*

k. The Max-Min and the Max-Max optimization of width, respectively,

$$\max_{Q\in\mathcal{Q}_k}\min_{t\in[0,1]}\mathsf{W}(Q,t) \qquad \textit{and} \qquad \max_{Q\in\mathcal{Q}_k}\max_{t\in[0,1]}\mathsf{W}(Q,t),$$

can be solved in $\mathcal{O}(n^4)$ *time.*

4.3 Axis-aligned Bounding Box

4.3.1 Min-Min

The Min-Min optimization of the axis-aligned bounding box area is of the form $\min_{Q\in\mathcal{Q}_k}\min_{t\in[0,1]}\mathsf{A}_B(Q,t)$, meanwhile the perimeter optimization is similar, but the objective function is the perimeter of the box, $\mathsf{P}_B(Q,t)$. Recall from Chapter 2 that optimizing $\mathsf{A}_B(Q,t)$ differs from optimizing $\mathsf{P}_B(Q,t)$ when $k < n$, as seen in the example of Figure 2.3. Any bounding box is realized by two, three, or four points on its boundary. In the case of the set S of n objects moving with constant velocity, the bounding box of S at some time t is realized by two, three, or four elements of S. Since the optimization searches for the smallest area or perimeter of a box enclosing k elements of S, a brute-force algorithm can be defined as follows.

For each combination of two, three, and four elements of S, let $B(t)$ be the axis-aligned box realized by these elements of S. That is, given that elements $a(t), b(t), c(t), d(t) \in S$ form the bounding box $B(t)$, the box has the following format:

$$B(t) = [x_{min}(t), x_{max}(t)] \times [y_{min}(t), y_{max}(t)]$$

such that:

$$x_{min}(t) = \min\{a_x(t), b_x(t), c_x(t), d_x(t)\} \quad x_{max}(t) = \max\{a_x(t), b_x(t), c_x(t), d_x(t)\}$$
$$y_{min}(t) = \min\{a_y(t), b_y(t), c_y(t), d_y(t)\} \quad x_{min}(t) = \min\{a_y(t), b_y(t), c_y(t), d_y(t)\}$$

and similarly for the cases where $B(t)$ is formed from two or three elements of S.

A Min-Min area solution is to find all subintervals of the time domain $[0, 1]$ such that $B(t)$ encloses k elements of S. Among all possible boxes $B(t)$, select the box B_{min} with the smallest area or perimeter in the time subintervals in which B_{min} encloses k elements of S. Further details on this algorithm are discussed below.

The first step in developing the algorithm is how to define a strategy to find time intervals where the given bounding box encloses k elements of S. Recall that $\mathcal{Q}_k$ denotes the collection of subsets of S whose cardinality is $k \leq n$. Let $B(t)$ be formed by elements of $\mathcal{Q}_2 \cup \mathcal{Q}_3 \cup \mathcal{Q}_4$, that is, B is realized by two, three, or four objects from S. Let $x_{min}(t)$ be the minimum x-coordinate of B at time t and $x_{max}(t)$ the maximum x-coordinate of $B(t)$ at time t. Similarly, let $y_{min}(t)$ and $y_{max}(t)$ be the minimum and maximum y-coordinate of $B(t)$ at time t. An object $p_i(t) = (p_{i,x}(t), p_{i,y}(t))$ from S is inside $B(t)$ at time t if and only if $x_{min}(t) \leq p_{i,x}(t) \leq x_{max}(t)$ and $y_{min}(t) \leq p_{i,y}(t) \leq y_{max}(t)$.

Consider that the inequality $x_{min}(t) \leq p_{i,x}(t) \leq x_{max}(t)$ is valid in the subinterval $[t_{i,x}, t'_{i,x}]$ and that $y_{min}(t) \leq p_{i,y}(t) \leq y_{max}(t)$ is valid in the subinterval $[t_{i,y}, t'_{i,y}]$. The objects p will be inside the box $B(t)$ in the time subinterval $[t_i, t'_i] = \big([t_{i,x}, t'_{i,x}] \cap [t_{i,y}, t'_{i,y}]\big)$. In order to find the times $t_{i,x}$ and $t'_{i,x}$ as the intersection of the polynomial $p_{i,x}(t)$ with $x_{min}(t)$ and $x_{max}(t)$, respectively. Similarly, find the times $t_{i,y}$

and $t'_{i,y}$ as the intersections of $p_{i,y}(t)$ with $y_{min}(t)$ and $y_{max}(t)$, respectively. Observe that $x_{min}(t)$ and $x_{max}(t)$ are piecewise-linear functions since they are corresponding x and y coordinates of the objects in motion defining the box $B(t)$. Hence, these subintervals can be computed in linear time.

Lemma 5. *Given a bounding box $B(t)$ and the set S of objects moving with constant velocity, each objects $p(t) \in S$ enters or exits $B(t)$ a constant number of times.*

Since objects move on linear trajectories with constant velocity, each object $p(t) \in S$ will be able to move into and out of the box $B(t)$ a constant number of times. Recall the definition of $B(t)$ above. Since $x_{max}(t)$ can be written in the piecewise form:

$$x_{max}(t) = \begin{cases} a_x(t), & t_1 \leq t < t_2 \\ b_x(t), & t_2 \leq t < t_3 \\ c_x(t), & t_3 \leq t < t_4 \\ d_x(t), & t_4 \leq t \leq t_5 \end{cases}$$

each object $p(t)$ can intersect $x_{max}(t)$ up to four times. The case is similar for $x_{min}(t), y_{max}(t)$, and $y_{min}(t)$. That is, since $B(t)$ is composed of four segments, each object $p(t) \in S$ can only cross each segment up to four times, given its linear trajectory and constant velocity. Then, in total, $p(t) \in S$ can cross $B(t)$ a maximum of 16 times.

Hence, each object $p_i(t)$ will have an associated subinterval of time $[t_i, t'_i]$ or an empty interval such that $p_i(t)$ is inside the box $B(t)$. Now, the idea is to find the intervals such that k objects of S are inside the box $B(t)$. For this, it is possible to use a simple sweep line algorithm to find the intersection of one-dimensional time

intervals such that $B(t)$ contains k elements of S.

The idea is simple: sweep the imaginary vertical line ℓ from left ($t = 0$) to right ($t = 1$). The line stops at a set of discrete events. In this case, the discrete events are the endpoints of the subintervals $[t_i, t_i']$ associated to each point p_i that corresponds to the time that p_i is inside the box B. If the line ℓ hits a starting point t_i of some subinterval, then one more point moves into the box B. Meanwhile, if ℓ hits the end point t_i', then the point p_i exits the box B. Then, maintain a counter which starts with the number of points inside B at time $t = 0$ and increments the counter when the beginning of a subinterval t_i is hit or decrement the counter when the end of a subinterval t_i' is hit.

Let H and H' be two minimum binary heaps such that H stores the starting points t_i of the subintervals discussed above and H' stores the corresponding end points t_i'. Consider the counter also defined above. Remove the minimum between the roots of H and H'. If the element was removed from H, then an object of S moved into B and the counter is incremented. If the element was removed from H', then the counter should be decremented as an element of S moved out of B.

Whenever the counter sums to k at some time t_i, then B encloses k elements of S, the time t is then stored as the initial time of B containing k objects. The subsequent event of the sweep line t' will change the counter so that it will be different from k, then also store the time t' to form the subinterval $[t, t']$ such that B encloses k elements of S. For each interval $[t, t']$ from the previous step, compute the minimum area or perimeter of the box B. For this, we define the routine AABB-Area-Interval which returns the minimum area of the box B on the closed interval $[t, t']$, when $t \neq -1$

and $t \neq -1$ (non-degenerate intervals), using Calculus, since the function of area of B in terms of time t is a piecewise parabola. Repeat this process until both heaps are empty and maintain the minimum among the minimum area of each box B. The idea is summarized in Algorithms 4.5 and 4.6.

```
MIN-MIN-AABB(S, k)
 1  A_min = ∞
 2  Let H and H' be two minimum binary heaps
 3  for each B in (Q_2 ∪ Q_3 ∪ Q_4)
 4        x_max(t) = maximum x-coordinate of B at time t
 5        x_min(t) = minimum x-coordinate of B at time t
 6        y_min(t) = maximum y-coordinate of B at time t
 7        y_min(t) = minimum y-coordinate of B at time t
 8        for each p_i(t) = (p_{i,x}(t), p_{i,y}(t)) in S
 9              [t_{i,x}, t'_{i,x}] = time such that p_{i,x}(t) is between x_max(t) and x_min(t)
10              [t_{i,y}, t'_{i,y}] = time such that p_{i,y}(t) is between y_max(t) and y_min(t)
11              [t_i, t'_i] = [t_{i,x}, t'_{i,x}] ∩ [t_{i,y}, t'_{i,y}] ∩ [0, 1]
12              Insert t_i in H
13              Insert t'_i in H'
14        A = MIN-AREA-INTERVALS(B, H, H', k)
15        A_min = min{A, A_min}
16  return A_min
```

Algorithm 4.5: Min-Min optimization solution for axis-aligned bounding box area in dimension two.

Observe that the loop in Algorithm 4.6 runs $\mathcal{O}(n)$ times since both H and H' have size n. Inside this loop, all the operation takes constant time except for root removal of H and H' which takes $\mathcal{O}(\log n)$ time. Hence, this algorithm takes $\mathcal{O}(n \log n)$ time.

In Algorithm 4.5 the loop in lines 3 to 15 is repeated $|\, \mathcal{Q}_2 \cup \mathcal{Q}_3 \cup \mathcal{Q}_4 \,| = \mathcal{O}(n^4)$ times. The inner loop on lines 8 to 13 is run exactly n where lines 12 and 13 takes $\mathcal{O}(\log n)$ time to execute. In this case, the for loop on lines 8 to 13 takes a total of $\mathcal{O}(n \log n)$ time. Also, as discussed above, line 14 calls Algorithm 4.6 which also takes

```
MIN-AREA-INTERVALS(B, H, H', k)
 1  A_min = ∞
 2  points_inside_B = 0
 3  heap_root = −1
 4  t = −1
 5  t' = −1
 6  while (H is not empty) or (H' is not empty)
 7       if H.root < H'.root
 8            points_inside_B = points_inside_B + 1
 9            heap_root = H.root
10            Remove root of H
11       else
12            points_inside_B = points_inside_B − 1
13            heap_root = H'.root
14            Remove root of H'
15       if t = −1 and points_inside_B = k
16            t = heap_root
17       if t ≠ −1 and points_inside_B ≠ k
18            t' = heap_root
19       A = AABB-AREA-INTERVAL(B, t, t')
20       A_min = min{A, A_min}
21  return A_min
```

Algorithm 4.6: Compute the minimum area of bounding box B when B encloses elements of S.

$\mathcal{O}(n \log n)$ time, given that procedure AABB-Area-Interval uses Calculus to find the minimum of a parabola (minimum area of box B) in a closed interval of time in $\mathcal{O}(1)$ time. Therefore, the Min-Min solution of the bounding box area or perimeter will take a total of $\mathcal{O}(n^5 \log n)$ time.

Theorem 15. *Let S be the set of n moving objects with constant velocity in the Euclidean plane, $\mathbb{R}^2$. Let $\mathcal{Q}_k$ be the collection of all subsets of S with cardinality k. The Min-Min optimization of the area and the perimeter of the axis-aligned bounding box, respectively,*

$$\min_{Q\in\mathcal{Q}_k}\min_{t\in[0,1]}\mathsf{A}_B(Q,t) \qquad \textit{and} \qquad \min_{Q\in\mathcal{Q}_k}\min_{t\in[0,1]}\mathsf{P}_B(Q,t)$$

can be solved in $\mathcal{O}(n^5 \log n)$ *time.*

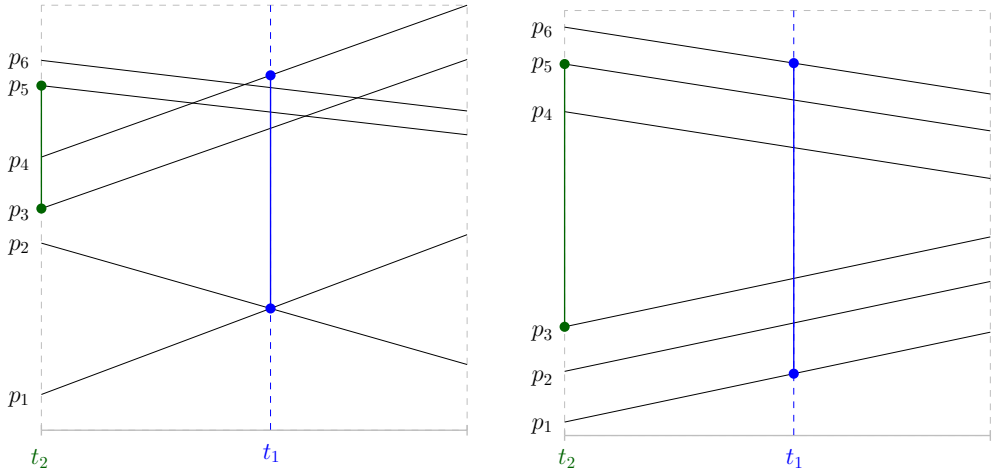
4.3.2 Max-Min

The Max-Min optimization for axis-aligned bounding box area and perimeter cannot be derived from Theorem 8. The fact that the minimum axis-aligned bounding box of S at time t can be realized by two, three, or four points invalidates the usage of Theorem 8 in some cases, which are discussed as follows.

Let S_A be the set of points such that $\min_{t\in[0,1]}\mathsf{A}_B(S,t)$ is realized. In the case where $|S_\mathsf{A}| > 2$ and $k = 2$, the sets of the collection $\mathcal{Q}_2$ are composed of two elements and a solution can be derived from the Euclidian distance between the objects of a set of $\mathcal{Q}_2$. That is, the Max-Min solution of the two-dimensional diameter would also solve the area and perimeter of the axis-aligned bounding box from the sets in $\mathcal{Q}_2$.

However, when $|S_\mathsf{A}| > 3$ and $k = 3$, there is no guarantee that the minimum axis-aligned bounding box for $k = 3$ is realized by the subset S_A. A simple counterexample is depicted in Figure 4.5. In this example, $\min_{t\in[0,1]}\mathsf{A}_B(S,t)$ is solved at $t = t_1$ and realized by $S_\mathsf{A} = \{p_1, p_2, p_4, p_6\}$. The Max-Min optimization for $k = 3$ is then realized at $t = t_2$ by $\{p_3, p_4, p_5\}$ which is not a subset of S_A. Hence, the conclusion of Theorem 8 does not apply to this case.

A solution to Max-Min optimization of the area or perimeter of axis-aligned bounding box can be derived similarly to the Min-Min solution. That is, for each $Q \in \mathcal{Q}_2 \cup \mathcal{Q}_3 \cup \mathcal{Q}_4$ find the intervals of time such that Q encloses k elements of S.



(a) $\mathcal{A}(S_x)$ (b) $\mathcal{A}(S_y)$

Figure 4.5: Minimum axis-aligned bounding of S and $Q \in \mathcal{Q}_3$ are realized by different elements of S.

Then, for each of those time subintervals, compute the minimum area or perimeter of the box. Among the minimum areas, maintain the maximum. That is, the maximum over the minimum areas (perimeters) over time will be the result of the Max-Min optimization of bounding box area or perimeter.

Algorithm 4.5 can be modified to accommodate Max-Min optimization. For this, line 1 in the algorithm will be replaced by $A_{max} = 0$ and line 15 will maintain the maximum, i.e., line 15 will be of the form $\mathsf{A}_{max} = \max\{A, A_{max}\}$. Since the modifications made are in constant-time operations, the total time complexity will not change and the Max-Min optimization will take a total of $\mathcal{O}(n^5 \log n)$ time.

Theorem 16. *Let S be the set of n moving objects with constant velocity in the Euclidean plane, $\mathbb{R}^2$. Let $\mathcal{Q}_k$ be the collection of all subsets of S with cardinality k. The Max-Min optimization of the area and the perimeter of the axis-aligned bounding box, respectively,*

$$\max_{Q \in \mathcal{Q}_k} \min_{t \in [0,1]} \mathsf{A}_B(Q,t) \qquad \textit{and} \qquad \max_{Q \in \mathcal{Q}_k} \min_{t \in [0,1]} \mathsf{P}_B(Q,t)$$

can be solved in $\mathcal{O}(n^5 \log n)$ *time.*

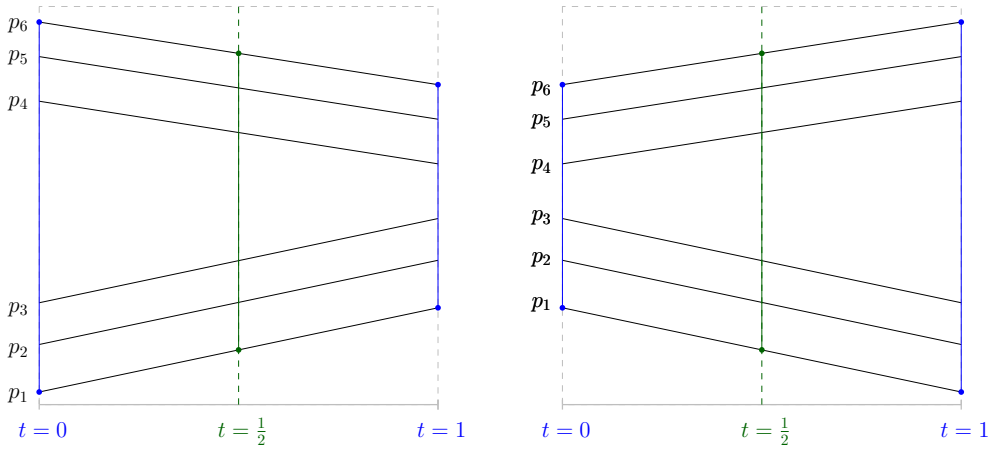
4.3.3 Min-Max and Max-Max

The main consideration when solving the maximization of a measure M over time is whether this measure is a convex function or not. If M is a convex function, then maximization occurs at the boundary of the domain. Recall that $\mathcal{A}(S_x)$ and $\mathcal{A}(S_y)$ are the arrangements of lines corresponding to the movement of the objects in S but projected onto the x and y-axes. Also, recall that $\ell_x(t)$ and $\ell_y(t)$ denote the vertical line segment connecting the upper and lower envelopes of $\mathcal{A}(S_x)$ and $\mathcal{A}(S_y)$, respectively.

Given the linear motion of S, the perimeter of the minimum axis-aligned bounding box, $\mathsf{P}_B(S,t) = \ell_x(t) + \ell_y(t)$, is a linear function. Hence, $\mathsf{P}_B(S,t)$ is maximized at the boundary of the time domain. The same property is not guaranteed for the area of a bounding box. The area of the bounding box can be defined as $\mathsf{A}_B(S,t) = |\ell_x(t)| \cdot |\ell_y(t)|$. The second derivative of the bounding box area with respect to time t is of the form:

$$\frac{\partial^2}{\partial t^2} \mathsf{A}_B(S,t) = 2 \frac{\partial |\ell_x(t)|}{\partial t} \cdot \frac{\partial |\ell_y(t)|}{\partial t}$$

which is not guaranteed to be zero. Indeed, it is possible to derive an example such that the area of the bounding box is maximized in the interior of $[0,1]$. Figure 4.6 shows an example where area of the bounding box is maximized when $t = \frac{1}{2}$. Hence,



(a) $\mathcal{A}(S_x)$ (b) $\mathcal{A}(S_y)$

Figure 4.6: Example of axis-aligned bounding box area is maximized in the interior of the time domain. At $t = 0$ and $t = 1$ the area of the bounding box is 41.4 while at $t = \frac{1}{2}$ the area is 44.0.

the entirety of the time domain must be considered for area maximization.

The Min-Max and Max-Max optimization of axis-aligned bounding box area must consider all the possible combinations of two, three, and four points that realize the bounding box. This process is similar to the Min-Min optimization of the bounding box area. For a box $B \in \mathcal{Q}_2 \cup \mathcal{Q}_3 \cup \mathcal{Q}_4$, given the intervals such that B encloses k elements of S, compute the maximum area of B in this interval. Hence, Algorithm 4.6 returns the maximum area instead of the minimum. Hence, denote this modified algorithm by Max-Area-Intervals. Since the number of iterations of the loop in Algorithm 4.6 remains the same and the binary root removal is still needed, the process Max-Area-Intervals also takes $\mathcal{O}(n \log n)$ time.

The Min-Max solution can be derived from Algorithm 4.5 with a single modification in line 14 such that this line will now call Max-Area-Intervals defined above. In the case of Max-Max optimization the modifications of Algorithm 4.5 are the following:

(i) Line 1 is replaced by $A_{max} = 0$.

(ii) Line 14 calls the procedure MAX-AREA-INTERVALS above.

(iii) Line 15 maintains the maximum of the areas computed in the previous line.

Hence, in both cases the total time complexity is not changed. Min-Max and Max-Max of axis-aligned bounding box area are solved in $\mathcal{O}(n^5 \log n)$ time.

Theorem 17. *Let S be the set of n constant-velocity moving objects in $\mathbb{R}^2$. Let $\mathcal{Q}_k$ be the collection of all subsets of S with cardinality k. The Min-Max and Max-Max optimization of the area of the minimum axis-aligned bounding box, respectively,*

$$\min_{Q \in \mathcal{Q}_k} \max_{t \in [0,1]} \mathsf{A}_B(Q,t) \qquad \textit{and} \qquad \max_{Q \in \mathcal{Q}_k} \max_{t \in [0,1]} \mathsf{A}_B(Q,t)$$

can be solved in $\mathcal{O}(n^5 \log n)$ time.

Recall from the arguments at the beginning of this section that Theorem 8 does not apply to the perimeter P_B of the minimum axis-aligned bounding boxes. In the case of Min-Max and Max-Max perimeter optimizations, the perimeter over time is maximized at the boundary, i.e., at $t = 0$ or $t = 1$. A line sweep strategy can be applied to solve these two cases, similar to the Min-Min diameter optimization in the one-dimensional case. However, the sweep is restricted to the times $t = 0$ and $t = 1$ only. That is, let T be an array containing the pairs $(p, 0)$ and $(p, 1)$ for each $p \in S$. Compute $\ell_x^+(p)$ and $\ell_x^-(p)$ which are the vertical line segments in $\mathcal{A}(S_x)$ starting at p and intersecting $k-1$ lines, respectively, above and below p. Then, compute sets $Q_{p,x}^+$ and $Q_{p,x}^+$ containing the elements of $A(S_x)$, respectively, intersected by $\ell_x^+(p)$ and $\ell_x^-(p)$. Recall that this operation takes $\mathcal{O}(n)$ time.

In the case of Min-Max, the problem is equivalent to $\max_{(p,t)\in T} \min\{\mathsf{P}_B(Q^+_{p,x}, t), \mathsf{P}_B(Q^-_{p,x}, t)\}$. Then, let t be a time from T, i.e. either $t = 0$ or $t = 1$, compute the minimum of the perimeter of Q^+_p and Q^-_p. Repeat the same process for $\mathcal{A}(S_y)$, compute corresponding vertical line segments $\ell^+_y(p)$ and $\ell^+_y(p)$. Also, compute the subsets $Q^+_{p,y}$ and $Q^-_{p,y}$ and its corresponding perimeters. Among the minima of perimeter computed at each t from T, return the maximum which is the Min-Max perimeter solution. Since $|T| = 2n$ in this case, the sweep line algorithm takes $\mathcal{O}(n^2)$ time.

```
MIN-MAX-BBOX(S, k)
 1  P_min = ∞
 2  Compute arrangement A(S_x)
 3  Compute arrangement A(S_y)
 4  Let T be the array containing all pairs (p, 0) and (p, 1) for each p ∈ S
 5  for (p, t) in T
 6        ℓ⁺_x(p) = vertical segment from p, intersecting k − 1 lines above p in A(S_x)
 7        ℓ⁻_x(p) = vertical segment from p, intersecting k − 1 lines below p in A(S_x)
 8        ℓ⁺_y(p) = vertical segment from p, intersecting k − 1 lines above p in A(S_y)
 9        ℓ⁻_y(p) = vertical segment from p, intersecting k − 1 lines below p in A(S_y)
10        Compute sets Q⁺_{p,x}, Q⁻_{p,x}, Q⁺_{p,y}, Q⁻_{p,y}
11        P = max{P(Q⁺_{p,x}, t), P(Q⁻_{p,x}, t), P(Q⁺_{p,y}, t), P(Q⁻_{p,y}, t)}
12        P_min = min{P_min, P}
13  return P_min
```

Algorithm 4.7: Compute the axis-aligned bounding box perimeter Min-Max optimization.

Observe also that the optimization $\max_{(p,t)\in T} \max\{\mathsf{P}_B(Q^+_{p,x}, t), \mathsf{P}_B(Q^-_{p,x}, t), \mathsf{P}_B(Q^+_{p,y}, t), \mathsf{P}_B(Q^-_{p,y}, t)\}$ is equivalent to Max-Max perimeter optimization. Hence, this optimization solution can be derived similarly to Algorithm 4.7. However, in the loop from lines 5 to 12 in Algorithm 4.7 should maintain the maximum in line 12. That is, return the maximum among the maxima of perimeter at each time t in T. Also, as $|T| = 2n$ in this case, the line sweep algorithm takes $\mathcal{O}(n^2)$ time.

Theorem 18. *Let S be the set of n constant-velocity moving objects in $\mathbb{R}^2$. Let $\mathcal{Q}_k$ be the collection of all subsets of S with cardinality k. The Min-Max and Max-Max optimization of the perimeter of the minimum axis-aligned bounding box, respectively,*

$$\min_{Q \in \mathcal{Q}_k} \max_{t \in [0,1]} \mathsf{P}_B(Q,t) \qquad \textit{and} \qquad \max_{Q \in \mathcal{Q}_k} \max_{t \in [0,1]} \mathsf{P}_B(Q,t)$$

can be solved in $\mathcal{O}(n^2)$ time.

4.4 Minimum Enclosing Disk

4.4.1 Min-Min

The minimum enclosing disk $C(t)$ of a set of objects moving with constant velocity S can be defined by two or three objects of S on the boundary of $C(t)$. The Min-Min optimization of minimum enclosing disk area searches for a solution of $\min_{Q \in \mathcal{Q}_k} \min_{t \in [0,1]} \mathsf{A}(Q,t)$, where $\mathsf{A}(Q,t)$ is the area of the minimum enclosing disk of the subset $Q \subset S$ at time t and $\mathcal{Q}_k$ refers to the collection of subsets of S such that the subsets have cardinality k.

The proposed solution to this problem is a brute-force algorithm. Given a pair or a triple of objects of S forming a disk $C(t)$, find the time subinterval $[t_1, t_2] \subset [0,1]$ such that $C(t)$ encloses k elements of S, then find the minimum area of the minimum enclosing circle in the closed interval $[t_1, t_2]$. Hence, the Min-Min optimization can be derived from the minimum among the minimum areas found in the previous step. For this, an important observation is that the number of time subintervals such that $C(t)$ encloses k elements of S is linear.

Lemma 6. *Given a disk $C(t)$ formed by two or three moving objects from S whose elements move with constant velocity, and a distinct object $p(t) \in S$. The object $p(t)$ enters or exits $C(t)$ a constant number of times.*

The proof of Lemma 6 can be discussed in two cases: when $C(t)$ is formed from two objects of S and when $C(t)$ is formed from three objects S. The proof idea is discussed as follows.

Consider the possible disks formed from the objects $u(t)$ and $v(t)$ from S. The minimum-area disk $C(t)$ containing $u(t)$ and $v(t)$ on its boundary is the disk whose diameter is given by the distance $\mathrm{d}(u, v, t)$ from $u(t) = (u_x(t), u_y(t))$ to $v(t) = (v_x(t), v_y(t))$ at time t. In this case, the coordinates $c(t) = (c_x(t), c_y(t))$ of the center of the disk $C(t)$ and its radius $r(t)$ are in the form:

$$(c_x(t), c_y(t)) = \left(\frac{1}{2}(u_x(t) + v_x(t)), \frac{1}{2}(u_y(t) + v_y(t))\right) \qquad \text{and} \qquad r(t) = \frac{1}{2}\,\mathrm{d}(u, v, t)$$

Given the constant-velocity motion of the objects in S, the coordinates $c_x(t)$ and $c_y(t)$ are modeled by degree-one polynomials. An objects $p(t) \in S$ enters or exits $C(t)$ at the time t when $\mathrm{d}(p, c, t) = r(t)$. Observe that the distance function and the radius of $C(t)$, as defined above, are rational functions in terms of t. It is similar to find the time $p(t)$ enters or exits $C(t)$ in terms of the squared distance, $\mathrm{d}^2(p, c, t)$, and the squared radius, $r^2(t) = [r(t)]^2$. Equivalently, $p(t)$ enters or exits $C(t)$ at the times t which satisfy the equation $\mathrm{d}^2(p, c, t) - r^2(t) = 0$. Hence, $\mathrm{d}^2(p, c, t) - r^2(t)$ is a polynomial of degree two with respect to t and there are at most two real roots of this polynomial. This means that $p(t)$ can enter or exit $C(t)$ at most two times.

Now, consider that $C(t)$ is the enclosing disk which contains the objects $u(t), v(t)$, and $w(t)$ from the set S on the disk boundary. In this case, $C(t)$ is uniquely defined at each time t by the three objects $u(t), v(t)$, and $w(t)$. The center $c(t) = (c_x(t), c_y(t))$ of the disk $C(t)$ and its radius $r(t)$, as discussed by Schneider and Eberly [2002], is given by:

$$c_x(t) = u_x(t) + \frac{(w_y(t) - u_y(t))\,\mathrm{d}^2(u, v, t) - (v_y(t) - u_y(t))\,\mathrm{d}^2(u, w, t)}{2[(v_x(t) - u_x(t))(w_y(t) - u_y(t)) - (w_x(t) - u_x(t))(v_y(t) - u_y(t))]} \tag{4.4}$$

$$c_y(t) = u_y(t) + \frac{(v_x(t) - u_x(t))\,\mathrm{d}^2(u, w, t) - (w_x(t) - u_x(t))\,\mathrm{d}^2(u, v, t)}{2[(v_x(t) - u_x(t))(w_y(t) - u_y(t)) - (w_x(t) - u_x(t))(v_y(t) - u_y(t))]} \tag{4.5}$$

$$r(t) = \sqrt{(c_x(t) - u_x(t))^2 + (c_y(t) - u_y(t))^2} \tag{4.6}$$

As in the first case, an object $p(t) \in S$ enters or exits the disk $C(t)$ at time t when the distance from $p(t)$ to the center of $C(t)$ is equal to the radius $r(t)$ of $C(t)$, i.e., $\mathrm{d}^2(p, c, t) - r^2(t) = 0$. The subtraction of polynomials $\mathrm{d}^2(p, c, t) - r^2(t)$, from the definitions above, will result in a rational function in terms of t such that the polynomial in the numerator has degree six and the polynomial in the denominator has degree four. Searching for the zeros of this polynomial subtraction, only the numerator needs be equal zero. Hence, the zeros of a polynomial of degree six will yield up to six real solutions. Therefore, the objects $p(t)$ can enter or exit $C(t)$ a

constant number of times.

Since the objects in $p(t)$ enter and exits the disk $C(t)$ a constant number of times and S contains n objects moving with constant-velocity, there are up to $\mathcal{O}(n)$ intervals of time such that $C(t)$ encloses k objects of S, for all combinations of two and three objects of S forming the disk $C(t)$. In this case, a Min-Min solution can be derived by searching the subintervals of time such that $C(t)$ encloses k objects of S.

However, it is important to observe that, in the case that $C(t)$ is formed by three objects $u(t), v(t), w(t) \in S$, the time such that $u(t), v(t)$, and $w(t)$ are aligned has to be considered in the subintervals described above. Let t_0 be time such that $u(t_0), v(t_0)$, and $w(t_0)$ are aligned. That is, at t_0, u, v, and w form a line $\overline{uvw}$ which divides the plane into two oppositely oriented half-planes, say H and H'. For some $\varepsilon > 0$, from time $t = t_0 - \varepsilon$ to $t = t_0 + \varepsilon$, the circle $C(t)$ move from one half-plane to the other, e.g., from H to H'. In this case, the number of points enclosed by $C(t)$ may abruptly change from before and after the time t_0. Figure 4.7 illustrates the idea.

Lemma 7. *Given three objects $u(t)$, $v(t)$, and $w(t)$ moving with constant velocity, the number of times that these objects can form a line segment $\overline{uvw}$ is bounded above by a constant.*

The sketch of the proof of Lemma 7 is discussed as follows. Given three objects $u(t), v(t)$, and $w(t)$ moving with constant velocity, let $u(t)$ and $v(t)$ form the segment $\overline{uv}(t)$ for each time t. Then, the object $w(t)$ will be aligned with $\overline{uv}(t)$ when the following equation is satisfied:

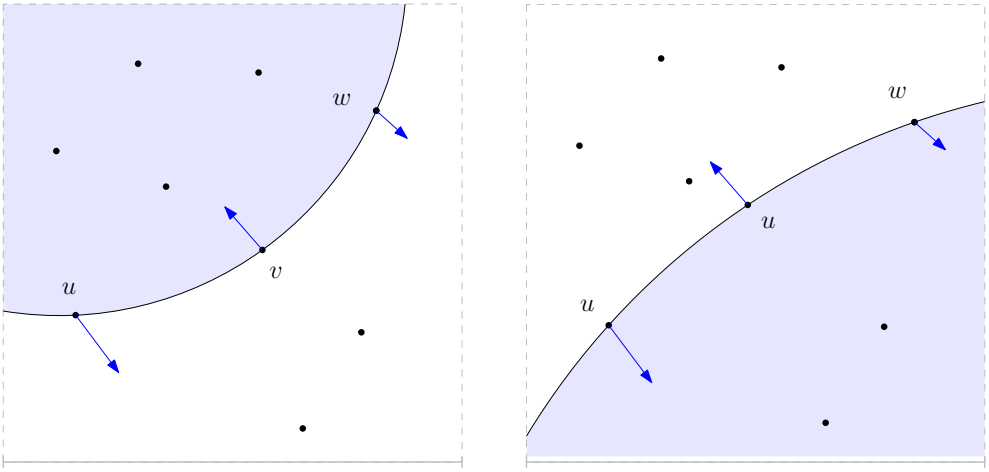


Figure 4.7: The objects enclosed by $C(t)$ may abruptly change if $C(t)$ is formed by three objects of S and the objects are aligned at certain time t_0. For some $\varepsilon > 0$, the image on the left show the points at some time $t_0 - \varepsilon$ and the left image is at time $t_0 + \varepsilon$.

$$\det \begin{bmatrix} u_x(t) - w_x(t) & u_y(t) - w_y(t) \\ v_x(t) - w_x(t) & v_y(t) - w_y(t) \end{bmatrix} = 0$$

Given the constant velocity of the objects, the determinant above will result in a polynomial with degree up to two with respect to t. Hence, there are up to two real values of t which satisfy the equation above. In this case, $w(t)$ can intersect $\overline{uv}(t)$ up to two times. This case is similar to $v(t)$ intersecting $\overline{uw}(t)$ and also $u(t)$ intersecting $\overline{vw}(t)$. Therefore, the objects $u(t), v(t)$, and $w(t)$ can be aligned into a line segment up to a constant number of times.

As previously discussed in the beginning of this section, the Min-Min optimization of minimum enclosing disk area can be found by exploring the subintervals of time in which a disk $C(t)$ encloses k objects of S. For a disk $C(t)$ formed from objects of S, compute the center $c(t)$ and the radius $r(t)$ of $C(t)$. Let T be an empty array.

Compute the time t such that $\mathrm{d}^2(p, c, t) - r^2(t) = 0$ is satisfied via some stable numerical algorithm, and add each solution t to the array T as long as t is inside the time domain $[0, 1]$. In case $C(t)$ is formed by three objects, compute the times such that the three objects are aligned and add those times to T.

The times in T will be the endpoints of the subintervals in which $C(t)$ encloses the same number of objects from S. Then, sort T and find the number of objects in each subinterval $[T[i], T[i+1]]$ for i ranging from 0 to $|T| - 2$. In the case that $C(t)$ encloses k objects in the subinterval $[T[i], T[i+1]]$, find the minimum of $r(t)$ in this closed subinterval using a numerical solver for the equations of $r(t)$ defined above, e.g., the steepest descent method. Repeat this process for every possible $C(t)$ and maintain the minimum of the minimum radius found in the previous step. This solution is presented in Algorithm 4.8.

In Algorithm 4.8, the lines 3 to 9 and also lines 12 to 13 run in $\mathcal{O}(1)$ time. Since the solution of a polynomial equation of fixed degree can be numerically approximated in $\mathcal{O}(1)$ time, the loop in lines 10 to 11 runs in $\mathcal{O}(n)$ time. Observe that the loop in lines 16 to 19 runs in $\mathcal{O}(n)$ time, and the loop in lines 14 to 22 runs in $\mathcal{O}(n^2)$ total time given that $|T| = \mathcal{O}(n)$ from Lemmas 6 and 7. Hence, since $|\,\mathcal{Q}_3\,| = \mathcal{O}(n^3)$, the Algorithm 4.8 takes a total of $\mathcal{O}(n^5)$ time.

Theorem 19. *Let S be the set of n objects moving with constant velocity in the Euclidean place, $\mathbb{R}^2$. Also, let $\mathcal{Q}_k$ be the collection of subsets of S with cardinality k. The Min-Min optimization of the minimum enclosing disk area, i.e., $\min_{Q \in \mathcal{Q}_k} \min_{t \in [0,1]} \mathsf{A}(Q, t)$, can be solved in $\mathcal{O}(n^5)$ time.*

```
MIN-MIN-ENCLOSING-DISK(S, k)
 1  r_min = ∞
 2  for C ∈ (Q_2 ∪ Q_3)
 3       if |C| == 2
 4            Let u(t) and v(t) be the elements of C
 5       else
 6            Let u(t), v(t), and w(t) be the elements of C
 7       c(t) = the center of C(t) containing elements of C on the boundary
 8       r(t) = the corresponding radius of C(t)
 9       T = empty array // Will contain times where p(t) ∈ S enters or exits C(t)
10       for p(t) ∈ S \ (C ∪ {p})
11            Add solutions of d^2(p, c, t) − r^2(t) = 0 to array T, if t is in [0, 1]
12       if |C| == 3
13            Add times t to T when elements of C are aligned, if t is in [0, 1]
14       for i = 0..(|T| − 2)
15            points_in_C = 0
16            for p(t) ∈ S \ C
17                 // Use halfpoint of time interval
18                 if d^2(p, c, (T[i] + T[i + 1])/2) ≤ r^2(t)
19                      points_in_C = points_in_C + 1
20            if points_in_C == k − |C|
21                 r = find minimum of r(t) in the closed subinterval [T[i], T[i + 1]]
22                 r_min = min{r, r_min}
23  return π r_min^2
```

Algorithm 4.8: Min-Min optimization solution for minimum enclosing disk area in two dimension.

4.4.2 Max-Min

The main idea to solve Max-Min optimization for minimum enclosing disk is also to use Theorem 8. That is, the minimization of the disk area over time for S will also give the solution for Max-Min optimization for all $Q \in \mathcal{Q}_k$ and $k \leq n$. This is an application of Theorem 8 with the fact that the minimum enclosing disk can be defined by two or three objects of S.

Let $\mathsf{A}(S, t)$ denote the area of the minimum disk enclosing S at time t. Denote

by S_{A} the set containing the points that realize the solution of $\min_{t\in[0,1]} \mathsf{A}(S,t)$. Then, $|S_{\mathsf{A}}| = 2$ or $|S_{\mathsf{A}}| = 3$. Theorem 8 is only valid when there is at least one $Q \in \mathcal{Q}_k$ such that $S_{\mathsf{A}} \subseteq Q$. Then, if $|S_{\mathsf{A}}| = 2$, since the area of the enclosing disk of elements in $\mathcal{Q}_1$ is zero, Theorem 8 can be applied for any $2 \leq k \leq n$. When $|S_{\mathsf{A}}| = 3$, the idea in Theorem 8 can be applied when $3 \leq k \leq n$, but not when $k = 2$ as $S_{\mathsf{A}} \not\subset Q$ for every $Q \in \mathcal{Q}_2$. Thus, the special case of $k = 2$ and $|S_{\mathsf{A}}| = 3$ must be considered separately.

Let $|S_{\mathsf{A}}| = 3$. For $k = 2$, there are only two points in every $Q \in \mathcal{Q}_2$. The Euclidean distance between the two points in Q will be the diameter of the minimum enclosing disk and also the diameter of the set Q. Then, for this special case, it is possible to use the Min-Max solution of the two-dimensional diameter to solve the Min-Max of minimum enclosing disk area, which takes $\mathcal{O}(n \log n)$ time from the algorithm in Chan [2004].

The remaining cases are $k = 2$ and $|S_{\mathsf{A}}| = 2$, or $|S_{\mathsf{A}}| = 3$ and $3 \leq k \leq n$. Then, Theorem 8 is valid and Min-Max for the minimum enclosing disk is solved by $\min_{t\in[0,1]} \mathsf{A}(S,t)$. In this case, Algorithm 4.8 can be considered to solve this problem, with the difference that it needs to maintain the maximum in line 22. That is, line 3 is replaced by $r_{max} = 0$ and the line 22 is replaced by $r_{max} = \max\{r_{max}, r\}$. Given that there is no significant modification in the running time of the algorithm, this solution also runs in $\mathcal{O}(n^5)$ time.

Theorem 20. *Let S be the set of n objects moving with constant velocity in the Euclidean place, $\mathbb{R}^2$. Also, let $\mathcal{Q}_k$ be the collection of subsets of S with cardinality k. The Max-Min optimization of the minimum enclosing disk area, i.e., $\max_{Q\in\mathcal{Q}_k} \min_{t\in[0,1]} \mathsf{A}(Q,t)$, can be solved in $\mathcal{O}(n^5)$ time.*

4.4.3 Min-Max and Max-Max

Observe that when the minimum enclosing disk $C(t)$ is composed by three objects moving with constant velocity in linear trajectory, the center $c(t)$ and radius $r(t)$ of $C(t)$ are defined as in Equations 4.4, 4.5, and 4.6. In this case, the squared radius $r^2(t) = [r(t)]^2$ of $C(t)$ is a rational function with respect to t whose polynomial in the numerator has degree six and the polynomial in the denominator has degree four.

Hence, there is no guarantee that the minimum enclosing disk area $\mathsf{A}(t) = \pi r^2(t)$ is a convex function. A counter-example showing that the area is not a convex function is not easy to obtain, given the possible degrees of freedom of the movement of $C(t)$. Therefore, Min-Max and Max-Max solution needs to consider that the maximization of the minimum enclosing disk area may occur in the interior of closed intervals of time.

Min-Max

A solution for the Min-Max problem of the form $\min_{Q \in \mathcal{Q}_k} \max_{t \in [0,1]} \mathsf{A}(Q, t)$, where $\mathcal{Q}_k$ is the collection of subsets of cardinality k from the objects moving with constant velocity in S, can be derived from Algorithm 4.8. As Algorithm 4.8 solves the Min-Min optimization, a solution to Min-Max needs to maintain the maximum of area over the subintervals of time. That is, replace the line 21 with r as the maximum area of the minimum enclosing circle in the given subinterval of time. This would solve the Min-Max optimization. Thus, this is also a $\mathcal{O}(n^5)$-time solution.

Max-Max

Recall that the Max-Max optimization is formulated as $\max\limits_{Q\in\mathcal{Q}_k}\max\limits_{t\in[0,1]}\mathsf{A}(Q,t)$. Observe that the maximum of area is sought on the time interval and also among the subsets in $\mathcal{Q}_k$. Using the Min-Min solution in Algorithm 4.8, the line 21 has to be replaced by the radius r which gives maximum area of the minimum enclosing circle in the given subinterval of time. Also, the solution needs to maintain maximum among the subsets in $\mathcal{Q}_k$. For this, the line 3 in Algorithm 4.8 is replaced by $r_{max} = 0$, and the line 22 is replaced by the maximization, i.e., $r_{max} = \max\{r, r_{max}\}$. Therefore, the Max-Max solution also takes $\mathcal{O}(n^5)$ time.

Theorem 21. *Let S be the set of n constant-velocity moving objects in the plane, i.e., $\mathbb{R}^2$. Also, let $\mathcal{Q}_k$ denote the collection of subsets of S with cardinality k. The Min-Max and the Max-Max optimization of the minimum enclosing disk area, respectively,*

$$\min_{Q\in\mathcal{Q}_k}\max_{t\in[0,1]}\mathsf{A}(Q,t) \qquad \textit{and} \qquad \max_{Q\in\mathcal{Q}_k}\max_{t\in[0,1]}\mathsf{A}(Q,t)$$

can be solved in $\mathcal{O}(n^5)$ time.

Chapter 5

Conclusion

The optimization of Minimum Moving Spanning Trees over a set S of objects moving in constant-velocity linear trajectory in $\mathbb{R}^2$ proposed by Akitaya et al. [2021] opened ground for optimizing other geometric measures on S. This thesis proposed linear-time algorithmic solutions for optimizing the diameter, width, minimum alignment of the axis boundary box and minimum envelope disk of the set S. Future work can also be determined in terms of other measures of extent and proximity. Also, the proposal of other restrictions for the space and movement of elements in S can lead to new developments.

Another measure that could be considered is the area and perimeter of the convex hull of S. Let $\mathcal{Q}_k$ be the collections of subsets of S with cardinality k. It is not trivial to determine whether minimizing the area of the convex hull of Q over $Q \in \mathcal{Q}_k$ and over the time domain can be solved by a polynomial-time algorithm. The same would apply to the perimeter of the convex hull of Q.

The movement in three-dimension could also be considered as future work. Some

of the solutions presented in this thesis can be extended to $\mathbb{R}^3$. However, the time complexity will scale accordingly; for example, the arrangement of lines in $\mathbb{R}^3$ can be constructed in optimal $\Theta(n^3)$ time, as seen in de Berg et al. [2008], via an incremental algorithm increasing the time complexity of the solutions. Another idea is to consider the movement of S as polynomials of degree greater than one. In this case, the arrangement of curves will contain a possible higher number of intersections between pairs of curves. Hence, $\lambda_s(n)$ will increase its complexity as s increases, consequently resulting in a higher time complexity to construct arrangements, as well as to find envelopes and k-levels.

The examples cited above are a few of the possibilities of new research ideas that can be considered from the developments in Akitaya et al. [2021]. More recently, Wachholz and Suri [2023] shows hardness results for a minimum moving spanning tree when optimizing the sum of edge lengths. That is, the hardness of optimization of other geometric measures can also be a new path of research. Therefore, optimization of geometric measures of objects in movement may be considered a promising topic to be explored.

Bibliography

P. K. Agarwal, L. J. Guibas, J. Hershberger, and E. Veach. Maintaining the extent of a moving point set. *Discrete & Computational Geometry*, 2001.

H. A. Akitaya, A. Biniaz, P. Bose, J.-L. De Carufel, A. Maheshwari, L. F. S. X. da Silveira, and M. Smid. The minimum moving spanning tree problem. In *Algorithms and Data Structures: 17th International Symposium, WADS 2021*. Springer-Verlag, 2021.

A. Banik, B. B. Bhattacharya, and S. Das. Minimum enclosing circle of a set of fixed points and a mobile point. *Computational Geometry*, 2014.

J. Basch, L. J. Guibas, C. D. Silverstein, and L. Zhang. A practical evaluation of kinetic data structures. In *Proceedings of the thirteenth annual symposium on Computational geometry*, 1997.

T. M. Chan. Dynamic planar convex hull operations in near-logarithmic amortized time. In *40th Annual Symposium on Foundations of Computer Science*, 1999.

T. M. Chan. Approximating the diameter, width, smallest enclosing cylinder, and

minimum-width annulus. *International Journal of Computational Geometry & Applications*, 2002.

T. M. Chan. On levels in arrangements of curves. *Discrete & Computational Geometry*, 2003.

T. M. Chan. An optimal randomized algorithm for maximum tukey depth. In *Symposium on Discrete Algorithms (SODA)*, 2004.

T. M. Chan. On levels in arrangements of curves, ii: A simple inequality and its consequences. *Discrete & Computational Geometry*, 2005.

T. M. Chan. On levels in arrangements of curves, iii: further improvements. In *Proceedings of the twenty-fourth annual symposium on Computational geometry*, 2008.

M.-K. Chiu, S. Felsner, M. Scheucher, P. Schnider, R. Steiner, and P. Valtr. On the average complexity of the k-level. *arXiv preprint arXiv:1911.02408*, 2019.

S. Cicerone, A. Di Fonso, G. Di Stefano, and A. Navarra. Arbitrary pattern formation on infinite regular tessellation graphs. In *Proceedings of the 22nd International Conference on Distributed Computing and Networking*, 2021.

K. Clarkson. Algorithms for the minimum diameter of moving points and for the discrete 1-center problem. *Manuscript, http://cm.bell-labs.com/who/clarkson/moving_diam.html*, 1997.

R. Cole, M. Sharir, and C. K. Yap. On k-hulls and related problems. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, 1984.

T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms, fourth edition*. MIT Press, 2022.

M. de Berg, O. Cheong, M. v. Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag TELOS, 2008.

S. L. Devadoss and J. O'Rourke. *Discrete and Computational Geometry*. Princeton University Press, 2011.

T. K. Dey. Improved bounds on planar k-sets and k-levels. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*. IEEE, 1997.

T. K. Dey. Improved bounds for planar k-sets and related problems. *Discrete & Computational Geometry*, 1998.

H. Edelsbrunner and E. Welzl. Constructing belts in two-dimensional arrangements with applications. *SIAM Journal on Computing*, 1986.

M. Elbanhawi and M. Simic. Sampling-based robot motion planning: A review. *IEEE Access*, 2014.

L. Guibas. Kinetic data structures. In *Handbook of Data Structures and Applications*. Chapman and Hall/CRC, 2018.

L. Guibas, F. Xie, and L. Zhang. Kinetic collision detection: algorithms and experiments. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation*, 2001.

L. J. Guibas. Kinetic data structures—a state of the art report. In *Proc. Workshop Algorithmic Found. Robot*, 1998.

P. Gupta, R. Janardan, and M. Smid. Fast algorithms for collision and proximity problems involving moving geometric objects. *Computational Geometry*, 1996.

D. Halperin and M. Sharir. Arrangements. In *Handbook of discrete and computational geometry*. Chapman and Hall/CRC, 2017.

S. Har-Peled. A practical approach for computing the diameter of a point set. In *Proceedings of the Seventeenth Annual Symposium on Computational Geometry*, SCG 2001. Association for Computing Machinery, 2001.

S. Har-Peled, H. Kaplan, and M. Sharir. Approximating the k-level in three-dimensional plane arrangements. *A Journey Through Discrete Mathematics: A Tribute to Jiří Matoušek*, 2017.

B. Lévy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generation. *Seminal Graphics Papers: Pushing the Boundaries*, 2002.

L. Liu, L. Zhang, Y. Xu, C. Gotsman, and S. J. Gortler. A local/global approach to mesh parameterization. In *Computer graphics forum*. Wiley Online Library, 2008.

N. Megiddo. Linear-time algorithms for linear programming in R^3 and related problems. *SIAM journal on computing*, 1983.

W. Meulemans, K. Verbeek, and J. Wulms. Stability analysis of kinetic oriented bounding boxes. In *35th European Workshop on Computational Geometry (EuroCG 2019)*, 2019.

K. Mulmuley. *Computational Geometry: An Introduction Through Randomized Algorithms*. Pearson, 1993.

F. P. Preparata and M. I. Shamos. *Computational geometry: an introduction*. Springer Science & Business Media, 2012.

F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, 1985.

Z. Rahmati and A. Zarei. Kinetic euclidean minimum spanning tree in the plane. *Journal of Discrete Algorithms*, 2012. Selected papers from the 22nd International Workshop on Combinatorial Algorithms (IWOCA 2011).

E. A. Ramos. Deterministic algorithms for 3-d diameter and some 2-d lower envelopes. In *SCG*, 2000.

P. J. Schneider and D. Eberly. *Geometric Tools for Computer Graphics*. Elsevier Science Inc., 2002.

M. Sharir and P. K. Agarwal. *Davenport-Schinzel sequences and their geometric applications*. Cambridge university press, 1995.

M. Sharir and C. Ziv. On the complexity of the k-level in arrangements of pseudoplanes. *Discrete Mathematics*, 2021.

H. Tamaki and T. Tokuyama. How to cut pseudo-parabolas into segments. In *Proceedings of the eleventh annual symposium on Computational geometry*, 1995.

G. Tóth. Point sets with many k-sets. In *Proceedings of the sixteenth annual symposium on Computational geometry*, 2000.

G. Toussaint. Applications of the rotating calipers to geometric problems in two and three dimensions. *International Journal of Digital Information and Wireless Communications (IJDIWC)*, 2014.

G. T. Toussaint. Solving geometric problems with the rotating calipers. In *Proc. IEEE Melecon*, 1983.

N. Wachholz and S. Suri. Spanning tree, matching, and tsp for moving points: Complexity and regret. In *Proceedings of the 35th Canadian Conference on Computational Geometry (CCCG 2023)*, 2023.

E. Welzl. Smallest enclosing disks (balls and ellipsoids). In *New Results and New Trends in Computer Science: 1991 Proceedings*. Springer, 2005.