Faster algorithms for geometric clustering and competitive facility-location problems

Faster Algorithms for Geometric Clustering and Competitive Facility-Location Problems

Mehran Mehr

This research was supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 022.005.025.

The work in the thesis has been carried out under the auspices of the research school IPA (Institute for Programming research and Algorithmics).

A catalogue record is available from the Eindhoven University of Technology Library
ISBN 978-90-386-4625-1
Printed by Gildeprint, Enschede

Plants and fences in the cover: Designed by Brgfx / Freepik

The cover image is a visual allusion to the minimizing the sum of perimeters problem studied in Chapters 2 and 3 where the goal is to find some closed curves with minimum total length enclosing a given set of points. This problem and similar problems in this category are often referred to as fencing problems in the literature.

Faster Algorithms for Geometric Clustering and Competitive Facility-Location Problems

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
rector magnificus prof.dr.ir. F.P.T. Baaijens, voor een
commissie aangewezen door het College voor
Promoties, in het openbaar te verdedigen op
woensdag 5 december 2018 om 13:30 uur

door

Mehran Mehr

geboren te Urmia, Iran

Acknowledgments

I would like to express my sincere gratitude to my supervisor Mark de Berg. His support and care as an advisor, enthusiasim and passion as a researcher, and patience and kindness as a teacher made these four years a successful and pleasant experience for me. It was a great opportunity for me to collaborate with him. He devoted plenty of time to assist me in doing research, writing papers, and preparing presentations. I would also like to thank him for helping me to broaden my professional network by encouraging me to participate in conferences, schools, and to pay visit to prosperous universties in the field of theoretical computer science.

I also wish to thank the other members of my thesis committee, Kevin Buchin, Marc van Kreveld, Christian Knauer, Nikhil Bansal, and Frits Spieksma, for reviewing my thesis and coming to my defence. Especially, I appreciate having Kevin Buchin as my co-supervisor, colleague, collaborator, and coauthor. I would also like to thank all my coauthors and collaborators, Johan van Leeuwaarden, Joachim Gudmundsson, Mikkel Abrahamsen, Ali Mehrabi, Anna Adamaszek, Karl Bringmann, Vincent Cohen-Addad, Eva Rotenberg, Alan Roytman, Mikkel Thorup, Sándor Kisfaludi-Bak, Mahnaz Ghaffari, and Mohammad Ali Abam, for the time they devoted and for the intereseting results we obtained. I really enjoyed working with them. I would like to thank Mikkel Thorup, the then head of the Center for Efficient Algorithms and Data structures (EADS), and his colleagues in the University of Copenhagen once more for their hospitality and the valuable research and learning experience I had during my visit there.

Thanks to all my colleagues and friends in the department, Ali, Meivan, Wouter, Arthur, Maximilian, Quirijn, Jorn, Bart, Kevin, Herman, Bettina, Irina, Mahmoud, Aleks, Taher, Amir, Mahdi, Marek, Greg, Siddique, Mehdi, Sarmen, Tim, Kevin, Astrid, Sándor, Willem, Thom, Mikkel, Jules, Max, Morteza, Sudeshna, Mahnaz, Daniel, Ignaz, Marcel and many others who made Eindhoven a nice place to live and work. A special thanks goes to my officemates, Aleks, Astrid, and Sándor for a friendly atmosphere in our office. I would also like to thank Aleks for organizing many social events, movie nights, and Ali for being a roommate in many trips and a close friend.

Above all, thanks to my mother and sister for their patience and understanding, and to all my friends and relatives in Iran for their constant support.

Contents

Chapter 1

Introduction

The continuous increase in processing power and storage capacities enable individuals, companies, and organizations to collect and store more and more data. As a result, data analysis has become one of the main challenges nowadays. Geometric data analysis is a type of data analysis that deals with geometric data sets. Some data sets such as geographic data sets are geometric by themselves, and for some others the data elements can be viewed as points in a geometric space. Similarity between the data elements is then often measured by considering the distance between the points. Hence, geometric data analysis is important in many areas.

Cluster analysis is a fundamental task in data analysis that involves partitioning a given data set into subsets called *clusters*, such that similar elements end up in the same cluster (see Figure 1.1). Cluster analysis has many applications in areas such as market research, information retrieval, bio informatics, and pattern recognition. For example, market researchers use cluster analysis when working with multivariate data from surveys and test panels, to partition the general population of consumers into market segments.

The facility-location problem is a closely-related type of problem which is concerned with finding an optimal placement of a number of facilities that must serve a given collection of "demands". In the basic version of the problem, there is a cost associated with opening each facility and a cost associated with serving a demand by a certain facility; the latter is usually proportional to the demand's distance from the serving facility (which is typically simply the nearest facility). The goal is now to minimize the total cost of opening facilities and serving all demands. The facility-location problem can be viewed as a type of clustering problem, where the demands served by each facility form a cluster.

In the competitive version of the facility-location problem, different entities want to place facilities and each entity tries to maximize its market share (that is, the number of demands served by its facilities). This model has also been

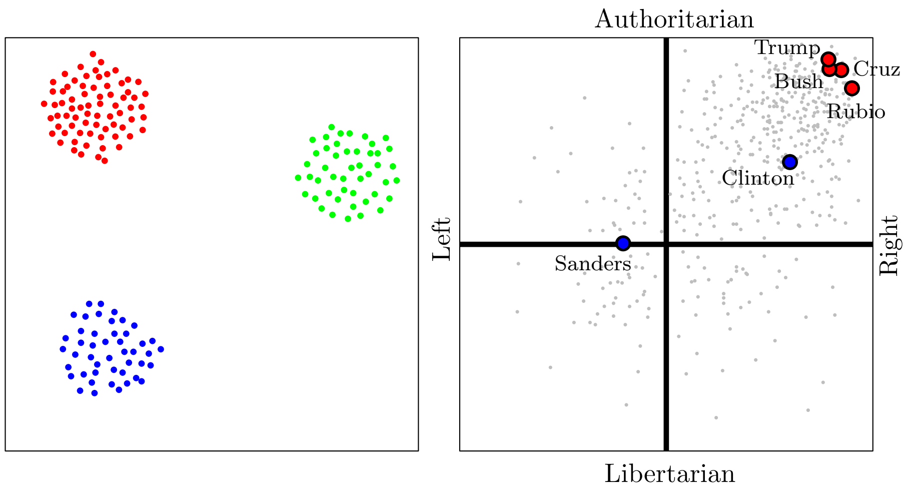


Figure 1.1: Example of a geometric data set consisting of three clusters (left), and the approximate political position of the candidates in the US presidential election, 2016 (right).

studied in social choice and voting theory. In an election, the opinion of the voters and the candidates can be represented by points in a Euclidean space where each axis represents an aspect of their political views (see[1] Figure 1.1). Assuming voters vote for the candidate nearest to their opinion, it is then interesting to see if there is a position that guarantees that a candidate would win against any (single) opponent. Such a position is called a plurality point; we will discuss it more extensively later in this chapter.

In this thesis, we consider several geometric variants of cluster analysis and competitive facility-location problems that can be formulated as optimization problems where we want to maximize profit or minimize a cost function. There are different approaches for solving optimization problems. In the field of machine learning, heuristic approaches are common. These approaches typically do not give any guarantees on the quality of the computed solutions. Another approach, which is common in theoretical computer science, is the algorithmic approach where the performance of an algorithm is measured based on its computational complexity and guaranteed accuracy. This is the approach we take in this thesis.

[1]The idea of the chart is taken from the website `http://www.politicalcompass.com/`, the position of the candidates and the voters are not accurate.

1.1 Geometric Clustering

The clustering problem is to partition a given data set into clusters (that is, subsets) according to some measure of optimality. Clustering problems can be categorized based on their cluster model. These models include:

- Centroid-based models: In these models, clusters are represented by an object (the cluster center), not necessarily a member of the data set, that plays the role of the center of that cluster. The goal is then to find a set of cluster centers that minimizes some cost function of the clusters.

- Connectivity-based (hierarchical) models: The idea of these models is to form clusters in a hierarchical way, where the decision which clusters to merge at each step depends on the distances between them. This results in a hierarchical structure, often called a dendrogram, where the leaves of the tree represent the objects in the data set and each internal node represents the cluster formed by merging the clusters of its children.

- Distribution-based models: In this approach, statistical distributions are used to model clusters. The underlying assumption is that those data points are samples from some (unknown) distribution.

- Density-based models: In these models, higher density areas of the data set are defined as clusters. Usually the objects in the sparse area are considered as noise or border points.

We are interested in clustering problems where the data set is a set S of points in Euclidean space, which are usually called *geometric clustering problems*. Many geometric clustering problems fall into one of two categories: problems where the maximum cost of a cluster is given and the goal is to find a clustering consisting of a minimum number of clusters, and problems where the number of clusters is given and the goal is to find a clustering of minimum total cost. In this thesis, we consider basic problems of the latter type.

More formally, we focus on clustering problems of the following type. Let S be a set of n points in $\mathbb{R}^d$, and let $k \geqslant 2$ be a natural number. A *k-clustering* of S is a partitioning $\mathcal{C}$ of S into at most k clusters. Let $\Phi(\mathcal{C})$ denote the *cost* of $\mathcal{C}$. The goal is now to find a clustering $\mathcal{C}$ that minimizes $\Phi(\mathcal{C})$. Many well-known geometric clustering problems are of this type. A well known example is the k-center problem. In the *Euclidean k-center problem*, $\Phi(\mathcal{C})$ is the maximum cost of any of the clusters $C \in \mathcal{C}$, where the cost of C is the radius of its smallest enclosing ball. Hence, in the Euclidean k-center problem we want to cover the point set S by k congruent balls of minimum radius. The k-center problem, including the important special case of the 2-center problem, has been studied extensively (e.g. [5, 23, 38, 49, 50, 65]). The k-means and k-medians problems are two other clustering problems that fit in this framework. In the k-means problem, $\Phi(\mathcal{C})$ is the sum of the squares of the distance of each point to the mean of the points in its cluster. And in the k-medians problem, $\Phi(\mathscr{C})$ is the sum of the distances of each point to the median of the points in its cluster.

The *minimum perimeter-sum problem* is another clustering problem that fits in this framework. This problem was already studied by Capoyleas, Rote, and Woeginger [22] and Arkin, Khuller, and Mitchell [10] in the early 90s. Given a set S of n points in the plane, we aim at finding a set of at most k closed curves such that (1) each point is enclosed by a curve and (2) the total length of the curves is minimized. Note that the shortest curve surrounding a given set of points is always the convex hull of those points. Hence, we can think of this as dividing the points into k clusters $\mathscr{C}$ and then $\Phi(\mathscr{C})$ is defined as the sum of perimeters of the convex hulls of the clusters. For this reason, we refer to this variant as *minimizing the sum of perimeters*. Below this variant, which is one of the problems we will focus on in this thesis, will be discussed in more detail.

An important special case of k-clustering problems (which we will also study in this thesis) is when $k = 2$. When $k = 2$, a clustering problem in the aforementioned framework reduces to the problem of finding a bipartition (C_1, C_2) of a point set S. Bipartition problems are not only interesting in their own right, but also because bipartition algorithms can form the basis of hierarchical clustering methods. There are many possible variants of the bipartition problem on planar point sets, which differ in how the cost of a clustering is defined. A variant that received a lot of attention is the 2-center problem [23, 34, 38, 52, 65], where as stated earlier the cost of a partition (C_1, C_2) of the given point set S is defined as the maximum of the radii of the smallest enclosing disks of C_1 and C_2. Other cost functions that have been studied include the maximum diameter of the two point sets [12] and the sum of the diameters [48]; see also the survey by Agarwal and Sharir [6] for some more variants.

1.1.1 Minimizing the Sum of Perimeters

In Chapters 2 and 3, we study the problem of partitioning a planar point set into clusters in order to minimize the sum of the perimeters of the clusters, which was already mentioned briefly above. Next we discuss the previous work on this problem. We start with the case $k = 2$ and then discuss the general case. After discussing the previous work, we summarize our results on the problem.[2]

As mentioned, the minimum perimeter-sum problem asks for minimizing the sum of the perimeters of the convex hulls of the clusters. (The perimeter of $\text{CH}(C_i)$, the convex hull of cluster C_i, is the length of the boundary $\partial\,\text{CH}(C_i)$.) One can also define the size of the convex hull (which determined the cost of the cluster) in a different way. For the cases $k = 2$, this class of cost functions

[2]Throughout the text, we assume that we are working in the real-RAM model, so that it is possible to compare the costs of two different clusterings exactly (even though this might involve comparing sums of square roots). Note that this is a standard assumption in computational geometry.

was already studied in 1991 by Mitchell and Wynters [59]. They studied four problem variants: minimize the sum of the perimeters, the maximum of the perimeters, the sum of the areas, or the maximum of the areas. In three of the four variants the convex hulls $\text{CH}(C_1)$ and $\text{CH}(C_2)$ in an optimal solution may intersect [59, full version]—only in the *minimum perimeter-sum problem* the optimal bipartition is guaranteed to be a so-called *line partition*, that is, a solution with disjoint convex hulls. For each of the four variants they gave an $O(n^3)$ algorithm that uses $O(n)$ storage and that computes an optimal line partition; for all except the minimum area-maximum problem they also gave an $O(n^2)$ algorithm that uses $O(n^2)$ storage. Note that (only) for the minimum perimeter-sum problem the computed solution is an optimal bipartition. Mitchell and Wynters mentioned the improvement of the space requirement of the quadratic-time algorithm as an open problem, and they stated the existence of a subquadratic algorithm for any of the four variants as the most prominent open problem.

Rokne *et al.* [62] made progress on the first question, by presenting an $O(n^2 \log n)$ algorithm that uses only $O(n)$ space for the line-partition version of each of the four problems. Devillers and Katz [32] gave algorithms for the min-max variant of the problem, both for area and perimeter, which run in $O((n+k)\log^2 n)$ time. Here k is a parameter that is only known to be in $O(n^2)$, although Devillers and Katz suspected that k is subquadratic. They also gave linear-time algorithms for these problems when the point set P is in convex position and given in cyclic order. Segal [64] proved an $\Omega(n \log n)$ lower bound for the min-max problems. Very recently, and apparently unaware of some of the earlier work on these problems, Bae *et al.* [13] presented an $O(n^2 \log n)$ time algorithm for the minimum-perimeter-sum problem and an $O(n^4 \log n)$ time algorithm for the minimum-area-sum problem (considering all partitions, not only line partitions). Despite these efforts, the main question is still open: is it possible to obtain a subquadratic algorithm for any of the four bipartition problems based on convex-hull size?

We now turn our attention to the general case, $k > 2$. Many classical clustering problems are NP-hard when k is given as part of the input, though there are some notable exceptions. In 2012, Gibson et al. [43] devised a polynomial time algorithm for finding k disks, each centered at a point in S, such that the sum of the radii of the disks is minimized subject to the constraint that their union must cover S. In their paper, they used a dynamic-programming approach to get a running time of $O(n^{881})$. In another work, Behsaz and Salavatipour [19] studied the objective of minimizing the sum of diameters subject to the constraint of having at most k clusters. They gave a polynomial time approximation scheme for points in the plane.

In early 90s, Capoyleas, Rote, and Woeginger [22] presented an algorithm for the minimum perimeter-sum problem that runs in time $\rho(k)n^{O(k)}$, where $\rho(k)$ is the number of nonisomorphic planar graphs on k nodes. This yields a polynomial running time when k is fixed. Arkin *et al.* [11] studied the same problem and gave a similar algorithm. Arkin *et al.* [10] also conjectured the problem to be NP-hard when k is part of the input, but neither an NP-hardness proof nor a polynomial time algorithm has been found so far.

Our results. We improve the previous results for both the general case of the minimum perimeter-sum problem and its bipartition version.

The bipartition case. We answer the question above, about the existence of a subquadratic algorithm for the bipartition problems, affirmatively by presenting a subquadratic algorithm for the minimum perimeter-sum bipartition problem in the plane.

As mentioned, an optimal solution (C_1, C_2) to the minimum perimeter-sum bipartition problem must be a line partition. A straightforward algorithm would generate all $\Theta(n^2)$ line partitions and compute the value $\text{per}(C_1) + \text{per}(C_2)$ for each of them. If the latter is done from scratch for each partition, the resulting algorithm runs in $O(n^3 \log n)$ time. The algorithms by Mitchell and Wynters [59] and Rokne *et al.* [62] improve on this by using the fact that the different line bipartitions can be generated in an ordered way, such that subsequent line partitions differ in at most one point. Thus the convex hulls do not have to be recomputed from scratch, but they can be obtained by updating the convex hulls of the previous bipartition. To obtain a subquadratic algorithm a fundamentally new approach is necessary: we need a strategy that generates a subquadratic number of candidate partitions, instead of considering all line partitions. We achieve this as follows.

We start by proving that an optimal bipartition (C_1, C_2) has the following property: either there is a set of $O(1)$ canonical orientations such that C_1 can be separated from C_2 by a line with a canonical orientation, or the distance between $\text{CH}(C_1)$ and $\text{CH}(C_2)$ is $\Omega(\min(\text{per}(C_1), \text{per}(C_2))$. There are only $O(1)$ bipartitions of the former type, and finding the best among them is relatively easy. The bipartitions of the second type are much more challenging. We show how to employ a compressed quadtree to generate a collection of $O(n)$ canonical 5-gons—intersections of axis-parallel rectangles and canonical halfplanes—such that the smaller of $\text{CH}(C_1)$ and $\text{CH}(C_2)$ (in a bipartition of the second type) is contained in one of the 5-gons.

It then remains to find the best among the bipartitions of the second type. Even though the number of such bipartitions is linear, we cannot afford to compute their perimeters from scratch. We therefore design a data structure to quickly compute $\text{per}(S \cap Q)$, where Q is a query canonical 5-gon. Brass *et al.* [21] presented such a data structure for the case where Q is an axis-parallel rectangle. Their structure uses $O(n \log^2 n)$ space and has $O(\log^5 n)$ query time; it can be extended to handle canonical 5-gons as queries, at the cost of increasing the space usage to $O(n \log^3 n)$ and the query time to $O(\log^7 n)$. Our data structure improves upon this: it has $O(\log^4 n)$ query time for canonical 5-gons (and $O(\log^3 n)$ for rectangles) while using the same amount of space. Using this data structure to find the best bipartition of the second type we obtain our main result: an exact algorithm for the minimum perimeter-sum bipartition problem that runs in $O(n \log^4 n)$ time. As our model of computation we use the real RAM (with the capability of taking square roots) so that we can compute the exact perimeter of a convex polygon—this is necessary to compare the costs of two competing clusterings. We furthermore make the (standard) assumption that the model of computation allows us to compute a compressed quadtree of n points in $O(n \log n)$ time; see footnote 3 in Section 2.1.2.

Besides our exact algorithm, we present a linear-time $(1+\varepsilon)$-approximation algorithm. Its running time is $O(n + T(1/\varepsilon^2)) = O(n + 1/\varepsilon^2 \cdot \log^4(1/\varepsilon))$, where $T(1/\varepsilon^2)$ is the running time of an exact algorithm on an instance of size $1/\varepsilon^2$.

The general case. For the minimum perimeter-sum problem, we present an algorithm that is polynomial in both n and k (Theorem 3.25). In particular, this refutes the conjectured hardness by Arkin *et al.* [10].

Our algorithm for the minimum perimeter-sum problem shares some similarities with the work of Gibson et al. [43], in which a dynamic-programming approach was used to solve the minimum radius sum problem. One main difference is that the running time complexity they obtain is $O(n^{881})$, whereas our approach works in $O(n^{27})$ time. Their technique is to divide a given instance (S, k) of the minimum radius sum problem into subproblems, where S is the set of points to be clustered and k is a constraint on the number of clusters we can use. The problem (S, k) is solvable if we have a solution to all subproblems. However, the number of subproblems is exponential. To get a polynomial time algorithm, they showed that a solution can be found after considering only a polynomial number of subproblems.

We also use a dynamic-programming approach for our problem, although we need new techniques. Observe that the number of candidate clusters for the minimum radius sum problem is only $O(n^2)$ (as each disk is determined by two points in S, one determining the center and the other determining the radius). In contrast, our problem has an exponential number of candidate clusters (dictated by all subsets of S). We define subproblems based on *boxes*, which are rectangles that cover some portion of the plane and some number of input points from S. Our key observation is that there is some separator of each box (i.e., a vertical line segment or a horizontal line segment) that splits the box into two strictly smaller boxes such that an optimal solution only has a constant number of line segments that intersect this separator (in fact, we give a bound of two on the number of such intersecting line segments).

A dynamic-programming approach then naturally follows: we simply guess the position of such a separator (for which we argue there are $O(n)$ choices), and then guess which segments crossing this separator belong to an optimal solution. We first obtain solutions for smaller boxes, and then glue together solutions for smaller boxes to obtain solutions for larger boxes.

1.1.2 A Generic Method for Finding Coresets for Clustering Problems

After studying exact algorithms for the minimum perimeter-sum problem, we turn our attention in Chapter 4 to approximate solutions for a rather general class of cost functions, which we call *regular costs functions*.

Our results. Our main result is a general method to find an ε-coreset for such clustering problems. Given a set of points S, a cost function Φ, the number of clusters k, a value $\varepsilon > 0$, an ε-coreset is a set $R \subseteq S$ such that $\Phi(\mathcal{C}'_{\text{opt}}) \geqslant (1-\varepsilon)\cdot\Phi(\mathcal{C}_{\text{opt}})$, where $\mathcal{C}'_{\text{opt}}$ is an optimal clustering for R and $\mathcal{C}_{\text{opt}}$ is an optimal clustering for S.

We find an ε-coreset of size $O(k\left(f(k)/\varepsilon\right)^d)$ for a set of points S in $\mathbb{R}^d$, where $f(k)$ is a function that only depends on the number of clusters. The algorithm runs in linear time for constant k, in the appropriate model of computation. This is similar to the approach taken by Har-Peled and Mazumdar [47], who solve the approximate k-means and k-median problem efficiently by generating a coreset of size $O((k/\varepsilon^d)\cdot\log n)$.

Our method applies to a large class of clustering problems including the k-center problem in any L_p-metric, variants of the k-center problem where we want to minimize the sum (rather than maximum) of the cluster radii, and the 2-dimensional problem where we want to minimize the maximum or sum of the perimeters of the clusters.

1.2 Competitive Facility Location

As mentioned earlier, facility-location problems are closely related to (centroid-based) clustering problems. Voronoi games and plurality points are two related competitive facility-location problems that we consider in this thesis.

Voronoi games. Voronoi games, as introduced by Ahn *et al.* [7], can be viewed as competitive facility-location problems in which two players $\mathcal{P}$ and $\mathcal{Q}$ want to place their facilities in order to maximize their market area. The Voronoi game of Ahn *et al.* is played in a bounded region $R \subset \mathbb{R}^2$, and the facilities of the players are modeled as points in this region. Each player gets the same number, k, of facilities, which they have to place alternatingly. The market area of $\mathcal{P}$ (and similarly of $\mathcal{Q}$) is now given by the area of the region of all points $q \in R$ whose closest facility was placed by $\mathcal{P}$, that is, it is the total area of the Voronoi cells of $\mathcal{P}$'s facilities in the Voronoi diagram of the facilities of $\mathcal{P}$ and $\mathcal{Q}$. Ahn *et al.* proved that for $k > 1$ and when the region R is a circle or a segment, the second player can win the game by a payoff of $1/2+\varepsilon$, for some $\varepsilon > 0$, where the first player can ensure ε is arbitrarily small.

The one-round Voronoi game introduced by Cheong *et al.* [26] is similar to the Voronoi game of Ahn *et al.*, except that the first player must first place all his k facilities, after which the second player places all her k facilities. They considered the problem where R is a square, and they showed that when k is large enough the first player can always win a fraction $1/2+\alpha$ of the area of R for some $\alpha > 0$. Fekete and Meijer [40] considered the problem on a rectangle R of aspect ratio $\rho \leqslant 1$. They showed that the first player wins more than half the area of R, unless $k \geqslant 3$ and $\rho > \sqrt{2}/n$, or $k=2$ and $\rho > \sqrt{3}/2$. They also showed that if R is a polygon with holes, then computing the locations of the facilities for the second player that maximize the area she wins, against a given set of facilities of the first player is NP-hard.

One-round discrete Voronoi games. In this thesis, we are interested in *discrete (Euclidean) one-round Voronoi games*, where the players do not compete for area but for a discrete set of points. That is, instead of the region R one is given a set V of n points in a geometric space, and a point $v \in V$ is won by the player owning the facility closest to v. (Another discrete variant of Voronoi games is played on graphs [66, 68] but we restrict our attention to the geometric variant.) More formally, the problem we study is defined as follows.

Let V be a multiset of n points in $\mathbb{R}^d$, which we call *voters* from now on, and let $k \geqslant 1$ and $\ell \geqslant 1$ be two integers. The one-round discrete Voronoi game defined by the triple $\langle V, K, \ell\rangle$ is a single-turn game played between two players $\mathscr{P}$ and $\mathscr{Q}$. First, player $\mathscr{P}$ places a set P of k points in $\mathbb{R}^d$, then player $\mathscr{Q}$ places a set Q of ℓ points in $\mathbb{R}^d$. (These points may coincide with the voters in V.) We call the set P the *strategy of* $\mathscr{P}$ and the set Q the *strategy of* $\mathscr{Q}$. Player $\mathscr{P}$ wins a voter $v \in V$ if $\text{dist}(v,P) \leqslant \text{dist}(v,Q)$, where $\text{dist}(v,P)$ and $\text{dist}(v,Q)$ denote the minimum distance between a voter v and the sets P and Q, respectively. Note that this definition favors player $\mathscr{P}$, since in case of a tie a voter is won by $\mathscr{P}$. We now define $V[P \succcurlyeq Q] := \{v \in V : \text{dist}(v,P) \leqslant \text{dist}(v,Q)\}$ to be the multiset of voters won by player $\mathscr{P}$ when he uses strategy P and player $\mathscr{Q}$ uses strategy Q. Player $\mathscr{P}$ wins the game $\langle V, K, \ell\rangle$ if he wins at least half the voters in V, that is, when $\big|V[P \succcurlyeq Q]\big| \geqslant n/2$; otherwise $\mathscr{Q}$ wins the game. Here $\big|V[P \succcurlyeq Q]\big|$ denotes the size of the multiset $V[P \succcurlyeq Q]$ (counting multiplicities). We now define $\Gamma_{k,\ell}(V)$ as the maximum number of voters that can be won by player $\mathscr{P}$ against an optimal opponent:

$$\Gamma_{k,\ell}(V) := \max_{P\subset\mathbb{R}^d,\,|P|=k} \ \min_{Q\subset\mathbb{R}^d,\,|Q|=\ell} \big|V[P \succcurlyeq Q]\big|.$$

For a given multiset V of voters, we want to decide if[3] $\Gamma_{k,\ell}(V) \geqslant n/2$. In other words, we are interested in determining for a given game $\langle V, K, \ell\rangle$ if $\mathscr{P}$ has a *winning strategy*, which is a set of k points such that $\mathscr{P}$ wins the game no matter where $\mathscr{Q}$ places her points.

An important special case, which has already been studied in spatial voting theory for a long time, is when $k=\ell=1$ [57]. Here the coordinates of a point in

[3]One can also require that $\Gamma_{k,\ell}(V) > n/2$; with some small modifications, all the results in this thesis can be applied to the case with strict inequality as well.

V represent the preference of the voter on certain topics, and the point played by $\mathscr{Q}$ represents a certain proposal. If the point played by $\mathscr{P}$ wins against all possible points played by $\mathscr{Q}$, then the $\mathscr{P}$'s proposal will win the vote against any other proposal. Note that in the problem definition we gave above, voters at equal distance from P and Q are won by $\mathscr{P}$, and $\mathscr{P}$ has to win at least half the voters. This is the definition typically used in papers of Voronoi games [14–17]. In voting theory other variants are studied as well, for instance where points at equal distance to P and Q are not won by either of them, and $\mathscr{P}$ wins the game if he wins more voters than $\mathscr{Q}$; see the paper by McKelvey and Wendell [57] who use the term *majority points* for the former variant and the term *plurality points* for the latter variant. Next we discuss the latter variant in more detail.

Plurality points. In social choice and voting theory, the concept of plurality point is defined as follows. Let V be a set of *n voters* and let $\mathscr{C}$ be a space of possible choices. Each voter $v \in V$ has a utility function indicating how much v likes a certain choice, i.e. the utility function of v determines for any two choices from $\mathscr{C}$ which one is preferred by v or whether both choices are equally preferable. A (weak) plurality point is now defined as a choice $p \in \mathscr{C}$ such that no alternative $p' \in \mathscr{C}$ is preferred by more voters.

When there are different issues on which the voters can decide, then the space $\mathscr{C}$ becomes a multi-dimensional space. This has led to the study of plurality points in the setting where $\mathscr{C} = \mathbb{R}^d$ and each voter has an ideal choice which is a point in $\mathbb{R}^d$. To simplify the presentation, from now on we will not distinguish the voters from their ideal choice and so we view each voter $v \in V$ as being a point in $\mathbb{R}^d$, the so-called *spatial model* in voting theory [57]. Thus the utility of a point $p \in \mathbb{R}^d$ for a voter v is inversely proportional to $\text{dist}(v, p)$, the distance from v to p under a given distance function, and v prefers a point p over a point p' if $\text{dist}(v, p) < \text{dist}(v, p')$. Now a point $p \in \mathbb{R}^d$ is a plurality point if for any point $p' \in \mathbb{R}^d$ we have $|\{v \in V : \text{dist}(v, p) < \text{dist}(v, p')\}| \geqslant |\{v \in V : \text{dist}(v, p') < \text{dist}(v, p)\}|$.

Plurality points and related concepts were already studied in the 1970s in voting theory [37, 44, 45, 57, 61, 70]. McKelvey and Wendell [57] define three different notions of plurality points—majority Condorcet, plurality Condorcet, and majority core—and for each notion they define a weak and a strong variant. Under certain assumptions on the utility functions, which are satisfied for the L_2 norm, the three notions are equivalent. Thus for the L_2 norm we only have two variants: weak plurality points (which should be at least as popular as any alternative) and strong plurality points (which should be strictly more popular than any alternative). We focus on weak plurality points, since they are more challenging from an algorithmic point of view. From now on, whenever we speak of plurality points we refer to weak plurality points.

Plurality points represent a stable choice with respect to the opinions of the voters. One can also look at the concept from the viewpoint of competitive facility location. Here one player wants to place a facility in the space $\mathscr{C}$ such that she always wins at least as many clients (voters) as her competitor, no matter where the competitor places his facility. Competitive facility-location problems have been studied widely in a discrete setting, where the clients and the possible locations for the facilities are nodes in a network; see the survey by Kress and Pesch [54]. We focus on the geometric setting.

Previous work on discrete Voronoi games and plurality points. Besides algorithmic problems concerning the one-round discrete Voronoi game one can also consider combinatorial problems. In particular, one can ask for bounds on $\Gamma_{k,\ell}(V)$ as a function of n, k, and ℓ. It is known that for any set V in $\mathbb{R}^2$ and $k = \ell = 1$ we have $\lfloor n/3 \rfloor \leqslant \Gamma_{1,1}(V) \leqslant \lceil n/2 \rceil$. This result is based on known bounds for maximum Tukey depth, where the lower bound can be proven using Helly's theorem. It is also known [17] that there is a constant c such that $k = c\ell$ points suffice for $\mathscr{P}$ to win the game, that is, $\Gamma_{c\ell,\ell}(V) \geqslant n/2$ for any V.

In this thesis, we focus on the algorithmic problem of computing $\Gamma_{k,\ell}(V)$ for given V, k, and ℓ. Some existing algorithmic results are for the setting where the players already placed all but one of their points, and one wants to compute the best locations for the last point of $\mathscr{P}$ and of $\mathscr{Q}$. Banik *et al.* [16] gave algorithms that find the best location for $\mathscr{P}$ in $O(n^8)$ time and for $\mathscr{Q}$ in $O(n^2)$ time. For the two-round variant of the problem, with $k = \ell = 2$, polynomial algorithms for finding the optimal strategies of both players are also known [15].

Banik *et al.* [14] studied the problem of computing $\Gamma_{k,\ell}(V)$ in $\mathbb{R}^1$. They considered the case of arbitrarily large k and ℓ, but where $k = \ell$ (and V is a set instead of a multiset). For this case they showed that depending on the set V either $\mathscr{P}$ or $\mathscr{Q}$ can win the game, and they presented an algorithm to compute $\Gamma_{k,\ell}(V)$ in time $O(n^{k-\lambda_k})$, where $0 < \lambda_k < 1$ is a constant dependent only on k. This raises the question: is the problem NP-hard when k is part of the input?

When the L_2 norm defines the distance between voters and potential plurality points, then plurality points can be defined in terms of Tukey depth [69]. The *Tukey depth* of a point $p \in \mathbb{R}^d$ with respect to a given set V of n points is defined as the minimum number of points from V lying in any closed halfspace containing p. A point of maximum Tukey depth is called a *Tukey median*. It is known that for any set V, the depth of the Tukey median is at least $\lceil n/(d+1) \rceil$ and at most $\lceil n/2 \rceil$. Wu *et al.* [72] showed that a point $p \in \mathbb{R}^d$ is a plurality point in the L_2 norm if and only if any open halfspace with p on its boundary contains at most $n/2$ voters. This is equivalent to saying that the Tukey depth of p is $\lceil n/2 \rceil$. They used this observation to present an algorithm that decides in $O(n^{d-1}\log n)$ time if a plurality point exists for a given set V of n voters in $\mathbb{R}^d$. A slightly better result can be obtained using a randomized algorithm by Chan [24], which computes a Tukey median (together with its depth) in $O(n\log n + n^{d-1})$ time.

As is clear from the relation to Tukey depth, a plurality point in the L_2 norm does not always exist. In fact, the set V of voters must, in a certain sense, be highly symmetric to admit a plurality point. For example, when the number of voters is even, any line containing a plurality point p must contain the same number of voters on either side of p. This led Lin *et al.* [56] to study the *minimum-cost plurality problem*. Here each voter is assigned a cost, and the goal is to find a minimum-cost subset $W \subset V$ of voters such that if we ignore the voters in W—that is, if we consider $V \setminus W$—then a plurality point exists. Lin *et al.* gave an $O(n^5 \log n)$ algorithm for the planar version of the problem; whether the problem in $\mathbb{R}^3$ can be solved in polynomial time was left as an open problem.

In the voting-theory literature plurality points in the L_1 norm have also been considered [57, 70, 71]. One advantage of the L_1 norm is that in $\mathbb{R}^2$ a plurality point always exists and can easily be found in $O(n)$ time: any point (p_x, p_y) such that p_x and p_y are medians of the multisets of x-coordinates and y-coordinates of the voters in V, respectively, is a plurality point. Unfortunately, this is no longer true when $d > 2$ [57, 70]. We are not aware of any existing algorithms for deciding whether a given set V in $\mathbb{R}^d$ admits a plurality point in the L_1 norm.

Our results. We improved the existing results for both plurality point and discrete Voronoi game problems.

Plurality points. Currently the fastest algorithm for deciding whether a plurality point exists runs in $O(n \log n + n^{d-1})$ randomized time and actually computes a Tukey median. However, in the case of plurality points we are only interested in the Tukey median if its depth is the maximum possible, namely $\lceil n/2 \rceil$. Wu *et al.* [72] exploited this to obtain a deterministic algorithm, but their running time is $O(n^{d-1} \log n)$. This raises the question: can we decide whether a plurality point exists faster than by computing the depth of the Tukey median? We show that this is indeed possible: we present a deterministic algorithm that decides if a plurality point exists (and, if so, computes one) in $O(n \log n)$ time.[4]

We then turn our attention to the minimum-cost plurality problem. We solve the open problem of Lin *et al.* [56] by presenting an algorithm that solves the problem in $O(n^4)$ time. Note that this even improves on the $O(n^5 \log n)$ running time for the planar case. We also consider the following problem for unit-cost voters in $\mathbb{R}^d$: given a parameter k, find a minimum-cost set W of size at most k such that $V \setminus W$ admits a plurality point, if such a set exists. Our algorithm for this case runs in $O(k^3 n \log n)$ time when $d = 2$ and in $O(k^5 \log k + k^3 n \log n)$ expected time when $d > 2$.

Ignoring some voters in order to have a plurality point is undesirable when almost all voters must be ignored. Instead of ignoring voters we can work with *plurality balls*, as defined next. The idea is that if two points p and p' are very similar, then voters do not care much whether p or p' is chosen. Thus we define a ball $b(p, r)$ centered at p and of radius r to be a plurality ball if the

[4] Here and in the other bounds stated in the introduction we assume the dimension d is a fixed constant, to make it easier to compare our results to earlier work. In the theorems stated later in Chapter 5, we also analyze the dependency on d.

following holds: there is no point p' outside $b(p,r)$ that is preferred by more voters than p. Note that a plurality point is a plurality ball of zero radius. We show that in the plane, the minimum-radius plurality ball can be computed in $O(T(n))$ time, where $T(n)$ is the time needed to compute the $\lfloor n/2 \rfloor$-level in an arrangement of n lines.

Recall that the different dimensions represent different issues on which the voters can express their preferences. It is then natural to allow the voters to give different weights to these issues. This leads us to introduce what we call the *personalized L_1 norm*. Here each voter $v \in V$ has a *preference vector* $\langle w_1(v), \ldots, w_d(v) \rangle$ of non-negative weights that specifies the relative importance of the various issues. The distance of a point $p \in \mathbb{R}^d$ to a voter v is now defined as $\mathrm{dist}_{\mathrm{w}}(v,p) := \sum_{i=1}^{d} w_i(v) \cdot |x_i(v) - x_i(p)|$, where $x_i(\cdot)$ denotes the i-th coordinate of a point. We present an algorithm that decides in $O(n^{d-1})$ time whether a set V of n voters admits a plurality point with respect to the personalized L_1 norm. For the special case when all preference vectors are identical—this case reduces to the normal case of using the L_1 norm—the running time improves to $O(n)$.

Voronoi games. We present an algorithm that computes $\Gamma_{k,\ell}(V)$ in $\mathbb{R}^1$ in polynomial time. Our algorithm works when V is a multiset, and it does not require k and ℓ to be equal. Our algorithm computes $\Gamma_{k,\ell}(V)$ and finds a strategy for $\mathscr{P}$ that wins this many voters in time $O(kn^4)$. The algorithm can be extended to the case when the voters are weighted, at a slight increase in running time.

The algorithm by Banik *et al.* [14] discretizes the problem, by defining a finite set of potential locations for $\mathscr{P}$ to place his points. However, to ensure an optimal strategy for $\mathscr{P}$, the set of potential locations has exponential size. To overcome this problem we need several new ideas. First of all, we essentially partition the possible strategies into various classes such that for each class we can anticipate the behavior of the optimal strategy for $\mathscr{Q}$. To compute the best strategy within a certain class we use dynamic programming, in a non-standard (and, unfortunately, rather complicated) way. The subproblems in our dynamic-program are for smaller points sets and smaller values of k and ℓ (actually we will need several other parameters) where the goal of $\mathscr{P}$ will be to push his rightmost point as far to the right as possible to win a certain number of points. One complication in the dynamic program is that it is unclear which small subproblems I' can be used to solve a given subproblem I. The opposite direction—determining for I' which larger subproblems I may use I' in their solution—is easier however, so we use a sweep approach: when the solution to some I' has been determined, we update the solution to larger subproblems I that can use I'.

After establishing that we can compute $\Gamma_{k,\ell}(V)$ in polynomial time in $\mathbb{R}^1$, we turn to the higher-dimensional problem. We show that deciding if $\mathscr{P}$ has a winning strategy is Σ_2^P-hard in $\mathbb{R}^2$. We also show that for fixed k and ℓ this problem can be solved in polynomial time. Our solution combines algebraic methods [18] with a previously known result that one can construct a polynomial-size boolean formula for the majority function using a special kind of sorting network, called the AKS sorting network [8]. The latter result is essential to avoid the appearance of n in the exponent. As a byproduct of the algebraic method, we show that the problem is contained in the complexity class $\exists\forall\mathbb{R}$; see [33] for more information on this complexity class.

1.3 Publications and Outline of the Thesis

Chapter 2 describes the results for the bipartion case of the minimum perimeter-sum problem and is based on the following publication

> Mikkel Abrahamsen, Mark de Berg, Kevin Buchin, Mehran Mehr, and Ali D. Mehrabi. Minimum perimeter-sum partitions in the plane. In *Proceedings of the 33rd International Symposium on Computational Geometry*, pages 4:1–4:15, 2017

Chapter 3 describes the results for the generic case of the minimum perimeter-sum problem and is based on (parts of)

> Mikkel Abrahamsen, Anna Adamaszek, Karl Bringmann, Vincent Cohen-Addad, Mehran Mehr, Eva Rotenberg, Alan Roytman, and Mikkel Thorup. Fast fencing. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing*, pages 564–573, 2018

Chapter 4 describes the results for finding coresets for clustering problems and is based on

> Mikkel Abrahamsen, Mark de Berg, Kevin Buchin, Mehran Mehr, and Ali D. Mehrabi. A generic method for finding coresets for clustering problems. In *Abstracts of European Workshop on Computational Geometry*, pages 249–252, 2017

This chapter also includes two lemmas from the following joint work

> Mikkel Abrahamsen, Mark de Berg, Kevin Buchin, Mehran Mehr, and Ali D. Mehrabi. Range-clustering queries. In *Proceedings of the 33rd International Symposium on Computational Geometry*, pages 5:1–5:16, 2017

Chapter 5 describes the results for plurality points and is based on

> Mark de Berg, Joachim Gudmundsson, and Mehran Mehr. Faster algorithms for computing plurality points. *ACM Transactions on Algorithms*, 14(3):36:1–36:23, 2018

Chapter 6 describes the results for the Voronoi game and is based on

> Mark de Berg, Sándor Kisfaludi-Bak, and Mehran Mehr. On one-round discrete voronoi games. Manuscript, 2018

Chapter 2

Minimizing the Sum of Perimeters: The Bipartition Case

In this chapter, we consider the bipartition version of the minimum perimeter-sum problem. Here, a set P of points in the plane is given and the goal is to find a bipartition (P_1, P_2) of the set P such that the sum of perimeters of the convex hulls $\text{CH}(P_1)$ and $\text{CH}(P_2)$, i.e. $\text{per}(\text{CH}(P_1)) + \text{per}(\text{CH}(P_2))$, is minimized. We show that a subquadratic algorithm for this problem exists by presenting an algorithm that solves this problem in $O(n \log^4 n)$ time. Besides our exact algorithm, we present a linear-time $(1+\varepsilon)$-approximation algorithm. Its running time is $O(n + T(1/\varepsilon^2)) = O(n + 1/\varepsilon^2 \cdot \log^4(1/\varepsilon))$, where $T(1/\varepsilon^2)$ is the running time of an exact algorithm on an instance of size $1/\varepsilon^2$.

2.1 The Exact Algorithm

In this section we present an exact algorithm for the minimum-perimeter-sum partition problem. We first prove a separation property that an optimal solution must satisfy, and then we show how to use this property to develop a fast algorithm.

Let P be the set of n points in the plane for which we want to solve the minimum-perimeter-sum partition problem. An optimal partition (P_1, P_2) of P has the following two basic properties: P_1 and P_2 are non-empty, and the convex hulls $\text{CH}(P_1)$ and $\text{CH}(P_2)$ are disjoint [59, full version]. In the remainder, whenever we talk about a partition of P, we refer to a partition with these two properties.

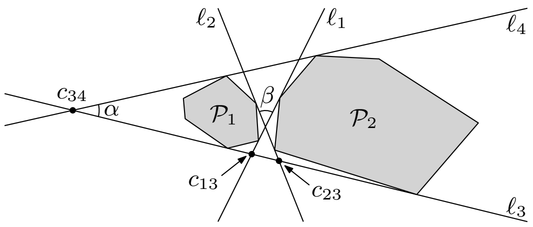


Figure 2.1: The angles α and β.

2.1.1 Geometric Properties of an Optimal Partition

Consider a partition (P_1, P_2) of P. Define $\mathcal{P}_1 := \text{CH}(P_1)$ and $\mathcal{P}_2 := \text{CH}(P_2)$ to be the convex hulls of P_1 and P_2, respectively, and let ℓ_1 and ℓ_2 be the two inner common tangents of $\mathcal{P}_1$ and $\mathcal{P}_2$. The lines ℓ_1 and ℓ_2 define four wedges: one containing P_1, one containing P_2, and two empty wedges. We call the opening angle of the empty wedges the *separation angle* of P_1 and P_2. Furthermore, we call the distance between $\mathcal{P}_1$ and $\mathcal{P}_2$ the *separation distance* of P_1 and P_2.

Theorem 2.1. *Let P be a set of n points in the plane, and let (P_1, P_2) be a partition of P that minimizes* $\text{per}(P_1) + \text{per}(P_2)$*. Then the separation angle of P_1 and P_2 is at least $\pi/6$ or the separation distance is at least* $c_{\text{sep}} \cdot \min(\text{per}(P_1), \text{per}(P_2))$*, where* $c_{\text{sep}} := 1/250$.

The remainder of this section is devoted to proving Theorem 2.1. To this end let (P_1, P_2) be a partition of P that minimizes $\text{per}(P_1) + \text{per}(P_2)$. Let ℓ_3 and ℓ_4 be the outer common tangents of $\mathcal{P}_1$ and $\mathcal{P}_2$. We define α to be the angle between ℓ_3 and ℓ_4. More precisely, if ℓ_3 and ℓ_4 are parallel we define $\alpha := 0$, otherwise we define α as the opening angle of the wedge defined by ℓ_3 and ℓ_4 containing $\mathcal{P}_1$ and $\mathcal{P}_2$. We denote the separation angle of P_1 and P_2 by β; see Fig. 2.1.

The idea of the proof is as follows. Suppose that the separation distance and the separation angle β are both relatively small. Then the region A in between $\mathcal{P}_1$ and $\mathcal{P}_2$ and bounded from the bottom by ℓ_3 and from the top by ℓ_4 is relatively narrow. But then the left and right parts of ∂A (which are contained in $\partial \mathcal{P}_1$ and $\partial \mathcal{P}_2$) would be longer than the bottom and top parts of ∂A (which are contained in ℓ_3 and ℓ_4), thus contradicting the assumption that (P_1, P_2) is an optimal partition. To make this idea precise, we first prove that if the separation angle β is small, then the angle α between ℓ_3 and ℓ_4 must be large. Second, we show that there is a value $f(\alpha)$ such that the distance between $\mathcal{P}_1$ and $\mathcal{P}_2$ is at least $f(\alpha) \cdot \min(\text{per}(P_1), \text{per}(P_2))$. Finally we argue that this implies that if the separation angle is smaller than $\pi/6$, then (to avoid the contradiction mentioned above) the separation distance must be relatively large. Next we present our proof in detail.

Let c_{ij} be the intersection point between ℓ_i and ℓ_j, where $i < j$. If ℓ_3 and ℓ_4 are parallel, we choose c_{34} as a point at infinity on ℓ_3. Assume without loss of generality that neither ℓ_1 nor ℓ_2 separate $\mathscr{P}_1$ from c_{34}, and that ℓ_3 is the outer common tangent such that $\mathscr{P}_1$ and $\mathscr{P}_2$ are to the left of ℓ_3 when traversing ℓ_3 from c_{34} to an intersection point in $\ell_3 \cap \mathscr{P}_1$. Assume furthermore that c_{13} is closer to c_{34} than c_{23}.

For two lines, rays, or segments r_1, r_2, let $\angle(r_1, r_2)$ be the angle we need to rotate r_1 in a counterclockwise direction until r_1 and r_2 are parallel. For three points a, b, c, let $\angle(a, b, c) := \angle(ba, bc)$. For $i = 1, 2$ and $j = 1, 2, 3, 4$, let s_{ij} be a point in $P_i \cap \ell_j$. Let $\partial\mathscr{P}_i$ denote the boundary of $\mathscr{P}_i$ and $\mathrm{per}(\mathscr{P}_i)$ the perimeter of $\mathscr{P}_i$. Furthermore, let $\partial\mathscr{P}_i(x, y)$ denote the portion of $\partial\mathscr{P}_i$ from $x \in \partial\mathscr{P}_i$ counterclockwise to $y \in \partial\mathscr{P}_i$, and $\mathrm{length}(\partial\mathscr{P}_i(x, y))$ denote the length of $\partial\mathscr{P}_i(x, y)$.

Lemma 2.2. *Let p_0 and q be points and $\mathbf{v}$ be a unit vector. Let $p(t) := p_0 + t \cdot \mathbf{v}$ and $d(t) := |p(t)q|$ and assume that $p(t) \neq q$ for all $t \in \mathbb{R}$. Then $d'(t) = \cos(\angle(q, p(t), p(t) + \mathbf{v}))$ if the points $q, p(t), p(t) + \mathbf{v}$ make a left-turn and $d'(t) = -\cos(\angle(q, p(t), p(t) + \mathbf{v}))$ otherwise.*[1]

Proof. We prove the lemma for an arbitrary value $t = t_0$. By reparameterizing p, we may assume that $t_0 = 0$. Furthermore, by changing the coordinate system, we can without loss of generality assume that $p_0 = (0, 0)$ and $q = (x, 0)$ for some value $x > 0$.

Let $\phi := \angle((x, 0), (0, 0), \mathbf{v})$. Assume that $\mathbf{v}$ has positive y-coordinate—the case that $\mathbf{v}$ has negative y-coordinate can be handled analogously. We have proven the lemma if we manage to show that $d'(0) = -\cos\phi$. Note that since $\mathbf{v}$ has positive y-coordinate, we have $p(t) = (t\cos\phi, t\sin\phi)$ for every $t \in \mathbb{R}$. Hence

$$d(t) = \sqrt{\left(t\cos\phi - x\right)^2 + t^2\sin^2\phi}.$$

and

$$d'(t) = \frac{t - x\cos\phi}{\sqrt{t^2 - 2tx\cos\phi + x^2}}.$$

Evaluating in $t = 0$, we get

$$d'(0) = -\frac{x\cos\phi}{|x|} = -\cos\phi,$$

where the last equality follows since $x > 0$. $\square$

Lemma 2.3. *We have $\alpha + 3\beta \geqslant \pi$.*

[1]Note that $\angle(q, p(t), p(t) + \mathbf{v}) = \angle(q, p(t), p(t) - \mathbf{v})$ by the definition of $\angle(\cdot, \cdot, \cdot)$ which is the reason that there are two cases in the lemma.

Proof. Since $\mathrm{per}(\mathscr{P}_1)+\mathrm{per}(\mathscr{P}_2)$ is minimum, we know that

$$\mathrm{length}(\partial\mathscr{P}_1(s_{13},s_{14}))+\mathrm{length}(\partial\mathscr{P}_2(s_{24},s_{23})) \leqslant \Psi,$$

where $\Psi := |s_{13}s_{23}|+|s_{14}s_{24}|$. Furthermore, we know that $s_{11},s_{12}\in\partial\mathscr{P}_1(s_{13},s_{14})$ and $s_{21},s_{22}\in\partial\mathscr{P}_1(s_{24},s_{23})$. We thus have

$$\mathrm{length}(\partial\mathscr{P}_1(s_{13},s_{14}))+\mathrm{length}(\partial\mathscr{P}_2(s_{24},s_{23})) \geqslant \Phi,$$

where $\Phi := |s_{13}s_{11}|+|s_{11}s_{12}|+|s_{12}s_{14}|+|s_{24}s_{21}|+|s_{21}s_{22}|+|s_{22}s_{23}|$. Hence, we must have

$$\Phi \leqslant \Psi. \tag{2.1}$$

Now assume that $\alpha+3\beta<\pi$. We will show that this assumption, together with inequality (2.1), leads to a contradiction, thus proving the lemma. To this end we will argue that if (2.1) holds, then there exist points s'_{ij} for $i=1,2$ and $j=1,2,3,4$, where s'_{ij} is a point on ℓ_j, with the following properties:

(i) $\Phi'\leqslant\Psi'$, where Φ' and Ψ' are defined as Φ and Ψ when each point s_{ij} is replaced by s'_{ij},

(ii) s'_{21} or s'_{22} coincides with c_{12}, and

(iii) s'_{11} or s'_{12} coincides with c_{12}.

Note that the point s'_{ij} is not required to be contained in P_i. In particular, the points s'_{13} and s'_{14} will in some cases be on the other side of c_{34} than the points s_{13} and s_{14}.

To finish the proof it then suffices to observe that properties (i)–(iii) together contradict the triangle inequality.

To prove the existence of the points s'_{ij} with the claimed properties, we initially define $s'_{ij}:=s_{ij}$, so that property (i) is satisfied. Then we will move the points s'_{ij} (where each s'_{ij} moves on ℓ_j) such that property (i) is preserved throughout the movements and properties (ii) and (iii) are satisfied at the end of the movements.

We first show how to create a situation where (ii) holds, and (i) still holds as well. Let $\gamma_{ij}:=\angle(\ell_i,\ell_j)$. We consider two cases.

- *Case (A): $\gamma_{32}<\pi-\beta$.*

  We observe that moving s'_{23} along ℓ_3 away from s'_{13} increases Ψ' more than it increases Φ', so property (i) is preserved by such a movement. Note that $\angle(xs'_{23},\ell_2)\geqslant\gamma_{32}$ for any $x\in s'_{22}c_{12}$. However, by moving s'_{23} sufficiently far away we can make $\angle(xs'_{23},\ell_2)$ arbitrarily close to γ_{32}. We therefore move s'_{23} so far away that $\angle(xs'_{23},\ell_2)<\pi-\beta$ for any point $x\in s'_{22}c_{12}$. We now

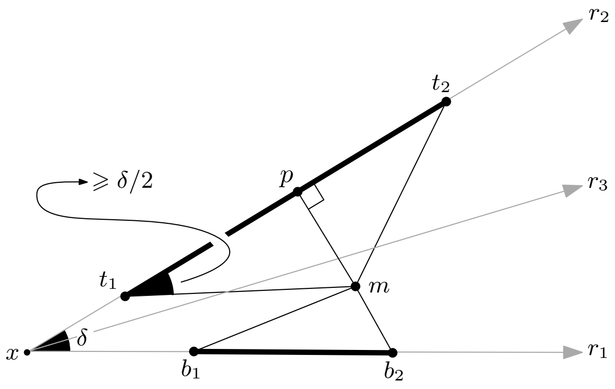


Figure 2.2: Illustration for Lemma 2.4. Φ is the total length of the four segments $t_1 m$, $t_2 m$, $b_1 m$, $b_2 m$, and Ψ is the total length of the two fat segments.

consider what happens as we let a point x move at unit speed from s'_{22} towards c_{12}. To be more precise, let $T := |s'_{22} c_{12}|$, let $\mathbf{v}$ be the unit vector with direction from c_{23} to c_{12}, and for any $t \in [0,T]$ define $x(t) := s'_{22} + t \cdot \mathbf{v}$. Note that $x(0) = s'_{22}$ and $x(T) = c_{12}$.

Let $a(t) := |x(t)s'_{23}|$ and $b(t) := |x(t)s'_{21}|$. Lemma 2.2 gives that

$$a'(t) = -\cos(\angle(x(t)s'_{23}, \ell_2)) \text{ and } b'(t) = \cos(\angle(\ell_2, x(t)s'_{21})).$$

Since $\angle(x(t)s'_{23}, \ell_2) < \pi - \beta$ for any value $t \in [0,T]$, we get $a'(t) < -\cos(\pi - \beta)$. Furthermore, we have $\angle(\ell_2, x(t)s'_{21}) \geqslant \pi - \beta$ and hence $b'(t) \leqslant \cos(\pi - \beta)$. Therefore, $a'(t) + b'(t) < 0$ for any t and we conclude that $a(T) + b(T) \leqslant a(0) + b(0)$. This is the same as $|s'_{21} c_{12}| + |c_{12} s'_{23}| \leqslant |s'_{21} s'_{22}| + |s'_{22} s'_{23}|$, so we now move s'_{22} to c_{12} and are ensured that (i) still holds.

- *Case (B):* $\gamma_{32} \geqslant \pi - \beta$.

  Using our assumption $\alpha + 3\beta < \pi$ we get $\gamma_{32} > \alpha + 2\beta$. Note that $\gamma_{14} = \pi - \gamma_{32} + \alpha + \beta$. Hence, $\gamma_{14} < \pi - \beta$. By first moving s'_{24} away from s'_{14} and then s'_{21} towards c_{12}, we can in a similar way as in Case (A) argue that we can reach a situation where (i) still holds and s'_{21} coincides with c_{12}.

We conclude that in both cases we can ensure (ii) without violating (i).

Since $\gamma_{13} \leqslant \gamma_{14}$ and $\gamma_{42} \leqslant \gamma_{32}$, we likewise have $\gamma_{13} < \pi - \beta$ or $\gamma_{42} < \pi - \beta$. Hence, by first moving s'_{13} or s'_{14} and since then s'_{11} or s'_{12}, we can in a similar way reach a situation where s'_{11} or s'_{12} coincides with c_{12} without violating (i), thus ensuring (iii) and finishing the proof. □

The following lemma is illustrated in Fig. 2.2.

Lemma 2.4. *Let x be a point and r_1 and r_2 be two rays starting at x such that $\angle(r_1, r_2) = \delta$, and assume that $\delta \leqslant \pi$. Let $b_1, b_2 \in r_1$ and $t_1, t_2 \in r_2$ be such that $b_1 \in xb_2$ and $t_1 \in xt_2$, and let m be a point in the wedge bounded by r_1 and r_2. Then*

$$\Phi - \Psi \geqslant \frac{(1-\cos(\delta/2))\cdot\sin(\delta/2)}{1+\sin(\delta/2)} \cdot (|b_1m| + |t_1m|),$$

where $\Phi := |b_1m| + |t_1m| + |b_2m| + |t_2m|$ and $\Psi := |b_1b_2| + |t_1t_2|$.

Proof. First note that

$$|b_1m| + |b_2m| \geqslant |b_1b_2| \tag{2.2}$$

and

$$|t_1m| + |t_2m| \geqslant |t_1t_2|. \tag{2.3}$$

Let r_3 be the angular bisector of r_1 and r_2. Assume without loss of generality that m lies in the wedge defined by r_1 and r_3. Then $\angle(m, t_1, t_2) \geqslant \delta/2$.

We now consider two cases.

- *Case (A):* $|t_1m| \geqslant \frac{\sin(\delta/2)}{1+\sin(\delta/2)} \cdot (|b_1m| + |t_1m|)$.

  Our first step is to prove that

  $$|t_1m| + |t_2m| - |t_1t_2| \geqslant (1-\cos(\delta/2)) \cdot |t_1m|. \tag{2.4}$$

  Let p be the orthogonal projection of m on r_2. Note that $|t_2m| \geqslant |t_2p|$. Consider first the case that p is on the same side of t_1 as x. In this case $|t_2p| \geqslant |t_1t_2|$ and therefore

  $$|t_1m| + |t_2m| - |t_1t_2| \geqslant |t_1m| \geqslant (1-\cos(\delta/2)) \cdot |t_1m|,$$

  which proves (2.4).

  Assume now that p is on the same side of t_1 as t_2. In this case, we have $\angle(m, t_1, t_2) \leqslant \pi/2$ and thus $|t_1p| = \cos(\angle(m, t_1, t_2)) \cdot |t_1m| \leqslant \cos(\delta/2) \cdot |t_1m|$. Hence we have

  $$\begin{aligned} |t_1m| + |t_2m| - |t_1t_2| &\geqslant |t_1m| + |t_2p| - (|t_1p| + |t_2p|) \\ &\geqslant (1-\cos(\delta/2)) \cdot |t_1m|, \end{aligned}$$

  and we have proved (2.4).

  We now have

  $$\begin{aligned} \Phi - \Psi &= |b_1m| + |t_1m| + |b_2m| + |t_2m| - |b_1b_2| - |t_1t_2| \\ &\geqslant |b_1m| + |b_2m| - |b_1b_2| + (1-\cos(\delta/2)) \cdot |t_1m| \quad \text{by (2.4)} \\ &\geqslant (1-\cos(\delta/2)) \cdot \tfrac{\sin(\delta/2)}{1+\sin(\delta/2)} \cdot (|b_1m| + |t_1m|) \quad \text{by (2.2)} \end{aligned}$$

  where the last step uses that we are in Case (A). Thus the lemma holds in Case (A).

- *Case (B):* $|t_1 m| < \frac{\sin(\delta/2)}{1+\sin(\delta/2)} \cdot (|b_1 m| + |t_1 m|)$.

  The condition for this case can be rewritten as

$$|b_1 m| \; > \; \frac{1}{1+\sin\delta/2} \cdot (|b_1 m| + |t_1 m|). \tag{2.5}$$

  To prove the lemma in this case we first argue that $\angle(b_2, b_1, m) > \pi/2$. To this end, assume for a contradiction that $\angle(b_2, b_1, m) \leqslant \pi/2$. It is easy to verify that for a given length of $t_1 m$ (and assuming $\angle(b_2, b_1, m) \leqslant \pi/2$), the fraction $|b_1 m|/(|b_1 m| + |t_1 m|)$ is maximized when segment $t_1 m$ is perpendicular to r_2, and $m \in r_3$, and $b_1 = x$. But then

$$\frac{|b_1 m|}{|b_1 m| + |t_1 m|} \leqslant \frac{1}{1+\sin\delta/2},$$

  which would contradict (2.5). Thus we indeed have $\angle(b_2, b_1, m) > \pi/2$. Hence, $|b_2 m| \geqslant |b_1 b_2|$, and so $|b_1 m| + |b_2 m| - |b_1 b_2| \geqslant |b_1 m|$. We can now derive

$$\begin{array}{rcll} \Phi - \Psi & = & |b_1 m| + |t_1 m| + |b_2 m| + |t_2 m| - |b_1 b_2| - |t_1 t_2| & \\ & \geqslant & |b_1 m| + |t_1 m| + |t_2 m| - |t_1 t_2| & \text{by the above} \\ & \geqslant & \frac{1}{1+\sin\delta/2} \cdot \big(|b_1 m| + |t_1 m|\big) & \text{by (2.3) and (2.5)} \\ & \geqslant & \big(\sin(\delta/2) \cdot (1 - \cos(\delta/2))\big) \cdot \frac{1}{1+\sin\delta/2} \cdot \big(|b_1 m| + |t_1 m|\big) & \end{array}$$

  Thus the lemma also holds in Case (B).

□

Let $\text{dist}(\mathscr{P}_1, \mathscr{P}_2) := \min_{(p,q) \in \mathscr{P}_1 \times \mathscr{P}_2} |pq|$ denote the separation distance between $\mathscr{P}_1$ and $\mathscr{P}_2$. Recall that α denotes the angle between the two common outer tangents of $\mathscr{P}_1$ and $\mathscr{P}_2$; see Fig. 2.1.

Lemma 2.5. *We have*

$$\textit{dist}(\mathscr{P}_1, \mathscr{P}_2) \; \geqslant \; f(\alpha) \cdot \text{per}(\mathscr{P}_1), \tag{2.6}$$

where $f\colon [0, \pi] \longrightarrow \mathbb{R}$ *is the increasing function*

$$f(\varphi) \; := \; \frac{\sin(\varphi/4)}{1+\sin(\varphi/4)} \cdot \frac{\sin(\varphi/2)}{1+\sin(\varphi/2)} \cdot \frac{1-\cos(\varphi/4)}{2}.$$

Proof. The statement is trivial if $\alpha = 0$ so assume $\alpha > 0$. Let $p \in \mathscr{P}_1$ and $q \in \mathscr{P}_2$ be points so that $|pq| = \text{dist}(\mathscr{P}_1, \mathscr{P}_2)$ and assume without loss of generality that pq is a horizontal segment with p being its left endpoint. Let ℓ_1^{vert} and ℓ_2^{vert} be vertical lines containing p and q, respectively. Note that $\mathscr{P}_1$ is in the closed half-plane to the left of ℓ_1^{vert} and $\mathscr{P}_2$ is in the closed half-plane to the right of ℓ_2^{vert}. Recall that s_{ij} denotes a point on $\partial\mathscr{P}_i \cap \ell_j$.

Claim: There exist two convex polygons $\mathscr{P}_1'$ and $\mathscr{P}_2'$ satisfying the following conditions:

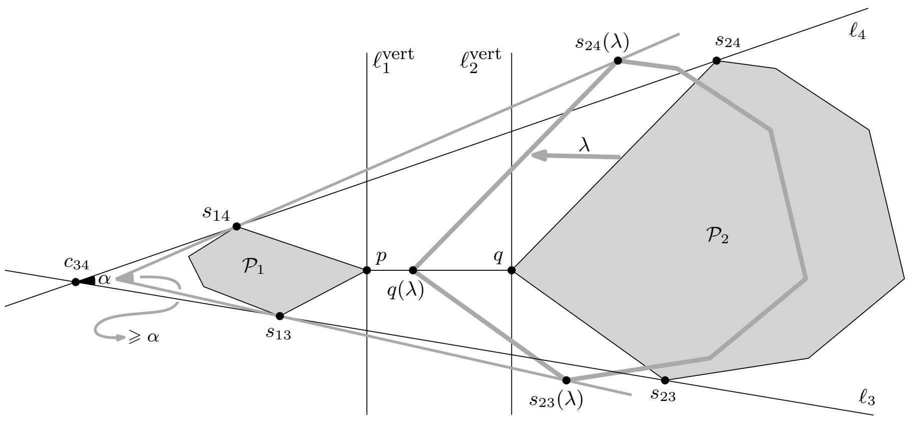


Figure 2.3: Illustration for the proof of Lemma 2.5.

1. $\mathscr{P}_1'$ and $\mathscr{P}_2'$ have the same outer common tangents as $\mathscr{P}_1$ and $\mathscr{P}_2$, namely ℓ_3 and ℓ_4.
2. $\mathscr{P}_1'$ is to the left of ℓ_1^{vert} and $p \in \partial\mathscr{P}_1'$; and $\mathscr{P}_2'$ is to right of ℓ_2^{vert} and $q \in \partial\mathscr{P}_2'$.
3. $\text{per}(\mathscr{P}_1') = \text{per}(\mathscr{P}_1)$.
4. $\text{per}(\mathscr{P}_1') + \text{per}(\mathscr{P}_2') \leqslant \text{per}(\text{CH}(\mathscr{P}_1' \cup \mathscr{P}_2'))$.
5. There are points $s_{ij}' \in \mathscr{P}_i' \cap \ell_j$ for all $i \in \{1,2\}$ and $j \in \{3,4\}$ such that $\partial\mathscr{P}_1'(s_{13}', p)$, $\partial\mathscr{P}_1'(p, s_{14}')$, $\partial\mathscr{P}_2'(s_{24}', q)$, and $\partial\mathscr{P}_2'(q, s_{23}')$ each consist of a single line segment.
6. Let $s_{2j}'(\lambda) := s_{2j}' - (\lambda, 0)$ and let $\ell_j'(\lambda)$ be the line through s_{1j}' and $s_{2j}'(\lambda)$ for $j \in \{3,4\}$. Then $\angle(\ell_3'(|pq|), \ell_4'(|pq|)) \geqslant \alpha/2$.

Proof of the claim. Let $\mathscr{P}_1' := \mathscr{P}_1$ and $\mathscr{P}_2' := \mathscr{P}_2$, and let s_{ij}' be a point in $\mathscr{P}_i' \cap \ell_j$ for all $i \in \{1,2\}$ and $j \in \{3,4\}$. We show how to modify $\mathscr{P}_1'$ and $\mathscr{P}_2'$ until they have all the required conditions. Of course, they already satisfy conditions 1–4. We first show how to obtain condition 5, namely that $\partial\mathscr{P}_1'(s_{13}', p)$ and $\partial\mathscr{P}_1'(p, s_{14}')$—and similarly $\partial\mathscr{P}_2'(s_{24}', q)$ and $\partial\mathscr{P}_1'(q, s_{23}')$—each consist of a single line segment, as depicted in Fig. 2.3. To this end, let v_{ij} be the intersection point $\ell_i^{\text{vert}} \cap \ell_j$ for $i \in \{1,2\}$ and $j \in \{3,4\}$. Let $s' \in s_{14}'v_{14}$ be the point such that $\text{length}(\partial\mathscr{P}_1'(p, s_{14}')) = |ps'| + |s's_{14}'|$. Such a point exists since

$$|ps_{14}'| \leqslant \text{length}(\partial\mathscr{P}_1'(p, s_{14}')) \leqslant |pv_{14}| + |v_{14}s_{14}'|.$$

We modify $\mathscr{P}_1'$ by replacing $\partial\mathscr{P}_1'(p, s_{14}')$ by the segments ps' and $s's_{14}'$. We can now redefine $s_{14}' := s'$ so that $\partial\mathscr{P}_1'(p, s_{14}') = ps_{14}'$ is a line segment. We can modify

$\mathscr{P}_1'$ in a similar way to ensure that $\partial\mathscr{P}_1'(s_{13}', p) = s_{13}'p$, and we can modify $\mathscr{P}_2'$ to ensure $\partial\mathscr{P}_2'(s_{24}', q) = s_{24}'q$ and $\partial\mathscr{P}_2'(q, s_{23}') = qs_{23}'$. Note that these modifications preserve conditions 1–4 and that condition 5 is now satisfied.

The only condition that $(\mathscr{P}_1', \mathscr{P}_2')$ might not satisfy is condition 6. Let $s_{2j}'(\lambda) := s_{2j}' - (\lambda, 0)$ and let $\ell_j(\lambda)$ be the line through $s_{2j}'(\lambda)$ and s_{1j}' for $j \in \{3,4\}$. Clearly, if the slopes of ℓ_3 and ℓ_4 have different signs (as in Fig. 2.3), the angle $\angle(\ell_3(\lambda), \ell_4(\lambda))$ is increasing for $\lambda \in [0, |pq|]$, and condition 6 is satisfied. However, if the slopes of ℓ_3 and ℓ_4 have the same sign, the angle might decrease.

Consider the case where both slopes are positive—the other case is analogous. Changing $\mathscr{P}_2'$ by replacing $\partial\mathscr{P}_2'(s_{23}', s_{24}')$ by the line segment $s_{23}'s_{24}'$ makes the sum $\text{per}(\mathscr{P}_1') + \text{per}(\mathscr{P}_2')$ and $\text{per}(\text{CH}(\mathscr{P}_1' \cup \mathscr{P}_2'))$ decrease equally much and hence condition 4 is preserved. This clearly has no influence on the other conditions. We thus assume that $\mathscr{P}_2'$ is the triangle $qs_{23}'s_{24}'$. Consider what happens if we move s_{23}' along the line ℓ_3 away from c_{34} with unit speed. Then $|s_{13}'s_{23}'|$ grows with speed exactly 1 whereas $|qs_{23}'|$ grows with speed at most 1. We therefore preserve condition 4, and the other conditions are likewise not affected.

We now move s_{23}' sufficiently far away so that $\angle(\ell_3, \ell_3(|pq|)) \leqslant \alpha/4$. Similarly, we move s_{24}' sufficiently far away from c_{34} along ℓ_4 to ensure that $\angle(\ell_4, \ell_4(|pq|)) \leqslant \alpha/4$. It then follows that $\angle(\ell_3(|pq|), \ell_4(|pq|)) \geqslant \angle(\ell_3, \ell_4) - \alpha/2 = \alpha/2$, and condition 6 is satisfied. $\square$

Note that condition 2 in the claim implies that $\text{dist}(\mathscr{P}_1', \mathscr{P}_2') = \text{dist}(\mathscr{P}_1, \mathscr{P}_2) = |pq|$, and hence inequality (2.6) follows from condition 3 if we manage to prove $\text{dist}(\mathscr{P}_1', \mathscr{P}_2') \geqslant f(\alpha)\cdot\text{per}(\mathscr{P}_1')$. Therefore, with a slight abuse of notation, we assume from now on that $\mathscr{P}_1$ and $\mathscr{P}_2$ satisfy the conditions in the claim, where the points s_{ij} play the role as s_{ij}' in conditions 5 and 6.

We now consider a copy of $\mathscr{P}_2$ that is translated horizontally to the left over a distance λ; see Fig. 2.3. Let $s_{24}(\lambda)$, $s_{23}(\lambda)$, and $q(\lambda)$ be the translated copies of s_{24}, s_{23}, and q, respectively, and let $\ell_j(\lambda)$ be the line through s_{1j} and $s_{2j}(\lambda)$ for $j \in \{3,4\}$. Furthermore, define

$$\Phi(\lambda) \;:=\; |s_{13}p| + |s_{14}p| + |s_{23}(\lambda)q(\lambda)| + |s_{24}(\lambda)q(\lambda)|$$

and

$$\Psi(\lambda) \;:=\; |s_{13}s_{23}(\lambda)| + |s_{14}s_{24}(\lambda)|.$$

Note that $\Phi(\lambda) = \Phi$ is constant. By conditions 4 and 5, we know that

$$\Phi \;\leqslant\; \Psi(0). \tag{2.7}$$

Note that $q(|pq|) = p$. We now apply Lemma 2.4 to get

$$\Phi - \Psi(|pq|) \;\geqslant\; \sin(\delta/2)\cdot\frac{1-\cos(\delta/2)}{1+\sin(\delta/2)}\cdot(|s_{13}p| + |s_{14}p|), \tag{2.8}$$

where $\delta := \angle(\ell_3(|pq|), \ell_4(|pq|))$. By condition 6, we know that $\delta \geqslant \alpha/2$. The function $\delta \longmapsto \sin(\delta/2) \cdot \frac{1-\cos(\delta/2)}{1+\sin(\delta/2)}$ is increasing for $\delta \in [0,\pi]$ and hence inequality (2.8) also holds for $\delta = \alpha/2$.

When λ increases from 0 to $|pq|$ with unit speed, the value $\Psi(\lambda)$ decreases with speed at most 2, i.e., $\Psi(\lambda) \geqslant \Psi(0) - 2\lambda$. Using this and inequalities (2.7) and (2.8), we get

$$2|pq| \;\geqslant\; \Psi(0) - \Psi(|pq|) \;\geqslant\; \Phi - \Phi + \sin(\alpha/4) \cdot \frac{1-\cos(\alpha/4)}{1+\sin(\alpha/4)} \cdot (|s_{13}p| + |s_{14}p|),$$

and we conclude that

$$|pq| \;\geqslant\; \frac{1}{2} \cdot \sin(\alpha/4) \cdot \frac{1-\cos(\alpha/4)}{1+\sin(\alpha/4)} \cdot (|s_{13}p| + |s_{14}p|). \tag{2.9}$$

By the triangle inequality, $|s_{13}p| + |s_{14}p| \geqslant |s_{13}s_{14}|$. Furthermore, for a given length of $s_{13}s_{14}$, the fraction $|s_{13}s_{14}|/(|s_{14}c_{34}| + |c_{34}s_{13}|)$ is minimized when $s_{13}s_{14}$ is perpendicular to the angular bisector of ℓ_3 and ℓ_4. (Recall that c_{34} is the intersection point of the outer common tangents ℓ_3 and ℓ_4; see Fig. 2.3.) Hence

$$|s_{13}s_{14}| \;\geqslant\; \sin(\alpha/2) \cdot (|s_{14}c_{34}| + |c_{34}s_{13}|). \tag{2.10}$$

We now conclude

$$\begin{aligned}
|s_{13}p| + |s_{14}p| &= \tfrac{\sin(\alpha/2)}{1+\sin(\alpha/2)} \cdot \left(\tfrac{|s_{13}p|+|s_{14}p|}{\sin(\alpha/2)} + |s_{13}p| + |s_{14}p| \right) \\
&\geqslant \tfrac{\sin(\alpha/2)}{1+\sin(\alpha/2)} \cdot \left(\tfrac{|s_{13}s_{14}|}{\sin(\alpha/2)} + |s_{13}p| + |s_{14}p| \right) \quad \text{triangle inequality} \\
&\geqslant \tfrac{\sin(\alpha/2)}{1+\sin(\alpha/2)} \cdot \left(|s_{14}c_{34}| + |c_{34}s_{13}| + |s_{13}p| + |s_{14}p| \right) \quad \text{by (2.10)} \\
&\geqslant \tfrac{\sin(\alpha/2)}{1+\sin(\alpha/2)} \cdot \operatorname{per}(\mathscr{P}_1),
\end{aligned}$$

where the last inequality follows because $\mathscr{P}_1$ is fully contained in the quadrilateral $s_{14}, c_{34}, x_{13}, p$. The statement (2.6) in the lemma now follows from (2.9). □

We are now ready to prove Theorem 2.1.

Proof of Theorem 2.1. If the separation angle of P_1 and P_2 is at least $\pi/6$, we are done. Otherwise, Lemma 2.3 gives that $\alpha > \pi/2$, and Lemma 2.5 gives that $\operatorname{dist}(\mathscr{P}_1, \mathscr{P}_2) \;\geqslant\; f(\pi/2) \cdot \operatorname{per}(\mathscr{P}_1) \geqslant (1/250) \cdot \min(\operatorname{per}(\mathscr{P}_1), \operatorname{per}(\mathscr{P}_2))$. □

2.1.2 The Algorithm

Theorem 2.1 suggests to distinguish two cases when computing an optimal partition: the case where the separation angle is large (namely at least $\pi/6$) and the case where the separation distance is large (namely at least $c_{\text{sep}} \cdot \min(\operatorname{per}(P_1), \operatorname{per}(P_2))$). As we will see, the first case can be handled in $O(n \log n)$ time and the second case in $O(n \log^4 n)$ time, leading to the following theorem.

Theorem 2.6. *Let P be a set of n points in the plane. Then we can compute a partition (P_1,P_2) of P that minimizes* $\mathrm{per}(P_1)+\mathrm{per}(P_2)$ *in $O(n\log^4 n)$ time using $O(n\log^3 n)$ space.*

The Best Partition with Large Separation Angle

Define the *orientation* of a line ℓ, denoted by $\phi(\ell)$, to be the counterclockwise angle that ℓ makes with the positive y-axis. If the separation angle of P_1 and P_2 is at least $\pi/6$, then there must be a line ℓ separating P_1 from P_2 that does not contain any point from P and such that $\phi(\ell) = j\cdot\pi/7$ for some $j \in \{0,1,\dots,6\}$. For each of these seven orientations we can compute the best partition in $O(n\log n)$ time, as explained next.

Without loss of generality, consider separating lines ℓ with $\phi(\ell) = 0$, that is, vertical separating lines. Let X be the set of all x-coordinates of the points in P. For any x-value $x \in X$ define $P_1(x) := \{p \in P \mid p_x \leqslant x\}$, where p_x denotes the x-coordinate of a point p, and define $P_2(x) := P \setminus P_1(x)$. Our task is to find the best partition of the form $(P_1(x),P_2(x))$ over all $x \in X$. To this end we first compute the values $\mathrm{per}(P_1(x))$ for all $x \in X$ in $O(n\log n)$ time in total, as follows. We compute the lengths of the upper hulls of the point sets $P_1(x)$, for all $x \in X$, using Graham's scan [28], and we compute the lengths of the lower hulls in a second scan. (Graham's scan goes over the points from left to right and maintains the upper (or lower) hull of the encountered points; it is trivial to extend the algorithm so that it also maintains the length of the hull.) By combining the lengths of the upper and lower hulls, we get the values $\mathrm{per}(P_1(x))$.

Computing the values $\mathrm{per}(P_2(x))$ can be done similarly, after which we can easily find the best partition of the form $(P_1(x),P_2(x))$ in $O(n)$ time. Thus the best partition with large separation angle can be found in $O(n\log n)$ time.

The Best Partition with Large Separation Distance

Next we show how to compute the best partition with large separation distance. We assume without loss of generality that $\mathrm{per}(P_2) \leqslant \mathrm{per}(P_1)$. It will be convenient to treat the case where P_2 is a singleton separately.

Lemma 2.7. *The point $p \in P$ minimizing* $\mathrm{per}(P\setminus\{p\})$ *can be computed using $O(n\log n)$ time.*

Proof. The point p we are looking for must be a vertex of $\mathrm{CH}(P)$. First we compute $\mathrm{CH}(P)$ in $O(n\log n)$ time [28]. Let $v_0,v_1,\dots,v_{m-1}$ denote the vertices of $\mathrm{CH}(P)$ in counterclockwise order. Let Δ_i be the triangle with vertices $v_{i-1}v_iv_{i+1}$ (with indices taken modulo m) and let P_i denote the set of points lying inside Δ_i, excluding v_i but including v_{i-1} and v_{i+1}. Note that any point $p \in P$ is present in at most two sets P_i. Hence, $\sum_{i=0}^{m}|P_i| = O(n)$. It is not hard to compute the sets P_i in $O(n \log n)$ time in total. After doing so, we compute all convex hulls $\text{CH}(P_i)$ in $O(n \log n)$ time in total. Since

$$\text{per}(P \setminus \{v_i\}) = \text{per}(P) - |v_{i-1}v_i| - |v_i v_{i+1}| + \text{per}(P_i) - |v_{i-1}v_{i+1}|,$$

we can now find the point p minimizing $\text{per}(P \setminus \{p\})$ in $O(n)$ time. □

It remains to compute the best partition (P_1, P_2) with $\text{per}(P_2) \leqslant \text{per}(P_1)$ whose separation distance is at least $c_{\text{sep}} \cdot \text{per}(P_2)$ and where P_2 is not a singleton. Let (P_1^*, P_2^*) denote this partition. Define the *size* of a square[2] σ to be its edge length. A square σ is a *good square* if (i) $P_2^* \subset \sigma$, and (ii) $\text{size}(\sigma) \leqslant c^* \cdot \text{per}(P_2^*)$, where $c^* := 18$. Our algorithm globally works as follows.

1. Compute a set S of $O(n)$ squares such that S contains a good square.
2. For each square $\sigma \in S$, construct a set H_σ of $O(1)$ halfplanes such that the following holds: if $\sigma \in S$ is a good square then there is a halfplane $h \in H_\sigma$ such that $P_2^* = P(\sigma \cap h)$, where $P(\sigma \cap h) := P \cap (\sigma \cap h)$.
3. For each pair (σ, h) with $\sigma \in S$ and $h \in H_\sigma$, compute $\text{per}(P \setminus P(\sigma \cap h)) + \text{per}(P(\sigma \cap h))$, and report the partition $(P \setminus P(\sigma \cap h), P(\sigma \cap h))$ that gives the smallest sum.

Step 1: Finding a good square. To find a set S that contains a good square, we first construct a set S_{base} of so-called *base squares*. The set S will then be obtained by expanding the base squares appropriately.

We define a base square σ to be *good* if (i) σ contains at least one point from P_2^*, and (ii) $c_1 \cdot \text{diam}(P_2^*) \leqslant \text{size}(\sigma) \leqslant c_2 \cdot \text{diam}(P_2^*)$, where $c_1 := 1/4$ and $c_2 := 4$ and $\text{diam}(P_2^*)$ denotes the diameter of P_2^*. Note that $2 \cdot \text{diam}(P_2^*) \leqslant \text{per}(P_2^*) \leqslant 4 \cdot \text{diam}(P_2^*)$. For a square σ, define $\overline{\sigma}$ to be the square with the same center as σ and whose size is $(1 + 2/c_1) \cdot \text{size}(\sigma)$.

Lemma 2.8. *If σ is a good base square then $\overline{\sigma}$ is a good square.*

Proof. The distance from any point in σ to the boundary of $\overline{\sigma}$ is at least

$$\frac{\text{size}(\overline{\sigma}) - \text{size}(\sigma)}{2} \geqslant \text{diam}(P_2^*).$$

Since σ contains a point from P_2^*, it follows that $P_2^* \subset \overline{\sigma}$. Since $\text{size}(\sigma) \leqslant c_2 \cdot \text{diam}(P_2^*)$, we have

$$\text{size}(\overline{\sigma}) \leqslant (2/c_1 + 1) \cdot c_2 \cdot \text{diam}(P_2^*) = 36 \cdot \text{diam}(P_2^*) \leqslant c^* \cdot \text{per}(P_2^*).$$

□

[2]Whenever we speak of squares, we always mean axis-parallel squares.

To obtain S it thus suffices to construct a set S_{base} that contains a good base square. To this end we first build a compressed quadtree for P. For completeness we briefly review the definition of compressed quadtrees; see also Fig. 2.4 (left).

Assume without loss of generality that P lies in the interior of the unit square $U := [0,1]^2$. Define a *canonical square* to be any square that can be obtained by subdividing U recursively into quadrants. A *compressed quadtree* [46] for P is a hierarchical subdivision of U, defined as follows. In a generic step of the recursive process we are given a canonical square σ and the set $P(\sigma) := P \cap \sigma$ of points inside σ. (Initially $\sigma = U$ and $P(\sigma) = P$.)

- If $|P(\sigma)| \leqslant 1$ then the recursive process stops and σ is a square in the final subdivision.
- Otherwise there are two cases. Consider the four quadrants of σ. The first case is that at least two of these quadrants contain points from $P(\sigma)$. (We consider the quadrants to be closed on the left and bottom side, and open on the right and top side, so a point is contained in a unique quadrant.) In this case we partition σ into its four quadrants—we call this a *quadtree split*—and recurse on each quadrant. The second case is that all points from $P(\sigma)$ lie inside the same quadrant. In this case we compute the smallest canonical square, σ', that contains $P(\sigma)$ and we partition σ into two regions: the square σ' and the so-called *donut region* $\sigma \setminus \sigma'$. We call this a *shrinking step*. After a shrinking step we only recurse on the square σ', not on the donut region.

A compressed quadtree for a set of n points can be computed in $O(n \log n)$ time in the appropriate model of computation[3] [46]. The idea is now as follows. Let $p, p' \in P_2^*$ be a pair of points defining $\text{diam}(P_2^*)$. The compressed quadtree hopefully allows us to zoom in until we have a square in the compressed quadtree that contains p or p' and whose size is roughly equal to $|pp'|$. Such a square will be then a good base square. Unfortunately this does not always work since p and p' can be separated too early. We therefore have to proceed more carefully: we need to add five types of base squares to S_{base}, as explained next and illustrated in Fig. 2.4 (right).

(B1) Any square σ that is generated during the recursive construction—note that this not only refers to squares in the final subdivision—is put into S_{base}.

(B2) For each point $p \in P$ we add a square σ_p to S_{base}, as follows. Let σ be the square of the final subdivision that contains p. Then σ_p is a smallest square that contains p and that shares a corner with σ.

[3] In particular we need to be able to compute the smallest canonical square containing two given points in $O(1)$ time. See the book by Har-Peled [46] for a discussion.

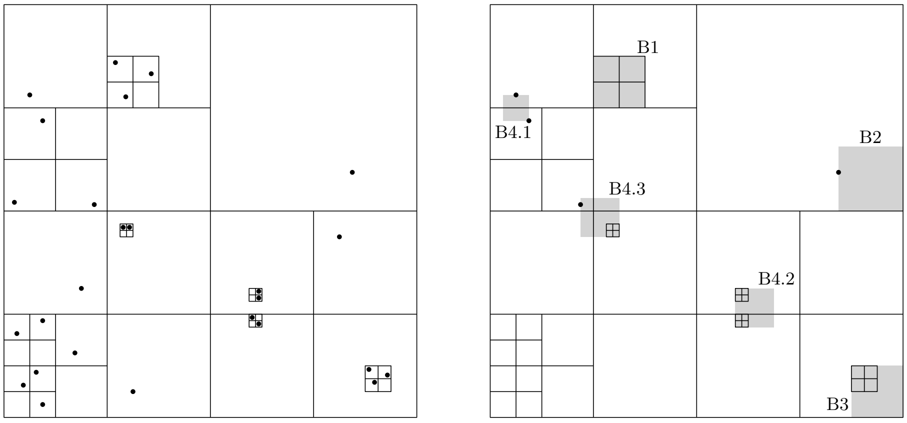


Figure 2.4: A compressed quadtree and some of the base squares generated from it. In the right figure, only the points are shown that are relevant for the shown base squares.

(B3) For each square σ that results from a shrinking step we add an extra square σ' to S_{base}, where σ' is the smallest square that contains σ and that shares a corner with the parent square of σ.

(B4) For any two regions in the final subdivision that touch each other—we also consider two regions to touch if they only share a vertex—we add at most one square to S_{base}, as follows. If one of the regions is an empty square, we do not add anything for this pair. Otherwise we have three cases.

(B4.1) If both regions are non-empty squares containing single points p and p', respectively, then we add a smallest enclosing square for the pair of points p, p' to S_{base}.

(B4.2) If both regions are donut regions, say $\sigma_1 \setminus \sigma'_1$ and $\sigma_2 \setminus \sigma'_2$, then we add a smallest enclosing square for the pair σ'_1, σ'_2 to S_{base}.

(B4.3) If one region is a non-empty square containing a single point p and the other is a donut region $\sigma \setminus \sigma'$, then we add a smallest enclosing square for the pair p, σ' to S_{base}.

Lemma 2.9. *The set S_{base} has size $O(n)$ and contains a good base square. Furthermore, S_{base} can be computed in $O(n \log n)$ time.*

Proof. A compressed quadtree has size $O(n)$ so we have $O(n)$ base squares of type (B1) and (B3). Obviously there are $O(n)$ base squares of type (B2). Finally, the number of pairs of final regions that touch is $O(n)$—this follows because we have a planar rectilinear subdivision of total complexity $O(n)$—and so the number of base squares of type (B4) is $O(n)$ as well. The fact that we can compute S_{base} in $O(n \log n)$ time follows directly from the fact that we can compute the compressed quadtree in $O(n \log n)$ time [46].

It remains to prove that S_{base} contains a good base square. We call a square σ *too small* when $\text{size}(\sigma) < c_1 \cdot \text{diam}(P_2^*)$ and *too large* when $\text{size}(\sigma) > c_2 \cdot \text{diam}(P_2^*)$; otherwise we say that σ has the *correct size*. Let $p, p' \in P_2^*$ be two points with $|pp'| = \text{diam}(P_2^*)$, and consider a smallest square $\sigma_{p,p'}$, in the compressed quadtree that contains both p and p'. Note that $\sigma_{p,p'}$ cannot be too small, since $c_1 = 1/4 < 1/\sqrt{2}$. If $\sigma_{p,p'}$ has the correct size, then we are done since it is a good base square of type (B1). So now suppose $\sigma_{p,p'}$ is too large.

Let $\sigma_0, \sigma_1, \ldots, \sigma_k$ be the sequence of squares in the recursive subdivision of $\sigma_{p,p'}$ that contain p; thus $\sigma_0 = \sigma_{p,p'}$ and σ_k is a square in the final subdivision. Define $\sigma'_0, \sigma'_1, \ldots, \sigma'_{k'}$ similarly, but now for p' instead of p. Suppose that none of these squares has the correct size—otherwise we have a good base square of type (B1). There are three cases.

- *Case (i): σ_k and $\sigma'_{k'}$ are too large.*

  We claim that σ_k touches $\sigma'_{k'}$. To see this, assume without loss of generality that $\text{size}(\sigma_k) \leqslant \text{size}(\sigma'_{k'})$. If σ_k does not touch $\sigma'_{k'}$ then $|pp'| \geqslant \text{size}(\sigma_k)$, which contradicts the assumption that σ_k is too large. Hence, σ_k indeed touches $\sigma'_{k'}$. But then we have a base square of type (B4.1) for the pair p, p' and since $|pp'| = \text{diam}(P_2^*)$ this is a good base square.

- *Case (ii): σ_k and $\sigma'_{k'}$ are too small.*

  In this case there are indices $0 < j \leqslant k$ and $0 < j' \leqslant k'$ such that σ_{j-1} and $\sigma'_{j'-1}$ are too large and σ_j and $\sigma'_{j'}$ are too small. Note that this implies that both σ_j and $\sigma'_{j'}$ result from a shrinking step, because $c_1 < c_2/2$ and so the quadrants of a too-large square cannot be too small. We claim that σ_{j-1} touches $\sigma'_{j'-1}$. Indeed, similarly to Case (i), if σ_{j-1} and $\sigma'_{j'-1}$ do not touch then $|pp'| > \min(\text{size}(\sigma_{j-1}), \text{size}(\sigma'_{j'-1}))$, contradicting the assumption that both σ_{j-1} and $\sigma'_{j'-1}$ are too large. We now have two subcases.

  - The first subcase is that the donut region $\sigma_{j-1} \setminus \sigma_j$ touches the donut region $\sigma'_{j'-1} \setminus \sigma_{j'}$. Thus a smallest enclosing square for σ_j and $\sigma'_{j'}$ has been put into S_{base} as a base square of type (B4.2). Let σ^* denote this square. Since the segment pp' is contained in σ^* we have

    $$c_1 \cdot \text{diam}(P_2^*) \; < \; \text{diam}(P_2^*)/\sqrt{2} \; = \; |pp'|/\sqrt{2} \; \leqslant \; \text{size}(\sigma^*).$$

    Furthermore, since σ_j and $\sigma'_{j'}$ are too small we have

    $$\text{size}(\sigma^*) \; \leqslant \; \text{size}(\sigma_j) + \text{size}(\sigma'_{j'}) + |pp'| \; \leqslant \; 3 \cdot \text{diam}(P_2^*) \; < \; c_2 \cdot \text{diam}(P_2^*),$$

(2.11)

and so σ^* is a good base square.

– The second subcase is that $\sigma_{j-1} \setminus \sigma_j$ does not touch $\sigma'_{j'-1} \setminus \sigma_{j'}$. This can only happen if σ_{j-1} and $\sigma'_{j'-1}$ just share a single corner, v. Observe that σ_j must lie in the quadrant of σ_{j-1} that has v as a corner, otherwise $|pp'| \geqslant \text{size}(\sigma_{j-1})/2$ and σ_{j-1} would not be too large. Similarly, $\sigma'_{j'}$ must lie in the quadrant of $\sigma'_{j'-1}$ that has v as a corner. Thus the base squares of type (B3) for σ_j and $\sigma'_{j'}$ both have v as a corner. Take the largest of these two base squares, say σ_j. For this square σ^* we have

$$c_1 \cdot \text{diam}(P_2^*) \;<\; \text{diam}(P_2^*)/2\sqrt{2} \;=\; |pp'|/2\sqrt{2} \;\leqslant\; \text{size}(\sigma^*),$$

since $|pp'|$ is contained in a square of twice the size of σ^*. Furthermore, since σ_j is too small and $|pv| < |pp'|$ we have

$$\text{size}(\sigma^*) \;\leqslant\; \text{size}(\sigma_j)+|pv| \;\leqslant\; (c_1+1)\cdot\text{diam}(P_2^*) \;<\; c_2\cdot\text{diam}(P_2^*). \quad (2.12)$$

Hence, σ^* is a good base square.

- *Case (iii): neither (i) nor (ii) applies.*

  In this case σ_k is too small and $\sigma'_{k'}$ is too large (or vice versa). Thus there must be an index $0 < j \leqslant k$ such that σ_{j-1} is too large and σ_j is too small. We can now follow a similar reasoning as in Case (ii): First we argue that σ_j must have resulted from a shrinking step and that σ_{j-1} touches $\sigma'_{k'}$. Then we distinguish two subcases, namely where the donut region $\sigma_j \setminus \sigma_{j-1}$ touches $\sigma'_{k'}$ and where it does not touch $\sigma'_{k'}$. The arguments for the two subcases are similar to the subcases in Case (ii), with the following modifications. In the first subcase we use base squares of type (B4.3) and in (2.11) the term $\text{size}(\sigma'_{j'})$ disappears; in the second subcase we use a type (B3) base square for σ_j and a type (B2) base square for p', and when the base square for p' is larger than the base square for σ_j then (2.12) becomes $\text{size}(\sigma^*) \leqslant 2\,|p'v| < c_2 \cdot \text{diam}(P_2^*)$.

□

Step 2: Generating halfplanes. Consider a good square $\sigma \in S$. Let Q_σ be a set of $4 \cdot c^*/c_{\text{sep}} + 1 = 18001$ points placed equidistantly around the boundary of σ. Note that the distance between two neighbouring points in Q_σ is less than $c_{\text{sep}}/c^* \cdot \text{size}(\sigma)$. For each pair q_1, q_2 of points in Q_σ, add to H_σ the two halfplanes defined by the line through q_1 and q_2.

Lemma 2.10. *For any good square $\sigma \in S$, there is a halfplane $h \in H_\sigma$ such that $P_2^* = P(\sigma \cap h)$.*

Proof. In the case where $\sigma \cap P_1^* = \emptyset$, two points in Q_σ from the same edge of σ define a half-plane h such that $P_2^* = P(\sigma \cap h)$, so assume that σ contains one or more points from P_1^*.

We know that the separation distance between P_1^* and P_2^* is at least $c_{\text{sep}} \cdot \text{per}(P_2^*)$. Moreover, $\text{size}(\sigma) \leqslant c^* \cdot \text{per}(P_2^*)$. Hence, there is an empty open strip O with a width of at least $c_{\text{sep}}/c^* \cdot \text{size}(\sigma)$ separating P_2^* from P_1^*. Since σ contains a point from P_1^*, we know that $\sigma \setminus O$ consists of two pieces and that the part of the boundary of σ inside O consists of two disjoint portions B_1 and B_2 each of length at least $c_{\text{sep}}/c^* \cdot \text{size}(\sigma)$. Hence the sets $B_1 \cap Q_\sigma$ and $B_2 \cap Q_\sigma$ contain points q_1 and q_2, respectively, that define a half-plane h as desired. $\square$

Step 3: Evaluating candidate solutions. In this step we need to compute for each pair (σ, h) with $\sigma \in S$ and $h \in H_\sigma$, the value $\text{per}(P \setminus P(\sigma \cap h)) + \text{per}(P(\sigma \cap h))$. We do this by preprocessing P into a data structure that allows us to quickly compute $\text{per}(P \setminus P(\sigma \cap h))$ and $\text{per}(P(\sigma \cap h))$ for a given pair (σ, h). Recall that the bounding lines of the halfplanes h we must process have $O(1)$ different orientations. We construct a separate data structure for each orientation.

Consider a fixed orientation ϕ. We build a data structure $\mathscr{D}_\phi$ for range searching on P with ranges of the form $\sigma \cap h$, where σ is a square and h is a halfplane whose bounding line has orientation ϕ. Since the edges of each σ are axis-parallel and the bounding lines of the halfplanes h have a fixed orientation, we can use a standard three-level range tree [28] for this. Constructing this tree takes $O(n \log^2 n)$ time and the tree has $O(n \log^2 n)$ nodes.

Each node ν of the third-level trees in $\mathscr{D}_\phi$ is associated with a *canonical subset* $P(\nu)$, which contains the points stored in the subtree rooted at ν. We preprocess each canonical subset $P(\nu)$ as follows. First we compute the convex hull $\text{CH}(P(\nu))$. Let $v_1, \dots, v_k$ denote the convex-hull vertices in counterclockwise order. We store these vertices in order in an array, and for each vertex v_i, we store $\text{length}(\partial P(v_1, v_i))$, that is, the length of the part of $\partial\, \text{CH}(P(\nu))$ from v_1 to v_i in counterclockwise order. Note that the convex hull $\text{CH}(P(\nu))$ can be computed in $O(|P(\nu)|)$ from the convex hulls at the two children of ν. Hence, the convex hulls $\text{CH}(P(\nu))$ (and the values $\text{length}(\partial P(v_1, v_i))$) can be computed in $\sum_{\nu \in \mathscr{D}_\phi} O(|P(\nu)|) = O(n \log^3 n)$ time in total, in a bottom-up manner.

Now suppose we want to compute $\text{per}(P(\sigma \cap h))$, where the orientation of the bounding line of h is ϕ. We perform a range query in $\mathscr{D}_\phi$ to find a set $N(\sigma \cap h)$ of $O(\log^3 n)$ nodes such that $P(\sigma \cap h)$ is equal to the union of the canonical subsets of the nodes in $N(\sigma \cap h)$. Standard range-tree properties guarantee that the convex hulls $\text{CH}(P(\nu))$ and $\text{CH}(P(\mu))$ of any two nodes $\nu, \mu \in N(\sigma \cap h)$ are disjoint. Note that $\text{CH}(P(\sigma \cap h))$ is equal to the convex hull of the set of convex hulls $\text{CH}(P(\nu))$ with $\nu \in N(\sigma \cap h)$. Lemma 2.12 given below implies that we can compute $\text{per}(P(\sigma \cap h))$ in $O(\log^4 n)$ time.

Observe that $P \setminus P(\sigma \cap h)$ can also be expressed as the union of $O(\log^3 n)$ canonical subsets with disjoint convex hulls, since $\mathbb{R}^2 \setminus (\sigma \cap h)$ is the disjoint union of $O(1)$ ranges of the right type. Hence, we can compute $\mathrm{per}(P \setminus P(\sigma \cap h))$ in $O(\log^4 n)$ time. We thus obtain the following result, which finishes the proof of Theorem 2.6.

Lemma 2.11. *Step 3 can be performed in $O(n \log^4 n)$ time and using $O(n \log^3 n)$ space.*

Lemma 2.12. *Let $\mathcal{Q}$ be a set of k pairwise disjoint convex polygons with m vertices in total. Suppose each $Q \in \mathcal{Q}$ is represented by an array storing its vertices in counterclockwise order, and suppose for each vertex v_i of Q the value $\mathrm{length}(\partial Q(v_1, v_i))$ is known. Let $\mathbf{Q} := \bigcup_{Q \in \mathcal{Q}} Q$. Then we can compute the perimeter of $\mathrm{CH}(\mathbf{Q})$ in $O(k \log m)$ time.*

Proof. Any ordered pair (Q_i, Q_j) of disjoint convex polygons has two outer common tangents: the *left outer tangent*, which is the one having Q_i and Q_j on its right when directed from Q_i to Q_j, and the *right outer tangent*. The *bridge* $B(Q_i, Q_j)$ from Q_i to Q_j is the minimum-length segment $q_i q_j$ contained in the left outer tangent of Q_i and Q_j and connecting points in Q_i and Q_j. The boundary $\partial\,\mathrm{CH}(\mathbf{Q})$ consists of portions of boundaries ∂Q, where $Q \in \mathcal{Q}$, that are connected by bridges.

The *upper convex hull* of a set of points S, denoted by $\mathrm{UH}(S)$, is the part of $\partial\,\mathrm{CH}(S)$ from the rightmost to the leftmost point in S in counterclockwise direction. We compute a list $\mathcal{L}$ that represents $\mathrm{UH}(\mathbf{Q})$. $\mathcal{L}$ consists of the polygons in $\mathcal{Q}$ having corners on $\mathrm{UH}(\mathbf{Q})$ in the order they are encountered as we traverse $\mathrm{UH}(\mathbf{Q})$ from left to right. We denote the length of $\mathcal{L}$ as $|\mathcal{L}|$ and the entries as $\mathcal{L}[1], \dots, \mathcal{L}[|\mathcal{L}|]$, and do similarly for other lists. Consecutive polygons $\mathcal{L}[i], \mathcal{L}[i+1]$ should always be different, but the same polygon $Q \in \mathcal{Q}$ can appear in $\mathcal{L}$ multiple times, since several portions of ∂Q can appear on $\mathrm{UH}(\mathbf{Q})$ interrupted by portions of boundaries of other polygons.

The *upper envelope* of a set of points S, denoted $\mathrm{ENV}(S)$, is the subset $\{(x, y) \in S \mid \forall (x, y') \in S\colon y' \leqslant y\}$. In order to compute $\mathcal{L}$, we first compute $\mathrm{ENV}(\mathbf{Q})$. Clearly, if a portion of the boundary of a polygon $Q \in \mathcal{Q}$ is on $\mathrm{UH}(\mathbf{Q})$, then the same portion is also on $\mathrm{ENV}(\mathbf{Q})$. We thus have $\mathrm{UH}(\mathbf{Q}) = \mathrm{UH}(\mathrm{ENV}(\mathbf{Q}))$. The envelope $\mathrm{ENV}(\mathbf{Q})$ can be computed with a simple sweep-line algorithm, as described next.

Define the x-range of a polygon $Q \in \mathcal{Q}$ to be the interval

$$I_x(Q) := [x_{\min}(Q), x_{\max}(Q)],$$

where $x_{\min}(Q)$ and $x_{\max}(Q)$ denote the minimum and maximum x-coordinate of Q, respectively. For an interval $I \subseteq I_x(Q)$, define $Q[I]$ to be the intersection of Q with the vertical slab $I \times (-\infty, +\infty)$. We call $Q[I]$ a *vertical slice* of Q. Our representation of Q allows us to do the following using the algorithm described by Kirkpatrick and Snoeyink [53]: given vertical slices $Q[I]$ and $Q'[I']$, compute the bridge $B(Q[I], Q'[I'])$.

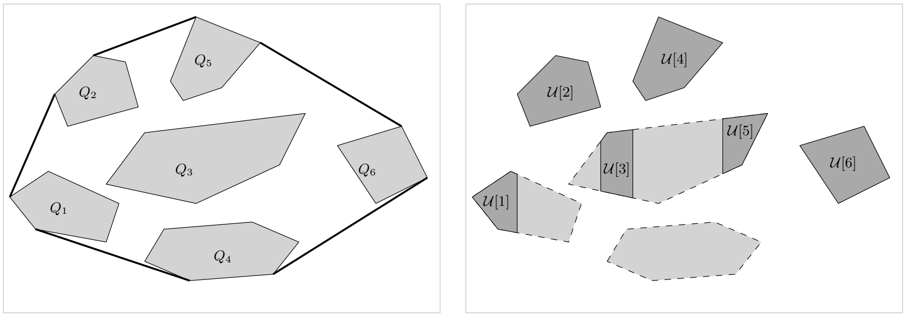


Figure 2.5: A collection of disjoint polygons $\mathcal{Q}$ (left) and the vertical slices in the corresponding list $\mathcal{U}$ which appear on the upper envelope (right). Note that polygon Q_3 defines two slices that contribute to the upper envelope.

Consider the upper envelope ENV($\mathbf{Q}$). It consists of portions of the upper boundaries of the polygons in $\mathbf{Q}$. Each maximal boundary portion of some polygon Q that shows up on ENV($\mathbf{Q}$) defines a vertical slice of Q, namely the slice whose top boundary is exactly the envelope portion. We create a list $\mathcal{U}$ that stores these vertical slices in left-to-right order; see Fig. 2.5. Consecutive slices $\mathcal{U}[i], \mathcal{U}[i+1]$ are always from different polygons, but multiple slices from the same polygon $Q \in \mathcal{Q}$ can appear in $\mathcal{U}$, since several portions of ∂Q can appear on ENV($\mathbf{Q}$) interrupted by portions of boundaries of other polygons.

As mentioned, we will compute ENV($\mathbf{Q}$) using a sweep-line algorithm. As the sweep line ℓ moves from left to right, we maintain a data structure Σ containing all the polygons intersecting ℓ from top to bottom. Let Σ^{top} be the topmost polygon in Σ. In case Σ is empty, so is Σ^{top}. We implement Σ as a red-black tree [27]. Note that since the polygons are disjoint, the vertical order of any two polygons in Σ is invariant, and so Σ only needs to be updated when ℓ starts or stops intersecting a polygon in $\mathcal{Q}$. Thus, to find the sorted set of *events* we simply find the leftmost point L_i and the rightmost point R_i of each polygon $Q_i \in \mathcal{Q}$ and sort these points from left to right.

An event $e_j \in E$ is now handled as follows.

- If $e_j = L_i$, we insert Q_i to Σ. This requires $O(\log k)$ comparisons between Q_i and polygons currently stored in Σ, to find the position where Q should be inserted. Each such comparison can be done in $O(1)$ time since Q_i is above Q_j if and only if $L_i R_i$ is above $L_j R_j$.

  If Σ^{top} changes from some polygon Q_h to Q_i, then we add the appropriate vertical slice of Q_h to $\mathcal{U}$. (This slice ends at the current position of the sweep line ℓ, and it starts at the most recent position of ℓ at which Q_h became Σ^{top}.)

- If $e_j = R_i$ then we delete Q_i from Σ in $O(\log k)$ time. If Σ^{top} was equal to

Q_i before the event, we add the appropriate vertical slice of Q_i to $\mathscr{U}$.

There are $2k$ events to handle, each taking $O(\log k)$ time, so the total time used to compute $\mathscr{U}$ is $O(k \log k)$.

We now proceed to the algorithm computing the list $\mathscr{L}$ representing the upper convex hull of the vertical slices in $\mathscr{U}$. In the sequel, we think of $\mathscr{U}$ as a list of polygons with disjoint x-ranges sorted from left to right. Let $\mathscr{M}$ be a subsequence of $\mathscr{U}$, and let b_i be the bridge between $\mathscr{M}[i]$ and $\mathscr{M}[i+1]$. We say that a triple $\mathscr{M}[i-1], \mathscr{M}[i], \mathscr{M}[i+1]$ is a *valid triple* if either

(a) the right endpoint of b_{i-1} lies strictly to the left of the left endpoint of b_i, or

(b) the right endpoint of b_{i-1} coincides with the left endpoint of b_i, and b_{i-1} and b_i make a right turn.

We need the following claim.

Claim: Suppose $\mathscr{M}$ satisfies the following conditions:

(i) All triples $\mathscr{M}[i-1], \mathscr{M}[i], \mathscr{M}[i+1]$ in $\mathscr{M}$ are valid triples.

(ii) Every polygon $\mathscr{U}[i]$ that is not in $\mathscr{M}$ lies completely below one bridge b_i between consecutive polygons in $\mathscr{M}$. (Note that this condition implies that the first element in $\mathscr{M}$ is $\mathscr{U}[1]$ and the last element is $\mathscr{U}[|\mathscr{U}|]$.)

Then $\mathscr{M}$ correctly represents $\text{UH}(\mathscr{U})$.

Proof of the Claim. Observe that condition (i), together with the definition of a valid triple, implies that the bridges between consecutive polygons in $\mathscr{U}$ together with the relevant boundary pieces—namely, for each polygon in $\mathscr{U}$ the piece of its upper boundary in between the bridges to the previous and the next polygon in $\mathscr{U}$—form a convex x-monotone chain. Hence, $\mathscr{M}$ represents the upper hull of all polygons that appear in $\mathscr{M}$. On the other hand, a polygon that does not appear in $\mathscr{M}$ cannot contribute to $\text{UH}(\mathscr{U})$ by condition (ii). We conclude that $\mathscr{M}$ correctly represents $\text{UH}(\mathscr{U})$. □

We now describe the algorithm computing $\mathscr{L}$, and we prove its correctness by showing that it satisfies the conditions from the claim.

The algorithm is essentially the same as Andrew's version of Graham's scan [28] for point sets, except that the standard right-turn check for points is replaced by a valid-triple check for polygons. Thus it works as follows. We handle the polygons from $\mathscr{U}$ to $\mathscr{L}$ one by one in order from $\mathscr{U}[1]$ to $\mathscr{U}[|\mathscr{U}|]$. To handle $\mathscr{U}[i]$ we first append $\mathscr{U}[i]$ to $\mathscr{L}$. Next, we check if the last three polygons in $\mathscr{L}$ define a valid triple. If not, we remove the middle of the three polygons, and check if the new triple at the end of $\mathscr{L}$ is valid, remove the middle polygon if the triple is invalid, and so on. This continues until either

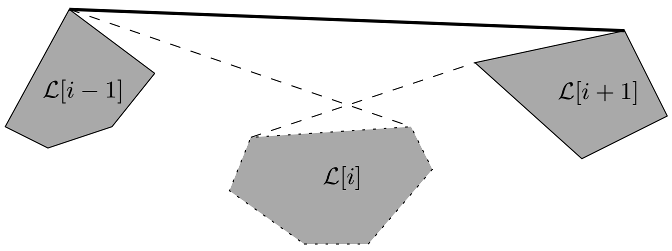


Figure 2.6: An invalid triple of polygons.

the last triple in the list is valid, or we have only two polygons left in $\mathscr{L}$. We then proceed to handle the next polygon, $\mathscr{U}[i+1]$.

We claim that the algorithm satisfies the following invariant: When we have added $\mathscr{U}[1],\ldots,\mathscr{U}[i]$ to $\mathscr{L}$, then $\mathscr{L}$ defines the upper convex hull $\textsc{ch}(\mathscr{U}[1,\ldots,i])$. It clearly follows from this invariant that when we have handled the last polygon in $\mathscr{U}$, then $\mathscr{L}$ correctly defines $\textsc{uh}(\mathscr{U})$.

We prove the invariant by induction. Assume therefore that it holds when we have added the polygons $\mathscr{U}[1,\ldots,i]$ to $\mathscr{L}$ and consider what happens when we add $\mathscr{U}[i+1]$ to $\mathscr{L}$. By our invalid-triple removal procedure, after we have handled $\mathscr{U}[i+1]$ all triples $\mathscr{U}[j-1],\mathscr{U}[j],\mathscr{U}[j+1]$ that remain in $\mathscr{L}$ must be valid, either because the triple was already in the list before the addition of $\mathscr{U}[i+1]$, or because it is a triple involving $\mathscr{U}[i+1]$ (in which case it was explicitly checked). Thus condition (i) is satisfied. To establish condition (ii) we only need to argue that every polygon that is removed from $\mathscr{L}$ is completely below some bridge. This is true because the middle polygon of an invalid triple lies below the bridge between the first and last polygon of the triple—see Fig. 2.6. Hence, the resulting list $\mathscr{L}$ satisfies conditions (ii) as well. This completes the proof of the correctness of the algorithm.

Since $\mathscr{U}$ has size $O(k)$, we need to do $O(k)$ checks for invalid triples. Each such check involves the computation of two bridges, which takes $O(\log m)$ time. Thus the whole procedure takes $O(k \log m)$ time. It is easy to compute the length of $\textsc{uh}(\mathbf{Q})$ within the same time bounds. Similarly, we can compute the lower convex hull of $\mathbf{Q}$ and its length in $O(k \log m)$ time. This finishes the proof of the lemma. $\square$

2.2 The Approximation Algorithm

Theorem 2.13. *Let P be a set of n points in the plane and let (P_1^*, P_2^*) be a partition of P minimizing* $\text{per}(P_1^*) + \text{per}(P_2^*)$. *Suppose we have an exact algorithm for the minimum perimeter-sum problem running in $T(k)$ time for instances with k points. Then for any given $\varepsilon > 0$ we can compute a partition (P_1, P_2) of P such* *that* $\text{per}(P_1) + \text{per}(P_2) \leqslant (1+\varepsilon)\cdot\big(\text{per}(P_1^*) + \text{per}(P_2^*)\big)$ *in* $O(n + T(1/\varepsilon^2))$ *time.*

Proof. Consider the axis-parallel bounding box B of P. Let w be the width of B and let h be its height. Assume without loss of generality that $w \geqslant h$. Our algorithm works in two steps.

- *Step 1: Check if* $\text{per}(P_1^*) + \text{per}(P_2^*) \leqslant w/16$. *If so, compute the exact solution.*

  We partition B vertically into four strips with width $w/4$, denoted B_1, B_2, B_3, and B_4 from left to right. If B_2 or B_3 contains a point from P, we have $\text{per}(P_1^*) + \text{per}(P_2^*) \geqslant w/2 > w/16$ and we go to Step 2. If B_2 and B_3 are both empty, we consider two cases.

  - *Case (i):* $h \leqslant w/8$.

    In this case we simply return the partition $(P \cap B_1, P \cap B_4)$. To see that this is optimal, we first note that any subset $P' \subset P$ that contains a point from B_1 as well as a point from B_4 has $\text{per}(P') \geqslant 2\cdot(3w/4) = 3w/2$. On the other hand, $\text{per}(P \cap B_1) + \text{per}(P \cap B_4) \leqslant 2\cdot(w/2 + 2h) \leqslant 3w/2$.

  - *Case (ii):* $h > w/8$.

    We partition B horizontally into four rows with height $h/4$, numbered R_1, R_2, R_3, and R_4 from bottom to top. If R_2 or R_3 contains a point from P, we have $\text{per}(P_1^*) + \text{per}(P_2^*) \geqslant h/2 > w/16$, and we go to Step 2. If R_2 and R_3 are both empty, we overlay the vertical and the horizontal partitioning of B to get a 4×4 grid of cells $C_{ij} := B_i \cap R_j$ for $i, j \in \{1,\dots,4\}$. We know that only the corner cells $C_{11}, C_{14}, C_{41}, C_{44}$ contain points from P. If three or four corner cells are non-empty, $\text{per}(P_1^*) + \text{per}(P_2^*) \geqslant 6h/4 > w/16$, and we go to Step 2. Hence, we may without loss of generality assume that any point of P is in C_{11} or C_{44}. We now return the partition $(P \cap C_{11}, P \cap C_{44})$, which is easily seen to be optimal.

- *Step 2: Handle the case where* $\text{per}(P_1^*) + \text{per}(P_2^*) > w/16$.

  The idea is to compute a subset $\widehat{P} \subset P$ of size $O(1/\varepsilon^2)$ such that an exact solution to the minimum perimeter-sum problem on $\widehat{P}$ can be used to obtain a $(1+\varepsilon)$-approximation for the problem on P.

  We subdivide B into $O(1/\varepsilon^2)$ rectangular cells of width and height at most $c := \varepsilon w/(64\pi\sqrt{2})$. For each cell C where $P \cap C$ is non-empty we pick an arbitrary point in $P \cap C$, and we let $\widehat{P}$ be the set of selected points. For a point $p \in \widehat{P}$, let $C(p)$ be the cell containing p. Intuitively, each point $p \in \widehat{P}$ represents all the points $P \cap C(p)$. Let $(\widehat{P}_1, \widehat{P}_2)$ be a partition of $\widehat{P}$ that minimizes $\text{per}(\widehat{P}_1) + \text{per}(\widehat{P}_2)$. We assume we have an algorithm that can compute such an optimal partition in $T(|\widehat{P}|)$ time. For $i = 1, 2$, define

  $$P_i := \bigcup_{p \in \widehat{P}_i} P \cap C(p).$$

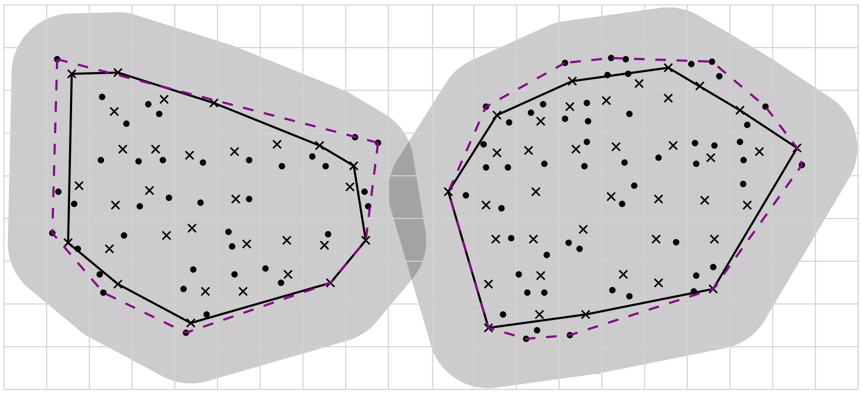


Figure 2.7: The crossed points are the points of $\widehat{P}$. The left gray region is $\widetilde{P}_1$ and the right gray region is $\widetilde{P}_2$. The left dashed polygon is the convex hull of P_1 and the right dashed polygon is the convex hull of P_2.

Our approximation algorithm returns the partition (P_1, P_2). (Note that the convex hulls of P_1 and P_2 are not necessarily disjoint.) It remains to prove the approximation ratio.

First, note that $\text{per}(\widehat{P}_1)+\text{per}(\widehat{P}_2) \leqslant \text{per}(P_1^*)+\text{per}(P_2^*)$ since $\widehat{P} \subseteq P$. For $i = 1, 2$, let $\widetilde{P}_i$ consist of all points in the plane (not only points in P) within a distance of at most $c\sqrt{2}$ from $\text{CH}(\widehat{P}_i)$. In other words, $\widetilde{P}_i$ is the Minkowksi sum of $\text{CH}(\widehat{P}_i)$ with a disk D of radius $c\sqrt{2}$ centered at the origin; see Fig. 2.7. Note that if $p \in \widehat{P}_i$, then $q \in \widetilde{P}_i$ for any $q \in P \cap C(p)$, since any two points in $C(p)$ are at most $c\sqrt{2}$ apart from each other. Therefore $P_i \subset \widetilde{P}_i$ and hence $\text{per}(P_i) \leqslant \text{per}(\widetilde{P}_i)$. Note also that $\text{per}(\widetilde{P}_i) = \text{per}(\widehat{P}_i) + 2c\pi\sqrt{2}$. These observations yield

$$\begin{aligned}
\text{per}(P_1)+\text{per}(P_2) &\leqslant \text{per}(\widetilde{P}_1)+\text{per}(\widetilde{P}_2) \\
&= \text{per}(\widehat{P}_1)+\text{per}(\widehat{P}_2)+4c\pi\sqrt{2} \\
&\leqslant \text{per}(P_1^*)+\text{per}(P_2^*)+4c\pi\sqrt{2} \\
&= \text{per}(P_1^*)+\text{per}(P_2^*)+4\pi\sqrt{2}\cdot\big(\varepsilon w/(64\pi\sqrt{2})\big) \\
&\leqslant \text{per}(P_1^*)+\text{per}(P_2^*)+\varepsilon w/16 \\
&\leqslant (1+\varepsilon)\cdot(\text{per}(P_1^*)+\text{per}(P_2^*)).
\end{aligned}$$

As all the steps can be done in linear time, the time complexity of the algorithm is $O(n + T(n_\varepsilon))$ for some $n_\varepsilon = O(1/\varepsilon^2)$.

□

2.3 Concluding Remarks

We presented the first sub-quadratic algorithm for the bipartition case. The algorithm works in $O(n \log^4 n)$ time. It would be interesting to see if improving this result is possible. Finding a lower bound is another open problem. Existence of subquadratic algorithms for other variants studied by Mitchell and Wynters [59] is still an open problem.

We also presented a linear-time $(1+\varepsilon)$-approximation algorithm for this case with running time $O(n+T(1/\varepsilon^2)) = O(n+1/\varepsilon^2 \cdot \log^4(1/\varepsilon))$, where $T(1/\varepsilon^2)$ is the running time of an exact algorithm on an instance of size $1/\varepsilon^2$.

Chapter 3

Minimizing the Sum of Perimeters: The General Case

In this chapter we consider the general case of the minimum perimeter-sum problem where a set S of n points in the plane and a number $k \in \{1, \ldots, n\}$ is given, and we aim at finding a partition of the points in S to k clusters such that the sum of the perimeters of the convex hulls of the clusters is minimized. We disprove the conjecture made by Arkin et al. [10] that this problem is NP-hard when k is part of the input by presenting an algorithm that solves this problem in $O(n^{27})$ time.

3.1 Problem Definition and Preliminaries

In the minimum perimeter-sum problem, we are given a set of n points S in the plane and an integer k as an input instance I, which we denote by the pair $I := (S, k)$. Our main approach is to use dynamic programming to solve this problem. Our subproblems are based on the notion of boxes (i.e., rectangles in the plane), which contain some subset of the input points that we would like to cluster optimally. Great care is needed in order to ensure that there are only polynomially many subproblems we need to consider. To this end, we prove some structural properties regarding an optimal solution that enable us to reduce the complexity of the subproblems that we solve. We first describe some relevant notation and preprocessing that we do to the input.

Given two points $p, q, \in \mathbb{R}^2$, with $p \neq q$, we use pq to denote the directed edge from p to q. We call p the *head* of the edge and we call q its *tail*. A loop o is defined by a single point $p \in \mathbb{R}^2$ and is denoted by $o := pp$. An *edge set* is defined to be a set of edges or a set containing a single loop. For a convex polygon P, define ∂P to be the boundary of P and $\text{per}(P)$ to be the perimeter of

P. We use $\mathscr{E}(P)$ to denote the edge set defining ∂P oriented in counterclockwise order. For the special case where P is a single point $p \in \mathbb{R}^2$, $\mathscr{E}(P) := \{pp\}$.

For a set of geometric objects O, define $\textsc{ch}(O)$ to be the convex hull of O. For a set S of points in the plane, with some abuse of notation, we define $\partial S := \partial\textsc{ch}(S)$, $\text{per}(S) := \text{per}(\textsc{ch}(S))$, and $\mathscr{E}(S) := \mathscr{E}(\textsc{ch}(S))$. For the special case where S is a single point p, define $\mathscr{E}(\{p\}) := \{pp\}$. We denote by $\mathscr{G}(S)$ the set of the $O(n^2)$ oriented segments defined by any ordered pair of points in S.

We define a *k-clustering* of S as a partition $\mathscr{C} := \{S_1, \ldots, S_k\}$ of S into k subsets or *clusters* $S_1, \ldots, S_k \subseteq S$. Let $\text{Part}(S)$ be the set of all possible clusterings of a set S. Let $\Phi(\mathscr{C}) := \sum_{i=1}^{k} \text{per}(S_i)$ be the cost of the clustering $\mathscr{C}$. An *optimal k*-clustering is a k-clustering $\mathscr{C}_k^{\textsc{Opt}}(S)$ such that $\Phi(\mathscr{C}_k^{\textsc{Opt}}(S)) \leqslant \Phi(\mathscr{C})$ for any k-clustering $\mathscr{C}$. (If there are multiple optimal clusterings, we let $\mathscr{C}_k^{\textsc{Opt}}(S)$ denote an arbitrary one.) In the following, whenever we talk about the edges of $\mathscr{C}_k^{\textsc{Opt}}(S)$ we refer to the edges induced by the convex hulls of the clusters in $\mathscr{C}_k^{\textsc{Opt}}(S)$. Let $\textsc{Opt}_k(S) := \Phi(\mathscr{C}_k^{\textsc{Opt}}(S))$.

A clustering $\mathscr{C} := \{S_1, \ldots, S_k\}$ is called a *disjoint clustering* if the convex hulls of any two clusters $S_i, S_j \in \mathscr{C}$ are disjoint, i.e., $\textsc{ch}(S_i) \cap \textsc{ch}(S_j) = \emptyset$ for all $i \neq j$.

Observation 3.1. *Given a set of points S in the plane, $\mathscr{C}_k^{\textsc{Opt}}(S)$ is a disjoint clustering.*

We first ensure that no two points in S have the same x- or y-coordinate. This is possible to do in $O(n^2)$ time by computing the slopes of segments between all pairs of points in S. If one of the slopes is vertical or horizontal, we apply a slight rotation to the coordinate system to eliminate all the horizontal and vertical slopes without introducing new ones.

Let $p_1, \ldots, p_n$ be the points in S sorted by their x-coordinates $x_1 < \cdots < x_n$. For each point p_i we define two vertical lines, v_i^- and v_i^+ such that v_i^- is infinitessimally to the left of p_i and v_i^+ is infinitessimally to the right of p_i.

We construct two vertical lines v_i^- and v_i^+ with each x-coordinate x_i, such that v_i^- formally is to the left of p_i and v_i^+ formally is to the right of p_i.

Thus, an edge $p_j p_i$ for $j < i$ intersects v_i^- at p_i but does not intersect v_i^+, whereas $p_i p_j$ for $i < j$ intersects v_i^+ at p_i but does not intersect v_i^-. The set of all lines v_i^-, v_i^+ for all $i \in \{1, \ldots, n\}$ are the *vertical main lines*. Between any two consecutive vertical main lines v_i^+, v_{i+1}^-, we define 19999 *vertical help lines* with x-coordinates $x_i + \frac{x_{i+1} - x_i}{20000} \cdot j$ for $j \in \{1, \ldots, 19999\}$. That is, these 19999 vertical help lines induce 20000 intervals between x_i and x_{i+1}, each of which has the same length given by $\frac{x_{i+1} - x_i}{20000}$. In a similar way, we define two horizontal main lines with the y-coordinate of each input point p, one formally below p and the other formally above p. Let $h_1^-, h_1^+, \ldots, h_n^-, h_n^+$ be the horizontal main lines sorted by ascending y-coordinate. We also define 19999 equidistant horizontal help lines between any two consecutive horizontal main lines. [1]

[1] The value 20,000 is nothing special but just chosen sufficiently large to make things work in the sequel.

Boxes. Let $\mathcal{B}(S)$ be the set of all closed rectangles with edges contained in main or help lines. Note that the size of $\mathcal{B}(S)$ is $O(n^4)$. We use S_B as an abbreviation for $B \cap S$ where $B \in \mathcal{B}(S)$. We denote by $l(B)$ and $r(B)$ the left and right vertical edge of B and by $b(B)$ and $t(B)$ the bottom and top edge of B. We denote by $w(B)$ the *width* of B, i.e., the difference in the x-coordinates of $r(B)$ and $l(B)$. Similarly, we denote by $h(B)$ the *height* of B, i.e., the difference in the y-coordinates of $t(B)$ and $b(B)$. We define the *length* of a box $B \in \mathcal{B}(S)$ as $\max\{w(B), h(B)\}$ and denote it by length(B). Consider an arbitrary k-clustering $\mathcal{C} := \{S_1, \ldots, S_k\}$. For a box $B \in \mathcal{B}(S)$, consider for each $S_i \in \mathcal{C}$ the part of the boundary of $\textsc{ch}(S_i)$ that is in B. The cost of $\mathcal{C}$ in B is denoted as $\Phi_B(\mathcal{C})$, and we define it to be the total length of these parts for all the clusters $S_1, \ldots, S_k$.

Consider a box B and a vertical line ℓ lying strictly between $l(B)$ and $r(B)$. We say that the vertical line segment $s := \ell \cap B$ is a *vertical separator* of B. A vertical separator of a box B is *good* if it intersects at most two edges in $\mathcal{C}_k^{\text{OPT}}(S)$. We also analogously define a *horizontal separator*, along with the notion of a *good horizontal separator*. We call a box *elementary* if there are no vertical help or main lines that lie strictly in between $l(B)$ and $r(B)$, and no horizontal help or main lines that lie strictly in between $b(B)$ and $t(B)$. In other words, an elementary box is a cell of the grid induced by the main and help lines.

Finally, we define a *vertical strip* of a box B to be a rectangle T contained in B where the bottom edge of T is contained in $b(B)$ and the top edge of T is contained in $t(B)$ (i.e., the top and bottom edges of T lie on the boundary of B) and whose left and right edges are contained in a main line or a help line. Similarly, a *horizontal strip* of a box B is a rectangle H contained in B where the left edge of H is contained in $l(B)$ and the right edge of H is contained in $r(B)$ and whose top and bottom edges are contained in a main line or a help line. For a vertical strip T of a box B, we denote by $w(T)$ the length of the bottom (or top) edge of T, and for a horizontal strip H of a box B, we denote by $h(H)$ the length of the left (or right) edge of H.

We interpret vertical strips of a box B as the portion of B that lies between two consecutive vertical help lines, or possibly between a vertical help line and a vertical main line. We similarly interpret horizontal strips of a box to be the portion of the box between two consecutive horizontal help lines, or possibly between a horizontal help line and a horizontal main line. Note that, with this interpretation, elementary boxes are precisely those that consist of one vertical strip and one horizontal strip.

3.2 Structural Properties

The main structural property we aim to show is that in a subproblem for a box $B \in \mathcal{B}(S)$, there is a good vertical or horizontal separator contained in a main or help line. In particular, in order to ensure subproblems of low complexity, we aim to maintain the following invariant.

Definition 3.2. A box $B \in \mathscr{B}(S)$ satisfies the *box invariant* if each of its edges is intersected by at most two edges of $\mathscr{C}_k^{\text{OPT}}(S)$.

Hence, we make some observations and prove some lemmas that aid us in this goal. The following observation exploits the general position assumption that no two points in S have the same x- or y-coordinate.

Observation 3.3. *Let $B, B_1, B_2 \in \mathscr{B}(S)$ be boxes such that $B = B_1 \cup B_2$ and B_1 and B_2 have disjoint interiors. Let $\mathscr{C}$ be any clustering of S. Then $\Phi_B(\mathscr{C}) = \Phi_{B_1}(\mathscr{C}) + \Phi_{B_2}(\mathscr{C})$.*

We also have the property that no point lies on the boundary of a box B. Due to the way we have defined the main lines,

Observation 3.4. *No box $B \in \mathscr{B}(S)$ contains a point from S on its boundary. Furthermore, if the convex hull C of a cluster of $\mathscr{C}_k^{\text{OPT}}(S)$ (or an edge e of such a convex hull) intersects the boundary of a box $B \in \mathscr{B}(S)$, then C (or the edge e) is not fully contained in B.*

The following lemma states that, for an optimal solution, the total cost of the portion of its edges that lie in a box B cannot exceed the perimeter of the box.

Lemma 3.5. *Let $B \in \mathscr{B}(S)$. Then $\Phi_B(\mathscr{C}_k^{\text{OPT}}(S)) \leqslant 2w(B) + 2h(B)$.*

Proof. Suppose for a contradiction that this is not the case. Let $S_1, \dots, S_k$ be the clusters of $\mathscr{C}_k^{\text{OPT}}(S)$ whose convex hulls intersect B. Let $P := B \cup \bigcup_{i=1}^k \text{CH}(S_i)$. Now, P has a perimeter strictly smaller than $\sum_{i=1}^k \text{per}(S_i)$. However, the merged cluster $\bigcup_{i=1}^k S_i$ has a convex hull with a perimeter at most as large as the boundary P, which is a contradiction. $\square$

The following lemma is a key ingredient in our algorithm, and it is the main structural property we show in this section.

Lemma 3.6. *If $B \in \mathscr{B}(S)$ is an elementary box, then there are at most two edges of $\mathscr{C}_k^{\text{OPT}}(S)$ intersecting B (in particular, B satisfies the box invariant). Furthermore, any non-elementary box $B \in \mathscr{B}(S)$ satisfying the box invariant has a good vertical or horizontal separator s contained in a main or help line such that s divides B into two boxes B_l and B_r, both of which satisfy the box invariant.*

We prove the following sequence of lemmas, which we later show how to combine to yield a proof of Lemma 3.6.

Lemma 3.7. *Consider any box $B \in \mathscr{B}(S)$. Let X be an arbitrary partition of B into vertical strips, and let Y be an arbitrary partition of B into horizontal strips. Moreover, let $A := \{A_1, \dots, A_m\}, Z = \{Z_1, \dots, Z_q\}$ be arbitrary nonempty subsets of X, Y, respectively. Then either there is a vertical separator of B contained in some vertical strip $A_i \in A$ or a horizontal separator of B contained in some horizontal strip $Z_j \in Z$ that intersects at most $2\sqrt{2}\frac{w(B)+h(B)}{\sum_{A_i \in A} w(A_i) + \sum_{Z_j \in Z} h(Z_j)}$ edges of $\mathscr{C}_k^{\text{OPT}}(S)$.*

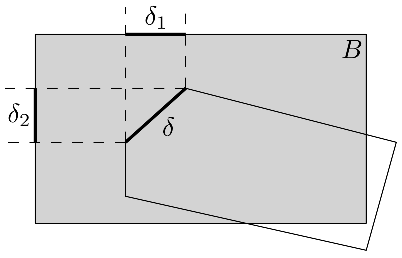


Figure 3.1: Contribution of an edge of length δ to the sum of integrals in the proof of Lemma 3.7 is at most $\sqrt{2}\delta$.

Proof. Define x_1, x_2 to be the values such that the edges $l(B)$ and $r(B)$ are contained in the vertical lines $x = x_1$ and $x = x_2$, respectively. Similarly, define y_1, y_2 to be the values such that the edges $b(B)$ and $t(B)$ are contained in the horizontal lines $y = y_1$ and $y = y_2$, respectively. Define l_i and r_i to be the x-coordinates of the left and right edge, respectively, of the vertical strip A_i. Similarly, define b_j and t_j to be the y-coordinates of the bottom and top edge, respectively, of the horizontal strip Z_j.

Suppose for a contradiction that all vertical separators that are contained in some vertical strip $A_i \in A$ and all horizontal separators that are contained in some horizontal strip $Z_j \in Z$ intersect strictly more than $2\sqrt{2}\frac{w(B)+h(B)}{\sum_{A_i \in A} w(A_i)+\sum_{Z_j \in Z} h(Z_j)}$ edges of $\mathscr{C}_k^{\text{OPT}}(S)$. Define the function $f(x)$ to be the number of edges of $\mathscr{C}_k^{\text{OPT}}(S)$ that intersect the vertical separator with x-coordinate x (for each $x_1 \leqslant x \leqslant x_2$). Similarly, define the function $g(y)$ to be the number of edges of $\mathscr{C}_k^{\text{OPT}}(S)$ that have at least some portion in B and intersect the horizontal separator with y-coordinate y (for each $y_1 \leqslant y \leqslant y_2$).

Consider an edge e (or portion of an edge) of $\mathscr{C}_k^{\text{OPT}}(S)$ that lies in B, and suppose it has length δ. We want to understand its contribution to the sum of integrals $\int_{x_1}^{x_2} f(x)dx + \int_{y_1}^{y_2} g(y)dy$. In particular, we want to understand the maximum contribution possible by such a line segment (i.e., edge). To this end, consider the right triangle formed by the two endpoints of the line segment (which is the hypotenuse) and the intersection of the vertical line passing through the upper endpoint and the horizontal line passing through the lower endpoint. Let δ_1 denote the length of one leg and δ_2 denote the length of the other leg, and observe that $\delta_1 + \delta_2$ is precisely the contribution of edge e to the sum of integrals. Hence, we wish to maximize $\delta_1 + \delta_2$ subject to the constraint $\delta^2 = \delta_1^2 + \delta_2^2$, which yields $\delta_1 = \delta_2 = \frac{\delta}{\sqrt{2}}$ (obtained when the triangle is a right isosceles triangle). This implies that an edge of length δ contributes at most $\sqrt{2}\delta$ to the sum of integrals (see Figure 3.1).

By Lemma 3.5, we know that $\Phi_B(\mathscr{C}_k^{\text{OPT}}(S)) \leqslant 2w(B) + 2h(B)$. Hence, we obtain a contradiction as follows:

$$\begin{aligned}
2w(B)+2h(B) &\geqslant \Phi_B(\mathscr{C}_k^{\mathrm{OPT}}(S))\\
&\geqslant \frac{\int_{x_1}^{x_2} f(x)dx + \int_{y_1}^{y_2} g(y)dy}{\sqrt{2}}\\
&\geqslant \frac{\sum_{i=1}^{m}\int_{l_i}^{r_i} f(x)dx + \sum_{j=1}^{q}\int_{b_j}^{t_j} g(y)dy}{\sqrt{2}}\\
&> \left(2\frac{w(B)+h(B)}{\sum_{A_i\in A} w(A_i)+\sum_{Z_j\in Z} h(Z_j)}\right)\left(\sum_{A_i\in A} w(A_i)+\sum_{Z_j\in Z} h(Z_j)\right)\\
&= 2w(B)+2h(B).
\end{aligned}$$

Here, the second inequality follows from the maximum amount that an edge (or portion of an edge) of $\mathscr{C}_k^{\mathrm{OPT}}(S)$ can contribute to the sum of integrals, the third inequality follows since we are integrating over a smaller domain, and the last inequality follows from our assumption that each vertical separator that belongs to some vertical strip intersects many edges (and similarly for such horizontal separators). □

We first consider boxes that have many small vertical strips and many small horizontal strips (i.e., many main lines and help lines). We argue that such boxes have either a good vertical separator or a good horizontal separator in the following lemma.

Lemma 3.8. *Let B be any box satisfying the box invariant and consider the vertical strips induced by the vertical help and main lines going through B, along with the horizontal strips induced by the horizontal help and main lines going through B. Moreover, suppose that for every such vertical strip T we have $w(T) \leqslant \frac{1}{200}w(B)$ and for all such horizontal strips H we have $h(H) \leqslant \frac{1}{200}h(B)$. Then there exists a good vertical separator of B that is contained in a vertical help or main line, or there exists a good horizontal separator of B with similar properties.*

Proof. Consider any box B with the given properties. We consider each portion of B that lies between two consecutive vertical help lines (or between a vertical help line and a vertical main line) to be a vertical strip of B. We similarly consider the horizontal strips of B induced by the corresponding horizontal help and main lines.

One of the vertical strips has a left edge that is precisely $l(B)$, and another one of the vertical strips has a right edge that is precisely $r(B)$. Since we aim to split the box B into two smaller boxes with our separator, we discard these two vertical strips. Moreover, we also discard any vertical strip T that has an edge of $\mathscr{C}_k^{\mathrm{OPT}}(S)$ intersecting the top edge of T or the bottom edge of T. Because B satisfies the box invariant, there are at most four such edges of $\mathscr{C}_k^{\mathrm{OPT}}(S)$. Each edge of $\mathscr{C}_k^{\mathrm{OPT}}(S)$ can result in discarding at most two vertical strips (in case the edge intersects the corner of a vertical strip, in which case two strips are affected). Hence, in all, we discard at most 10 vertical strips. Similar reasoning applies to horizontal strips, resulting in discarding at most 10 horizontal strips. Note that the total width of vertical strips that are not discarded is at least $(1-\frac{10}{200})w(B) = \frac{19}{20}w(B)$, and similarly the total height of horizontal strips that are not discarded is at least $\frac{19}{20}h(B)$.

By Lemma 3.7, either there exists a vertical separator of B contained in some non-discarded vertical strip or a non-discarded horizontal separator of B contained in some horizontal strip that intersects at most $2\sqrt{2}\frac{w(B)+h(B)}{\frac{19}{20}(w(B)+h(B))} < 3$ edges of $\mathscr{C}_k^{\text{OPT}}(S)$. Suppose there is some non-discarded vertical strip T containing a good vertical separator (as the other case is symmetric). Since T was not discarded, its left edge is not $l(B)$ and its right edge is not $r(B)$. Moreover, since there are no edges of $\mathscr{C}_k^{\text{OPT}}(S)$ intersecting the top or bottom edge of T, there can be no edge of $\mathscr{C}_k^{\text{OPT}}(S)$ that intersects the left or right edge of T without also intersecting the good vertical separator contained in T. Hence, either of the left or right edge of T serves as a good vertical separator of B. □

We now argue that boxes that consist of many vertical strips (induced by the corresponding vertical help or main lines) and are much wider than they are tall have a good vertical separator. A similar result holds in the horizontal direction.

Lemma 3.9. *Let B be any box satisfying the box invariant and consider the vertical strips induced by the vertical help (and main) lines going through B, along with the horizontal strips induced by the horizontal help (and main lines) going through B. Moreover, suppose that $w(B) > 4h(B)$ and all vertical strips T have $w(T) \leqslant \frac{1}{200}w(B)$. Then there exists a good vertical separator of B that is contained in a vertical help or main line. Likewise, if $h(B) > 4w(B)$ and all horizontal strips H have $h(H) \leqslant \frac{1}{200}h(B)$, then there exists a good horizontal separator of B with similar properties.*

Proof. Consider any such box B, and suppose $w(B) > 4h(B)$, as the other case is proved similarly. We consider each portion of B that lies between two consecutive vertical help lines (or between a vertical help line and a vertical main line) to be a vertical strip of B. Assume that all such vertical strips T satisfy $w(T) \leqslant \frac{1}{200}w(B)$. We proceed in a manner similar to the proof of Lemma 3.8.

One of these vertical strips has a left edge that is precisely $l(B)$, and another one of these vertical strips has a right edge that is precisely $r(B)$. Since we aim to split the box B into two smaller boxes with our separator, we discard these two vertical strips. Moreover, we also discard any vertical strip T that has an edge of $\mathscr{C}_k^{\text{OPT}}(S)$ intersecting the top edge of T or the bottom edge of T. Because B satisfies the box invariant, there are at most four such edges of $\mathscr{C}_k^{\text{OPT}}(S)$ (since at most two such edges intersect $t(B)$, and at most two such edges intersect $b(B)$). Each such edge of $\mathscr{C}_k^{\text{OPT}}(S)$ can result in discarding at most two vertical strips (in case the edge intersects the corner of a vertical strip, in which case two strips are affected). Hence, in all, we discard at most 10 vertical strips. Note that the total width of vertical strips that are not discarded is at least $\frac{19}{20}w(B)$.

Now, suppose for a contradiction that all vertical separators in the remaining vertical strips intersect at least three edges of $\mathscr{C}_k^{\text{OPT}}(S)$. This implies that the total length of such edges is at least $\frac{57}{20}w(B)$. By Lemma 3.5, we know that $\Phi_B(\mathscr{C}_k^{\text{OPT}}(S)) \leqslant 2w(B) + 2h(B)$. Hence, we obtain a contradiction:

$$\frac{57}{20}w(B) \leqslant \Phi_B(\mathscr{C}_k^{\text{OPT}}(S)) \leqslant 2w(B) + 2h(B) \leqslant 2w(B) + \frac{w(B)}{2} = \frac{5}{2}w(B).$$

Hence, there is some remaining vertical strip T that contains a good vertical separator which implies as before that the left or right edge of T serves as a good vertical separator. □

We now consider the case regarding boxes that either have some wide vertical strip and are not much taller than they are wide, or have some tall horizontal strip and are not much wider than they are tall.

Lemma 3.10. *Let B be any box and consider the vertical strips induced by the vertical help (and main) lines going through B, along with the horizontal strips induced by the horizontal help (and main lines) going through B. Moreover, suppose that $h(B) \leqslant 4w(B)$ and there exists some vertical strip T with $w(T) > \frac{1}{200}w(B)$. Then the left edge of T has at most two edges of $\mathscr{C}_k^{\text{OPT}}(S)$ intersecting it, and the same holds for the right edge of T. Similarly, if $w(B) \leqslant 4h(B)$ and there exists some horizontal strip H with $w(H) > \frac{1}{200}h(B)$, then each of the bottom and top edges of H has at most two edges of $\mathscr{C}_k^{\text{OPT}}(S)$ intersecting them.*

Proof. Consider any such box B, and suppose $h(B) \leqslant 4w(B)$, as the other case is proved similarly. We consider each portion of B that lies between two consecutive vertical help lines (or between a vertical help line and a vertical main line) to be a vertical strip of B. Assume that there exists some vertical strip T with $w(T) > \frac{1}{200}w(B)$.

Now, consider the two points p_1, p_2 that give rise to the vertical strip T induced by the corresponding vertical help and main lines, and let x_1, x_2 be the x-coordinates of p_1, p_2 (respectively), with $x_1 < x_2$. Observe that $h(B) \leqslant 4w(B) < 800w(T) = 800\frac{x_2-x_1}{20000} = \frac{(x_2-x_1)}{25}$, where the inequalities follow from our assumptions in the lemma and the first equality follows from the fact that we divide the interval $[x_1, x_2]$ into 20000 segments of equal length (induced by the vertical help lines).

Suppose for a contradiction that the left or right edge of T has at least three edges of $\mathscr{C}_k^{\text{OPT}}(S)$ intersecting it. Let s denote this vertical edge, and let u denote the x-coordinate of s. We must have that at least one of x_1, x_2 must

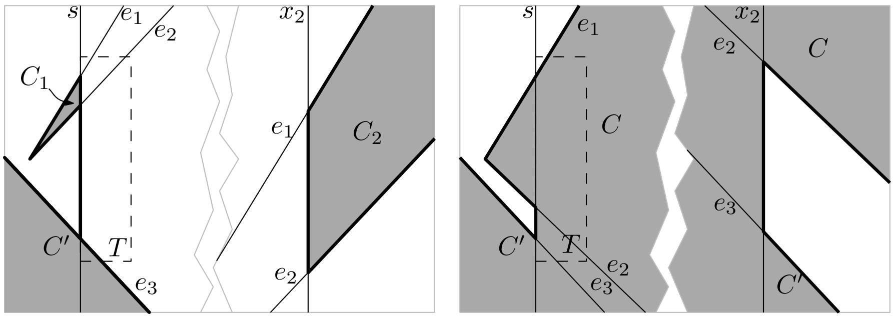


Figure 3.2: The two cases in the proof of Lemma 3.10, $\theta_{1,2} \leqslant \frac{\pi}{2}$ (left) and $\theta_{2,3} \leqslant \frac{\pi}{2}$ (right).

be far away from u (note that $x_1 \leqslant u \leqslant x_2$). In particular, either $x_2 \geqslant u + \frac{x_2 - x_1}{2}$ or $x_1 \leqslant u - \frac{x_2 - x_1}{2}$. We consider the former case as the latter is symmetric, so suppose $x_2 \geqslant u + \frac{x_2 - x_1}{2}$.

Let e_1, e_2, e_3 denote three consecutive edges of $\mathscr{C}_k^{\text{OPT}}(S)$ that intersect the line segment s, sorted in decreasing order according to the height of the point of intersection with s (e.g., e_1 is above e_2, which is above e_3 at $x = u$). Note that it cannot be the case that all three edges are contained in the boundary of the convex hull of the same cluster, and hence the three edges lie on the boundaries of two convex hulls (induced by two clusters of $\mathscr{C}_k^{\text{OPT}}(S)$). Without loss of generality, assume that there exists an optimal cluster C such that $\text{CH}(C)$ contains both e_1 and e_2 (the case when e_2 and e_3 are on the same boundary is symmetric). Moreover, let C' denote the cluster giving rise to the boundary on which e_3 lies.

We first consider the case when the slope of e_1 is strictly more than that of e_2, and the slope of e_2 is strictly more than that of e_3. We imagine extending the line segments e_1, e_2 to the left until they meet at some point $r_{1,2}$ (they must meet at some point), and denote by $\theta_{1,2}$ the angle in radians formed as a result of this extension. We can do a similar process for edges e_2, e_3 to obtain the angle $\theta_{2,3}$, and also for edges e_1, e_3 to obtain an angle $\theta_{1,3}$. Observe that $\theta_{1,3} \leqslant \pi$, implying that either $\theta_{1,2} \leqslant \frac{\pi}{2}$ or $\theta_{2,3} \leqslant \frac{\pi}{2}$ (see Figure 3.2).

Consider the scenario where $\theta_{1,2} \leqslant \frac{\pi}{2}$. We transform the solution as follows. First, we split the cluster C into two clusters, where we take all points in C with an x-coordinate of at most u to one cluster C_1, and all points in C with an x-coordinate of at least x_2 to another cluster C_2 (note that there are no points with an x-coordinate in the open interval (u, x_2)). Lastly, we merge the clusters C_1 and C' by taking the union $C_1 \cup C'$. We note that the total number of clusters, after performing the split and merge, remains unchanged. Hence, we need only argue that the cost after splitting and merging does not result in an increase in cost.

To this end, observe that the points in C_1 all lie inside the polygon P_1

obtained by considering the original cluster C, cutting it with the vertical line $x = u$, and taking the left portion (i.e., all points in $\mathrm{CH}(C)$ that have an x-coordinate of at most u). By a similar argument, the points in C_2 all lie inside the polygon P_2 obtained by considering the original cluster C, cutting it with the vertical line $x = x_2$, and taking the right portion (i.e., all points in $\mathrm{CH}(C)$ that have an x-coordinate of at least x_2). Now, consider the polygon Q obtained by merging the polygons P_1 and $\mathrm{CH}(C')$ as follows: we consider a vertical line segment at $x = u$ going from edge e_2 to edge e_3 (note that both edges intersect line s at $x = u$). Observe that Q contains the points in $C_1 \cup C'$ and P_2 contains the points in C_2, and hence $\mathrm{per}(C_1 \cup C')$ is at most the perimeter of Q and $\mathrm{per}(C_2)$ is at most the perimeter of P_2. Hence, we need only bound the perimeter of Q and P_2.

Observe that the combined perimeter of Q and P_2 is at most $\mathrm{per}(C)+\mathrm{per}(C')$, plus the lengths of the vertical line segments at $x = u$ and $x = x_2$ going from edge e_1 to e_2, plus twice the length of the vertical line segment at $x = u$ going from edge e_2 to e_3, minus the lengths of the portion of edges e_1 and e_2 with x-coordinates in between $x = u$ and $x = x_2$. For ease of notation, we denote by $b_{1,2}$ the length of the vertical line segment at $x = x_2$ (going from e_1 to e_2), by ℓ_1 the length of e_1 in the interval $[u, x_2]$, and by ℓ_2 the length of e_2 in the interval $[u, x_2]$. First, the vertical line segments at $x = u$ (which, joined together, go from e_1 to e_3 at $x = u$) contribute at most $2h(B)$ to the perimeter of Q. Now, we upper bound $b_{1,2}$ by considering the triangle formed by the following three points: $r_{1,2}$, the intersection of e_1 with the vertical line $x = x_2$, and the intersection of e_2 with the vertical line $x = x_2$. Let β denote the angle formed at the intersection of e_1 with $x = x_2$, and γ denote the angle formed at the intersection of e_2 with $x = x_2$. We have that $b_{1,2} = h(B)+\ell_1\cos(\beta)+\ell_2\cos(\gamma)$. Since $\theta_{1,2} \leqslant \frac{\pi}{2}$, we know that $\max\{\beta,\gamma\} \geqslant \frac{\pi}{4}$. This implies $b_{1,2} = h(B) + \ell_1\cos(\beta) + \ell_2\cos(\gamma) \leqslant h(B) + \max\{\ell_1 + \frac{\ell_2}{\sqrt{2}}, \frac{\ell_1}{\sqrt{2}} + \ell_2\}$. Hence, the new solution's cost, given by the sum of the perimeters of Q and P_2, is at most $\mathrm{per}(C) + \mathrm{per}(C') + 3h(B) + \max\{\ell_1 + \frac{\ell_2}{\sqrt{2}}, \frac{\ell_1}{\sqrt{2}} + \ell_2\} - \ell_1 - \ell_2$. We bound this expression as follows:

$$\begin{aligned}
&3h(B) + \max\left\{\ell_1 + \frac{\ell_2}{\sqrt{2}}, \frac{\ell_1}{\sqrt{2}} + \ell_2\right\} - \ell_1 - \ell_2 \\
&\qquad \leqslant 3h(B) - \left(1 - \frac{1}{\sqrt{2}}\right)\min\{\ell_1,\ell_2\} \leqslant 3h(B) - \left(1 - \frac{1}{\sqrt{2}}\right)(x_2 - u),
\end{aligned}$$

where the last inequality follows from the fact that both ℓ_1 and ℓ_2 are at least $x_2 - u$. Hence, as long as $3h(B) - (1 - \frac{1}{\sqrt{2}})(x_2 - u) < 0$, we have a contradiction. This holds as $h(B) < \frac{(x_2-x_1)}{25}$ and $x_2 - u \geqslant \frac{x_2-x_1}{2}$, implying $3h(B) - (1 - \frac{1}{\sqrt{2}})(x_2 - u) < 0$.

In the case that $\theta_{1,2} > \frac{\pi}{2}$, then we know $\theta_{2,3} \leqslant \frac{\pi}{2}$. In this setting, we obtain a new solution by merging the two clusters C and C' to get $C \cup C'$, and argue that this new solution is cheaper. Consider the polygon P obtained by removing the portion of edges e_2 and e_3 in between $x = u$ and $x = x_2$, and then joining

C and C' on the left by a vertical line segment going from the point at which e_2 intersects s to the point at which e_3 intersects s (at $x = u$). Similarly, we join C and C' on the right by a vertical line segment between the two points at which edges e_2 and e_3 intersect at $x = x_2$. Now, since P contains all points in $C \cup C'$, we have $\text{per}(C \cup C')$ is at most the perimeter of P. Moreover, we have that the perimeter of P is at most $\text{per}(C) + \text{per}(C')$, plus the lengths of the left and right vertical segments we added, minus ℓ_2 and ℓ_3. The left vertical segment contributes at most $h(B)$ to the perimeter (since the left vertical line segment is contained in s). By a similar argument as earlier, the right vertical segment is at most $h(B) + \max\{\ell_2 + \frac{\ell_3}{\sqrt{2}}, \frac{\ell_2}{\sqrt{2}} + \ell_3\}$ (using the fact that $\theta_{2,3} \leqslant \frac{\pi}{2}$ in this case). Hence, using similar reasoning as before, the perimeter of P is at most $\text{per}(C) + \text{per}(C') + 2h(B) + \max\left\{\ell_2 + \frac{\ell_3}{\sqrt{2}}, \frac{\ell_2}{\sqrt{2}} + \ell_3\right\} - \ell_2 - \ell_3$. We bound this expression as follows:

$$\begin{aligned} 2h(B) + \max\left\{\ell_2 + \frac{\ell_3}{\sqrt{2}}, \frac{\ell_2}{\sqrt{2}} + \ell_3\right\} - \ell_2 - \ell_3 &\leqslant 2h(B) - \left(1 - \frac{1}{\sqrt{2}}\right)\min\{\ell_2, \ell_3\} \\ &\leqslant 2h(B) - \left(1 - \frac{1}{\sqrt{2}}\right)(x_2 - u). \end{aligned}$$

We again have that this quantity is strictly less than zero, a contradiction.

In the case when either the slope of e_1 is at most the slope of e_2, or the slope of e_2 is at most the slope of e_3, the proof is simpler. In particular, in the former case, we can more easily bound the vertical line segment we add along the line $x = x_2$ by $h(B)$ (instead of some function of ℓ_1, ℓ_2). This holds since the gap between edges e_1 and e_2 shrinks when going from $x = u$ to $x = x_2$. The same holds in the latter case when the slope of e_2 is at most the slope of e_3. □

Finally, as a type of base case, we argue that boxes that are not much taller than they are wide and consist of one vertical strip have at most two edges of $\mathscr{C}_k^{\text{OPT}}(S)$ intersecting the whole box. Likewise, boxes that are not much wider than they are tall and consist of one horizontal strip satisfy the same property.

Lemma 3.11. *Let B be any box with $h(B) \leqslant 4w(B)$ such that no vertical help or main line has an x-coordinate strictly in between the x-coordinates induced by $l(B)$ and $r(B)$. Then there are at most two edges of $\mathscr{C}_k^{\text{OPT}}(S)$ intersecting box B. Similarly, if B is any box with $w(B) \leqslant 4h(B)$ such that no horizontal help or main line has a y-coordinate strictly in between the y-coordinates induced by $b(B)$ and $t(B)$, then there are at most two edges of $\mathscr{C}_k^{\text{OPT}}(S)$ intersecting box B.*

Proof. Consider any such box B with $h(B) \leqslant 4w(B)$ (as the proof of the other case is symmetric). We consider each portion of B that lies between two consecutive vertical help lines (or between a vertical help line and a vertical main line) to be a vertical strip of B. In particular, the assumption in the lemma regarding vertical help and main lines implies that B consists of exactly one vertical strip T (induced by $l(B)$ and $r(B)$), and hence we have $w(B) = w(T)$. The following proof uses similar ideas as in the proof of Lemma 3.10.

Now, consider the two points p_1, p_2 that give rise to the vertical strip T, and let x_1, x_2 be the x-coordinates of p_1, p_2 (respectively), with $x_1 < x_2$. Observe that $h(B) \leqslant 4w(B) = 4w(T) = 4\frac{x_2-x_1}{20000} < \frac{x_2-x_1}{200}$, where the first inequality and the first equality follow from our assumptions in the lemma, and the second equality follows from the fact that we divide the interval $[x_1, x_2]$ into 20000 segments of equal length (induced by the vertical help lines).

Suppose for a contradiction that there are (at least) three edges of $\mathscr{C}_k^{\mathrm{OPT}}(S)$ intersecting the interior of B. Let u_1, u_2 denote the x-coordinates induced by the line segments $l(B)$ and $r(B)$, respectively. We must have either $x_2 \geqslant \frac{u_1+u_2}{2} + \frac{x_2-x_1}{2}$ or $x_1 \leqslant \frac{u_1+u_2}{2} - \frac{x_2-x_1}{2}$. Otherwise, we get a contradiction: $x_2 - x_1 < \frac{u_1+u_2}{2} + \frac{x_2-x_1}{2} - \frac{u_1+u_2}{2} + \frac{x_2-x_1}{2} = x_2 - x_1$. We consider the former case as the latter is symmetric, so suppose $x_2 \geqslant \frac{u_1+u_2}{2} + \frac{x_2-x_1}{2}$. In the former case, we focus on the vertical line $x = u_1$, namely the line containing $l(B)$ (in the latter case, we focus on the vertical line $x = u_2$, namely the line containing $r(B)$).

Let e_1, e_2, e_3 denote three edges of $\mathscr{C}_k^{\mathrm{OPT}}(S)$ that intersect the interior of box B. Each such edge must also intersect the vertical lines $x = u_1$ and $x = x_2$. We consider three such edges that are consecutive, sorted in decreasing order according to the height of the point of intersection with the line $x = u_1$ (e.g., e_1 is above e_2, which is above e_3 at $x = u_1$). Note that it cannot be the case that all three edges are contained in the boundary of the convex hull of the same cluster, and hence the three edges lie on the boundaries of at least two convex hulls (induced by at least two clusters of $\mathscr{C}_k^{\mathrm{OPT}}(S)$). Without loss of generality, assume that there exists an optimal cluster C such that $\mathrm{CH}(C)$ contains both e_1 and e_2 (the case when e_2 and e_3 are on the same boundary is symmetric). We also denote by C' the cluster giving rise to the boundary on which e_3 lies.

For ease of notation, we denote by ℓ_1, ℓ_2, ℓ_3 the lengths of e_1, e_2, e_3 in the interval $[u_1, x_2]$, and by g_1, g_2, g_3 the lengths of e_1, e_2, e_3 in an arbitrary interval of length $\frac{x_2-x_1}{20000}$ in $[u_1, x_2]$. We also denote by $a_{1,2}$ the length of the vertical line segment at $x = u_1$ connecting e_1 and e_2, by $a_{2,3}$ the segment at $x = u_1$ connecting e_2 and e_3, and $a_{1,3} = a_{1,2} + a_{2,3}$ (i.e., the length of the vertical line segment at $x = u_1$ going from e_1 to e_3). We define analogous lengths of vertical line segments $b_{1,2}$ and $b_{2,3}$ at $x = x_2$ between edges e_1, e_2 and edges e_2, e_3, respectively. We note that $\ell_i \geqslant 1000 g_i$ for $1 \leqslant i \leqslant 3$, since there are at least 1000 disjoint intervals of length $\frac{x_2-x_1}{20000}$ in the interval $[u_1, x_2]$ (using the fact that $x_2 \geqslant \frac{u_1+u_2}{2} + \frac{x_2-x_1}{2}$), and in each such interval the length of e_i is precisely g_i.

We first argue that $a_{i,j} \leqslant g_i + g_j + h(B)$ for all $1 \leqslant i < j \leqslant 3$. For any two such edges e_i, e_j we argue this by considering the following three line segments. We can go along e_i from the intersection of edge e_i with $x = u_1$ to any intersection point of e_i with box B, and then to any intersection point of e_j with box B, and finally go along e_j back to $x = u_1$. The cost of connecting e_i to e_j along $x = u_1$ is at most the lengths of the projection of these three line segments onto $x = u_1$, which sum to at most $g_i + g_j + h(B)$.

Moreover, we show a tighter bound of $a_{i,j} \leqslant \min\{g_i, g_j\} + h(B)$ for all $1 \leqslant i < j \leqslant 3$ satisfying the property that the slope of e_i is strictly more than that

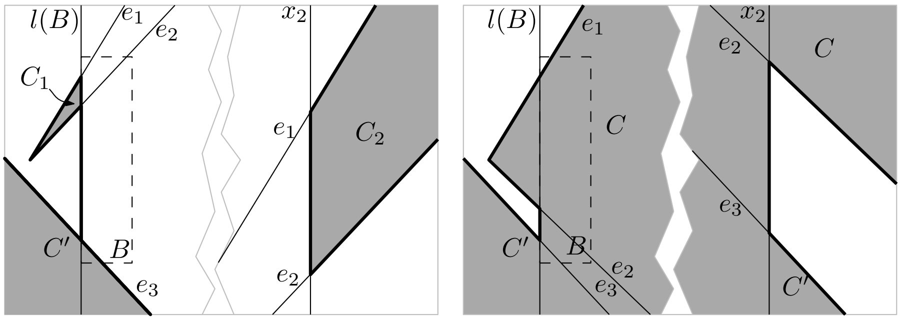


Figure 3.3: The two cases in the proof of Lemma 3.11, $\theta_{1,2} \leqslant \frac{\pi}{2}$ (left) and $\theta_{2,3} \leqslant \frac{\pi}{2}$ (right).

of e_j. If the slopes of e_i, e_j are positive and negative, respectively, then the intersections of e_i, e_j with $x = u_1$ must both lie in B, and hence the cost of connecting them via the vertical line segment at $x = u_1$ is $a_{i,j} \leqslant h(B)$. If the slope of e_i is negative, in which case the slope of e_j is also negative, then the intersection points of both edges with $x = u_1$ must lie above $b(B)$, and hence we can obtain the bound via the following two line segments. We can go from the intersection of edge e_i with $x = u_1$ to any point at which it intersects box B, and then go to the intersection of $b(B)$ with $x = u_1$. The cost of connecting e_i to e_j along $x = u_1$ is at most the lengths of the projection of these two line segments onto $x = u_1$, which sum to at most $g_i + h(B)$ (note that $g_i = \min\{g_i, g_j\}$, since both slopes are negative and the slope of e_i is strictly more than that of e_j). A symmetric argument shows that, in the case that e_j is positive (in which case the slope of e_i is also positive), we have $a_{i,j} \leqslant g_j + h(B)$ (we symmetrically have $g_j = \min\{g_i, g_j\}$).

We first consider the case when the slope of e_1 is strictly more than that of e_2, and the slope of e_2 is strictly more than that of e_3. This implies $a_{1,2} \leqslant \min\{g_1, g_2, g_3\} + h(B)$ and $a_{2,3} \leqslant \min\{g_1, g_2, g_3\} + h(B)$. In particular, we know $a_{1,2} \leqslant \min\{g_1, g_2\} + h(B)$, and we also know $a_{1,2} \leqslant a_{1,3} \leqslant \min\{g_1, g_3\} + h(B)$, implying $a_{1,2} \leqslant \min\{g_1, g_2, g_3\} + h(B)$. Symmetrically, we have $a_{2,3} \leqslant \min\{g_1, g_2, g_3\} + h(B)$.

Now, we imagine extending the line segments e_1, e_2 to the left until they meet at some point $r_{1,2}$ (they must meet at some point), and denote by $\theta_{1,2}$ the angle in radians formed as a result of extension. We can do a similar process for edges e_2, e_3 to obtain the angle $\theta_{2,3}$ (again, extending line segments e_2 and e_3 to the left must result in an intersection), and also for edges e_1, e_3 to obtain an angle $\theta_{1,3}$. Observe that $\theta_{1,3} \leqslant \pi$, implying that either $\theta_{1,2} \leqslant \frac{\pi}{2}$ or $\theta_{2,3} \leqslant \frac{\pi}{2}$ (see Figure 3.3).

Consider the scenario where $\theta_{1,2} \leqslant \frac{\pi}{2}$. We transform the solution as follows.

First, we split the cluster C into two clusters, where we take all points in C with an x-coordinate of at most u_1 to one cluster C_1, and all points in C with an x-coordinate of at least x_2 to another cluster C_2 (note that there are no points with an x-coordinate in the open interval (u_1, x_2)). Lastly, we merge the clusters C_1 and C' by taking the union $C_1 \cup C'$. We note that the total number of clusters, after performing the split and merge, remains unchanged. Hence, we need only argue that the cost after splitting and merging does not result in an increase in cost.

To this end, observe that the points in C_1 all lie inside the polygon P_1 obtained by considering the original cluster C, cutting it with the vertical line $x = u_1$, and taking the left portion (i.e., all points in $\mathrm{CH}(C)$ that have an x-coordinate of at most u_1). By a similar argument, the points in C_2 all lie inside the polygon P_2 obtained by considering the original cluster C, cutting it with the vertical line $x = x_2$, and taking the right portion (i.e., all points in $\mathrm{CH}(C)$ that have an x-coordinate of at least x_2). Now, consider the polygon Q obtained by merging the polygons P_1 and $\mathrm{CH}(C')$ via a vertical line segment at $x = u_1$ going from edge e_2 to edge e_3. Observe that Q contains the points in $C_1 \cup C'$ and P_2 contains the points in C_2, and hence $\mathrm{per}(C_1 \cup C')$ is at most the perimeter of Q and $\mathrm{per}(C_2)$ is at most the perimeter of P_2. Moreover, the combined perimeters of Q and P_2 is at most $\mathrm{per}(C) + \mathrm{per}(C') + a_{1,2} + 2a_{2,3} + b_{1,2} - \ell_1 - \ell_2$.

Now, we upper bound $b_{1,2}$ by considering the triangle formed by the following three points: $r_{1,2}$, the intersection of e_1 with the vertical line $x = x_2$, and the intersection of e_2 with the vertical line $x = x_2$. Let β denote the angle formed at the intersection of e_1 with $x = x_2$, and γ denote the angle formed at the intersection of e_2 with $x = x_2$. We have that $b_{1,2} = a_{1,2} + \ell_1 \cos(\beta) + \ell_2 \cos(\gamma)$. Since $\theta_{1,2} \leqslant \frac{\pi}{2}$, we know that $\max\{\beta, \gamma\} \geqslant \frac{\pi}{4}$. This implies $b_{1,2} = a_{1,2} + \ell_1 \cos(\beta) + \ell_2 \cos(\gamma) \leqslant a_{1,2} + \max\{\ell_1 + \frac{\ell_2}{\sqrt{2}}, \frac{\ell_1}{\sqrt{2}} + \ell_2\}$. Hence, we obtain

$$\begin{aligned}
a_{1,2} + 2a_{2,3} + b_{1,2} - \ell_1 - \ell_2 &\leqslant 2(a_{1,2} + a_{2,3}) + \max\left\{\ell_1 + \frac{\ell_2}{\sqrt{2}}, \frac{\ell_1}{\sqrt{2}} + \ell_2\right\} - \ell_1 - \ell_2 \\
&\leqslant 4(\min\{g_1, g_2\} + h(B)) - \left(1 - \frac{1}{\sqrt{2}}\right)\min\{\ell_1, \ell_2\} \\
&\leqslant 4(\min\{g_1, g_2\} + 4w(B)) - 1000\left(1 - \frac{1}{\sqrt{2}}\right)\min\{g_1, g_2\} \\
&< 0,
\end{aligned}$$

where the first inequality follows from substituting our upper bound on $b_{1,2}$, the second inequality follows from substituting for our upper bounds on $a_{1,2}$ and $a_{2,3}$ and simplifying, the third inequality follows by the assumption $h(B) \leqslant 4w(B)$, along with $\ell_1 \geqslant 1000g_1$ and $\ell_2 \geqslant 1000g_2$, and the last inequality follows from the observation that $w(B) \leqslant \min\{g_1, g_2\}$. Hence, the perimeter sum of Q and P_2 is at most $\mathrm{per}(C) + \mathrm{per}(C') + a_{1,2} + 2a_{2,3} + b_{1,2} - \ell_1 - \ell_2 < \mathrm{per}(C) + \mathrm{per}(C')$, a contradiction.

In the case that $\theta_{1,2} > \frac{\pi}{2}$, then we know $\theta_{2,3} \leqslant \frac{\pi}{2}$. In this setting, we obtain a new solution by merging the two clusters C and C' to get $C \cup C'$, and argue that this new solution is cheaper. Consider the polygon P obtained by removing the portion of edges e_2 and e_3 in between $x = u_1$ and $x = x_2$, and then joining C and C' on the left by a vertical line segment going from the point at which e_2 intersects $x = u_1$ to the point at which e_3 intersects $x = u_1$. Similarly, we join C and C' on the right by a vertical line segment between the two points at which edges e_2 and e_3 intersect at $x = x_2$. Now, since P contains all points in $C \cup C'$, we have $\text{per}(C \cup C')$ is at most the perimeter of P. Moreover, we have that the perimeter of P is at most $\text{per}(C) + \text{per}(C') + a_{2,3} + b_{2,3} - \ell_2 - \ell_3$.

By a similar argument as earlier, we have $b_{2,3} \leqslant a_{2,3} + \max\{\ell_2 + \frac{\ell_3}{\sqrt{2}}, \frac{\ell_2}{\sqrt{2}} + \ell_3\}$ (using the fact that $\theta_{2,3} \leqslant \frac{\pi}{2}$ in this case). Hence, using similar reasoning as before, we obtain

$$\begin{aligned}
a_{2,3} + b_{2,3} - \ell_2 - \ell_3 &\leqslant 2a_{2,3} + \max\left\{\ell_2 + \frac{\ell_3}{\sqrt{2}}, \frac{\ell_2}{\sqrt{2}} + \ell_3\right\} - \ell_2 - \ell_3 \\
&\leqslant 2(\min\{g_2, g_3\} + h(B)) - \left(1 - \frac{1}{\sqrt{2}}\right)\min\{\ell_2, \ell_3\} \\
&\leqslant 2(\min\{g_2, g_3\} + 4w(B)) - 1000\left(1 - \frac{1}{\sqrt{2}}\right)\min\{g_2, g_3\} < 0,
\end{aligned}$$

where the first inequality follows from substituting for $b_{2,3}$, the second inequality follows from substituting for $a_{2,3}$ and simplifying, the third inequality follows from the assumption $h(B) \leqslant 4w(B)$, along with $\ell_2 \geqslant 1000g_2$ and $\ell_3 \geqslant 1000g_3$, and the last inequality follows from $w(B) \leqslant \min\{g_2, g_3\}$. Thus, the perimeter of P is at most $\text{per}(C) + \text{per}(C') + a_{2,3} + b_{2,3} - \ell_2 - \ell_3 < \text{per}(C) + \text{per}(C')$, a contradiction.

In the case that the slope of e_2 is at most that of e_3, then we also argue that merging C and C' to get $C \cup C'$ yields a cheaper solution. In particular, in this case, we know that $b_{2,3} \leqslant a_{2,3}$ (as we go from $x = u_1$ to $x = x_2$, the vertical gap between the two edges shrinks since the slope of e_2 is smaller than the slope of e_3). Moreover, we have $a_{2,3} \leqslant g_2 + g_3 + h(B)$. By similar reasoning as before, we obtain

$$\begin{aligned}
a_{2,3} + b_{2,3} - \ell_2 - \ell_3 \leqslant 2a_{2,3} - \ell_2 - \ell_3 &\leqslant 2(g_2 + g_3 + 4w(B)) - \ell_2 - \ell_3 \\
&\leqslant 12\max\{g_2, g_3\} - 1000g_2 - 1000g_3 < 0,
\end{aligned}$$

which again yields a contradiction.

In the case that the slope of e_1 is at most that of e_2, but the slope of e_2 is strictly more than that of e_3, we do a similar transformation as previously seen by splitting C into two clusters C_1 and C_2, and then merging C_1 with C' to obtain the cluster $C_1 \cup C'$. As before, we need to argue $a_{1,2} + 2a_{2,3} + b_{1,2} - \ell_1 - \ell_2 < 0$. Note that $b_{1,2} \leqslant a_{1,2}$, since the vertical gap between e_1 and e_2 shrinks when going from $x = u_1$ to $x = x_2$. Moreover, we have $a_{2,3} \leqslant \min\{g_2, g_3\} + h(B) \leqslant g_2 + h(B)$

and $a_{1,2} \leqslant g_1 + g_2 + h(B)$. Hence, we have

$$\begin{aligned} a_{1,2} + 2a_{2,3} + b_{1,2} - \ell_1 - \ell_2 &\leqslant 2(a_{1,2} + a_{2,3}) - \ell_1 - \ell_2 \leqslant 2(g_1 + 2g_2 + 2h(B)) - \ell_1 - \ell_2 \\ &\leqslant 2(g_1 + 2g_2 + 8w(B)) - 1000g_1 - 1000g_2 \\ &\leqslant 20g_2 + 2g_1 - 1000g_1 - 1000g_2 < 0, \end{aligned}$$

yielding a contradiction. □

We are now ready to prove Lemma 3.6.

Proof of Lemma 3.6. Given any box $B \in B(S)$, we consider each portion of B that lies between two consecutive vertical help lines (or between a vertical help line and a vertical main line) to be a vertical strip of B. Similarly, we consider the horizontal strips induced by the horizontal help and main lines in B.

First, we consider the case when B is an elementary box. In this case, B is composed of precisely one vertical strip and one horizontal strip. In particular, we can apply Lemma 3.11 to get that at most two edges in $\mathscr{C}_k^{\text{OPT}}(S)$ intersect the elementary box B (clearly, we must have either $h(B) \leqslant w(B) \leqslant 4w(B)$ or $w(B) \leqslant h(B) \leqslant 4h(B)$).

Now, suppose B is a box satisfying the box invariant. If B is not an elementary box, then it has at least two vertical strips or at least two horizontal strips. If all vertical strips T satisfy $w(T) \leqslant \frac{w(B)}{200}$ and all horizontal strips H satisfy $h(H) \leqslant \frac{h(B)}{200}$, then we can apply Lemma 3.8 to get that there exists either a good vertical separator of B or a good horizontal separator of B (and is contained in a help or main line). Since the separator is good, at most two edges of $\mathscr{C}_k^{\text{OPT}}(S)$ intersect it, and hence the box invariant continues to be satisfied for each of the two smaller boxes.

Hence, suppose there is some vertical strip T satisfying $w(T) > \frac{w(B)}{200}$ or there is some horizontal strip H satisfying $h(H) > \frac{h(B)}{200}$. If the box B is much wider than it is tall and has many small vertical strips (i.e., $w(B) > 4h(B)$ and $w(T) \leqslant \frac{w(B)}{200}$ for all vertical strips T), then we can apply Lemma 3.9. We get that there is a good vertical separator (contained in a vertical help or main line) that splits B into two smaller boxes such that the box invariant continues to hold on each of the smaller boxes (since at most two edges of $\mathscr{C}_k^{\text{OPT}}(S)$ intersect the vertical separator). Likewise, if the box is much taller than it is wide and has many small horizontal strips (i.e., $h(B) > 4w(B)$ and $h(H) \leqslant \frac{h(B)}{200}$ for all horizontal strips H), then we can again apply Lemma 3.9. We get that there is a good horizontal separator (contained in a horizontal help or main line) that splits B into two smaller boxes, each satisfying the box invariant.

The only other case to consider is when either the box is not much taller than it is wide and there is some wide vertical strip T, or the box is not much wider than it is tall and there is some tall horizontal strip H. That is, either $h(B) \leqslant 4w(B)$ and there is some vertical strip T satisfying $w(T) > \frac{w(B)}{200}$, or $w(B) \leqslant 4h(B)$ and there is some horizontal strip H satisfying $w(H) > \frac{h(B)}{200}$.

Suppose the former is the case, as the proof of the latter is symmetric. Then, we can apply Lemma 3.10 to get that the left edge of T is intersected by at most two edges of $\mathscr{C}_k^{\text{OPT}}(S)$ and the right edge of T is intersected by at most two edges of $\mathscr{C}_k^{\text{OPT}}(S)$ (in the latter case, we know the bottom and top edges of H satisfy a similar property).

If either the left edge of T or the right edge of T serves as a good vertical separator of B (i.e., if either edge splits B into two strictly smaller boxes), we are done since we already know such an edge of T is intersected by at most two edges of $\mathscr{C}_k^{\text{OPT}}(S)$. Hence, we need only consider the case when the box B consists of one vertical strip (i.e., the left and right edges of T are $l(B)$ and $r(B)$, respectively). Since $h(B) \leqslant 4w(B)$, we can apply Lemma 3.11 to get that at most two edges of $\mathscr{C}_k^{\text{OPT}}(S)$ intersect the entire box B. Since box B is not elementary, there is either a vertical help or main line that lies strictly in between $l(B)$ and $r(B)$, or there is a horizontal help or main line that lies strictly in between $b(B)$ and $t(B)$. Regardless, the separators induced by such lines are good.

Tying everything together, for boxes B that are not elementary and satisfy the box invariant, we get that we can always find a good horizontal or vertical separator (contained in a help or main line) that splits B into two boxes, both of which also satisfy the box invariant. □

Remark. We note that there is an unpublished solution [58] (i.e., polynomial time algorithm) for the rectilinear version of the problem, where we must enclose n points using k axis-parallel rectangles rather than convex hulls (as in our setting). The solution to the axis-parallel version uses similar ideas. In particular, it is possible to argue the existence of separators that do not cut through any clusters of an optimal solution in this setting. This is not possible for our problem. For the minimum perimeter-sum problem, it is possible that any such vertical or horizontal separator cuts at least one cluster, and allowing skew separators would result in subproblems of high complexity. In particular, consider an example with n sufficiently large and assume $k < n/3$. We have $n-k+1$ points spread evenly on a circle as the corners of a regular $(n-k+1)$-gon, and $k-1$ points spread evenly on a surrounding circle with the same center. The surrounding points are fairly close yet sufficiently far enough away that the optimal solution is to cluster the inner $n-k+1$ points together, and open a cluster for each of the $k-1$ points on the surrounding circle. Cutting away the $k-1$ points on the outside (with not necessarily axis-aligned separators) creates subproblems defined on polygonal regions with $k-1$ sides, resulting in high complexity subproblems.

3.3 Coverings and Signatures

In the rest of this section, we assume we are given a set of points S in the plane and a number k, and we want to solve the k-clustering problem for S.

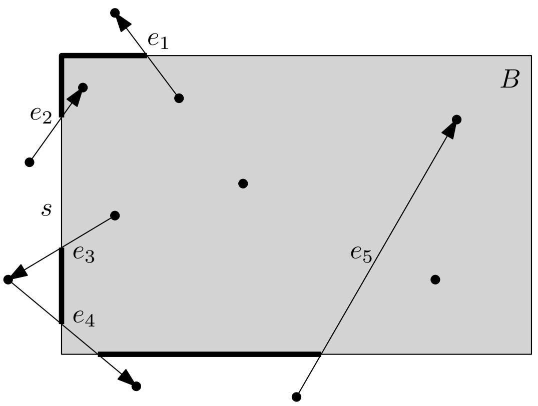


Figure 3.4: The cut number of the set of edges $E := \{e_1, \dots, e_5\}$ shown in this figure is 2, i.e. $\Theta_{B,s}(E) = 2$. Note that in this example the border set of E is E, i.e. $\mathcal{M}_B(E) = E$, and it is alternating.

Moreover, we assume the boxes we work with are in $\mathcal{B}(S)$, and the edges we work with are in $\mathcal{G}(S)$. For brevity, we define $\mathcal{C}^* := \mathcal{C}_k^{\text{OPT}}(S)$.

Let B be a box and E a set of edges and loops. E is called *interior-disjoint* if the interior of any edge $e \in E$ is not intersected by any other edge in E. Note that we do not allow an endpoint of one edge to be in the interior of another edge, but we do allow edges to share endpoints. All sets of edges (and loops) we work with in this section are interior-disjoint. The *border set* of E on B is denoted by $\mathcal{M}_B(E)$ and is defined as the set of all edges in E intersecting both the boundary and the interior of B. An intersection q of an edge $pp' \in \mathcal{M}_B(E)$ and ∂B is called *entering* if pq does not intersect the interior of B, and is called *exiting* if qp' does not intersect the interior of B.

Assuming E is interior-disjoint, all the intersections of the edges in its border set and ∂B are distinct because by definition of the main and help lines, there is no input point $p \in S$ on ∂B. Therefore, we can define a cyclic order on these intersections by sorting them in counterclockwise order. A border set is called *alternating* if no two consecutive intersections in this cyclic order are both entering or both exiting. Note that any two consecutive intersections define an "interval" on ∂B. Let a, a' be consecutive intersection points along ∂B (where we order the intersections in counterclockwise order) such that a is an exiting intersection and a' is an entering intersection. We call the interval (a, a') on ∂B in counterclockwise order an *exit-entry inteval.*

For an edge s of ∂B the *cut number* of E on the edge s is denoted by $\Theta_{B,s}(E)$ and is defined as the number of exit-entry intervals (if any) intersecting s (see Figure 3.4). The *border sequence* of E on B is defined as the sequence $e_1, \dots, e_t$ of all edges in $\mathcal{M}_B(E)$ sorted in counterclockwise order according to their intersection with the boundary of B. Note that some edges might appear

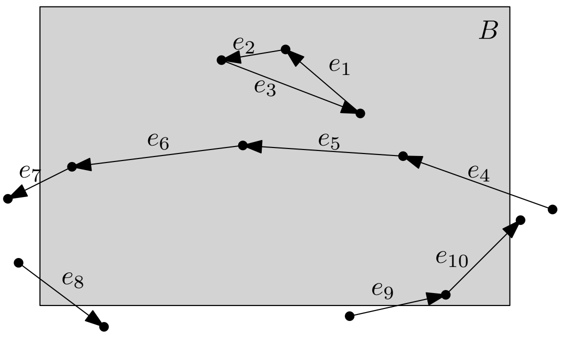


Figure 3.5: Example of an inner cycle $E = \{e_1, e_2, e_3\}$, three convex chains $E_1 = \{e_4, e_5, e_6, e_7\}$, $E_2 = \{e_8\}$, $E_3 = \{e_9, e_{10}\}$, and a border cycle $E' = E_1 \cup E_2 \cup E_3$.

twice in this sequence. Moreover, this sequence is not unique as the first edge in this sequence can be chosen arbitrarily.

Let B be a box. A nonempty set of edges E is called a *convex chain* on B of size t (see Figure 3.5), where $t := |E|$, if there exists a sequence of distinct points $p_1, \dots, p_{t+1}$ such that[2]

- $p_2, \dots, p_t \in S \cap B$ and $p_1, p_{t+1} \in S \setminus B$,
- $E = \{p_1p_2, p_2p_3, \dots, p_tp_{t+1}\}$,
- $\bigcup_{i=1}^{t} p_ip_{i+1}$ is a simple open curve intersecting the interior of B, and
- for any i, where $1 \leqslant i < t$, we have that $p_ip_{i+1}p_{i+2}$ is a left turn.

We call this sequence the *vertex sequence* of the convex chain E. The edge p_1p_2 is called the *starting* edge and the edge p_tp_{t+1} is called the *ending* edge of the convex chain. Note that the starting and ending edges of a convex chain are the same when $t = 1$.

Let B be a box. A *cycle* on B is defined to be a nonempty set E that is either an inner cycle or a border cycle, as defined next. We call the set E an *inner* cycle on B of size t, see Figure 3.5, if either $t = 1$ and E contains a single loop pp, where $p \in S \cap B$, or if there exists a sequence of distinct points $\sigma := (p_1, \dots, p_t)$ such that

- $p_1, \dots, p_t \in S \cap B$,
- $E = \{p_1p_2, p_2p_3, \dots, p_{t-1}p_t, p_tp_1\}$, and therefore $t = |E|$, and
- $p_1, \dots, p_t$ are the vertices of a convex polygon given in counterclockwise direction.

[2]Note that by definition there is no input point $p \in S$ on ∂B

An inner cycle consisting of a single loop is called a *loop* cycle on B. Moreover, we call the set E a *border* cycle on B, see Figure 3.5, if there exists a partition $E = E_1 \cup \cdots \cup E_x$, where $1 \leqslant x \leqslant 4$, such that

- each E_i, where $1 \leqslant i \leqslant x$, is a convex chain, and
- $e_1, e'_1, \ldots, e_x, e'_x$ is a border sequence of E, where e_i and e'_i are the starting and ending edges of the convex chain E_i.

Note that by the definition of main and help lines, no point in S can be on the boundary of B. It is not hard to verify that the border set of a border cycle is alternating. The *coverage* of a cycle E is denoted by $\mathscr{R}_B(E)$ and is defined either as the set $\{p\}$ if E is a loop cycle $\{pp\}$, or the intersection of all half planes defined by edges in E and the box B, i.e.,

$$\mathscr{R}_B(E) := B \cap \bigcap_{e \in E} HP(e),$$

where $HP(e)$ is the half-plane defined by the line passing through e located on the left of e. We say a point $b \in B$ is *covered* by E if it is in the coverage of E.

For a given box B, an interior-disjoint set E of edges and loops is called a *(nonempty) disjoint covering* or simply a *(nonempty) covering* on B of size t if there exists a partition $E = E_1 \cup \cdots \cup E_t$ such that

- each E_i is a cycle,
- when $1 \leqslant i < j \leqslant t$, the coverages of E_i and E_j are disjoint, i.e., $\mathscr{R}_B(E_i) \cap \mathscr{R}_B(E_j) = \emptyset$,
- $S \cap B \subset \mathscr{R}_B(E_1) \cup \cdots \cup \mathscr{R}_B(E_t)$.

It is not hard to verify that the border set of a covering is alternating. We denote the size of E by $\kappa_B(E)$. By definition, the empty set is a covering on B of size zero if $S \cap B = \emptyset$ and we call it the *empty covering* on B. The following lemma shows that the size of a covering is well defined.

Lemma 3.12. *For a nonempty covering E defined on an arbitrary box B, the satisfying partition, i.e. the partition satisfying the conditions for a nonempty covering on B, is unique.*

Proof. Note that the coverages of each cycle the satisfying partition is a convex polygon containing all parts of its edges in B. Therefore as all the coverages are disjoint, no two inner cycles or convex chains that belong to different border cycles intersect inside B. Moreover because of the condition on the border sequence of a border cycle, different convex chains of a border cycle cannot intersect in B. Hence, any other satisfying partition (if exists) consists of the same convex chains and inner cycles. Additionally, any two convex chain that are in the same border cycle in the satisfying partition must be in the same

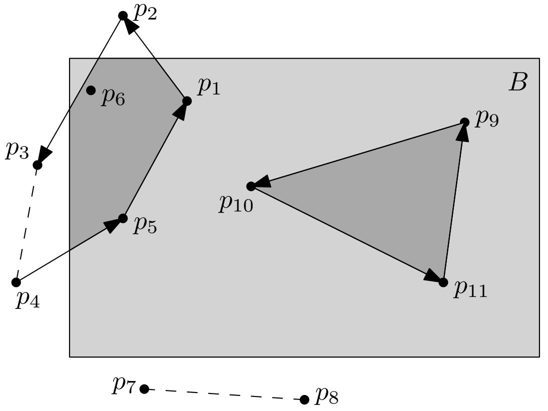


Figure 3.6: The signature of the clustering $\mathscr{C} := \{\{p_1,\dots,p_6\},\{p_7,p_8\},\{p_9,p_{10},p_{11}\}\}$ on B. The edges of the covering $\mathscr{C} \sqcap B$ are solid and the coverage of each cycle is shown in dark grey.

border cycle in any other satisfying partition as otherwise the coverages of their border cycles intersect. This shows that any other satisfying partition must have the same inner cycles and border cycles which completes the proof. □

It is not hard to verify that for an edge s of ∂B the *cut number* of E on an edge s of B, i.e. $\Theta_{B,s}(E)$, is equal to the number of the coverages of E intersecting s.

We say a cluster C *intersects* a box B if its convex hull intersects B, and we say it *intersects* the boundary ∂B of B if its convex hull intersects ∂B. The *signature* of C on B or its boundary is denoted by $C \sqcap B$ and $C \sqcap \partial B$ respectively and is defined as the set of all edges of its convex hull intersecting B or the boundary of it, i.e.,

$$C \sqcap B := \{e \in \mathscr{E}(C) \mid e \cap B \neq \emptyset\}, \qquad C \sqcap \partial B := \{e \in \mathscr{E}(C) \mid e \cap \partial B \neq \emptyset\}.$$

With some abuse of notation, the *signature* of a disjoint clustering $\mathscr{C}$ on B or its boundary is denoted by $\mathscr{C} \sqcap B$ and $\mathscr{C} \sqcap \partial B$ respectively and is defined as the set of all edges of convex hulls of its clusters intersecting B or the boundary of it, i.e.,

$$\mathscr{C} \sqcap B := \bigcup_{C \in \mathscr{C}} C \sqcap B, \qquad \mathscr{C} \sqcap \partial B := \bigcup_{C \in \mathscr{C}} C \sqcap \partial B.$$

Note that for any disjoint clustering $\mathscr{C}$, $\mathscr{C} \sqcap B$ is a covering on B (see Figure 3.6).

3.4 Subproblems

We consider subproblems of the following type. Given a box B, a set of edges M and integer k', we want to extend the set M into a covering on B of size k'

such that the total cost of the extension (which is measured as the sum of the lengths of the added edges) is minimized.

Let B be a box, M be an interior-disjoint set of edges with $|M| \leqslant 8$, let $k' \in \{0,\dots,k\}$, and $type \in \{\text{SPECIAL}, \text{NORMAL}\}$ a boolean value. The tuple $I := \langle B, M, k', type\rangle$ is called a *subproblem* on B if it it is either a special subproblem or a normal subproblem, as defined next. The subproblem I is called a *normal* subproblem if

- $type = \text{NORMAL}$,
- M is equal to its own border set, i.e. $M = \mathscr{M}_B(M)$, and
- the border set of M, i.e. M, is alternating.

A covering E on B is called a *solution* to a normal subproblem $\langle B, M, k', \text{NORMAL}\rangle$ if $\mathscr{M}_B(E) = M$ and $\kappa_B(E) = k'$. The *cost* of a solution is denoted by $\Phi_B(E)$ and is defined as the sum of the lengths of $e \cap B$ over all edges $e \in E$. A solution to I is called *optimal* if there is no solution to I with smaller cost. We say a point $b \in B$ is *covered* by a solution E if either b is covered by E or $type = \text{SPECIAL}$. Note that the coverage status of any point on the boundary of B is the same for all solutions. The tuple I is called a *special* subproblem if

- $type = \text{SPECIAL}$,
- $M = \emptyset$, and
- $k' = 1$.

We define the empty covering (on B) to be the only *solution* to a special subproblem $\langle B, \emptyset, 1, \text{SPECIAL}\rangle$.

The *cut number* of a subproblem $I := \langle B, M, k', type\rangle$ on an edge s of B is denoted by $\Theta_{B,s}(M, type)$ and is defined as

$$\Theta_{B,s}(M, type) := \begin{cases} 1, & type = \text{SPECIAL}, \\ \Theta_{B,s}(M), & \text{otherwise.} \end{cases}$$

A disjoint clustering $\mathscr{C}$ *respects* a subproblem $I := \langle B, M, k', type\rangle$ if either

- $type = \text{SPECIAL}$ and B is in the convex hull of a cluster in $\mathscr{C}$, or
- $type = \text{NORMAL}$ and $\mathscr{C} \sqcap B$ is a solution to I.

Observation 3.13. *Given a box B, the subproblem $I := \langle B, k', \mathscr{C}^* \sqcap \partial B, type\rangle$ is the unique subproblem on B respected by $\mathscr{C}^*$, where k' is the number of clusters in $\mathscr{C}^*$ intersecting B and $type = \text{SPECIAL}$ if B is contained in the convex hull of a cluster in $\mathscr{C}^*$ and $type = \text{NORMAL}$ otherwise. Moreover, $\mathscr{C}^* \sqcap B$ is a solution to I.*

Note that the original clustering problem is equivalent to the subproblem $I_0 := \langle B_0, \emptyset, k, \text{NORMAL}\rangle$, where B_0 is defined as the box with edges on $v_1^-, v_n^+, h_1^-, h_n^+$.

Observation 3.14. *For any solution E to I_0, there is a disjoint clustering $\mathscr{C} \in$ Part(S) such that $\mathscr{C} \sqcap B_0 = E$. Likewise, for any disjoint clustering $\mathscr{C} \in$ Part(S), it holds that $\mathscr{C} \sqcap B_0$ is a solution to I_0.*

The following property shows that the optimal clustering $\mathscr{C}^*$ is also optimal for subproblems.

Lemma 3.15. *Let $I := \langle B, M, k', type\rangle$ be a subproblem on a box B respected by $\mathscr{C}^*$. Then, for any solution E to I, we have that $\Phi_B(\mathscr{C}^*) \leqslant \Phi_B(E)$.*

Proof. Assume there exist a solution E to I such that $\Phi_B(E) < \Phi_B(\mathscr{C}^*)$. It is easy to verify that $E' := ((\mathscr{C}^* \sqcap B_0) \setminus (\mathscr{C}^* \sqcap B)) \cup E$ is a solution to I_0 such that $\Phi_{B_0}(E') < \Phi_{B_0}(\mathscr{C}^*)$. As E' consists of k inner cycles on B_0, the set E' is the boundary of a clustering $\mathscr{C}'$ with cost less than $\mathscr{C}^*$, which is a contradiction. □

3.5 Split and Merge

We say that a box B *splits* into B_l and B_r if there is a separator s that separates B to B_l and B_r. Here, the box B_l is the upper box if s is horizontal and the left box if s is vertical.

Consider two boxes B_1 and B_2 whose boundaries share an edge. We say two subproblems $\langle B_1, M_1, k'_1, type_1\rangle$ and $\langle B_2, M_2, k'_2, type_2\rangle$ are *compatible* if their edges match on their shared boundary, i.e.,

$$\{e \in M_1 | e \cap \partial B_2 \neq \emptyset\} = \{e \in M_2 | e \cap \partial B_1 \neq \emptyset\}.$$

Let B be a box that splits into B_l and B_r by a separator s. We say a subproblem $I := \langle B, M, k', type\rangle$ *splits* into two subproblems $I_l := \langle B_l, M_l, k'_l, type_l\rangle$ and $I_r := \langle B_r, M_r, k'_r, type_r\rangle$ if I_l and I_r are compatible, $\Theta_{B,s}(M_l, type_l) = \Theta_{B,s}(M_r, type_r)$, $k' = k'_l + k'_r - \Theta_{B,s}(M_l, type_l)$, and $type = type_l \wedge type_r$. Equivalently, we say that I_l and I_r *merge* to I.

Observation 3.16. *Let B be a box that splits into boxes B_l and B_r by a separator s, and $\mathscr{C}$ be a disjoint clustering. If $\mathscr{C}$ respects the three subproblems $I := \langle B, M, k', type\rangle$, $I_l := \langle B_l, M_l, k'_l, type_l\rangle$, and $I_r := \langle B_r, M_r, k'_r, type_r\rangle$, then the subproblem I splits into I_l and I_r.*

The following lemma provides sufficient conditions for a subproblem to have a solution and helps us to find a solution for a subproblem satisfying those conditions.

Lemma 3.17. *Let $I := \langle B, M, k', type\rangle$ be a subproblem that splits into subproblems $I_l := \langle B_l, M_l, k'_l, type_l\rangle$ and $I_r := \langle B_r, M_r, k'_r, type_r\rangle$. Suppose that E_l and E_r are solutions to I_l and I_r, respectively. Then, $E := E_l \cup E_r$ is a covering on B and a solution to I. Furthermore, $\Phi_B(I) = \Phi_{B_l}(E_l) + \Phi_{B_r}(E_r)$.*

Proof. First assume that none of the subproblems is special. As I_l and I_r are compatible and $\Theta_{B,s}(E_l, type_l) = \Theta_{B,s}(E_r, type_r)$, each border cycle in E_l intersected by s has a corresponding border cycle in E_r. The set E is a covering on B because each pair of corresponding border cycles intersected by s merges to a cycle on B, and all other cycles from E_l and E_r are also cycles in B. Moreover, as the number of the corresponding pairs of border cycles is $\Theta_{B,s}(E_l, type_l)$, the number of cycles in E is $\kappa_B(E) = k'_l + k'_r - \Theta_{B,s}(E_l, type_l)$. By the definition of a solution, the covering E is a solution to a subproblem $\langle B, \mathscr{M}_B(E), k'_l + k'_r - \Theta_{B,s}(E_l, type_l), \textsc{normal}\rangle$. However, as $E = E_l \cup E_r$, the border set $\mathscr{M}_B(E)$ is simply all the edges in M_l and M_r that intersect the boundary of B. As I splits into I_l and I_r, we have $M = \mathscr{M}_B(E)$ and $k' = k'_l + k'_r - \Theta_{B,s}(E_l, type_l)$. Moreover, as the edges of E are simply all the edges of E_l and E_r combined, the cost of the covering E satisfies the condition mentioned in the statement of the lemma.

It is easy to verify that in case one or both of the subproblems are special, the claim still holds, which completes the proof of the lemma. □

3.6 Elementary Subproblems

We call a subproblem *elementary* if it is defined on an elementary box. In this section, we show that any elementary subproblem $\langle B, M, k', type\rangle$ with at most two edges, i.e., $|M| \leqslant 2$, has at most one solution. Moreover, if such a subproblem has a solution we can recognize it and find its unique solution in constant time.

An elementary box contains either one point from S or is empty. Therefore, if an elementary box B is empty, all cycles on B are border cycles. If, on the other hand, $B \cap S = \{p\}$, then the only inner cycle on B is the loop cycle $\{pp\}$. Furthermore, as B is defined by two pairs of consecutive main lines, v_i^-, v_i^+ and h_j^-, h_j^+, and its width and height are infinitessimally small, any edge intersecting B contains p. The only covering on B with an inner cycle is $\{pp\}$, and all other coverings consists of border cycles only. Note that if $\{pp\}$ is a solution to a subproblem, there is no other solution, as in that case $M = \emptyset$ and the loop cycle $\{pp\}$ is the only way p can possibly be covered.

Now, consider a solution to an elementary subproblem without any inner cycles. As an elementary box B contains at most one point, the vertex sequence of a convex chain on B has a size at most three, which means it has at most two edges, and both edges must intersect the boundary of B. Therefore, for a covering E on an elementary box B that does not have any inner cycles, we have $\mathscr{M}_B(E) = E$.

The following observation summerizes the cases.

Observation 3.18. *An elementary subproblem $I := \langle B, M, k', type\rangle$ has a solution if and only if either*

- *$I = \langle B, \emptyset, 1, \text{NORMAL}\rangle$ and $S \cap B = \{p\}$,*
- *$I = \langle B, \emptyset, 0, \text{NORMAL}\rangle$ and $S \cap B = \emptyset$,*
- *$I = \langle B, \emptyset, 1, \text{SPECIAL}\rangle$, or*
- *$I = \langle B, M, k', \text{NORMAL}\rangle$, where M is a covering on B of size k'.*

Moreover, if a solution exists, it is unique and is equal to $\{pp\}$ in the first case and M otherwise.

Note that when $|M| \leqslant 2$, we can decide if a set of edges is a solution to an elementary subproblem in constant time.

3.7 A Dynamic-Programming Algorithm

Definition 3.19. A subproblem $I := \langle B, M, k', type\rangle$ is *compact* if for each edge of B, at most two edges in M intersect it.

Observation 3.20. *Let B be a box satisfying the box invariant and I be a subproblem on B such that $\mathscr{C}^*$ respects I. Then I is compact.*

Algorithm 1: Algorithm for Minimizing the Sum of Perimeters

```
for $b = 1, \ldots, (20001n + 1)^2$ do
    for all boxes $B$ consisting of $b$ elementary boxes do
        for all compact subproblems $I := \langle B, M, k', type\rangle$ do
            Table($I$) := Impossible
            if $b = 1$
                if $M = \emptyset$ and $k' = 1$ and $type = \text{NORMAL}$ and $B \cap S = \{p\}$
                    Table($I$) := $\{pp\}$
                else if $|M| \leqslant 2$ and $M$ is a solution to $I$
                    Table($I$) := $M$
            else
                for all compact subproblems $I_l$ and $I_r$ that merge to $I$ do
                    if Table($I_l$) $\neq$ Impossible and Table($I_r$) $\neq$ Impossible
                        $E :=$ Table($I_l$) $\cup$ Table($I_r$)
                        if Table($I$) = Impossible or $\Phi_B(E) \leqslant \Phi_B$(Table($I$))
                            Table($I$) := $E$
```

Algorithm 1 solves the minimum perimeter sum problem. For a subproblem $I := \langle B, M, k', type\rangle$, the entry Table($I$) either stores a solution to I or the value

"NotFound", which means we have not found any solution to I, though it does not mean a solution to I does not exist. After the execution of the algorithm, Table($\langle B_0, \emptyset, k, \text{NORMAL}\rangle$) contains the convex hull of an optimal k-clustering. The algorithm iterates over all compact subproblems sorted ordered by the size of the defining box (i.e., the number of elementary boxes contained in the box) and stores a solution for some of them in Table.

- For an elementary subproblem I that has a solution, the algorithm stores the unique solution in Table(I) only if $|M| \leqslant 2$. The algorithm uses the conditions stated in Observation 3.18 to find the unique solution to such subproblems. In the proof of Lemma 3.21, we show that we do not need to find a solution to elementary subproblem with $|M| > 2$.
- For a subproblem I defined on a non-elementary box, the algorithm finds all possible splits of I to two compact subproblems I_l and I_r. For each such split, it combines the coverings (if such exist) in Table(I_l) and Table(I_r) to get a covering E. Then, it stores E in Table(I) only if it is a better solution to I than already stored in Table(I). By Lemma 3.17, the union of the two solutions to I_l and I_r is a solution to I.

Lemma 3.21. *Let I be a subproblem on a box B. Consider the value of* Table(I) *as Algorithm 1 terminates. If B satisfies the box invariant and $\mathscr{C}^*$ respects I, the entry* Table(I) *contains a solution to I of cost $\Phi_B(\mathscr{C}^*)$. In particular,* Table(I_0) *contains a solution to $I_0 := \langle B_0, \emptyset, k, \text{NORMAL}\rangle$ of cost $\Phi_{B_0}(\mathscr{C}^*)$, and hence, the algorithm solves the minimum perimeter-sum problem.*

Proof. The proof is by induction on the number b of elementary boxes contained in B. Consider first the case $b = 1$. If I is respected by $\mathscr{C}^*$, Lemma 3.6 yields that B is intersected by at most two edges of the signature of $\mathscr{C}^*$. By Observation 3.4, there are no points from S on the boundary of B, which shows $|M| \leqslant 2$ and therefore I is compact and will be generated. Depending on the subproblem, the algorithm stores either $\{pp\}$ or M as a solution to I in Table(I) only if M is a solution to I. As by Observation 3.18 the solution to I is unique, it must be $\mathscr{C}^* \sqcap B$. If $|M| > 2$, the algorithm stores "NotFound" in Table(I) which completes the proof, for the base case.

Suppose the claim holds for boxes consisting of up to $b-1$ elementary boxes for some $b \geqslant 2$ and consider a subproblem $I := \langle B, M, k', \mathit{type}\rangle$ where B consists of b elementary boxes. Note that if Table(I) $\neq$ "NotFound", Table(I) must contain a value Table(I_l) $\cup$ Table(I_r) for some subproblems I_l and I_r that merge to I and are defined on boxes consisting of less than b elementary boxes. By the induction hypothesis, Table(I_l) and Table(I_r) contain solutions to I_l and I_r and by Lemma 3.17, Table(I) = Table(I_l) $\cup$ Table(I_r) is a solution to I.

If $\mathscr{C}^*$ respects I and B satisfies the box invariant, by Observation 3.20, I is compact. By Lemma 3.6, B has a separator s contained in a main or help line and s separates B into two boxes B_l and B_r both satisfying the box invariant. By Observation 3.13, there are unique subproblems I_l and I_r, on B_l and B_r, such that $\mathscr{C}^*$ respects them. By Observation 3.16, subproblem I splits into I_l and I_r. As B_l and B_r consists of less than b elementary boxes, by the induction hypothesis, $\text{Table}(I_l) = \Phi_{B_l}(\mathscr{C}^*)$ and $\text{Table}(I_r) = \Phi_{B_r}(\mathscr{C}^*)$. By Lemma 3.17,

$$\Phi(\text{Table}(I)) \leqslant \Phi_{B_l}(\mathscr{C}^*) + \Phi_{B_r}(\mathscr{C}^*) = \Phi_B(\mathscr{C}^*),$$

where the equality comes from Observation 3.3. Furthermore, by the above mentioned argument, $\text{Table}(I)$ is a solution to I and by Lemma 3.15,

$$\Phi(\text{Table}(I)) \geqslant \Phi_B(\mathscr{C}^*),$$

which completes the proof. □

3.8 Run-time Analysis

Data structures. We implement Table as a hash table where each subproblem I is hashed to a number, so that we can update and insert values at a given entry $\text{Table}(I)$ in time $O(1)$.

We use a special representation for solutions (set of edges and loops) by making a directed acyclic graph (DAG) to support constant time union operation on solutions (sets). Each solution E to a subproblem $\langle B, M, k', type\rangle$ is either the set M, where $|M| \leqslant 2$, or a set containing a single loop for elementary subproblems, or is the union of two solutions E_l and E_r to two subproblems defined on smaller boxes B_l and B_r for non-elementary subproblems. For elementary subproblems we simply make a node and store if the solution is a loop or not, and if it is a loop we store the loop in the node. For non-elementary subproblems we make a node and we store the (at most two) edges in the solution that intersect the shared edge of B_l and B_r but do not intersect ∂B. We also store two pointers to E_l and E_r in this node; these pointers form a DAG where the number of nodes visible to each node (solution) is linear and therefore it is possible to find the edges and loops of a solution in linear time by simply adding up all the loops and edges inside the visible nodes to M.

For a box B and an edge s of B, we can compute the value of $\Theta_{B,s}(M, type)$ for a subproblem $I := \langle B, M, k', type\rangle$ in $O(1)$ time as follows. For each subproblem, we store the coverage status of the four corners of the defining box. It is easy to compute those values for elementary subproblems and after merging two solutions we can easily merge the coverage status of their corners as well.[3] Let m_s be the number edges in M intersecting s. As we just work with compact subproblems, $0 \leqslant m_s \leqslant 2$. Having the coverage status of the four corners of B,

[3]To handle degenerate cases where an edge intersects a corner of a box, we can store two coverage values for each corner. One for the coverage of the corner point and a bit more of the boundary in counterclockwise direction and one for the coverage of the corner point and a bit more of the boundary in clockwise direction.

assuming s_1 and s_2 are the endpoints of s, the cut number is given by

$$\Theta_{B,s}(M, \mathit{type}) = \begin{cases} 2, & m_s = 2 \text{ and } s_1 \text{ and } s_2 \text{ are covered,} \\ 0, & m_s = 0 \text{ and } s_1 \text{ and } s_2 \text{ are not covered,} \\ 1, & \text{otherwise.} \end{cases}$$

Observation 3.22. *Given a box B, the number of compact subproblems $I := \langle B, M, k', \mathit{type}\rangle$ is at most*

$$O\left((n+1)\cdot\binom{O(n^2)}{8}\right) = O(n^{17}).$$

Here, $n+1$ is a bound on the number k' and the factor $\binom{O(n^2)}{8}$ is a bound on the number of ways the edges of M can be selected for a compact subproblem. There is also one special subproblem on box B that does not affect the upper bound $O(n^{17})$.

Observation 3.23. *Let B be a box that splits into B_l and B_r. The number of ways to split a compact subproblem $I := \langle B, M, k', \mathit{type}\rangle$ to two compact subproblems $I_l := \langle B_l, M_l, k'_l, \mathit{type}_l\rangle$ and $I_r := \langle B_r, M_r, k'_r, \mathit{type}_r\rangle$ is at most*

$$O\left(n\cdot\binom{O(n^2)}{2}\right) = O(n^5).$$

Here n is the number of ways we can choose a value for k'_l and $\binom{O(n^2)}{2}$ is the number subsets of at most two edges intersecting $\partial B_l \cap \partial B_r$.

We now describe how Algorithm 1 at line 11 iterates through the $O(n^5)$ pairs of subproblems (I_l, I_r) merging to I. At first, assume type_l and type_r have the value NORMAL. The edge set M specifies all edges in M_l and M_r except possibly some edges intersecting the edge $s := \partial M_l \cap \partial M_r$ and not other edges of ∂M_l or ∂M_r. As I_l and I_r are compact and compatible, M_l and M_r can have at most two such edges. Consider now one of these $\binom{O(n^2)}{2}$ choices of edges crossing s. We iterate through each value $k'_l \in \{0, \ldots, k'\}$. For a given value of k'_l, the subproblem I_l is completely specified. If I_l does not have any solution in Table(I_l), we can proceed with another choice of I_l. Otherwise, note that I_r is also specified except for the value k'_r. However, we can compute $\Theta_{B_l,s}(E, \mathit{type}_l)$, where E is the solution stored as Table(I_l), in constant time. In order for I_l and I_r to merge to I, we now define $k'_r := k' - k'_l + \Theta_{B_l,s}(M_l, \mathit{type}_l)$. There are four different choices for the values type_l and type_r that do not affect the upper bound $O(n^5)$.

Theorem 3.24. *Algorithm 1 solves the minimum perimeter sum problem in time $O(n^{27})$.*

Proof. Note that B_0, the box with edges on $v_1^-, v_n^+, h_1^-, h_n^+$, satisfies the box invariant. Since $\mathscr{C}^*$ respects $I_0 := \langle B_0, \emptyset, k, \text{NORMAL}\rangle$, Lemma 3.21 gives that as the algorithm terminates, Table(I_0) contains a solution E to I_0 of cost $\Phi_{B_0}(\mathscr{C}^*) = \Phi(\mathscr{C}^*)$. Moreover, Observation 3.14 states that there is a clustering $\mathscr{C}$ such that $\mathscr{C} \sqcap B_0 = E$, and hence, $\mathscr{C}$ is an optimal clustering. Furthermore, the clustering $\mathscr{C}$ can easily be computed given E.

The algorithm uses the conditions specified in Observation 3.18 to verify solutions to elementary subproblems $I := \langle B, M, k', \mathit{type}\rangle$ satisfying $|M| \leqslant 2$, which takes constant time and does not affect the asymptotic running time of the algorithm.

There are $O(n^4)$ different boxes in $\mathscr{B}(S)$. By Observation 3.22, the number of compact subproblems on a B is $O(n^{17})$. For each separator s of B, by Observation 3.23, the number of ways to split I to I_l and I_l is $O(n^5)$. As we showed in the beginning of this section, merging two solutions can be done in constant time. The cost of each solution can be stored as a single real number and comparing could be assumed to take only $O(1)$ time.The running time of the algorithm thus is

$$O(n^4) \cdot O(n^{17}) \cdot O(n) \cdot O(n^5) \cdot O(1) = O(n^{27}),$$

where $O(n^4)$ is the number of different boxes $B \in \mathscr{B}(S)$, the factor $O(n^{17})$ is the number of subproblems on B, $O(n)$ is the number of separators of B, $O(n^5)$ is the number of ways to split a subproblem into two subproblems for a given separator s, and $O(1)$ is the time needed to compare two solutions. □

Theorem 3.25 (The minimum perimeter-sum problem). *There is a polynomial time algorithm that, given any set S of n points in the plane and an integer k, finds a set of at most k closed curves such that each point in S is enclosed by a curve and the total length of the curves is minimized.*

3.9 Concluding Remarks

We presented the first polynomial time algorithm that solves the minimum perimeter-sum problem in $O(n^{27})$ time. One interesting question is if there exists an algorithm that solves this problem faster or a lower bound on the time complexity of such algorithm.

As mentioned in the introduction section, a polynomial time algorithm also exists for the minimum radii-sum. However, the case for minimum diameter-sum is still open. Another interesting problem therefore is closing the gap for this closely related problem by providing a lower bound or an algorithm that solves minimum diameter-sum in polynomial time.

Chapter 4

A Generic Method for Finding Coresets for Clustering Problems

In this chapter we consider clustering problems of the following type. Let S be a set of n points in $\mathbb{R}^d$, and let $k \geqslant 2$ be a natural number. A *k-clustering* of S is a partitioning $\mathscr{C}$ of S into at most k clusters. Let $\Phi(\mathscr{C})$ denote the *cost* of $\mathscr{C}$. The goal is now to find a clustering $\mathscr{C}$ that minimizes $\Phi(\mathscr{C})$. We present a general method to find an ε-coreset for a given clustering problem. Given a set of points S, a cost function Φ, the number of clusters k, a value $\varepsilon > 0$, an ε-coreset is a set $R \subseteq S$ such that $\Phi(\mathscr{C}'_{\text{opt}}) \geqslant (1-\varepsilon) \cdot \Phi(\mathscr{C}_{\text{opt}})$, where $\mathscr{C}'_{\text{opt}}$ is an optimal clustering for R and $\mathscr{C}_{\text{opt}}$ is an optimal clustering for S. We present an algorithm that finds a coreset of size $O(k\big(f(k)/\varepsilon\big)^d)$ for a set of points S in $\mathbb{R}^d$, where $f(k)$ is a function that only depends on the number of clusters. The algorithm runs in linear time for constant k, assuming a certain model of computing. As we mentioned in the introduction, our method applies to a large class of clustering problems including the k-center problem in any L_p-metric, variants of the k-center problem where we want to minimize the sum (rather than maximum) of the cluster radii, and the 2-dimensional problem where we want to minimize the maximum or sum of the perimeters of the clusters.

4.1 Regular Cost Functions

We start by defining the class of clustering problems to which our algorithm applies.

Let S be a set of n points in $\mathbb{R}^d$ and let $\text{Part}(S)$ be the set of all partitions of S. Let $\text{Part}_k(S)$ be the set of all partitions into at most k subsets, that is, all k-clusterings of S. Let $\Phi : \mathrm{Part}(S) \mapsto \mathbb{R}_{\geqslant 0}$ be the cost function defining our clustering problem, and define

$$\textsc{Opt}_k(S) := \min_{\mathscr{C} \in \mathrm{Part}_k(S)} \Phi(\mathscr{C})$$

to be the minimum cost of any k-clustering. An ε-coreset for this clustering problem defined by Φ is a set $R \subseteq S$ such that $|R|$ is independent of n and $\textsc{Opt}_k(R) \geqslant (1-\varepsilon) \cdot \textsc{Opt}_k(S)$.

To define the class of clusterings to which our method applies, we need the concept of r-packing [46]. Actually, we use a slightly weaker variant, which we define as follows. Let $|pq|$ denote the Euclidean distance between two points p and q. A subset $R \subseteq P$ of a point set P is called a *weak r-packing* for P, for some $r > 0$, if for any point $p \in P$ there exists a *packing point* $q \in R$ such that $|pq| \leqslant r$. (The difference with standard r-packings is that we do not require that $|qq'| > r$ for any two points $q, q' \in R$.) The clustering problems to which our method applies are the ones whose cost function is *regular*, as defined next.

Definition 4.1. A cost function $\Phi : \mathrm{Part}(S) \mapsto \mathbb{R}_{\geqslant 0}$ is called *$(c, f(k))$-regular,* if there is constant $c > 0$ and function $f : \mathbb{N}_{\geqslant 2} \mapsto \mathbb{R}_{\geqslant 0}$ such that the following holds.

- For any clustering $\mathscr{C} \in \mathrm{Part}(S)$, we have

  $$\Phi(\mathscr{C}) \geqslant c \cdot \max_{C \in \mathscr{C}} \mathrm{diam}(C),$$

  where $\mathrm{diam}(C) = \max_{p,q \in C} |pq|$ denotes the Euclidean diameter of the cluster C. We call this the *diameter-sensitivity* property.

- For any weak r-packing R of S, and any $k \geqslant 2$, we have that

  $$\textsc{Opt}_k(R) \leqslant \textsc{Opt}_k(S) \leqslant \textsc{Opt}_k(R) + r \cdot f(k).$$

Example. We show k-center problems have regular cost functions. For a cluster C, let $\mathrm{radius}_p(C)$ denote the radius of the minimum enclosing ball of C in the L_p-metric. In the L_∞ metric, for instance, $\mathrm{radius}_p(C)$ is half the edge length of a minimum enclosing axis-aligned cube[1] of C. Then the cost of a clustering $\mathscr{C}$ for the k-center problem in the L_p-metric is $\Phi_p^{\max}(\mathscr{C}) = \max_{C \in \mathscr{C}} \mathrm{radius}_p(C)$. One easily verifies that the cost function for the rectilinear k-center problem is $(1/(2\sqrt{d}), 1)$-regular, and for the Euclidean k-center problem it is $(1/2, 1)$-regular. (In fact $\Phi_p^{\max}(\mathscr{C})$ is regular for any p.)

[1]Throughout this chapter, when we speak of cubes (or squares, or rectangles, or boxes) we always mean axis-aligned cubes (or squares, or rectangles, or boxes).

4.2 The Algorithm

We start with a high-level overview of our approach. Let S be the given point set for which we want to find a coreset. Our algorithm is shown in Algorithm 2.

Algorithm 2: FINDCORESET(S, k, ε)

```
1 compute a lower bound LB on $\textsc{Opt}_k(S)$.
2 $r := \varepsilon \cdot \textsc{lb}/f(k)$
3 compute a weak $r$-packing $R$ on $S$.
4 return $R$
```

Note that R is the desired ε-coreset because Φ is $(c, f(k))$-regular for some constant $c > 0$ and some function $f : \mathbb{N}_{\geqslant 2} \mapsto \mathbb{R}_{\geqslant 0}$. The following lemma is immediate.

Lemma 4.2. $\textsc{Opt}_k(R) \geqslant (1-\varepsilon) \cdot \textsc{Opt}_k(S)$.

Next, we will show how to perform Steps 1 and 3: we will describe an algorithm that allows us to compute a suitable lower bound LB and a corresponding weak r-packing, such that the size of the r-packing depends only on ε and k but not on $|S|$.

Our lower bound and weak packing computation are based on so-called cube covers. A *cube cover* of S is a collection $\mathscr{B}$ of interior-disjoint cubes that together cover all the points in S and such that each $B \in \mathscr{B}$ contains at least one point from S (in its interior or on its boundary). Define the size of a cube B, denoted by size(B), to be its edge length. The following lemma follows immediately from the fact that the diameter of a cube B in $\mathbb{R}^d$ is $\sqrt{d} \cdot \mathrm{size}(B)$.

Lemma 4.3. *Let $\mathscr{B}$ be a cube cover of S such that* $\mathrm{size}(B) \leqslant r/\sqrt{d}$ *for all $B \in \mathscr{B}$. Then any subset $R \subseteq S$ containing a point from each cube $B \in \mathscr{B}$ is a weak r-packing for S.*

Our next lemma shows we can find a lower bound on $\textsc{Opt}_k(S)$ from a suitable cube cover.

Lemma 4.4. *Suppose the cost function Φ is $(c, f(k))$-regular. Let $\mathscr{B}$ be a cube cover of S such that $|\mathscr{B}| > k2^d$. Then* $\textsc{Opt}_k(S) \geqslant c \cdot \min_{B \in \mathscr{B}} \mathrm{size}(B)$.

Proof. For two cubes B, B' such that the maximum x_i-coordinate of B is at most the minimum x_i-coordinate of B', we say that B is *i-below* B' and B' is *i-above* B. We denote this relation by $B \prec_i B'$. Now consider an optimal k-clustering $\mathscr{C}_{\mathrm{opt}}$ of S. By the pigeonhole principle, there is a cluster $C \in \mathscr{C}_{\mathrm{opt}}$ containing points from at least $2^d + 1$ cubes. Let $\mathscr{B}_C$ be the set of cubes that contain at least one point in C.

Clearly, if there are cubes $B, B', B'' \in \mathscr{B}_C$ such that $B' \prec_i B \prec_i B''$ for some $1 \leqslant i \leqslant d$, then the cluster C contains two points at distance at least $\text{size}(B)$ from each other. Since Φ is $(c, f(k))$-regular this implies that $\Phi(\mathscr{C}_{\text{opt}}) \geqslant c \cdot \text{size}(B)$, which proves the lemma.

Now suppose for a contradiction that such a triple B', B, B'' does not exist. Then we can define a characteristic vector $\Gamma(B) = (\Gamma_1, \ldots, \Gamma_d(B))$ for each cube $B \in \mathscr{B}_C$, as follows:

$$\Gamma_i(B) = \begin{cases} 1 & \text{if no cube } B' \in \mathscr{B}_C \text{ is } i\text{-below } B \\ 0 & \text{if no cube } B'' \in \mathscr{B}_C \text{ is } i\text{-above } B \end{cases}$$

Since the number of distinct characteristic vectors is $2^d < |\mathscr{B}_C|$, there must be two cubes $B_1, B_2 \in \mathscr{B}_C$ with identical characteristic vectors. However, any two interior-disjoint cubes can be separated by an axis-parallel hyperplane, so there is at least one $i \in \{1, \ldots, d\}$ such that B_1 is i-below or i-above B_2. But this contradicts that $\Gamma(B_1) = \Gamma(B_2)$. □

Details of Steps 1 and 3. For simplicity, we continue the discussion for 2-dimensional case. Generalizing the results to higher dimension is not hard.

Let $G(\alpha)$ denote the *grid* in $\mathbb{R}^2$ whose cells have size α and for which the origin O is a grid point. We call α, the *width* of grid $G(\alpha)$. We define cells to be open on the right and top, and closed on the left and bottom, so a cell is of the form $[\alpha i, \alpha(i+1)) \times [\alpha j, \alpha(j+1))$ for integers i, j. Notice that Lemmas 4.3 and 4.4 are still valid for such partially open cells instead of closed cells.

For any value α, *id* of a point $p = (x, y)$ is defined as $\text{id}(p, \alpha) = (\lfloor x/\alpha \rfloor, \lfloor y/\alpha \rfloor)$. Clearly, only points in the same cell of $G(\alpha)$, have similar id values. Using their id, we can store and fetch a set of points inside a grid efficiently, by using hashing.

For an integer s, we define a *canonical grid* with parameter s to be $G(2^s)$ and denote it by G_s. Similarly, for a point $p \in \mathbb{R}^2$ we define $\text{id}_s(p) := \text{id}(p, 2^s)$. A *canonical square* is any cell of one of the grids G_s. W.l.o.g, we assume all the points in a given set of points S reside in a single canonical square of some canonical grid.

We define the *separation level* of two points $p_1, p_2 \in \mathbb{R}^2$ to be the smallest integer s, such that p_1 and p_2 reside in a single canonical square in G_s and denote it by $\mathit{sdist}(p_1, p_2)$. We can compute this number in constant time in a computational model where computing $\lg x$, 2^x, and $\lfloor x \rfloor$ takes constant time ($\lg x = \log_2 x$). See Section 2.2.2 of [46] for more details.

We define the *canonical closest pair* of a set of points S to be the pair of points $p_1, p_2 \in S$ with minimum separation level. We define the *canonical closest pair distance* of S to be $\mathit{sdist}(p_1, p_2)$ or equivalently the minimum integer s such that at least two points in S are in the same cell of G_s.

Throughout this chapter, we use a data-structure H that stores a set of points indexed on their id in an arbitrary grid G by using hashing. H can store just one point for each cell of G. We denote the number of points in H by $|H|$. These are the operations supported by H (all times are expected):

1. Get the size, $|H|$, in $O(1)$ time.
2. Test if a cell of G is empty in $O(1)$ time.
3. Add a point to an empty cell of G in $O(1)$ time.
4. Retrieve all the points in H in $O(|H|)$ time.

We can also 'Rebuild H for a new grid G' by retrieving all points from the current data structure H, and inserting them one by one into a new structure H based on the grid G. If we wish to insert a point p but we already inserted another point with the same id, then p is discarded.

Computing canonical closest pair distance. Algorithm 3 is similar to the algorithm for computing *Euclidean closest pair* distance described in Section 1.2 of [46].

Algorithm 3: CClosestPair(S)

```
compute a random permutation $p_1,\dots,p_n$ of $S$.
$d_0 := \infty$, $s := \infty$, and set $H$ to empty
for $i := 1$ to $n$ do
    if $H[\mathrm{id}_s(p_i)]$ is empty
        $d_i := d_{i-1}$
    else
        $d_i := sdist(p_i, H[\mathrm{id}_s(p_i)])$
        $s := d_i - 1$, and rebuild $H$ for $G_s$
    $H[\mathrm{id}_s(p_i)] := p_i$
return $d_n$
```

Let d_i be the canonical closest pair distance of $p_1,\dots,p_i$. To compute d_i from d_{i-1}, consider the points $p_1,\dots,p_{i-1}$ in the grid G_s where $s = d_{i-1} - 1$. Obviously, no two points are in the same cell by definition of closest canonical pair distance. Now, if the point p_i is the only point in its cell (of G_s), it means the separation level of p_i to any other point in $p_1,\dots,p_{i-1}$ is at least d_{i-1} and therefore $d_i = d_{i-1}$. Otherwise, if p_i and p_j, where $1 \leqslant j < i$, are in the same cell of G_s, then $d_i = sdist(p_i, p_j)$.

Lemma 4.5. *Algorithm* CClosestPair *computes the canonical closest pair distance of a set of n points S in expected $O(n)$ time.*

Proof. As discussed above, after execution of the i-th iteration, the canonical closest pair distance of $p_1,\dots,p_i$ is stored in the variable d_i. This proves the correctness of the algorithm.

For each point p_i, we only need to rebuild H in time $O(i)$ if $d_i < d_{i-1}$; otherwise we spend $O(1)$ time for handling it. As the points are in random order, we have $\Pr[d_i < d_{i-1}] \leqslant 2/i$ for $i \geqslant 3$; therefore, the expected running time is

$$O(n) + \sum_{i=3}^{n} \Pr[d_i < d_{i-1}] \cdot O(i) = O(n)$$

□

Step 1 of FINDCORESET. We start with the grid G_s where $s = -\infty$ and add points in S one by one to this grid. Whenever the number of non-empty cells exceeds $16k$, we increase the parameter s of the grid G_s so that the number of non-empty cells of the new grid G_s is still greater that $4k$ but not greater than $16k$. After adding all the points S to grid G_s we can use Lemma 4.4 to lower bound $\mathrm{OPT}_k(S)$.

Algorithm 4: LOWERBOUND(S,k)

```
1 compute a random permutation $p_1,\dots,p_n$ of $S$.
2 $s := -\infty$, and set $H$ to empty
3 for $i := 1$ to $n$ do
4     if $H[\mathrm{id}_s(p_i)]$ is empty
5         $H[\mathrm{id}_s(p_i)] := p_i$
6         if $|H| > 16k$
7             $s :=$ CCLOSESTPAIR$(H)$
8             rebuild $H$ for the new grid $G_s$
9 return $c \cdot 2^s$
```

Lemma 4.6. *Given a set of points S and a value $k \geqslant 2$, the algorithm* LOWERBOUND *computes a lower bound for* $\mathrm{OPT}_k(S)$ *in expected $O(n + k^2 \log(n/k))$ time. Moreover, after execution of this algorithm G_s contains at most $16k$ nonempty cells.*

Proof. After execution of the algorithm, if $s = -\infty$ the algorithm returns 0 which is an obvious lower bound. Otherwise we show that $|H| > 4k$.

If $s \neq -\infty$, it means the value of s has changed in line 7. Whenever the value of s changes in line 7, each cell of G_s contains at most 4 points of H, because the new value of s is canonical closest pair distance of $p_1,\dots,p_i$. Therefore, after rebuilding H in line 8 the number of points in H decreases by a factor of at most 4. Since before a rebuild operation, we have $|H| > 16k$, after rebuild we have $|H| > 4k$ and Lemma 4.4 and Definition 4.1 proves the correctness of the lower bound returned by the algorithm.

Notice that $|H| > 16k$ only after adding a point to an empty cell of G_s and even at that time $|H| = 16k + 1$ and after execution of lines 7 and 8, we have $|H| \leqslant 16k$, because in the new grid G_s at least one of the canonical closest pair will be in the same cell and thus will be ignored by H. Therefore, the claim that G_s contains at most $16k$ nonempty cells is correct.

To compute the running time of the algorithm, notice that the most time consuming operations of the algorithm are lines 7 and 8, both of them run in $O(k)$ time. For each point p_i the lines 7 and 8 can get executed only if p_i is alone in G_s, i.e. p_i is the only point in its cell (of G_s) among the points $p_1, \dots, p_i$; otherwise we handle p_i in constant time. We know that after adding p_i to H, we have $|H| \leqslant 16k+1$; therefore among $p_1, \dots, p_i$ there can be at most $16k+1$ points that are alone in their cell in G_s. As the points are in random order, $\Pr[p_i \text{ is alone}] \leqslant (16k+1)/i$, for $i > 16k$. Hence, the expected running time is

$$O(n) + \sum_{i=16k}^{n} \Pr[p_i \text{ is alone}] \cdot O(k) = O\left(n + k^2 \log \frac{n}{k}\right)$$

□

Step 3 of FINDCORESET. Algorithm 5 computes a weak r-packing in linear time by selecting one point from each nonempty cell of the grid $G(r/\sqrt{2})$. The correctness of the algorithm follows from Lemma 4.3.

Algorithm 5: WEAKPACKING(S, r)

```
1 $\alpha := r/\sqrt{2}$, and set $H$ to empty
2 for $i := 1$ to $n$ do
3     if $H[\mathrm{id}(p_i, \alpha)]$ is empty
4         $H[\mathrm{id}(p_i, \alpha)] := p_i$
5 return $H$
```

Putting everything together. The running time of Algorithm FINDCORESET is dominated by the running time of Algorithm LOWERBOUND, which is $O(n + k^2 \log(n/k))$. Assuming k is constant, FINDCORESET runs in linear time.

By Lemma 4.6, there exists a cube cover of S using at most $16k$ cells of the grid $G(\mathrm{LB}/c)$. Therefore, the grid $G(r/\sqrt{2})$ where $r := \varepsilon \cdot \mathrm{LB}/f(k)$ has at most $16k \cdot \left(\left(\sqrt{2} f(k)/(c\,\varepsilon)\right) + 1\right)^2$ nonempty cells and Algorithm WEAKPACKING cannot return more than $O(k\left(f(k)/\varepsilon\right)^2)$ points in Step 3 of FINDCORESET.

As stated earlier, it is easy to generalize the algorithms and lemmas described above to higher dimension. Therefore, we can state the following theorem,

Theorem 4.7. *Let S be a set of n points in $\mathbb{R}^d$ and let Φ be a $(c, f(k))$-regular cost function. Then, it is possible to find an ε-coreset R for k-clustering according to Φ, of size $O(k\left(f(k)/\varepsilon\right)^d)$ in expected $O(n + k^2 \log(n/k))$ time, for values $k \geqslant 2$ and $\varepsilon > 0$.*

Corollary 4.8. *Let S be a set of n points in $\mathbb{R}^d$. It is possible to find an ε-coreset R for Euclidean (rectilinear) k-center, of size $O(k/\varepsilon^d)$ in expected $O(n + k^2 \log(n/k))$ time, for values $k \geqslant 2$ and $\varepsilon > 0$.*

4.3 Concluding Remarks

We presented an algorithm to compute an ε-coreset of size $O(k\big(f(k)/\varepsilon\big)^d)$ in linear time for clustering problems defined on a set of points in $\mathbb{R}^d$. Our method applies to a large class of clustering problems including the k-center problem in any L_p-metric, variants of the k-center problem where we want to minimize the sum (rather than maximum) of the cluster radii, and the 2-dimensional problem where we want to minimize the maximum or sum of the perimeters of the clusters.

As mentioned in the introduction, Har-Peled and Mazumdar [47] also have a similar result for k-means and k-median problems. An interesting direction for future research is to generalize our result for other cost functions.

Chapter 5

Computing Plurality Points

In this chapter, we study computational problems concerning plurality points. For convenience, we first recall the definition of plurality point given in the introduction. Let $V \subset \mathbb{R}^d$ be a set of n *voters* and let $\mathbb{R}^d$ be a space of possible choices. The utility of a point (choice) $p \in \mathbb{R}^d$ for a voter v is inversely proportional to $\text{dist}(v, p)$, the distance from v to p under a given distance function, and v prefers a point p over a point p' if $\text{dist}(v, p) < \text{dist}(v, p')$. Now a point $p \in \mathbb{R}^d$ is a plurality point if for any point $p' \in \mathbb{R}^d$ we have $|\{v \in V : \text{dist}(v, p) < \text{dist}(v, p')\}| \geqslant |\{v \in V : \text{dist}(v, p') < \text{dist}(v, p)\}|$. When the L_2 norm as distance function, we present a deterministic algorithm that decides if a plurality point exists (and, if so, computes one) in $O(n \log n)$ time[1] which is an improvement over the previously best running time, which was $O(n^d \log n)$.

We also consider the minimum-cost plurality problem where each voter is assigned a cost, and the goal is to find a minimum-cost subset $W \subset V$ of voters such that if we ignore the voters in W—that is, if we consider $V \setminus W$—then a plurality point exists. We solve this problem by presenting an algorithm that solves the problem in $O(n^4)$ time which solves the open problem proposed by Lin *et al.* [56] about the existence of a polynomial-time algorithm that solves this problem in $\mathbb{R}^3$. This result even improves on the previous $O(n^5 \log n)$ running time for the planar case devised by Lin *et al.* [56].

We also consider the following problem for unit-cost voters in $\mathbb{R}^d$: given a parameter k, find a minimum-cost set W of size at most k such that $V \setminus W$ admits a plurality point, if such a set exists. Our algorithm for this case runs in $O(k^3 n \log n)$ time when $d = 2$ and in $O(k^5 \log k + k^3 n \log n)$ expected time when $d > 2$.

We also introduce a new concept called *plurality balls*, as defined next. We define a ball $b(p, r)$ centered at p and of radius r to be a plurality ball if the

[1] Here and in the other bounds stated in this page and the following page, we assume the dimension d is a fixed constant, to make it easier to compare our results to earlier work. In the theorems stated later in this chapter, we also analyze the dependency on d.

following holds: there is no point p' outside $b(p,r)$ that is preferred by more voters than p. Note that a plurality point is a plurality ball of zero radius. We show that in the plane, the minimum-radius plurality ball can be computed in $O(T(n))$ time, where $T(n)$ is the time needed to compute the $\lfloor n/2 \rfloor$-level in an arrangement of n lines.

We also introduce a new utility function we call the *personalized L_1 norm* where each voter $v \in V$ has a *preference vector* $\langle w_1(v), \dots, w_d(v) \rangle$ of non-negative weights that specifies the relative importance of the various issues. The distance of a point $p \in \mathbb{R}^d$ to a voter v is now defined as $\text{dist}_w(v,p) := \sum_{i=1}^{d} w_i(v) \cdot |x_i(v) - x_i(p)|$, where $x_i(\cdot)$ denotes the i-th coordinate of a point. We present an algorithm that decides in $O(n^{d-1})$ time whether a set V of n voters admits a plurality point with respect to the personalized L_1 norm. For the special case when all preference vectors are identical—this case reduces to the normal case of using the L_1 norm—the running time improves to $O(n)$.

5.1 Plurality Points in the L_2 Norm

Let V be a set of n voters in $\mathbb{R}^d$. In this section we show how to compute a plurality point for V with respect to the L_2 norm in $O(n \log n)$ time, if it exists. We start by proving several properties of the plurality point in higher dimensions, which generalize similar properties that Lin *et al.* [56] proved in $\mathbb{R}^2$. These properties imply that if a plurality point exists, it is unique (unless all points are collinear). Our algorithm then consists of two steps: first it computes a single candidate point $p \in \mathbb{R}^d$, and then it decides if p is a plurality point.

5.1.1 Properties of Plurality Points in the L_2 Norm

As remarked in the introduction, plurality points can be characterized as follows.

Proposition 5.1 (Wu *et al.* [72]). *A point p is a plurality point for a set V of n voters in $\mathbb{R}^d$ with respect to the L_2 norm if and only if every open halfspace with p on its boundary contains at most $n/2$ voters.*

Verifying the condition in Proposition 5.1 directly is not efficient. Hence, we will prove alternative conditions for a point p to be a plurality point in $\mathbb{R}^d$, which generalize the conditions Lin *et al.* [56] stated for the planar case. First, we define some concepts introduced by Lin *et al.*

Let V be a set of n voters in $\mathbb{R}^d$, and consider a point $p \in \mathbb{R}^d$. Let $L(p)$ be the set of all lines passing through p and at least one voter $v \neq p$. The point p partitions each line $\ell \in L(p)$ into two opposite rays, which we denote by $\rho(\ell)$ and $\overline{\rho}(\ell)$. (The point p itself is not part of these rays.) We say that a line $\ell \in L(p)$ is *balanced* if $|\rho(\ell) \cap V| = |\overline{\rho}(\ell) \cap V|$. When n is odd, then p turns out to be a plurality point if and only if every line $\ell \in L(p)$ is balanced (which implies that

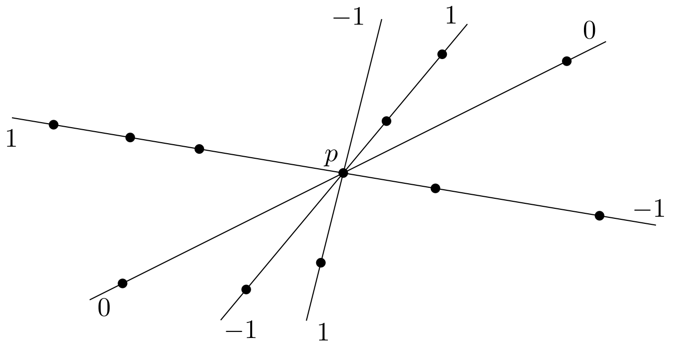


Figure 5.1: A plurality point p with alternating property

we must have $p \in V$). When n is even the situation is more complicated. Let $R(p)$ be the set of all rays $\rho(\ell)$ and $\overline{\rho}(\ell)$. Label each ray in $R(p)$ with an integer, which is the number of voters on the ray minus the number of voters from V on the opposite ray. Thus, a line ℓ is balanced if and only if its rays $\rho(\ell)$ and $\overline{\rho}(\ell)$ have label zero. Let $L^*(p)$ be the set of all unbalanced lines in $L(p)$ and let $R^*(p)$ be the corresponding set of rays. We now define the so-called alternating property, as introduced by Plott [61] and later by Lin *et al.* [56]. This property is restricted to the 2-dimensional setting, where we can order the rays in $R^*(p)$ around p. In this setting, the point p is said to have the *alternating property* if the following holds: the circular sequence of non-zero labels of the rays in $R^*(p)$, which we obtain when we visit the rays in $R^*(p)$ in clockwise order around p, alternates between labels $+1$ and -1 (see Figure 5.1). Note that if p has the alternating property then the number of unbalanced lines must be odd.

Recall that for $0 \leqslant k \leqslant d$ a *k-dimensional flat*, or a *k-flat* for short, is the affine hull of $k+1$ affinely independent points. For example, a line is a 1-flat and a hyperplane is a $(d-1)$-flat. The number k is called the dimension of the flat and is denoted by $\dim(f)$. The following theorem gives necessary and sufficient conditions for a point p to be a plurality point for a set V of voters.

Theorem 5.2. *Let V be a set of n voters in $\mathbb{R}^d$, with $d \geqslant 1$, and let p be an arbitrary point.*

a. If n is odd, p is a plurality point if and only if $p \in V$ and every line in $L(p)$ is balanced.

b. If n is even and $p \notin V$, then p is a plurality point if and only if every line in $L(p)$ is balanced.

c. If n is even and $p \in V$, then p is a plurality point if and only if all unbalanced lines in $L(p)$ are contained in a single 2-dimensional flat f and p has the alternating property for the set $f \cap V$.

For $d = 1$ the theorem is trivial, and for $d = 2$—the condition in case c then simply states that p has the alternating property—the theorem was proved by Lin *et al.* [56] and by Plott [61]. Our contribution is the extension to higher dimensions. Before proving Theorem 5.2, we need the following lemma regarding the robustness of plurality points to dimension reduction.

Lemma 5.3. *Let p be a plurality point for a set V in $\mathbb{R}^d$, with $d \geqslant 1$, and let f be any lower-dimensional flat containing p. Then p is a plurality point for $f \cap V$.*

Proof. We prove the statement by induction on $d - \dim(f)$.

Base case: $d - \dim(f) = 1$, that is, f is a hyperplane.

Let f^+ and f^- denote the open halfspaces bounded by f, and assume without loss of generality that $|f^+ \cap V| \geqslant |f^- \cap V|$. Suppose for a contradiction that p is not a plurality point for $f \cap V$. Then there must be a $(d-2)$-flat $g \subset f$ containing p such that, within the $(d-1)$-dimensional space f, the number of voters lying strictly to one side of g is greater than $|f \cap V|/2$. Let $g^+ \subset f$ denote the part of f lying to this side of g. Now imagine rotating f around g by an infinitesimal amount. Let $\hat{f}$ denote the rotated hyperplane, and $\hat{f}^+$ be the open halfspace that contains all voters in f^+. Observe that we can choose the direction of the rotation such that the voters in $g^+ \cap V$ end up in $\hat{f}^+$. But then $|\hat{f}^+ \cap V| \geqslant |f^+ \cap V| + |g^+ \cap V| > |f^+ \cap V| + |f \cap V|/2 \geqslant n/2$, which contradicts the assumption that p is a plurality point.

Induction step: $d - \dim(f) > 1$.

Let h be a hyperplane that contains f. From the base case we know that p must be a plurality point for $h \cap V$. Hence, we can apply our induction hypothesis to the flat f in the $(d-1)$-dimensional space h to conclude that p must be a plurality point for $f \cap V$.

□

Corollary 5.4. *Let V be a set of voters in $\mathbb{R}^d$, for $d \geqslant 2$, that are not collinear. Then V has at most one plurality point.*

Proof. Suppose for a contradiction that V has two distinct plurality points p_1 and p_2. Let f be a 2-flat containing p_1 and p_2, and a voter v not collinear with p_1 and p_2. By Lemma 5.3, both p_1 and p_2 are plurality points for $f \cap V$. But this contradicts the result by Wu *et al.* [72] that any set of voters in the plane admits at most one plurality point. □

Now we are ready to prove Theorem 5.2.

Proof of Theorem 5.2. Since the case $d = 2$ was already proved by Lin *et al.* [56], and the case $d = 1$ is trivial, we assume $d \geqslant 3$.

(a,⇐) and (b,⇐). Let p be a point such that all lines in $L(p)$ are balanced. Consider an arbitrary open halfspace h^+ whose bounding hyperplane h contains p, and let h^- be the open halfspace opposite to h^+. Since every line in $L(p)$ is balanced, there is a bijection from $h^+ \cap V$ to $h^- \cap V$. Hence, $|h^+ \cap V| \leqslant n/2$. Since this holds for any open halfspace h^+, the point p is a plurality point.

(b,⇒). Let $p \notin V$ be a plurality point, and consider a line $\ell \in L(p)$. By Lemma 5.3 the point p is also a plurality point on ℓ. Since $p \notin V$, this implies that ℓ is balanced.

(a,⇒). Assume n is odd, and let p be a plurality point. Suppose for a contradiction there is an unbalanced line $\ell \in L(p)$. First observe that we must have $p \in V$, otherwise p is not a plurality point on ℓ which contradicts Lemma 5.3. Let h be a hyperplane containing ℓ that contains no voters except those on ℓ, and assume without loss of generality that $|h^+ \cap V| \geqslant |h^- \cap V|$. Let $\rho(\ell)$ and $\overline{\rho}(\ell)$ denote the opposite rays along ℓ emanating from p. Since ℓ is unbalanced these rays do not contain the same number of voters, so we can assume $|\rho(\ell) \cap V| > |\overline{\rho}(\ell) \cap V|$. We consider two cases.

The first case is that $|h^+ \cap V| = |h^- \cap V|$. Since n is odd, this means that $|\cap V|$ is odd, and so $|(h \cap V) \setminus \{p\}|$ is even. But then $|\rho(\ell) \cap V| \geqslant |\overline{\rho}(\ell) \cap V| + 2$. Now imagine rotating h infinitesimally around the $(d-2)$-flat through p orthogonal to ℓ. For the resulting hyperplane $\hat{h}$ we then have $|\hat{h}^+ \cap V| \geqslant |\hat{h}^- \cap V| + 2$. Since p is the only voter on $\hat{h}$ this implies $|\hat{h}^+ \cap V| > n/2$, which contradicts that p is a plurality point.

The second case is that $|h^+ \cap V| > |h^- \cap V|$. Now we can use a similar argument: this time we only have $|\rho(\ell) \cap V| \geqslant |\overline{\rho}(\ell) \cap V| + 1$, but we also have $|h^+ \cap V| > |h^- \cap V|$ and so we still have $|\hat{h}^+ \cap V| \geqslant |\hat{h}^- \cap V| + 2$, leading to the desired contradiction.

(c,⇐). Assume n is even and let p be a point such that all unbalanced lines in $L(p)$ are contained in a single 2-dimensional flat f and p has the alternating property for the set $f \cap V$. Since all the lines $L(p)$ not contained in f are balanced, this mean $|f^+ \cap V| = |f^- \cap V|$, and so $|f \cap V|$ is even. Consider an arbitrary open halfspace h^+ whose bounding hyperplane h contains p, and let h^- be the opposite open halfspace. If h contains f then all unbalanced lines lie in h and so $|h^+ \cap V| = |h^- \cap V|$, which implies $|h^+ \cap V| \leqslant n/2$. If h does not contain f, we can argue as follows. Let $\ell := h \cap f$. Since the theorem is true for $d = 2$ and as we have the alternating property on f and $|f \cap V|$ is even, we know that p is a plurality point for $f \cap V$. Hence, the number of voters on f on either side of ℓ is at most $|f \cap V|/2$. But then we have $|h^+ \cap V| \leqslant n/2$, because all voters not in f lie on balanced lines. We conclude that for any open halfspace h^+ we have $|h^+ \cap V| \leqslant n/2$, and so p is a plurality point.

(c,⇒). Assume n is even and let p be a plurality point. We first argue that all unbalanced lines must lie on a single 2-flat. Assume for a contradiction that there are three unbalanced lines that do not lie on a common 2-flat. Let g be the 3-flat spanned by these lines, and let $L^*_g(p) \subset L^*(p)$ be the set of all unbalanced lines contained in g. Let $f_1 \subset g$ be a 2-flat not containing p and not parallel to any of the lines in $L^*_g(p)$. Each of the lines in $L^*_g(p)$ intersects f_1 in a single point, and these intersection points are not all collinear. According to the Sylvester-Gallai Theorem [42] this implies there is an ordinary line in f_1, that is, a line containing exactly two of the intersection points. Thus we have an ordinary 2-flat in g, that is, a flat f_2 containing exactly two lines from $L^*(p)$. This implies that $f_2 \cap V$ does not have the alternating property, and since we know by the result of Lin *et al.* that the theorem holds when $d = 2$ this implies that p is not a plurality point in f_2. However, this contradicts Lemma 5.3.

We just argued that all unbalanced lines must lie on a single 2-flat f. By Lemma 5.3 the point p is a plurality point on f. Since the theorem holds for $d = 2$, we can conclude that $f \cap V$ has the alternating property.

□

5.1.2 Finding Plurality Points in the L_2 Norm

We now turn our attention to finding a plurality point. Our algorithm needs a subroutine for finding a *median hyperplane* for V, which is a hyperplane h such that $|h^+ \cap V| < n/2$ and $|h^- \cap V| < n/2$, where h^+ and h^- denote the two open halfspaces bounded by h. The following lemma is easy to prove.

Lemma 5.5. *Let $v \in V$ be a voter that lies on a hyperplane h_0 such that no two voters strictly lie on opposite sides of h_0 (that is, either h_0^+ does not contain voters, or h_0^- does not contain voters). Then we can find a median hyperplane h containing v in $O(dn)$ time.*

Proof. Assume without loss of generality v is the origin and that h_0 is the hyperplane $x_d = 0$. Define $V_0 := V \cap h_0$ and $V_1 := V \setminus V_0$. Note that all voters in V_1 lie to the same side of h_0, say in h_0^+. If $|V_1| < n/2$ then h_0 is a median hyperplane. Otherwise, let f_0 be the $(d-2)$-flat $x_d = x_{d-1} = 0$, and note that f_0 contains v. We can now rotate a hyperplane h around f_0 until it satisfies the requirements. Next we make this precise.

Define n_0^- to be the number of voters in V_0 that lie (within the $(d-1)$-dimensional space h_0) strictly on the negative side of f_0. Project all voters in V_1 onto the hyperplane $h_1 : x_d = 1$, using v as the center of projection. Let X be the multi-set of all x_{d-1}-coordinates of the projected voters. Compute an element $x^* \in X$ such that

$$|\{x \in X : x < x^*\}| < n/2 - n_0^- \leqslant |\{x \in X : x \leqslant x^*\}|.$$

This can be done in $O(dn)$ time using a standard rank-selection algorithm. Note that there is at least one voter $v_1 \in V_1$ whose projection has x_{d-1}-coordinate equal to x^*. We claim that the hyperplane h containing f_0 and v_1 is a median hyperplane. To see this, we first observe that

$$|h^- \cap V| = n_0^- + |\{x \in X : x < x^*\}| < n/2.$$

Second, since h contains v as well as the voters in V_1 whose projection has x_{d-1}-coordinate x^* we have

$$|h^- \cap V| + |h \cap V| \geqslant n_0^- + (n/2 - n_0^-) + 1 = n/2 + 1.$$

Thus we have $|h^+ \cap V| < n/2$, which proves the claim and, hence, the lemma. □

Recall that for $d \geqslant 2$ the plurality point is unique, if it exists. The algorithm below either reports a single candidate point p—we show later how to test if the candidate is actually a plurality point or not—or it returns $\emptyset$ to indicate that it already discovered that a plurality point does not exist. When called with a set V of n collinear voters, the algorithm will return the set of all plurality points; if n is even the set is a segment connecting the two median voters, if n is odd the set is a degenerate segment consisting of the (in this case unique) median voter. We call this segment the *median segment*.

```
FindCandidates($V$)

1. If all voters in $V$ are collinear, then return the median segment of $V$.

2. Otherwise, proceed as follow.

    (a) Let $v_0 \in V$ be a voter with minimum $x_d$-coordinate. All voters in $V$
        lie either on the plane $h_0 : x_d = x_d(v_0)$ or in the halfspace above $h_0$,
        which we denote by $h_0^+$. Find a median hyperplane $m_0$ containing
        $v_0$ using Lemma 5.5, and let $cand_0 :=$ FindCandidates($m_0 \cap V$).

    (b) If $cand_0$ consists of a single point or $cand_0 = \emptyset$ then return $cand_0$.

    (c) If $cand_0$ is a (non-degenerate) segment then let $v_1 \in V$ be a voter
        whose distance to $m_0$ is maximized and $h_1$ be the hyperplane parallel
        to $m_0$ containing $v_1$. All voters in $V$ lie either on $h_1$ or in the
        halfspace defined by $h_1$ and containing $m_0$, which we denote by $h_1^+$.
        Find a median hyperplane $m_1$ containing $v_1$ using Lemma 5.5, and
        let $cand_1 :=$ FindCandidates($m_1 \cap V$). Return $cand_0 \cap cand_1$.
```

Lemma 5.6. *Algorithm* FindCandidates(V) *returns in* $O(dn)$ *time a set* cand *of candidate plurality points such that*

i. if all voters in V are collinear then cand *is the set of all plurality points of V;*

ii. *if not all voters in V are collinear then* cand *contains at most one point, and no other point can be a plurality point of V.*

Proof. We first prove the correctness of the algorithm, and then consider the time bound.

If all voters in V are collinear then the algorithm returns the correct result in Step 1, so assume not all voters are collinear. Consider the median hyperplane computed in Step 2a. Since $|m_0^+ \cap V| < n/2$ and $|m_0^- \cap V| < n/2$, for any point $p \notin m_0$ there is an open halfspace containing p and bounded by a hyperplane parallel to m_0 that contains more than $n/2$ voters. Hence, by Proposition 5.1 any plurality point for V must lie on m_0. By Lemma 5.3, if a plurality point exists for V it must also be a plurality point for $m_0 \cap V$. By induction we can assume that FINDCANDIDATES($m_0 \cap V$) is correct. Hence, the result of the algorithm is correct when $cand_0$ consists of a single point or $cand_0 = \emptyset$. Note that when $cand_0$ is a (non-degenerate) segment—this only happens when all voters in $m_0 \cap V$ are collinear—we must have $V \neq m_0 \cap V$, as otherwise V would be collinear and we would be done after Step 1. Hence, $v_1 \notin m_0$. By the same reasoning as above the median hyperplane m_1 must contain the plurality point of V (if it exists). But then the plurality point must lie in $cand_0 \cap cand_1$, and since $v_1 \notin m_0$ we know that $cand_0 \cap cand_1$ is either a single point or it is empty. This proves the correctness.

To prove the time bound, we note that we only have two recursive calls when the first recursive call reports a non-degenerate candidate segment. This only happens when all voters in $m_0 \cap V$ are collinear, which implies the recursive call just needs to compute a median segment in $O(dn)$ time—it does not make further recursive calls. Thus we can imagine adding this time to the original call, so that we never make more than one recursive call. Since the recursion depth is at most d, and each call needs $O(dn)$ time, the bound follows. □

Our algorithm to find a plurality point first calls FINDCANDIDATES(V). If all points in V are collinear we are done—FINDCANDIDATES(V) then reports the correct answer. Otherwise we either get a single candidate point p, or we already know that a plurality point does not exist. It remains to test if a candidate point p is a plurality point or not.

Lemma 5.7. *Given a set V of n voters in $\mathbb{R}^d$ and a candidate point p, we can test in $O(dn \log n)$ time if p is a plurality point in the L_2 norm.*

Proof. First compute the set $L(p)$ of lines containing p and at least one voter. We can compute $L(p)$, and for each line $\ell \in L(p)$ the number of voters on the rays $\rho(\ell)$ and $\overline{\rho}(\ell)$, in $O(dn \log n)$ time. (To this end, we take the line ℓ_v through p and v for each voter $v \neq p$, and group these into subsets of identical lines.) According to Theorem 5.2, we can now immediately decide if p is a plurality point when n is odd, or when n is even and $p \notin V$. When n is even and $p \in V$ we first check in $O(dn)$ time if all unbalanced lines lie in a 2-flat f. If not, then p is not a plurality point. Otherwise we check the alternating property in $O(dn\log n)$ time. □

We obtain the following theorem.

Theorem 5.8. *Let V be a set of n voters in $\mathbb{R}^d$, where $d \geqslant 2$. Then we can find in $O(dn\log n)$ time the plurality point for V in the L_2 norm, if it exists, and this time bound is optimal for $d = 2$.*

Proof. The time bound and correctness of our algorithm follow from Lemmas 5.6 and 5.7. The optimality of the algorithm follows from an easy reduction from SET EQUALITY, which takes $\Omega(n\log n)$ time in the algebraic decision-tree model [20]. To see this, consider an instance A, B of SET EQUALITY, where $A = \{a_1, \dots, a_n\}$ and $B = \{b_1, \dots, b_n\}$ are sets of n numbers. Define $V := \{(a_i, 1) : a_i \in A\} \cup \{(0,0)\} \cup \{(-b_i, -1) : b_i \in B\}$. Then $A = B$ if and only if V admits a plurality point (which must then be the point $(0,0)$), as follows immediately from Theorem 5.2a. □

5.2 Dealing With Point Sets That Do Not Admit a Plurality Point

Most point sets do not admit a plurality point in the L_2 norm. In this section we consider two ways of dealing with this. First, we present an algorithm to compute a minimum-cost subset $W \subset V$ such that $V \setminus W$ admits a plurality point, which can be useful when besides a few "outliers" the set V admits a plurality point. In general, however, the set W that needs to be removed can be quite large. We therefor also present an algorithm for computing a minimum-radius plurality ball in $\mathbb{R}^2$.

5.2.1 The Minimum-Cost Plurality Problem

Let V be a set of n voters in $\mathbb{R}^d$, where each voter v has a cost $cost(v) > 0$ associated to it. For a candidate plurality point p—here we consider all points in $\mathbb{R}^d$ as candidates—we define W_p to be a minimum-cost subset of V such that p is a plurality point for $V \setminus W_p$. We define the *price* of p to be the cost of W_p. Our algorithm will report a pair (p, W_p), where p is a cheapest candidate plurality point. The algorithm has two main parts: one finds the cheapest candidate that does not coincide with any voters, and the other finds the cheapest candidate that coincides with a voter.

Let $L(V)$ be the set of lines passing through at least two voters in V, and let $P(V)$ be the set of all intersection points of the lines in $L(V)$, excluding the intersection points coinciding with a voter. To find the cheapest candidate p that does not coincide with a voter, we only have to consider points in $P(V)$. Indeed, if all points in $V \setminus W_p$ are collinear then we can pick p to coincide with a voter; otherwise we know by Theorem 5.2b that p must be an intersection point of two lines in $L(V \setminus W_p)$, and so $p \in P(V)$. We will need the following lemma for the planar case.

Lemma 5.9 (Lin *et al.* [56]). *Let p be a candidate plurality point for a set V in $\mathbb{R}^2$. Then we can compute in $O(n \log n)$ time the price of p, together with the subset W_p.*

We also need the following lemma to compute all the intersections of each line with other lines in sorted order.

Lemma 5.10. *Given a set $\mathscr{L}$ of m lines in $\mathbb{R}^d$, we can list all intersection points of each line ℓ with the other lines in sorted order along ℓ in total time $O(d^2 m^2)$.*

Proof. Consider the set $\mathscr{F}$ of all 2-flats defined by any two axes x_i and x_j $(1 \leqslant i, j \leqslant d)$. We have $O(d^2)$ of them in total. For any such 2-flat $f \in \mathscr{F}$, we project all lines in $\mathscr{L}$ onto f and construct the arrangement in f in $O(m^2)$ time [36] and then check which intersections on f correspond to actual intersections in $\mathbb{R}^d$. Finally, for each line ℓ we merge the $O(d^2)$ lists of intersection points to obtain a complete, sorted list of intersection points along ℓ. In total, this takes $O(d^2 m^2)$ time.

Any two intersecting lines $\ell_1, \ell_2 \in \mathscr{L}$, define a 2-flat; as this 2-flat cannot be orthogonal to all the 2-flats in $\mathscr{F}$, it will be projected as a plane onto at least one of them. Therefore, we don't miss any intersection and this proves the correctness of the algorithm. □

In the algorithm below, we use $L(p, V')$ to denote the set of lines through a point p and at least one voter in a set $V' \subseteq V$.

MINCOSTPLURALITYPOINT(V)

1. Compute the set $L(V)$. If $|L(V)| = 1$, that is, all voters lie on a common line ℓ, then compute a median p along ℓ as a plurality point and report $(p, \emptyset)$.

2. Compute a cheapest candidate p that does not coincide with a voter, as follows. Compute the set $P(V)$. For each line $\ell \in L(V)$, sort the intersection points along ℓ. Using Lemma 5.10, this can be done in $O(d^2 n^4)$ time in total. Let $C := \sum_{v \in V} cost(v)$ be the total cost of all voters. For each intersection point $p \in P(V)$, let $\gamma(p)$ be the total cost of all voters v for which there is no line in $L(v, V \setminus \{v\})$ that contains p; we can compute $\gamma(p)$ in $O(n^4)$ time in total by determining the total cost of all voters that *do* have a line in $L(v, V \setminus \{v\})$ that contains p, and then subtracting this cost from C.

   (a) Traverse each line $\ell \in L(V)$, to visit the intersection points along ℓ in order. During the traversal, maintain the number of voters on ℓ on either side of the current intersection point p. Thus we know how many voters we have to remove to make ℓ balanced, and also from which side we should remove them. If we have to remove k voters, we have to remove the k cheapest voters on the relevant side. The subset $W_p(\ell)$ that we have to remove to make ℓ balanced only changes when p passes over a voter v on ℓ. When this happens we can compute the new $W_p(\ell)$ in linear time. In this way the traversal of ℓ takes $O(n^2)$ time in total, so over all $\ell \in L(V)$ we spend $O(n^4)$ time.

(b) For each intersection point p compute the price of p. This price has two components: the price to make every line $\ell \in L(V)$ that contains p balanced, and the price to remove any voter v for which the line $\ell(v,p)$ through v and p is not a line in $L(V)$. The first component equals $\sum_{\ell \ni p} cost(W_p(\ell))$. The second component equals $\gamma(p)$, which we precomputed.

3. Compute a cheapest candidate p that coincides with a voter $v \in V$. To this end, compute for each voter v the price of setting $p := v$—below we describe how to do this in $O(dn^3)$ time per voter—and take the cheapest of all n possibilities.

   Consider a candidate p coinciding with some $v \in V$. By Theorem 5.2 all unbalanced lines in $L(p, V \setminus W_p)$, if any, lie on a single 2-flat f. We will compute $price_p(f)$, the price to make p into a plurality point under the condition that all unbalanced lines lie in f, over all 2-flats f spanned by two lines from $L(p, V \setminus \{v\})$. Then we take the best of the results.

   Fix a 2-flat f spanned by two lines $\ell_1, \ell_2 \in L(p, V \setminus \{v\})$. Let L_f be the subset of lines from $L(p, V \setminus \{v\})$ contained in f, and let L'_f be the subset of lines not contained in f.

   (a) Compute $price_p(L_f)$, the price of making p a plurality point on f, using Lemma 5.9. This takes $O(n_f \log n_f)$ time, where $n_f := |f \cap V|$.

   (b) For each line $\ell \in L'_f$, compute $price_p(\ell)$, the price of making ℓ balanced. (This can easily be done in $O(n)$ time: we compute the smallest number k of voters we have to remove on ℓ to make ℓ balanced, and then find the k cheapest voters on the heavier of the two rays along ℓ and emanating from p.) Let $price_p(L'_f) := \sum_{\ell \in L'_f} price_p(\ell)$.

   (c) Set $price_p(f) := price_p(L_f) + price_p(L'_f)$.

4. Let p be the cheaper of the two candidates found in Steps 2 and 3, respectively. Compute the set W_p for this candidate—this takes $O(n \log n)$ time—and report (p, W_p).

The correctness of the algorithm follows from Theorem 5.2 and the discussion above. As for the running time, we note that Steps 1, 2, and 4 all run in $O(d^2 n^4)$

time. For Step 3, the time needed to compute the price of a single voter v is $\sum_f O(n_f \log n_f + dn)$, which is bounded by $O(dn^3 + \sum_f n_f \log n_f)$. Because every voter $v' \in V$ lies on at most n of the flats f (generated by the lines $\ell(v, v')$ and $\ell(v, v'')$ through v, v' and v, v'', respectively) we have $\sum_f n_f = O(n^2)$ and so $\sum_f n_f \log n_f = O(n^2 \log n)$. Hence, the whole algorithm runs in $O(d^2 n^4)$ time.

Theorem 5.11. *Let V be a set of n voters in $\mathbb{R}^d$, each with a positive cost, where $d \geqslant 2$. Then we can compute in $O(d^2 n^4)$ time a minimum-cost subset $W \subset V$ such that $V \setminus W$ admits a plurality point in the L_2 norm.*

Our algorithm for finding a minimum-cost plurality point checks $O(n^4)$ candidate points. The algorithm from the previous section for deciding if a plurality point exists avoids this, resulting in a near-linear running time. An obvious question is if a faster algorithm is also possible for the minimum-cost plurality-point problem. While we do not have the answer to this question, we can show that, even in the plane and when all voters have unit cost, it is unlikely that the problem can be solved in truly subquadratic time. We do this by a reduction from the problem THREE CONCURRENT LINES, which is to decide if a set of n lines has three or more lines meeting in a single point. THREE CONCURRENT LINES is 3SUM-hard [41] and has an $\Omega(n^2)$ lower bound if only sidedness tests are used [39].

Theorem 5.12. *Suppose we have an algorithm solving the minimum-cost plurality-point problem for any set of n unit-cost voters in the plane in time $T(n)$. Then there is a probabilistic algorithm solving* THREE CONCURRENT LINES *with probability 1 in $O(n \log n + T(n))$ time.*

Proof. Let L be a set of n lines for which we want to solve THREE CONCURRENT LINES. We will generate a set V of $2n$ voters such that L contains three or more concurrent lines if and only if there is a subset $W \subset V$ of size at most $2n-6$ such that $V \setminus W$ admits a plurality point. To this end we first compute in $O(n \log n)$ time a large circle C that contains all vertices of the arrangement $\mathscr{A}(L)$ in its interior. The circle C can be any circle containing the axis-parallel bounding box of the vertices of $\mathscr{A}(L)$. (It is well known that the bounding box of the vertices of a planar arrangement can be found in $O(n \log n)$ time; see Exercise 8.4 from [31].)

For each line $\ell \in L$ construct two small segments contained in ℓ, a segment $s_1(\ell) \subset \ell$ around the first intersection point of ℓ and C and another segment $s_2(\ell) \subset \ell$ around the second intersection point. Finally, we pick a voter uniformly at random in each segment $s_i(\ell)$ with $i \in \{1, 2\}$ and $\ell \in L$, thus creating a set V of $2n$ voters. We pick the segments $s_i(\ell)$ sufficiently small so that the set V is guaranteed to be in convex position. (Computing suitable lengths can be done in linear time, after we have sorted the intersection points of the lines in L with the circle C in clockwise order along C.)

If L has three or more lines meeting at a point p, then we can make p into a plurality point by removing at most $2n-6$ points on the lines not passing through p. To prove the other direction, note that to make any of the voters in V into a plurality point we have to delete all but one of the remaining $2n-1$ voters (since the voters are in convex position). Let $L(V) := \{\ell(v, v') : v, v' \in V\}$ be the set of all lines through a pair v, v' of voters. To finish the proof we observe that by picking the voters randomly on the segments $s_i(\ell)$, the following holds with probability 1: the set $L(V)$ has no triple intersections except those already present in L. □

5.2.2 A Fixed-Parameter Algorithm for Unit-Cost Voters

The proof of Theorem 5.12 uses a problem instance where many voters must be removed to obtain a plurality point. Below we show that a plurality point for which only a few voters have to be removed can be found in near-linear time for unit-cost voters. More precisely, we consider the case where we are given a set V of unit-cost voters in $\mathbb{R}^d$ and a parameter k, and we want to compute a smallest subset $W \subset V$ of size at most k (if it exists) such that $V \setminus W$ admits a plurality point.

Define a *k-line* to be a line ℓ such that for any hyperplane H containing ℓ, both open halfspaces bounded by H contain at most $n/2 + k$ voters. For a voter v, let $L(v)$ be the set of lines containing v and at least one other voter, and let $L_k(v)$ be the set of all k-lines in $L(v)$.

Lemma 5.13. *Let p be a plurality point of $V \setminus W$, for some subset W of size at most k. If $v \notin W$, then p lies on one of the lines in $L_k(v)$.*

Proof. Assume $v \notin W$. If $\ell(p, v)$, the line through p and v, is not a line in $L(v)$, then p does not coincide with a voter and $\ell(p, v)$ is unbalanced. But then p cannot be a plurality point, by Theorem 5.2. Hence, $\ell(p, v) \in L(v)$. If $\ell(p, v) \notin L_k(v)$ then there is an open halfspace bounded by a hyperplane containing p with more than $n/2 + k$ voters. But since $|W| \leqslant k$, such a point p cannot be a plurality point in $V \setminus W$, which implies we must have $\ell(p, v) \in L_k(v)$. □

The idea of our algorithm is now as follows. Consider a set $P := \{(v_{2i-1}, v_{2i}) : 1 \leqslant i \leqslant k+1\}$ of disjoint pairs of voters. Then there must be a pair $(v_{2i-1}, v_{2i}) \in P$ such that neither v_{2i-1} nor v_{2i} is in W. Hence, the point p we are looking for must lie on one of the lines in $L_k(v_{2i-1})$ and one of the lines in $L_k(v_{2i})$. So we check all intersection points between these lines, for every pair in P. The key to obtain an efficient algorithm is to generate P such that all sets $L_k(v_i)$ are small. There is one case that needs special attention, namely when there is a line—this must then be the line through v_{2i-1} and v_{2i}—that is present in both $L_k(v_{2i-1})$ and $L_k(v_{2i})$. This case is handled using the following lemma.

Lemma 5.14. *Suppose there exists a plurality point on $\ell := \ell(v, v')$ through two voters $v, v' \in V$ of price at most k. Let H be an arbitrary hyperplane containing ℓ but no other voter in $V \setminus \ell$. Let H^+ be any of the two open halfspaces bounded* *by H, and let $V^+ := H^+ \cap V$. Then either an intersection point of ℓ and a line in $\bigcup_{v'' \in V^+} L_k(v'')$, or one of the medians of the voters located on ℓ is a cheapest plurality point.*

Proof. Let p be a cheapest plurality point and W_p the corresponding subset of voters to be removed, where $|W_p| \leqslant k$. If there is a voter $v'' \in V^+ \setminus W_p$ then $p \in L_k(v'')$ by Lemma 5.13. Otherwise all voters in V^+ are in W_p. But then any line through p and a voter in $H^- \cap V$ is unbalanced, which implies that all voters in $H^- \cap V$ except at most one must be in W_p. If all voters in $H^- \cap V$ are in W_p, one of the medians of the voters located on ℓ is a cheapest plurality point. Otherwise, the number of voters located on ℓ must be odd and the median is the cheapest plurality point as any other point violates the alternating property. □

We now present our algorithm. We first explain the algorithm for the planar case, and then show how to extend it to higher dimensions. For technical reasons we assume $k \leqslant n/15$; see the proof of Theorem 5.16 for more details.

FIXEDPARAMETERMINCOSTPLURALITYPOINT(V, k)

1. Compute the set of convex layers of V. Let $V_1, V_2, \ldots$ be the sets of voters in these layers, where V_1 is the outermost layer, V_2 the next layer, and so on. Set $P := \emptyset$ and $i := 1$.

2. Visit the voters in V_i in clockwise order, starting at the lexicographically smallest voter in V_i. Put the first and second visited voters, the third and fourth visited voters, and so on as pairs into P until either P contains $k+1$ pairs or we run out of voters in V_i. In the former case we are done; in the latter case we start collecting pairs from the next layer, by setting $i := i+1$ and repeating the process. This continues until we have collected $k+1$ pairs. Note that in the latter case, exactly one voter remains unpaired if $|V_i|$ is odd.

3. Set $C := \emptyset$; the set C will contain candidates for the cheapest plurality points. For each pair $(v_{2i-1}, v_{2i}) \in P$, proceed as follows.

   (a) Compute the sets $L_k(v_{2i-1})$ and $L_k(v_{2i})$, and put all intersection points between two lines in $L_k(v_{2i-1}) \cup L_k(v_{2i})$ as candidates into C.

   (b) Let $\ell := \ell(v_{2i-1}, v_{2i})$. If ℓ is present in both $L_k(v_{2i-1})$ and $L_k(v_{2i})$, and ℓ contains at least $n/2 - 7k - 1$ voters, then proceed as follows. (We assume that the same line ℓ has not already been handled in this manner for a pair (v_{2j-1}, v_{2j}) with $j < i$, otherwise we can skip it now.) Assume without loss of generality that ℓ^+, the open halfplane above ℓ, contains at most as many voters as ℓ^-. For each voter $v \in \ell^+ \cap V$, compute $L_k(v)$ and add the intersection points of the lines in $L_k(v)$ with ℓ to the candidate set C. In addition, put a median voter along ℓ into C.

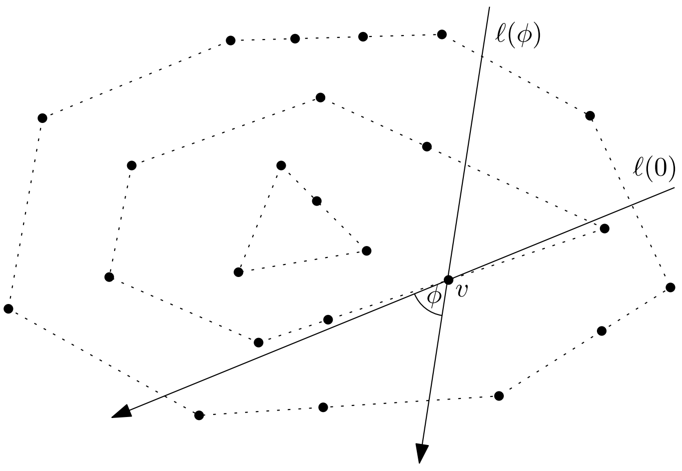


Figure 5.2: Sketch of the proof for Lemma 5.15

4. For each candidate point $p \in C$, compute a minimum-size subset W_p that makes p into a plurality point, using Lemma 5.9. Return the cheapest plurality point $p^* \in C$, provided $|W_{p^*}| \leqslant k$; if $|W_p| > k$ for all candidates, then report that it is not possible to obtain a plurality point by removing at most k voters.

The efficiency of our algorithm is based on the following lemma, which uses the fact that we constructed P using the convex layers of V.

Lemma 5.15. *Let v be a voter of some pair in P. Then $|L_k(v)| = O(k)$.*

Proof. Let V_i be the layer containing v, and let ℓ be a line tangent to $\textsc{ch}(V_i)$ at v, the convex hull of V_i. Assume without loss of generality that $\textsc{ch}(V_i)$ lies above ℓ. Then $\ell^- \cap V \subset V_1 \cup \cdots \cup V_{i-1}$, and so $|\ell^- \cap V| \leqslant 3k$. Moreover, the number of voters from V_i that lie on ℓ but "before" v is at most $2k$. (This is true because when we start visiting the voters in V_i in Step 2, we never start in the middle of a sequence of collinear voters.) Thus we can rotate ℓ slightly around v, so that v is the only voter on ℓ and $|\ell^- \cap V| \leqslant 5k$. Turn ℓ into a directed line in such a way that ℓ^- lies to the left of ℓ, and imagine rotating ℓ around v by π radians in counterclockwise direction. Let $\ell(\phi)$ be the line resulting from rotating ℓ over an angle ϕ, and define $F^-(\phi)$ and $F^+(\phi)$ to be the number of voters to the left and right of $\ell(\phi)$, respectively (see Figure 5.2). The line $\ell(\phi)$ is a k-line if it passes through a voter, $F^-(\phi) \leqslant n/2 + k$ and $F^+(\phi) \leqslant n/2 + k$. Note that $F^-(0) \leqslant 5k$ and $F^+(0) \geqslant n - 5k$, and $F^-(\pi) \geqslant n - 5k$ and $F^+(\pi) \leqslant 5k$. Since $F^-(\phi)$ can only decrease (and, similarly, $F^+(\phi)$ can only increase) when $\ell(\phi)$ passes through a voter that was to the left of $\ell(0)$, this implies that the number of times at which $\ell(\phi)$ is a k-line for v is $O(k)$. □

We can now prove the following theorem.

Theorem 5.16. *Let V be a set of n voters in the plane, and let k be a parameter with $k \leqslant n/15$. Then we can compute in $O(k^3 n \log n)$ time a minimum-size subset $W \subset V$ such that $V \setminus W$ admits a plurality point in the L_2 norm or report that there is no such subset W of size at most k.*

Proof. To prove the time bound, we note that we can compute the convex layers in $O(n \log n)$ time [25]. Step 2 runs in $O(k)$ time. Computing the set $L_k(v)$ for a voter v can easily be done in $O(n \log n)$ time by using a sweeping algorithm that rotates a line around v and keeps track of the number of points on each of the halfspaces defined by that line. By Lemma 5.15, Step 3a takes $O(n \log n + k^2)$ time. To bound the running time of Step 3b, we observe that there can be at most $O(1)$ pairs (v_{2i-1}, v_{2i}) to which this case applies. Indeed $\ell(v_{2i-1}, v_{2i})$ should contain at least $n/2 - 7k - 1$ voters, and since $k \leqslant n/15$ there can only be $O(1)$ such lines. Thus Step 3b needs $O(kn)$ time, and so Step 3 needs $O(k^2 n + kn \log n)$ time in total over all pairs in P, to generate $O(k^3)$ candidate points. Checking each of the candidates takes $O(n \log n)$, which proves the time bound.

The correctness of the algorithm follows from Lemmas 5.13 and 5.14, except for one thing: in Step 3b we only handle a line $\ell := \ell(v_{2i-1}, v_{2i})$ when it contains at least $n/2 - 7k - 1$ voters. This is allowed for the following reason. Note that ℓ is tangent to a convex hull $\mathrm{CH}(V_i)$ and on the side of ℓ that does not contain $\mathrm{CH}(V_i)$, say ℓ^+, there are at most $3k$ voters. Now consider a plurality point $p \in \ell$. Then there can be at most $3k+1$ voters in $(V \setminus W_P) \cap \ell^-$, by Theorem 5.2. Since $|W_p| \leqslant k$, we thus have $|V \cap \ell^-| \leqslant 4k+1$, which means that ℓ must contain at least $n - 7k - 1$ voters. □

Next we show how to generalize the algorithm to higher dimensions.

Let V be a set of voters in $\mathbb{R}^d$, with $d \geqslant 3$. Let f be any lower-dimensional flat in $\mathbb{R}^d$. Let v^f be the projection of voter v onto flat f and V^f be the set of voters obtained by projecting all the voters in V onto flat f. Assume no two voters in V are projected to a single voter in V^f. Let $L_k(v^f)$ be the set of k-lines of V^f on f containing v^f, and let $L_k^f(v)$ be all lines in $L(v)$ whose projection is in $L_k(v^f)$. We need the following lemma about the conservation of plurality point property under projection.

Lemma 5.17. *Let $V \in \mathbb{R}^d$ be a set of voters, where $d > 2$, and let f be a lower-dimensional flat in $\mathbb{R}^d$. Then*

i. $L_k(v) \subseteq L_k^f(v)$

ii. If p is a plurality point for V, then p^f is a plurality point for V^f.

Proof. Consider a line $\ell \in L(v)$ and its projection $\ell^f \in L(v^f)$ on f. If $\ell^f \notin L_k(v^f)$, then there is a hyperplane H^f in f such that the number of voters from V^f on one of the relatively open halfspaces bounded by H^f is greater than $n/2+k$. Let H be a hyperplane such that the projection of H on f is H^f. H contains ℓ and the number of voters on an open halfspace bounded by H is more than $n/2+k$ which means $\ell \notin L_k(v)$. This proves the first part of the lemma.

A point p is a plurality point for a set of voters V if and only if for any $v \in V$, all lines $\ell \in L(v)$ containing p are in $L_0(v)$. For any line $\ell \in L(v)$, by a similar argument to the previous part, $\ell^f \notin L_0(v^f)$ implies $\ell \notin L_0(v)$. This proves the second part of the lemma. □

Using this lemma we can extend Theorem 5.16 to higher dimensions.

Theorem 5.18. *Let V be a set of n voters in $\mathbb{R}^d$ where $d > 2$, and let k be a parameter with $k \leqslant n/15$. Then we can compute in $O(dk^5 \log k + dk^3 n \log n)$ expected time a minimum-size subset $W \subset V$ such that $V \setminus W$ admits a plurality point in the L_2 norm.*

Proof. We extend the algorithm for $\mathbb{R}^2$ to $\mathbb{R}^d$. The algorithm has four steps. In Steps 1 and 2 we find the set of pairs P and a set of lines passing through the voters in these pairs. In Step 3 we find the set of candidate points using these two sets. Finally in Step 4 we verify all the candidate points in order to find a plurality point (if one exists).

We modify Steps 1 and 2 for $\mathbb{R}^d$ as follows. We consider a 2-flat $f \subset \mathbb{R}^d$ and project all the voters in V onto flat f in order to get V^f. We will need certain non-degeneracy assumptions on the projections of the voters from V and on the projections of lines through pairs of voters onto f. If we take f to be a random 2-flat, then these assumptions are satisfied with probability 1. During the remainder of the algorithm, we will test for non-degeneracy at the appropriate places; see below for details. When we find that the flat f is not suitable after all, we start from scratch with a new random 2-flat f. Since the success probability is 1, we expect to succeed in 1 trial.

The first non-degeneracy assumption is that no two voters in V are mapped to a single voter in V^f. We can test this condition in $O(dn \log n)$ time. Now (assuming f passed the test) we apply the (2-dimensional version of) the algorithm on V^f, giving us a set P^f of $k+1$ pairs (v^f_{2i-1}, v^f_{2i}) and also the sets $L_k(v^f)$ for all v^f that appear in some pair in P^f.

Since the first non-degeneracy assumption is met, any voter $v^f \in V^f$ is the projection of a unique voter $v \in V$. Therefore, we can uniquely identify the set P of $k+1$ pairs (v_{2i-1}, v_{2i}) in V that are projected to P^f. However, a line $\ell \in L_k(v^f)$ or $\ell(v_{2i-1}, v_{2i})$ might still be a projection of many lines in $L^f_k(v)$. We need another non-degeneracy assumptions to get around this problem, as explained next.

Having the pairs P, the lines $\ell(v_{2i-1}, v_{2i})$ for the pairs $(v_{2i-1}, v_{2i}) \in P$, and the lines in $L^f_k(v)$ for all v appearing in these pairs, we can perform Step 3 of FIXEDPARAMETERMINCOSTPLURALITYPOINT, and find the set of all candidate points C. The only difference is that instead of $L_k(v)$ we work with $L^f_k(v)$.

Therefore, the second non-degeneracy assumption is that the number of voters on any line ℓ we work with in Step 3 of the algorithm, must be equal to the number of voters on the projected line ℓ^f. As the total number of lines to be tested is $O(k^3)$, we can test this condition in $O(dk^3 n)$ in total. (Again, we restart the algorithm if the test fails.)

Now each line in $L_k^f(v)$ uniquely corresponds to a line in $L_k(v^f)$, and $\left|L_k^f(v)\right| = \left|L_k(v^f)\right|$; moreover, each line $\ell(v_{2i-1}, v_{2i})$ has a unique projection $\ell(v_{2i-1}^f, v_{2i}^f)$. By Lemma 5.17, we have $L_k(v) \subseteq L_k^f(v)$. As each candidate point in C corresponds to a candidate point on f, the number of candidates in C is $O(k^3)$.

In Step 4, to verify a candidate point $p \in C$, we find the set $U = \{u_1, \ldots, u_{|U|}\}$ of all unbalanced lines containing p. According to Theorem 5.2, in order to have a plurality point, all unbalanced lines must be on a single 2-flat. We devise our verification algorithm based on the following two observations.

1. If a 2-flat f contains more than $k+1$ lines in U, then f is the only possible 2-flat for unbalanced lines. Indeed, any other 2-flat B' has at most one common unbalanced line with f, and to make at least $k+1$ remaining unbalanced lines in f balanced, we would have to remove more than k voters (which is not allowed).

2. For a 2-flat f, the number of lines in U that are not contained in f cannot be more than k—otherwise we would have to remove more than k voters (which is not allowed).

We define a set $\mathscr{F}$ of candidate 2-flats that together contain all unbalanced lines. We consider two cases.

Case A: $|U| \leqslant 2k+2$. In this case we simply put all 2-flats defined by any two lines in U in $\mathscr{F}$.

Case B: $|U| > 2k+2$. In this case we check all 2-flats defined by any two lines in $\{u_1, \ldots, u_{2k+3}\}$. If there is a 2-flat that contains more than $k+1$ lines in U, we put one such flat f into $\mathscr{F}$. By the first observation above, this is sufficient. Otherwise, the second observation implies that we cannot have a plurality point by removing only k voters.

Having the candidate 2-flats, we can compute the cost of the minimum-cost plurality point on each candidate 2-flat f. To this end, we first compute the price of p on f using Lemma 5.9. To get the real price of p, we add the cost of balancing the unbalanced lines that are not on f.

To prove the time bound, we note that the main difference between this algorithm and the algorithm for the 2D case is in Step 4. We can compute the set U in $O(dn \log n)$ time. If $\mathscr{F}$ contains a single candidate, the complexity of computing the price of p will be $O(dn \log n)$. Otherwise there are $O(k^2)$

candidate 2-flats. However, since each line appears in at most k 2-flats, the time complexity of the algorithm for each candidate plurality point will be $O(dk^2 \log k)$. For $O(k^3)$ candidate plurality points, the total complexity of algorithm is therefore $O(dk^5 \log k + dk^3 n \log n)$.

The correctness of Steps 1, 2, and 3 follows from Lemmas 5.13 and 5.14 and the correctness of the algorithm for the 2D case. The correctness of Step 4 follows from Lemma 5.9. □

5.2.3 The Minimum-Radius Plurality-Ball Problem

Let V be a set of n voters in $\mathbb{R}^d$. A closed ball $b(p,r)$ of radius r and centered at a point p is a *plurality ball* if for any point $q \notin b(p,r)$ the number of voters who prefer p over q is at least the number of voters who prefer q over p. Note that for any point p the ball $b(p,r)$ is a plurality ball if r is sufficiently large, and that a plurality ball with $r = 0$ is a plurality point. Below we describe an algorithm to compute a minimum-radius plurality ball for V. If all voters are collinear then any point on the median segment of V is a plurality ball of radius 0, so in the remainder we assume not all voters are collinear.

We define the *core* of a ball $b(p,r)$ as $b(p,r/2)$. Note that for any point $q \notin b(p,r)$ the locus of points that are strictly closer to q than p is an open halfspace that does not intersect the core $b(p,r/2)$, because obviously the bisector of p and q (that is, the hyperplane of all points equidistant to p and q) does not intersect $b(p,r/2)$. Hence, Proposition 5.1 can be generalized as follows.

Proposition 5.19. *A ball $b(p,r)$ is a plurality ball if and only if every open halfspace that does not intersect the core $b(p,r/2)$ contains at most $n/2$ voters.*

To check this condition we use the concept of k-set and k-level and their duality. A *k-set* of V, for some $0 \leqslant k \leqslant n-d$, is defined as a subset $V' \subset V$ of size k such that there is an open halfspace h^+ with $h^+ \cap V = V'$ and with at least d points from V on its boundary. Let V^* be the set of hyperplanes dual to the voters in V. The *k-level* in the arrangement $\mathscr{A}(V^*)$ is the set of points on the hyperplanes in V^* that have exactly k hyperplanes strictly below them (see Figure 5.3).

We call a k-dimensional feature of a d-dimensional arrangement (or some other d-dimensional structure) a *k-face*. A *$(d-1)$-face* is also called a *facet*. We associate each facet f of the k-level of $\mathscr{A}(V^*)$ to a cone in the primal space, as follows. Let $V^*(f)$ be the set of hyperplanes strictly below f, and consider the hyperplanes (in primal space) dual to the vertices of f. Then *cone*(f) is the closed cone defined by these hyperplanes that contains the k voters whose dual hyperplanes are in $V^*(f)$ plus, at its apex, the voter whose dual hyperplane contains f; see Figure 5.3. We call *cone*(f) a *k-cone*. A k-cone contains exactly $k+1$ voters including the voter at its apex, and the other $n-k-1$ voters all lie in the opposite cone.

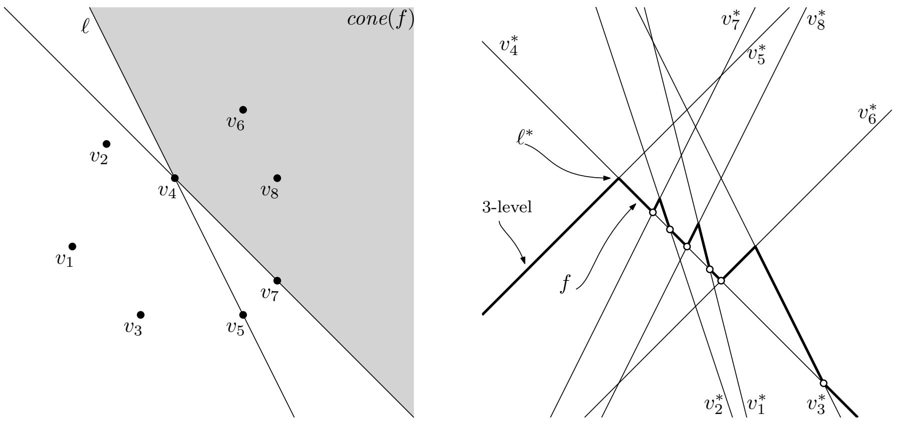


Figure 5.3: A 3-set, $\{v_6, v_7, v_8\}$, of $V = \{v_1, \ldots, v_8\}$ (left) and the 3-level in $\mathscr{A}(V^*)$ (right)

Lemma 5.20. *A ball $b(p,r)$ is a plurality ball if and only if its core $b(p,r/2)$ intersects all $\lfloor n/2 \rfloor$-cones of V.*

Proof. Assume all $\lfloor n/2 \rfloor$-cones are intersected by $b(p,r/2)$ and suppose for a contradiction that p is not a plurality point. By Proposition 5.19 there must be an open halfspace h^+ not intersecting $b(p,r/2)$ and containing more than $n/2$ voters. But then there must be a $\lfloor n/2 \rfloor$-cone contained inside the halfspace, which is a contradiction. On the other hand, if $b(p,r/2)$ does not intersect some $\lfloor n/2 \rfloor$-cone $cone(f)$ then, since any two disjoint convex sets can be separated by a hyperplane, there is an open halfspace h^+ not intersecting $b(p,r/2)$ containing all the points in $cone(f)$. Therefore $|h^+ \cap V| \geqslant \lfloor n/2 \rfloor + 1$, and so $b(p,r)$ is not a plurality ball. □

Our algorithm is now easy: We compute all $\lfloor n/2 \rfloor$-cones of V by computing the $\lfloor n/2 \rfloor$-level in the dual arrangement $\mathscr{A}(V^*)$. Then we compute the minimum-radius ball $b(p,r/2)$ intersecting all these cones, and report $b(p,r)$ as the minimum-radius plurality ball. Since in $\mathbb{R}^2$ a minimum-radius disk intersecting all the cones is computable in linear time [51] we obtain the following result.

Theorem 5.21. *Let V be a set of n voters in the plane. Then we can compute the minimum-radius plurality ball for V in $O(T(n))$ time, where $T(n)$ is the time needed to compute the $\lfloor n/2 \rfloor$-level in an arrangement of n lines in the plane.*

Computing the k-level in an arrangement of lines can be done in $O(n \log n + m \log^{1+\varepsilon} n)$ time, where m is the complexity of the level. Our algorithm then runs in $O(n^{4/3} \log^{1+\varepsilon} n)$ time. In higher dimensions, this problem fits within the generalized linear programming (GLP) framework [9]. However, as the complexity of $\lfloor n/2 \rfloor$-cones in higher dimensions is high, *violation test* and *basis* *computation* will be costly; therefore, a GLP-based algorithm would not be very efficient.

5.3 Plurality Points in the Personalized L_1 Norm

Let V be a set of n voters in $\mathbb{R}^d$, where each voter $v \in V$ has a preference vector $\langle w_1(v), \dots, w_d(v) \rangle$ of non-negative weights. Define $\text{dist}_w(v,p) := \sum_{i=1}^{d} w_i(v) \cdot |x_i(v) - x_i(p)|$. In this section we study plurality points for this personalized L_1 distance. As mentioned in the introduction, a plurality point in the L_1 norm always exists in $\mathbb{R}^2$, but not in higher dimensions [70]. Interestingly, in the personalized L_1 norm the statement already fails in the plane: in Section 5.3.3 we give an example of a weighted point set V in $\mathbb{R}^2$ that does not admit a plurality point in the personalized L_1 norm.

5.3.1 Properties of Plurality Points in the Personalized L_1 Norm

Our goal is to formulate conditions that help us to find candidate plurality points and to decide if a given candidate is actually a plurality point. For the L_2 norm we used Theorem 5.2 and Lemma 5.3 for this. Here we need a different approach. Recall that a candidate point $p \in \mathbb{R}^d$ is a plurality point if, for any point $q \in \mathbb{R}^d$, the number of voters who prefer p is at least the number of voters who prefer q. From now on we refer to the point q as a *competitor*.

For two points p and q, define $V[p \succ q] := \{v \in V : \text{dist}_w(v,p) < \text{dist}_w(v,q)\}$. We also define $V[p \sim q] := \{v \in V : \text{dist}_w(v,p) = \text{dist}_w(v,q)\}$ and $V[p \succsim q] := V[p \succ q] \cup V[p \sim q]$. Let p be a candidate plurality point. We call a point q a *non-degenerate competitor for* p if $V[p \sim q] = \emptyset$, and we say that q is *ε-close* to p if $|pq| < \varepsilon$, where $|pq|$ denotes the Euclidean distance between p and q. The following lemma implies that to test if a point p is a plurality point, we only have to consider non-degenerate competitors that are ε-close to p.

Lemma 5.22. *Let p be a candidate plurality point and let q be a competitor for p. For any $\varepsilon > 0$ there is a non-degenerate competitor q' that is ε-close to p such that $\big|V[q' \succ p]\big| \geqslant \big|V[q \succ p]\big| + \frac{1}{2} \cdot \big|V[q \sim p]\big|$.*

Proof. We prove the lemma in two steps: first we show there is an $(\varepsilon/2)$-close competitor q'' with $V[q \succ p] \subseteq V[q'' \succ p]$ and $V[q \succsim p] \subseteq V[q'' \succsim p]$, and then we show that this implies the lemma. We assume without loss of generality that p lies at the origin.

For the first step, let $\varepsilon' > 0$ be small enough so that $q'' := (\varepsilon' \cdot x_1(q), \dots, \varepsilon' \cdot x_d(q))$ is $(\varepsilon/2)$-close to p. Let v be a voter who prefers q over p, and define $b_v := b(v, \text{dist}_w(v,p))$ to be the ball (in the personalized L_1 metric of v) centered at v and with radius $\text{dist}_w(v,p)$. Then $p \in \partial b_v$ and $q \in \text{int}(b_v)$, where ∂b_v and $\text{int}(b_v)$ denote the boundary and interior of b_v, respectively. Since b_v is convex—this

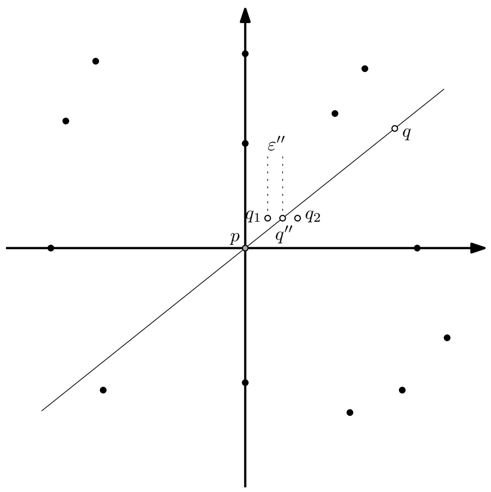


Figure 5.4: Sketch of the proof for Lemma 5.22

directly follows from the fact that balls in the standard L_1 norm are convex—we can conclude that $q'' \in \text{int}(b_v)$. Hence, $V[q \succ p] \subseteq V[q'' \succ p]$. The proof that $V[q \not\succ p] \subseteq V[q'' \not\succ p]$ is similar.

For the second step, define $B := \{b_v : v \in V[p \sim q'']\}$. Note that each ball $b_v \in B$ has both p and q'' on its boundary. Without loss of generality, we can assume $x_1(p) \neq x_1(q)$. Take two points q_1, q_2 such that $x_1(q_1) = x_1(q'') - \varepsilon''$, and $x_1(q_2) = x_1(q'') + \varepsilon''$, and $x_i(q_1) = x_i(q_2) = x_i(q'')$ for all $2 \leqslant i \leqslant d$ (see Figure 5.4). By taking $\varepsilon'' > 0$ sufficiently small we can ensure that (i) q_1 and q_2 are ε-close to p, and (ii) if $q'' \in \text{int}(b_v)$ for some ball $b_v \in B$ then $q_1, q_2 \in \text{int}(b_v)$, and if $q'' \in \text{ext}(b_v)$ then $q_1, q_2 \in \text{ext}(b_v)$ where $\text{ext}(b_v)$ is the set of points not on the boundary or interior of b_v. Now consider a voter v with $q'' \in \partial b_v$. Since q'', q_1, q_2 are ε-close to p, we can assume $x_1(v) \neq x_1(q'')$ and that either $x_1(v) < x_1(q_1) < x_1(q'') < x_1(q_2)$ or $x_1(q_1) < x_1(q'') < x_1(q_2) < x_1(v)$. This implies that either $\text{dist}_w(v, q_1) < \text{dist}_w(v, q'')$ and $\text{dist}_w(v, q_2) > \text{dist}_w(v, q'')$, or $\text{dist}_w(v, q_2) < \text{dist}_w(v, q'')$ and $\text{dist}_w(v, q_1) > \text{dist}_w(v, q'')$. But then q_1, q_2 are both non-degenerate, and each voter $v \in V[q'' \sim p]$ is either in $V[q_1 \succ p]$ or in $V[q_2 \succ p]$. Assume without loss of generality that at least half of the voters from $V[q'' \sim p]$ are in $V[q_1 \succ p]$. Then

$$\big|V[q_1 \succ p]\big| \;\geqslant\; \big|V[q'' \succ p]\big| + \frac{1}{2} \cdot \big|V[q'' \sim p]\big| \;\geqslant\; \big|V[q \succ p]\big| + \frac{1}{2} \cdot \big|V[q \sim p]\big|,$$

where the last inequality holds because $V[q \succ p] \subseteq V[q'' \succ p]$ and $V[q \not\succ p] \subseteq V[q'' \not\succ p]$. Hence, we can take $q' := q_1$. □

The following lemma helps us to narrow down our search for plurality points. Recall that a *multi-dimensional median* for V is a point $p \in \mathbb{R}^d$ such that, for all

$1 \leqslant i \leqslant d$, we have $\big|\{v \in V : x_i(v) < x_i(p)\}\big| \leqslant n/2$ and $\big|\{v \in V : x_i(v) > x_i(p)\}\big| \leqslant n/2$.

Lemma 5.23. *Let p be a plurality point for V in the personalized L_1 norm. Then p is a multi-dimensional median for V.*

Proof. Suppose for a contradiction that p is not a multi-dimensional median. Assume without loss of generality that p lies at the origin and that $\big|\{v \in V : x_1(v) > 0\}\big| > n/2$. Consider a competitor $q := (\varepsilon, 0, \dots, 0)$ for p, where ε is small enough so that no voter v has an x_1-coordinate in the range $(0, \varepsilon)$. For every voter v with $x_1(v) > 0$, we then have $|x_1(v) - x_1(q)| < |x_1(v) - x_1(p)|$ and $|x_i(v) - x_i(q)| = |x_i(v) - x_i(p)|$ for all $2 \leqslant i \leqslant d$. Hence, every such voter prefers q over p, contradicting the condition that p is a plurality point. □

The set M_V of all multi-dimensional medians for V is an axis-aligned hyperrectangle in $\mathbb{R}^d$, that is, it can be written as $M_V = I_1 \times \cdots \times I_d$, where each I_i is a closed interval that may degenerate into a single value. We call M_V the *median box* of V. Lemma 5.23 states that we only have to look at points in M_V when searching for plurality points. The next theorem implies that we only have to check which vertices of M_V are plurality points to fully classify the set of all plurality points. Let F be the set of all k-dimensional faces of M_V for $0 \leqslant k \leqslant d$, where each face $f \in F$ is considered relatively open. Note that $|F| = 3^{d'}$, where d' is the number of non-degenerate intervals defining M_V.

Theorem 5.24. *Let V be a set of voters in $\mathbb{R}^d$ and let f be a relatively open face of the median box M_V of V.*

a. *Either all points in f are plurality points in the personalized L_1 norm, or none of the points in f are.*

b. *The points in f are plurality points in the personalized L_1 norm if and only if all vertices of f are plurality points.*

Proof. Part (a) follows immediately from part (b). Next we prove part (b).

(b,⇒). Suppose $p \in f$ is a plurality point, and consider a vertex p' of f. We need to prove that p' is also a plurality point. Let $\varepsilon > 0$ be small enough so that for each $1 \leqslant i \leqslant d$ and every voter $v \in V$ with $x_i(v) \neq x_i(p)$ we have $|x_i(v) - x_i(p)| > \varepsilon$; define ε' similarly for p'. Let b and b' be the Euclidean balls of radius $\min(\varepsilon, \varepsilon')$ and centered at p and p', respectively. Since p is a plurality point we know that $|V[p \succ q]| \geqslant |V[q \succ p]|$ for all $q \in b$. Now consider an arbitrary point $q' \in b'$, and define $q := q' + (p - p')$. Note that $q \in b$ and that the relative position of q and p is the same as the relative position of q' and p'. In particular, $x_i(p) - x_i(q) = x_i(p') - x_i(q')$ for all $1 \leqslant i \leqslant d$. As shown in Lemma 5.25, this implies that $V[p \succ q] \subseteq V[p' \succ q']$ and $V[q' \succ p'] \subseteq V[q \succ p]$. Hence, $|V[p' \succ q']| \geqslant |V[p \succ q]| \geqslant |V[q \succ p]| \geqslant |V[q' \succ p']|$. Since q' is an arbitrary point in b', the point p' must be a plurality point according to Lemma 5.22.

(b,⇐). To prove this it suffices to show the following: if p is a point in the relative interior of f that is not a plurality point, then there is a vertex p' of f that is not a plurality point. To this end let q be an ε-close competitor (for a sufficiently small $\varepsilon > 0$) that beats p. We will argue that the vertex p' on the opposite side of p as compared to q'—this is defined more precisely below—is not a plurality point.

Let $M_V = I_1 \times \cdots \times I_d$, where $I_i = [\min_i, \max_i]$ (possibly with $\min_i = \max_i$). For each $1 \leqslant i \leqslant d$, we pick $x_i(p')$ as follows. If $x_i(p) = \min_i$ or $x_i(p) = \max_i$ then we set $x_i(p') := x_i(p)$. Otherwise we have $\min_i < x_i(p) < \max_i$; then, if $x_i(p) \leqslant x_i(q)$ we set $x_i(p') := \min_i$, and if $x_i(p) > x_i(q)$ we set $x_i(p') := \max_i$. Now consider the competitor q' of p' defined by $q' := q + (p' - p)$, so that q' has the same positive relative to p as q has relative to p. We claim the following: for any voter v and any $1 \leqslant i \leqslant d$ we have $|x_i(v) - x_i(p')| - |x_i(v) - x_i(q')| = |x_i(v) - x_i(p)| - |x_i(v) - x_i(q)|$. Since q beats p, this claim implies that q' beats p', thus finishing the proof. Indeed, if $x_i(p') = x_i(p)$ we have $x_i(q') = x_i(q)$ and the claim holds. Otherwise assume without loss of generality that $x_i(p') < x_i(p)$, and recall that q is ε-close to p. By taking ε sufficiently small we can thus ensure that $\min_i = x_i(p') < x_i(q') < x_i(p) < x_i(q) < \max_i$. Since there are no voters v with $\min_i < x_i(v) < \max_i$, this implies the claim.

□

The following lemma is used in the proof of Theorem 5.24.

Lemma 5.25. *$V[p \succ q] \subseteq V[p' \succ q']$ and $V[q' \succ p'] \subseteq V[q \succ p]$.*

Proof. The claim follows if we can show that for any voter v

$$\sum_{i=1}^{d} w_i(v) \cdot \Big(|x_i(v) - x_i(p)| - |x_i(v) - x_i(q)| \Big) \geqslant \sum_{i=1}^{d} w_i(v) \cdot \Big(|x_i(v) - x_i(p')| - |x_i(v) - x_i(q')| \Big).$$

(Indeed, the sign of the left-hand side is negative if and only if $v \in V[p,q]$ and positive if and only if $v \in V[q,p]$, and similarly for the right-hand side and $V[p',q']$ and $V[q',p']$.) Since all weights are non-negative this holds if for all $1 \leqslant i \leqslant d$ we have

$$|x_i(v) - x_i(p)| - |x_i(v) - x_i(q)| \geqslant |x_i(v) - x_i(p')| - |x_i(v) - x_i(q')|. \tag{5.1}$$

To prove this we consider three cases.

Case 1: $x_i(p) = x_i(p')$. Now $x_i(q) = x_i(q')$ and Inequality (5.1) holds with equality.

Case 2: $x_i(v) = x_i(p')$. Now we have

$$\begin{aligned} |x_i(v) - x_i(p)| - |x_i(v) - x_i(q)| &\geqslant -|x_i(p) - x_i(q)| \text{ (Triangle Inequality)} \\ &= -|x_i(p') - x_i(q')| \\ &= |x_i(v) - x_i(p')| - |x_i(p') - x_i(q')|, \end{aligned}$$

and so Inequality (5.1) also holds in this case.

Case 3: $x_i(p) \neq x_i(p')$ and $x_i(v) \neq x_i(p')$. Now $x_i(p)$ lies in the interior of I_i, the interval defining the i-th coordinate of M_V. Our choice of ε guarantees that either $x_i(v) < \min(x_i(p), x_i(q))$ or $x_i(v) > \max(x_i(p), x_i(q))$. Similarly, $x_i(v) < \min(x_i(p'), x_i(q'))$ or $x_i(v) > \max(x_i(p'), x_i(q'))$. (Note that $x_i(p)$ is an endpoint of I_i, and so we can also have $x_i(v) = x_i(p')$, but this was covered in Case 2.) Moreover, $x_i(v) < \min(x_i(p), x_i(p'))$ or $x_i(v) > \max(x_i(p), x_i(p'))$. Hence,

$$x_i(v) < \min\big(x_i(p), x_i(p'), x_i(q), x_i(q')\big) \quad \text{or} \quad x_i(v) > \max\big(x_i(p), x_i(p'), x_i(q), x_i(q')\big),$$

which proves Inequality (5.1) (as in Case 1 with equality) and, hence, the lemma. □

5.3.2 Finding All the Plurality Points in the Personalized L_1 Norm

Our algorithm for finding the set of all plurality points is quite simple: we compute the median box M_V in $O(dn)$ time, then we check for each vertex of M_V if it is a plurality point (as described in the proof of the theorem below), and finally we report the set of all plurality points using Theorem 5.24. The following theorem summarizes the result.

Theorem 5.26. *Let V be a set of n voters in $\mathbb{R}^d$, where $d \geqslant 2$. Then we can compute in $O(C_d n^{d-1})$ time the set of all plurality points for V in the personalized L_1 norm. When all voters have the same preferences the time bound reduces to $O(C_d + d2^d n)$. Here C_d is a constant depending on the dimension d.*

Proof. Let C'_d be a constant (depending on the dimension d) such that the arrangement defined by a set of n hyperplanes in $\mathbb{R}^d$ can be computed in $O(C'_d n^d)$ time. (The fact that an arrangement in $\mathbb{R}^d$ can be constructed in this time bound is well known [35, 36], but the constant C'_d has not been explicitly specified.) We will prove the theorem with $C_d := C'_{d-1} 2^{d^2}$.

Below we show that we can test if a given vertex p of M_V is a plurality point in $O(C'_{d-1}(2^d n)^{d-1})$ time (and in $O(C'_{d-1} 2^{d(d-1)} + dn)$ time if all voters have the same preferences), from which the theorem readily follows.

Assume without loss of generality that p lies at the origin. We need to check if for any competitor q we have $|V[p \succ q]| \geqslant |V[q \succ p]|$. By Lemma 5.22 we only have to consider non-degenerate competitors. Such a competitor q beats p if $|V[q \succ p]| > n/2$. Because p is at the origin, a voter v is in $V[q \succ p]$ if $\sum_{i=1}^{d} w_i(v) \cdot \Big(|x_i(v) - x_i(q)| - |x_i(v)|\Big) < 0$. When q is an ε-close competitor for p we have $|x_i(q)| < \varepsilon$, and then $|x_i(v) - x_i(q)| - |x_i(v)| \in \{-x_i(q), x_i(q)\}$. Hence, whether or not $v \in V[q \succ p]$ depends on the position of q relative to the hyperplane $h := \sum_{i=1}^{d} w_i(v)\alpha_i x_i = 0$, where $\alpha_i = +1$ if $x_i(q) > x_i(v)$ and $\alpha_i = -1$ if $x_i(q) < x_i(v)$. Each voter $v \in V$ thus generates a set of 2^d hyperplanes. Let

H be the total set of hyperplanes generated, that is, $H := \left\{\sum_{i=1}^{d} w_i(v)\alpha_i x_i = 0 : v \in V \text{ and } (\alpha_1,\dots,\alpha_d) \in \{-1,+1\}^d\right\}$. The discussion above implies that if two competitors q, q' have the same position relative to every hyperplane in H, then $V[q \succ p] = V[q' \succ p]$. Hence, we can proceed as follows.

We first compute the set H in $O(2^d n)$ time. Next we compute the arrangement $\mathscr{A}(H)$ defined by the hyperplanes in H. Since all hyperplanes pass through the origin, $\mathscr{A}(H)$ is effectively a $(d-1)$-dimensional arrangement, so it has complexity $O(2^d(2^d n)^{d-1})$ and it can be constructed in $O(C'_{d-1}(2^d n)^{d-1})$ time [35, 36]. Note that for any cell C of $\mathscr{A}(H)$, we have $V[q \succ p] = V[q' \succ p]$ for any two competitors q, q' in C. With a slight abuse of notation we denote this set by $V[C \succ p]$. The sets $V[C \succ p]$ and $V[C' \succ p]$ for neighboring cells C, C' differ by at most one voter (corresponding to the hyperplane that separates the cells).[2] Hence, we can compute for each cell C of $\mathscr{A}(H)$ the size of $V[C \succ p]$ in $O(2^d(2^d n)^{d-1})$ time in total, by performing a depth-first search on the dual graph of $\mathscr{A}(H)$ and updating the size as we step from one cell to the next. When we find a cell C with $|V[C \succ p]| > n/2$ we report that p is not a plurality point, otherwise we report that p is a plurality point.

When all preferences are equal—after appropriate scaling this reduces to the case where we simply use the standard L_1 norm—then all voters $v \in V$ define the same set of 2^d hyperplanes, and so $|H| = 2^d$. Hence, when all preferences are equal the algorithm for testing runs in $O(C'_{d-1}2^{d(d-1)} + dn)$ time. $\square$

5.3.3 A Plurality Point Does Not Always Exist in the Personalized L_1 Norm

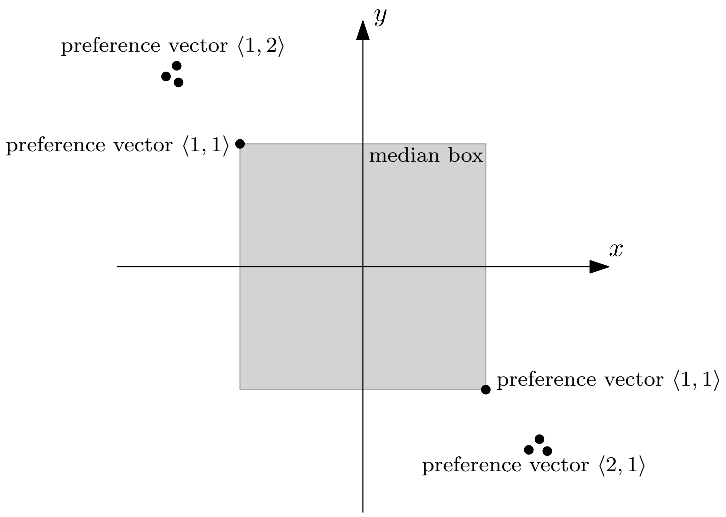
In this section we give an example showing that a plurality point doesn't always exist for the personalized L_1 norm in $\mathbb{R}^2$.

We define the set V of voters as follows (see Figure 5.5). First we add $(1,-1)$ and $(-1,1)$ to V, with preference vector $\langle 1,1\rangle$; these points will define opposite corners of the median box M_V. Then we add 6 points, as follows:

- 3 points in the $(+,-)$ quadrant of voter $(1,-1)$, with preference vector $\langle 2,1\rangle$;
- 3 points in the $(-,+)$ quadrant of voter $(-1,1)$, with preference vector $\langle 1,2\rangle$.

Now consider any point $p = (x_1(p), x_2(p))$ inside (or on the border of) the median box. It is evident that the competitor $q = (x_1(p)+\varepsilon, x_2(p)+\varepsilon)$ (for a sufficiently small $\varepsilon > 0$) falsifies that p is a plurality point as all the 6 lately

[2]Actually this is not quite true, as several voters could generate the same hyperplane. In this case the difference between $V[C \succ p]$ and $V[C' \succ p]$ can be more than one voter. Thus the time needed to step from C to C' is linear in the number of voters who generate the separating hyperplane of C and C'. It is easy to see that this does not influence the final time bound.



y
preference vector $\langle 1,2\rangle$
preference vector $\langle 1,1\rangle$
median box
x
preference vector $\langle 1,1\rangle$
preference vector $\langle 2,1\rangle$

Figure 5.5: An example where there are no plurality points in the general case

added points belong to $V[q \succ p]$. Therefore there will be no plurality point in this setting.

5.4 Concluding Remarks

We presented efficient algorithms for a number of problems concerning plurality points. It would be interesting to generalize these to the setting where the voters have weights—not to be confused with the weights defining the personal preferences—and a point is a plurality point if there is no other point that is preferred by a set of voters of higher total weight. This would also allow us to deal with multi-sets of voters, something which our current algorithms cannot do. Another direction for future research is to extend our fixed-parameter algorithm for the minimum-cost problem and the algorithm for plurality balls to the personalized L_1 norm.

Chapter 6

The Voronoi Game

In this chapter we consider *discrete (Euclidean) one-round Voronoi games*, defined as follows. Let V be a multiset of n points in $\mathbb{R}^d$, which we call *voters* from now on, and let $k \geqslant 1$ and $\ell \geqslant 1$ be two integers. The one-round discrete Voronoi game defined by the triple $\langle V, k, \ell \rangle$ is a single-turn game played between two players $\mathscr{P}$ and $\mathscr{Q}$. First, player $\mathscr{P}$ places a set P of k points in $\mathbb{R}^d$, then player $\mathscr{Q}$ places a set Q of ℓ points in $\mathbb{R}^d$. (These points may coincide with the voters in V.) We call the set P the *strategy of* $\mathscr{P}$ and the set Q the *strategy of* $\mathscr{Q}$. Player $\mathscr{P}$ wins a voter $v \in V$ if $\operatorname{dist}(v,P) \leqslant \operatorname{dist}(v,Q)$, where $\operatorname{dist}(v,P)$ and $\operatorname{dist}(v,Q)$ denote the minimum distance between a voter v and the sets P and Q, respectively. Note that this definition favors player $\mathscr{P}$, since in case of a tie a voter is won by $\mathscr{P}$. We now define $V[P \succcurlyeq Q] := \{v \in V : \operatorname{dist}(v,P) \leqslant \operatorname{dist}(v,Q)\}$ to be the multiset of voters won by player $\mathscr{P}$ when he uses strategy P and player $\mathscr{Q}$ uses strategy Q. Player $\mathscr{P}$ wins the game $\langle V, k, \ell \rangle$ if he wins at least half the voters in V, that is, when $\big|V[P \succcurlyeq Q]\big| \geqslant n/2$; otherwise $\mathscr{Q}$ wins the game. Here $\big|V[P \succcurlyeq Q]\big|$ denotes the size of the multiset $V[P \succcurlyeq Q]$ (counting multiplicities). We now define $\Gamma_{k,\ell}(V)$ as the maximum number of voters that can be won by player $\mathscr{P}$ against an optimal opponent:

$$\Gamma_{k,\ell}(V) := \max_{P \subset \mathbb{R}^d,\, |P|=k} \quad \min_{Q \subset \mathbb{R}^d,\, |Q|=\ell} \big|V[P \succcurlyeq Q]\big|.$$

For a given multiset V of voters, we want to decide if[1] $\Gamma_{k,\ell}(V) \geqslant n/2$. In other words, we are interested in determining for a given game $\langle V, k, \ell \rangle$ if $\mathscr{P}$ has a *winning strategy*, which is a set of k points such that $\mathscr{P}$ wins the game no matter where $\mathscr{Q}$ places her points.

We present an algorithm that computes $\Gamma_{k,\ell}(V)$ in $\mathbb{R}^1$ in polynomial time which is an improvement over the previous algorithm by Banik *et al.* [14] that works only for the case where $k = \ell$ and runs in exponential time when k is part

[1]One can also require that $\Gamma_{k,\ell}(V) > n/2$; with some small modifications, all the results in this chapter can be applied to the case with strict inequality as well.

of the input. Our algorithm works when V is a multiset, and it does not require k and ℓ to be equal. Our algorithm computes $\Gamma_{k,\ell}(V)$ and finds a strategy for $\mathscr{P}$ that wins this many voters in time $O(kn^4)$. The algorithm can be extended to the case when the voters are weighted, at a slight increase in running time.

After establishing that we can compute $\Gamma_{k,\ell}(V)$ in polynomial time in $\mathbb{R}^1$, we turn to the higher-dimensional problem. We show that deciding if $\mathscr{P}$ has a winning strategy is Σ_2^P-hard in $\mathbb{R}^2$. We also show that for fixed k and ℓ this problem can be solved in polynomial time. We also show that the problem is contained in the complexity class $\exists\forall\mathbb{R}$; see [33] for more information on this complexity class.

6.1 A Polynomial-Time Algorithm for $d = 1$

In this section we present a polynomial-time algorithm for the 1-dimensional discrete Voronoi game. Our algorithm will employ dynamic programming, and it will be convenient to use n, k, and ℓ as variables in the dynamic program. From now on, we therefore use n^* for the size of the original multiset V, and k^* and ℓ^* for the initial number of points that can be played by $\mathscr{P}$ and $\mathscr{Q}$, respectively.

6.1.1 Notation and Basic Properties

We denote the given multiset of voters by $V := \{v_1, \ldots, v_{n^*}\}$, where we assume the voters are numbered from left to right. We also always number the points in the strategies $P := \{p_1, \ldots, p_{k^*}\}$ and $Q := \{q_1, \ldots, q_{k^*}\}$ from left to right. For brevity we make no distinction between a point and its value (that is, its x-coordinate), so that we can for example write $p_1 < q_1$ to indicate that the leftmost point of P is located to the left of the leftmost point of player Q.

For a given game $\langle V, k, \ell \rangle$, we say that a strategy P of player $\mathscr{P}$ *realizes* a gain γ if $|V[P \succcurlyeq Q]| \geqslant \gamma$ for any strategy Q of player $\mathscr{Q}$. Furthermore, we say that a strategy P is *optimal* if it realizes $\Gamma_{k,\ell}(V)$, the maximum possible gain for $\mathscr{P}$, and we say a strategy Q is *optimal* against a given strategy P if $|V[P \succcurlyeq Q]| \leqslant |V[P \succcurlyeq Q']|$ for any strategy Q'.

Trivial, reasonable, and canonical strategies for $\mathscr{P}$. For $0 \leqslant n \leqslant n^*$, define $V_n := \{v_1, \ldots, v_n\}$ to be the leftmost n points in V. Suppose we want to compute $\Gamma_{k,\ell}(V_n)$ for some $1 \leqslant k \leqslant n$ and $0 \leqslant \ell \leqslant n$. The *trivial strategy* of player $\mathscr{P}$ is to place his points at the k points of V_n with the highest multiplicities—here we consider the multiset V_n as a set of distinct points, each with a multiplicity corresponding to the number of times it occurs in V_n—with ties broken arbitrarily. Let $\|V_n\|$ denote the number of distinct points in V_n. Then the trivial strategy is optimal when $k \geqslant \|V_n\|$ and also when $\ell \geqslant 2k$: in the former case $\mathscr{P}$ wins all voters with the trivial strategy, and in the latter case $\mathscr{Q}$ can always win all voters not coinciding with a point in P (namely by surrounding each point

$p_i \in P$ by two points sufficiently close to p_i) so the trivial strategy is optimal for $\mathscr{P}$. Hence, from now on we consider subproblems with $k < \|V_n\|$ and $\ell < 2k$.

We can without loss of generality restrict our attention to strategies for $\mathscr{P}$ that place at most one point in each half-open interval of the form $(v_i, v_{i+1}]$ with $v_i \neq v_{i+1}$, where $0 \leqslant i \leqslant n$, $v_0 := -\infty$, and $v_{n^*+1} := \infty$. Indeed, placing more than two points inside an interval $(v_i, v_{i+1}]$ is clearly not useful, and if two points are placed in some interval $(v_i, v_{i+1}]$ then we can always move the leftmost point onto v_i. (If v_i is already occupied by a point in P, then we can just put the point on any unoccupied voter; under our assumption that $k < \|V_n\|$ an unoccupied voter always exists.) We will call a strategy for $\mathscr{P}$ satisfying the property above *reasonable*.

Observation 6.1 (Banik *et al.* [14]). *Assuming $k < \|V_n\|$ there exist an optimal strategy for $\mathscr{P}$ that is reasonable and has $p_1 \in V$ (that is, p_1 coincides with a voter).*

We can define an ordering on strategies of the same size by sorting them in lexicographical order. More precisely, we say that a strategy $P = \{p_1, \dots, p_k\}$ is *greater than* a strategy $P' = \{p'_1, \dots, p'_k\}$, denoted by $P > P'$, if $\langle p_1, \dots, p_k \rangle >_{\text{lex}} \langle p'_1, \dots, p'_k \rangle$, where $>_{\text{lex}}$ denotes the lexicographical order. Using this ordering, the largest reasonable strategy P that is optimal—namely, that realizes $\Gamma_{k,\ell}(V_n)$—is called the *canonical strategy* of $\mathscr{P}$.

α-gains, β-gains, and gain sequences. Consider a strategy $P := \{p_1, \dots, p_k\}$. It will be convenient to add two extra points to P, namely $p_0 := -\infty$ and $p_{k+1} := \infty$; this clearly does not influence the outcome of the game. The strategy P thus induces $k+1$ open intervals of the form (p_i, p_{i+1}) where player $\mathscr{Q}$ may place her points. It is easy to see that there exists an optimal strategy for $\mathscr{Q}$ with the following property: Q contains at most two point in each interval (p_i, p_{i+1}) with $1 \leqslant i \leqslant k-1$, and at most one point in (p_0, p_1) and at one one point in (p_k, p_{k+1}). From now on we restrict our attention to strategies for $\mathscr{Q}$ with this property.

Now suppose that x and y are consecutive points (with $x < y$) in some strategy P, where x could be $-\infty$ and y could be ∞. As just argued, $\mathscr{Q}$ either places zero, one, or two points inside (x, y). When $\mathscr{Q}$ places zero points, then she obviously does not win any of the voters in $V_n \cap (x, y)$. The maximum number of voters $\mathscr{Q}$ can win from $V_n \cap (x, y)$ by placing a single point is the maximum number of voters in (x, y) that can be covered by an open interval of length $(y - x)/2$; see Banik *et al.* [14]. We call this value the *α-gain* of $\mathscr{Q}$ in (x, y) and denote it by $\text{gain}_\alpha(V_n, x, y)$. By placing two points inside (x, y), one immediately to the right of x and one immediately to the left of y, player $\mathscr{Q}$ will win all voters $V_n \cap (x, y)$. Thus the extra number of voters won by the second point in (x, y) as compared to just placing a single point is equal to $|V_n \cap (x, y)| - \text{gain}_\alpha(V_n, x, y)$. We call this quantity the *β-gain* of Q in (x, y) and denote it by $\text{gain}_\beta(V_n, x, y)$. Note that for intervals (x, ∞) we have $\text{gain}_\alpha(x, \infty) = |V_n \cap (x, \infty)|$ and $\text{gain}_\beta(x, \infty) = 0$; a similar statement holds for $(-\infty, y)$.

The following observation follows from the fact that $\text{gain}_\alpha(V_n, x, y)$ equals the maximum number of voters in (x, y) that can be covered by an open interval of length $(y - x)/2$.

Observation 6.2 (Banik *et al.* [14]). *For any x, y we have $\text{gain}_\alpha(V_n, x, y) \geqslant \text{gain}_\beta(V_n, x, y)$.*

If we let $a := \text{gain}_\alpha(V_n, x, y)$ and $b := \text{gain}_\beta(V_n, x, y)$, then player $\mathcal{Q}$ wins either 0, a, or $a + b$ points depending on whether she plays 0, 1, or 2 points inside then interval. It will therefore be convenient to introduce the notation $\oplus_j$ for $j \in \{0, 1, 2\}$, which is defined as

$$a \oplus_0 b := 0, \qquad a \oplus_1 b := a, \qquad a \oplus_2 b := a + b.$$

We assume the precedence of these operators are higher than addition.

Let $P := \{p_0, p_1, \ldots, p_k, p_{k+1}\}$ be a given strategy for $\mathcal{P}$, where by convention $p_0 = -\infty$ and $p_{k+1} = \infty$. Consider $\{\text{gain}_\alpha(V_n, p_i, p_{i+1}) : 0 \leqslant i \leqslant k\} \cup \{\text{gain}_\beta(V_n, p_i, p_{i+1}) : 0 \leqslant i \leqslant k\}$, the multiset of all α-gains and β-gains defined by the intervals $(p_i, p_i + 1)$. Sort this sequence in non-increasing order, using the following tie-breaking rules if two gains are equal:

- if one of the gains is for an interval (p_i, p_{i+1})—that is, the gain is either $\text{gain}_\alpha(V_n, p_i, p_{i+1})$ or $\text{gain}_\beta(V_n, p_i, p_{i+1})$—and the other gain is for an interval (p_j, p_{j+1}) with $j > i$, then the gain for (p_i, p_{i+1}) precedes the gain for (p_j, p_{j+1}).
- if both gains are for the same interval (p_i, p_{i+1}) then the α-gain precedes the β-gain.

We call the resulting sorted sequence the *gain sequence* induced by P on V_n. We denote this sequence by $\Sigma_{\text{gain}}(V_n, P)$ or, when P and V_n are clear from the context, sometimes simply by Σ_{gain}.

The canonical strategy of $\mathcal{Q}$ and sequence representations. Given the multiset V_n, a strategy P and value ℓ, player $\mathcal{Q}$ can compute an optimal strategy as follows. First she computes the gain sequence $\Sigma_{\text{gain}}(V_n, P)$ and chooses the first ℓ gains in $\Sigma_{\text{gain}}(V_n, P)$. Then for each $0 \leqslant i \leqslant k$ she proceeds as follows. When $\text{gain}_\alpha(V_n, p_i, p_{i+1})$ and $\text{gain}_\beta(V_n, p_i, p_{i+1})$ are both chosen, she places two points in (p_i, p_{i+1}) that win all voters in (p_i, p_{i+1}); when only $\text{gain}_\alpha(V_n, p_i, p_{i+1})$ is chosen, she places one point in (p_i, p_{i+1}) that win $\text{gain}_\alpha(V_n, p_i, p_{i+1})$ voters. (By Observation 6.2 and the tie-breaking rules, when $\text{gain}_\beta(V_n, p_i, p_{i+1})$ is chosen it is always the case that $\text{gain}_\alpha(V_n, p_i, p_{i+1})$ is also chosen.) The resulting optimal strategy Q is called the *canonical strategy* of $\mathcal{Q}$ with ℓ points against P on V_n.

From now one we restrict the strategies of player $\mathcal{Q}$ to canonical strategies. In an optimal strategy, player $\mathcal{Q}$ places at most two points in any interval induced by a strategy $P = \{p_0, \ldots, p_{k+1}\}$, and when we know that $\mathcal{Q}$ places a single point (and similarly when she places two points) then we also know where to place the point(s). Hence, we can represent an optimal strategy Q, for given V_n and P, by a sequence $M(V,P,Q) := \langle m_0,\dots,m_k\rangle$ where $m_i \in \{0,1,2\}$ indicates how many points $\mathscr{Q}$ plays in the interval (p_i, p_{i+1}). We call $M(V,P,Q)$ the *sequence representation* of the strategy Q against P on V_n. We denote the sequence representation of the canonical strategy of $\mathscr{Q}$ with ℓ points against P on V_n by $M(V,P,\ell)$. We have the following observation.

Observation 6.3. *The canonical strategy of $\mathscr{Q}$ with ℓ points against P is the optimal strategy Q with ℓ points against P whose sequence representation is maximal in the lexicographical order.*

6.1.2 The Subproblems for a Dynamic-Programming Solution

For clarity, in the rest of Section 6.1 we assume the multiset of voters V does not have repetitive entries, i.e we have a set of voters not a multiset. While all the results are easily extendible to multisets, dealing with them adds unnecessary complexity to the text.

Our goal is to develop a dynamic-programming algorithm to compute $\Gamma_{k^*,\ell^*}(V)$. Before we can define the subproblems on which the dynamic program is based, we need to introduce the concept of *thresholds*, which is a crucial ingredient in the subproblems.

Strict and loose thresholds. Consider an arbitrary gain sequence $\Sigma_{\text{gain}}(V_n, P) = \langle \tau_1,\dots,\tau_{2k+2}\rangle$. Recall that each τ_i is the α-gain or β-gain of some interval (p_i, p_{i+1}), and that these gains are sorted in non-increasing order. We call any integer value $\tau \in [\tau_{\ell+1}, \tau_\ell]$ an *ℓ-threshold* for $\mathscr{Q}$ induced by P on V_n, or simply a *threshold* if ℓ is clear from the context. We implicitly assume $\tau_0 := n$ so that talking about 0-threshold is also meaningful. Note that when $\tau_\ell \geqslant \tau > \tau_{\ell+1}$ then the canonical strategy for $\mathscr{Q}$ chooses all gains larger than τ and no gains smaller or equal to τ. Hence, we call τ a *strict* threshold if $\tau_\ell \geqslant \tau > \tau_{\ell+1}$. On the other hand, when $\tau = \tau_{\ell+1}$ then gains of value τ may or may not be chosen by the canonical strategy of $\mathscr{Q}$. (Note that in this case for gains of value τ to be picked, we would actually need $\tau_\ell = \tau = \tau_{\ell+1}$.) In this case we call τ a *loose* threshold.

The idea will be to guess the threshold τ in an optimal solution and then use the fact that fixing the threshold τ helps us to limit the strategies for $\mathscr{P}$ and anticipate the behavior of $\mathscr{Q}$. Let P_{opt} be the canonical strategy realizing $\Gamma_{k^*,\ell^*}(V)$. We call any ℓ^*-threshold of P_{opt} an *optimal threshold*. We devise an algorithm that gets a value τ as input and computes $\Gamma_{k^*,\ell^*}(V)$ correctly if τ is an optimal threshold, and computes a value not greater than $\Gamma_{k^*,\ell^*}(V)$, otherwise.

Clearly we only need to consider values of τ that are at most n^*. In fact, since each α-gain or β-gain in a given gain sequence corresponds to a unique subset of voters, the ℓ^*-th largest gain can be at most n^*/ℓ^*, so we only need to consider τ-values up to $\lfloor n^*/\ell^*\rfloor$. Observe that when there exists an optimal strategy that induces an ℓ^*-threshold equal to zero, then $\mathcal{Q}$ can win all voters not explicitly covered by P. In this case the trivial strategy is optimal for $\mathcal{P}$. Our global algorithm is now as follows.

1. For all thresholds $\tau \in \{1,\dots,\lfloor n^*/\ell^*\rfloor\}$, compute an upper bound on the number of voters $\mathcal{P}$ can win a strategy that has an ℓ^*-threshold τ. For the run where τ is an optimal threshold, the algorithm will return $\Gamma_{k^*,\ell^*}(V)$.
2. Compute the number of voters $\mathcal{P}$ wins in the game $\langle V,k^*,\ell^*\rangle$ by the trivial strategy.
3. Report the best of all solutions found.

The subproblems for a fixed threshold τ. From now on we consider a fixed threshold value $\tau \in \{1,\dots,\lfloor n^*/\ell^*\rfloor\}$. The subproblems in our dynamic-programming algorithm for the game $\langle V,k^*,\ell^*\rangle$ have several parameters.

- A parameter $n \in \{0,\dots,n^*\}$, specifying that the subproblem is on the voter set V_n.
- Parameters $k,\ell \in \{0,\dots,n\}$, specifying that $\mathcal{P}$ can use $k+1$ points and $\mathcal{Q}$ can use ℓ points.
- A parameter $\gamma \in \{0,\dots,n\}$, specifying the number of voters $\mathcal{P}$ must win.
- A parameter $\delta \in \{\text{strict},\text{loose}\}$, specifying the strictness of the fixed ℓ-threshold τ.

Intuitively, the subproblem specified by a tuple $\langle n,k,\ell,\gamma,\delta\rangle$ asks for a strategy P where $\mathcal{P}$ wins at least γ voters from V_n and such that P that induces an ℓ-threshold of strictness δ, against an opponent $\mathcal{Q}$ using ℓ points. Player $\mathcal{P}$ may use $k+1$ points and his objective will be to push his last point, p_{k+1} as far to the right as possible. The value of the solution to such a subproblem, which we denote by $X_{\max}(n,k,\ell,\gamma,\delta)$, will indicate how far to the right we can push p_{k+1}. Ultimately we will be interested in solutions where $\mathcal{P}$ can push p_{k^*+1} all the way to $+\infty$, which means he can actually win γ voters by placing only k^* points.

To formally define $X_{\max}(n,k,\ell,\gamma,\delta)$, we need one final piece of notation. Let $x \in \mathbb{R}\cup\{-\infty\}$ be a real value, let $n \in \{1,\dots,n^*\}$, and let a,b be integers. For convenience, define $v_{n^*+1} := \infty$. Now we define the *(a,b)-span of x to v_{n+1}*, denoted by $\text{span}(x,n,a,b)$, as

$$\text{span}(x,n,a,b) :=$$
$$\begin{cases} \text{the maximum real value } y \in (v_n, v_{n+1}] \text{ such that} \\ \text{gain}_\alpha(V,x,y) = a \text{ and } \text{gain}_\beta(V,x,y) = b & \text{if } x \neq -\infty \text{ and } y \text{ exists} \\ -\infty & \text{otherwise} \end{cases}$$

Definition 6.4. For parameters $n \in \{0,\dots,n^*\}$, and $k,\ell \in \{0,\dots,n\}$, and $\gamma \in \{0,\dots,n\}$ and $\delta \in \{\text{strict},\text{loose}\}$, we define the value $X_{\max}(n,k,\ell,\gamma,\delta)$ and what it means when a strategy P *realizes* this value, as follows.

- For $k = 0$, we call it an *elementary subproblem*, and define $X_{\max}(n,k,\ell,\gamma,\delta) = v_{n+1}$ if

  1. $\{v_{n+1}\}$ wins at least γ voters from V_n, and
  2. $\{v_{n+1}\}$ induces an ℓ-threshold τ with strictness δ on V_n.

  and we define $X_{\max}(n,k,\ell,\gamma,\delta) = -\infty$ otherwise. In the former case we say that $P := \{v_{n+1}\}$ *realizes* $X_{\max}(n,k,\ell,\gamma,\delta)$.

- For $k > 0$, we call it a *non-elementary subproblem*, and $X_{\max}(n,k,\ell,\gamma,\delta)$ is defined to be equal to the maximum real value $y \in (v_n, v_{n+1}]$ such that there exists a strategy $P := P' \cup \{y\}$ with $P' = \{p_1,\dots,p_k\}$, integer values n', a, b with $0 \leqslant n' < n$ and $0 \leqslant a, b \leqslant n$, an integer $j \in \{0,1,2\}$, and a $\delta' \in \{\text{strict}, \text{loose}\}$ satisfying the following conditions:

  1. P wins at least γ voters from V_n,
  2. P induces an ℓ-threshold τ with strictness δ on V_n,
  3. $\text{span}(p_k, n, a, b) = y$,
  4. P' realizes $X_{\max}(n', k-1, \ell - j, \gamma - n + n' + a \oplus_j b, \delta')$,
  5. Let $M(V_{n'}, P', \ell - j) := \langle m'_0, \dots, m'_k \rangle$ and $M(V_n, P, \ell) := \langle m_0, \dots, m_{k+1} \rangle$. Then $m'_i = m_i$ for all $0 \leqslant i < k$.

  When a set P satisfying the conditions exists, we say that P *realizes* $X_{\max}(n,k,\ell,\gamma,\delta)$. We define $X_{\max}(n,k,\ell,\gamma,\delta) = -\infty$ if no such P exists.

By induction we can show that if the parameters n, k, ℓ are not in a certain range, namely if one of the conditions $\ell < 2(k+1)$ or $k \leqslant \|V_n\|$ is violated, then $X_{\max}(n,k,\ell,\gamma,\delta) = -\infty$. The next lemma shows we can compute $\Gamma_{k^*,\ell^*}(V)$ from the solutions to our subproblems.

Lemma 6.5. *Let $V = \{v_1, \dots, v_{n^*}\}$ be a set of n^* voters in $\mathbb{R}^1$. Let $0 \leqslant k^* \leqslant n^*$ and $1 \leqslant \ell^* \leqslant n^*$ be two integers such that $\ell^* < 2k^*$ and $k^* < \|V\|$, and let τ be a fixed threshold. Then*

$$\Gamma_{k^*,\ell^*}(V) \geqslant \textit{the maximum value of } \gamma \textit{ with } 0 \leqslant \gamma \leqslant n^* \textit{ for which there exist a } \delta \in \{\text{loose}, \text{strict}\} \textit{ such that } X_{\max}(n^*, k^*, \ell^*, \gamma, \delta) = \infty. \tag{6.1}$$

Moreover, for an optimal threshold $\tau_{\text{opt}} > 0$, the inequality changes to equality.

Proof. It is clear that $\Gamma_{k^*,\ell^*}(V)$ is at least equal to the right-hand side in (6.1). Indeed, $X_{\max}(n^*, k^*, \ell^*, \gamma, \delta) = \infty$ implies by definition that there is a strategy P' of k^* points such that $P' \cup \{\infty\}$ wins at least γ voters against an opponent $\mathcal{Q}$ with ℓ^* points, and then P' must win γ voters as well. Next we prove the opposite direction for an optimal threshold τ_{opt}.

Let P_{opt} be the canonical strategy realizing $\Gamma_{k^*,\ell^*}(V)$. By definition, τ_{opt} is an ℓ^*-threshold induced by P_{opt} on V. Let Q_{opt} be the canonical strategy of $\mathcal{Q}$ with ℓ^* points against P_{opt} on V and let $M(V, P_{\mathrm{opt}}, \ell^*) = \langle m_0, \dots, m_{k^*}\rangle$ be its sequence representation. Define $P_k := \{p_1, \dots, p_k, p_{k+1}\}$ and $Q_k := Q_{\mathrm{opt}} \cap [-\infty, p_{k+1}]$, and let n_k be the largest index such that $v_{n_k} < p_{k+1}$. Let $\ell_k = |Q_k|$, and $\gamma_k = |V_{n_k}[P_k \succcurlyeq Q_k]|$, that is, γ_k is the number of voters from V_{n_k} won by P_k against Q_k. Note that τ_{opt} is an ℓ_k-threshold of P_k on V_{n_k}. let $\delta_k \in \{\text{strict}, \text{loose}\}$ indicate whether our fixed threshold τ_{opt} is a strict or loose threshold induced by P_k and ℓ_k on V_{n_k}, where $p_{k^*+1} := \infty$.

We will need the following claim.

Claim. Q_k is the canonical strategy of $\mathcal{Q}$ with ℓ_k points against P_k on V_{n_k}, and for all $0 \leqslant k \leqslant k^*$ we have $M(V_{n_k}, P_k, Q_k) = \langle m_0, \dots, m_k, 0\rangle$.

Proof of claim. The second part of the claim is obvious by definition of sequence representation and Q_k.

The first part of the claim can be shown by contradiction. Suppose Q'_k is the canonical strategy of $\mathcal{Q}$ with ℓ_k points against P_k on V_{n_k}, where $Q'_k \neq Q_k$. Let $M(V_{n_k}, P_k, Q'_k) = \langle m'_0, \dots, m'_k, 0\rangle$. Consider the strategy Q with sequence representation $\langle m'_0, \dots, m'_k, m_{k+1}, \dots, m_{k^*}\rangle$ of size ℓ^* against P_{opt}. We have two cases.

- Q_k is not an optimal strategy against P_k. This means Q wins more voters than Q_{opt} against P_{opt}, because winning a voter by $\mathcal{Q}$ is only dependent on the number of points $\mathcal{Q}$ places in the interval (p_i, p_{i+1}) of that voter. But this contradicts the fact that Q_{opt} is an optimal strategy against P_{opt}.
- Q_k is an optimal strategy against P_k, but Q_k is smaller in the lexicographical order than Q'_k. This mean Q is a strategy of equal gain as Q_{opt} whose sequence representation is lexicographically larger. But this contradicts the fact that Q_{opt} is a canonical strategy against P_{opt}.

$\lhd$

Let I_k denote the subproblem given by $\langle n_k, k, \ell_k, \gamma_k, \delta_k\rangle$ for $\tau = \tau_{\mathrm{opt}}$. Below we will show by induction on k that for any $0 \leqslant k \leqslant k^*$ we have

$$X_{\max}(I_k) = p_{k+1} \text{ and } X_{\max}(I_k) \text{ is realized by } P_k, \tag{6.2}$$

where we use $X_{\max}(I_k)$ as a shorthand for $X_{\max}(n_k, k, \ell_k, \gamma_k, \delta_k)$. Note that (6.2) finishes the proof of the lemma. Indeed, $p_{k^*+1} = \infty$ by definition, and $n_{k^*} = n^*$ which implies $V_{n_{k^*}} = V$. Hence, $\gamma_{k^*} = \Gamma_{k^*,\ell^*}(V)$ and $\ell_{k^*} = \ell^*$, and so there is a $\delta \in \{\text{loose}, \text{strict}\}$ such that $X_{\max}(n^*, k^*, \ell^*, \Gamma_{k^*,\ell^*}(V), \delta_{k^*}) = \infty$, thus finishing the proof. It remains to prove (6.2).

Base case: $k = 0$. By definition n_0 is such that $v_{n_0} < p_1$ and $v_{n_0+1} \geqslant p_1$. Moreover, $p_1 \in V$ by Observation 6.1. Hence, we have $v_{n_0+1} = p_1$, which establishes (6.2).

Induction step: $k > 0$. We first prove that $y := p_{k+1}$ satisfies all five conditions from Definition 6.4 for $a := \text{gain}_\alpha(V, p_k, p_{k+1})$ and $b := \text{gain}_\beta(V, p_k, p_{k+1})$, and for $j := m_k$ and $\delta' := \delta_k$. This implies that $X_{\max}(I_k) \geqslant p_{k+1}$. After that we argue there is no larger value of y satisfying all conditions, thus finishing the proof.

1. P_k wins at least γ_k voters from V_{n_k}.

   P_k wins γ_k voters against Q_k by definition. Moreover, by the Claim above, Q_k is an optimal strategy for $\mathcal{Q}$ against P_k on V_{n_k}. Hence, P_k can win γ_k voters from V_{n_k}.

2. P_k induces an ℓ-threshold τ_{opt} with strictness δ_k on V_{n_k}.

   This is true by definition.

3. $\text{span}(p_k, n_k, a, b) = p_{k+1}$.

   Obviously $\text{span}(p_k, n_k, a, b) \geqslant p_{k+1}$. Now let $y := \text{span}(p_k, n_k, a, b)$ and assume for a contradiction that $y > p_{k+1}$. We will argue that this implies that $P := \{p_1, \ldots, p_k, y, p_{k+2}, \ldots, p_{k^*}\}$ realizes $\Gamma_{k^*,\ell^*}(V)$. This gives the desired contradiction since P is then an optimal strategy that is lexicographically greater than the canonical strategy P_{opt}.

   Since the only difference between P and P_{opt} is that p_{k+1} is moved to the right to the position y, the intervals (p_i, p_{i+1}) only change for $i = k$ and for $i = k+1$. Thus the possible gains for $\mathcal{Q}$ in all intervals stay the same, except possibly in (p_k, p_{k+1}) and (p_{k+1}, p_{k+2}). Since $a = \text{gain}_\alpha(V, p_k, p_{k+1})$ and $b = \text{gain}_\beta(V, p_k, p_{k+1})$ and $y = \text{span}(p_k, n_k, a, b)$ by definition, the gains in (p_k, p_{k+1}) are the same as the gains in (p_k, y). Finally, since $(y, p_{k+2}) \subset (p_{k+1}, p_{k+2})$, it is clear that $\mathcal{Q}$ cannot win more voters in (y, p_{k+2}) against P than she can win in (p_{k+1}, p_{k+2}) against P_{opt} by playing a single point inside these intervals. Similarly, she cannot win more voters by playing two points in (y, p_{k+2}), as compared to playing two points in (p_{k+1}, p_{k+2}). Hence, P wins at least the same number of voters as P_{opt} and so P realizes $\Gamma_{k^*,\ell^*}(V)$.

4. P_{k-1} realizes $X_{\max}(n_{k-1}, k-1, \ell_k - j, \gamma_k - n_k + n_{k-1} + a \oplus_j b, \delta_{k-1})$ for $j = m_k$.

   By the induction hypothesis and since $\ell_{k-1} = \ell_k - j$ by definition, P_{k-1} realizes $X_{\max}(n_{k-1}, k-1, \ell_k - j, \gamma_{k-1}, \delta_{k-1})$. By the above Claim, Q_{k-1} is optimal against P_{k-1} on $V_{n_{k-1}}$ and Q_k is optimal against P_k on V_{n_k}. Hence, $\gamma_{k-1} = \gamma_k - n_k + n_{k-1} + a \oplus_j b$, which proves the condition.

5. Let $M(V_{n_{k-1}}, P_{k-1}, \ell_{k-1}) := \langle m'_0, \ldots, m'_k \rangle$ and $M(V_{n_k}, P_k, \ell_k) := \langle m_0, \ldots, m_{k+1} \rangle$. Then $m'_i = m_i$ for all $0 \leqslant i < k$.

   This is true by the above Claim.

We still need to show that p_{k+1} is the maximum value in $(v_n, v_{n+1}]$ satisfying the above mentioned conditions. This is obvious for $k = k^*$. For $k < k^*$, with a similar reasoning to the proof of the third condition, we can show that p_{k+1} is the maximum value in $(v_n, v_{n+1}]$ satisfying the conditions, as otherwise P_{opt} cannot be the canonical strategy realizing $\Gamma_{k^*,\ell^*}(V)$. Namely, we assume $y \in (v_n, v_{n+1}]$ with $y > p_{k+1}$ is the solution to I_k realized by $P'_k := \{p'_1, \ldots, p'_k, y\}$, and then we argue that $P := \{p'_1, \ldots, p'_k, y, p_{k+2}, \ldots, p_{k^*}\}$ is an optimal solution, thus obtaining a contradiction since P is lexicographically larger than P_{opt}. It remains to prove that P is optimal.

We show that any other strategy Q against P on V cannot win more voters than Q_{opt} wins against P_{opt}, which shows P realizes $\Gamma_{k^*,\ell^*}(V)$ and is therefore optimal. Let Q'_k be the canonical strategy of $\mathcal{Q}$ with ℓ_k points against P'_k on V_{n_k}. As P'_k wins at least γ_k voters and by definition of γ_k, P_k wins exactly γ_k voters, the number of voters Q'_k wins against P'_k cannot be more than the number of voters Q_k, which is part of Q_{opt}, wins against P_k.

Let $a := \text{gain}_\alpha(V, p_{k+1}, p_{k+2})$, and $b := \text{gain}_\beta(V, p_{k+1}, p_{k+2})$. Any strategy Q against P must select its gains from the gain sequence $\Sigma_{\text{gain}}(V, P)$. But, all the gains in $\Sigma_{\text{gain}}(V, P)$ are either the same gains chosen by Q'_k or Q_{opt} of a value at least τ_{opt}, or they have a value of at most τ_{opt} because of the threshold τ_{opt}. The only exceptions are the gains in interval (y, p_{k+2}) where by moving p_{k+1} to the right a does not increase, but b might increase. But, even in that case when b increases and its value gets bigger than τ_{opt} and it is chosen by Q, it means a is chosen by both Q_{opt} and Q. Hence, any extra voters won by Q through selecting the gain b are stolen from a which means the number of voters won by Q does not increase.

□

Remark. Usually in dynamic programming, subproblems have a clean non-recursive definition—the recursion only comes in when a recursive formula is given to compute the value of an optimal solution. Our approach is more complicated: we first present a recursive but “non-constructive” subproblem definition and later give a different recursive formula to compute the solutions to the subproblems.

6.1.3 Computing Solutions to Subproblems

The solution to an elementary subproblem follows fairly easily from the definitions, and can be computed in constant time.

Lemma 6.6. *Assuming the voter set V is given in sorted order, we can find the solution to an elementary subproblem in constant time.*

Proof. Consider an elementary subproblem $I = \langle n, 0, \ell, \gamma, \delta \rangle$. If $\ell = 0$ then $X_{\max}(I) = v_{n+1}$ if and only if the following two conditions hold: (i) $\gamma \leqslant n$ and (ii) $\tau > n$ or ($\tau = n$ and $\delta = \text{loose}$). If $\ell = 1$ then $X_{\max}(I) = v_{n+1}$ if and only if (i) $\gamma = 0$ and (ii) $\tau < n$ or ($\tau = n$ and $\delta = \text{strict}$). Otherwise $X_{\max}(I) = -\infty$. □

By definition, in order to obtain a strategy P realizing the solution to a non-elementary subproblem $I = \langle n, k, \ell, \gamma, \delta \rangle$ of size k, we need a solution to a smaller subproblem $I' = \langle n', k-1, \ell', \gamma', \delta' \rangle$ of size $k-1$ and add one point $y \in (v_n, v_{n+1}]$ to the strategy $P' = \{p_1, \dots, p_k\}$ realizing I'. Thus by adding y, we extend the solution to I' to get a solution to I. To find the "right" subproblem I', we guess some values for n', a, b, $j \in \{0,1,2\}$, and $\delta' \in \{\text{strict}, \text{loose}\}$; these values are enough to specify I'. We note that there are just a polynomial number of cases and therefore a polynomial number of values for the value $y \in (v_n, v_{n+1}]$ which we want to maximize. Namely, there are $O(n)$ choices for the values n', a, and b, three choices for j, and two choices for δ'. This makes $O(n^3)$ different cases to be considered for each subproblem I, in total. However, not all those subproblems can be extended to I. In the following definition, we list the triples (δ', j, δ) that provide all the valid combinations that guarantee the extendibility of I' to I.

Let a and b be the α-gain and β-gain of the interval (p_k, y) in a strategy $P = \{p_1, \dots, p_k, y\}$ with threshold τ. We define the following sets of triples depending on the relationship between a, b and τ:

$$\Delta(\tau, a, b) :=$$
$$\begin{cases} \{(\text{loose}, 2, \text{loose}), (\text{strict}, 2, \text{strict})\} & \text{if } a > \tau \wedge b > \tau \\ \{(\text{loose}, 1, \text{loose}), (\text{strict}, 1, \text{loose}), (\text{strict}, 2, \text{strict})\} & \text{if } a > \tau \wedge b = \tau \\ \{(\text{loose}, 1, \text{loose}), (\text{strict}, 1, \text{strict})\} & \text{if } a > \tau \wedge b < \tau \\ \{(\text{loose}, 0, \text{loose}), (\text{strict}, 0, \text{loose}), (\text{strict}, 1, \text{loose}), (\text{strict}, 2, \text{strict})\} & \text{if } a = \tau \wedge b = \tau \\ \{(\text{loose}, 0, \text{loose}), (\text{strict}, 0, \text{loose}), (\text{strict}, 1, \text{strict})\} & \text{if } a = \tau \wedge b < \tau \\ \{(\text{loose}, 0, \text{loose}), (\text{strict}, 0, \text{strict})\} & \text{if } a < \tau \wedge b < \tau. \end{cases}$$

Lemma 6.7. *Let $P' = \{p_1, \dots, p_k\}$ and $P := P' \cup \{y\}$, be two reasonable strategies on $V_{n'}$ and V_n, where $n' = \operatorname{argmax}_{1 \leqslant i \leqslant n^*} v_i < p_k$, $n = \operatorname{argmax}_{1 \leqslant i \leqslant n^*} v_i < y$, and $y \in (v_n, v_{n+1}]$. Let $a = gain_\alpha(V_n, p_k, y)$ and $b = gain_\beta(V_n, p_k, y)$, and assume $\tau > 0$ is an $(\ell - j)$-threshold of strictness δ' for $\mathcal{Q}$ induced by P' on $V_{n'}$, where $j \in \{0,1,2\}$. Then, there exists a triple $(\delta', j, \delta) \in \Delta(\tau, a, b)$ if and only if*

1. *P induces an ℓ-threshold τ with strictness δ on V_n,*
2. *Let $M(V_{n'}, P', \ell - j) := \langle m'_0, \ldots, m'_k \rangle$ and $M(V_n, P, \ell) := \langle m_0, \ldots, m_{k+1} \rangle$. Then $m'_i = m_i$ for all $0 \leqslant i < k$.*

Moreover, this triple is unique if it exists.

Proof. We prove the lemma for the case $\tau = a = b$. The proof for the other cases is similar.

Let Q' and Q be canonical strategies associated with $M(V_{n'}, P', \ell - j)$ and $M(V_n, P, \ell)$, respectively, and let $\Sigma_{\text{gain}}(V_{n'}, P') = \langle \tau'_1, \ldots, \tau'_{2k+2} \rangle$. We note that $\Sigma_{\text{gain}}(V_n, P)$ can be constructed by inserting a and b after the last τ value in $\Sigma_{\text{gain}}(V_{n'}, P')$. There are six cases,

- $\delta' = \text{loose}$ and $j = 0$.

  In this case $(\text{loose}, 0, \text{loose})$ is the only corresponding triple in $\Delta(\tau, a, b)$. As $\delta' = \text{loose}$, we have $\tau'_{\ell-j+1} = \tau$, and therefore a and b are not selected by Q. Therefore $Q' = Q$ and the strictness status also does not change. Hence both conditions are satisfied and $\delta = \text{loose}$.

- $\delta' = \text{strict}$ and $j = 0$.

  In this case $(\text{strict}, 0, \text{loose})$ is the only corresponding triple in $\Delta(\tau, a, b)$. As $\delta' = \text{strict}$, we have $\tau'_{\ell-j+1} < \tau$, and therefore a and b are not selected by Q; however, the strictness of the ℓ-threshold τ for P is different. Hence both conditions are satisfied and $\delta = \text{loose}$.

- $\delta' = \text{loose}$ and $j = 1$.

  As $\delta' = \text{loose}$, we have $\tau'_{\ell-j+1} = \tau$, and therefore a and b are not selected by Q. However, the gain $\tau'_{\ell-j+1}$ is selected by Q which violates the condition 2. We also note that there is no corresponding triple in $\Delta(\tau, a, b)$ for this case.

- $\delta' = \text{strict}$ and $j = 1$.

  In this case $(\text{strict}, 1, \text{loose})$ is the only corresponding triple in $\Delta(\tau, a, b)$. As $\delta' = \text{strict}$, we have $\tau'_{\ell-j+1} < \tau$, and therefore a is selected by Q and b is not selected, and the strictness of the ℓ-threshold τ for P is different. Hence both conditions are satisfied and $\delta = \text{loose}$.

- $\delta' = \text{strict}$ and $j = 2$.

  In this case $(\text{strict}, 2, \text{strict})$ is the only corresponding triple in $\Delta(\tau, a, b)$. As $\delta' = \text{strict}$, we have $\tau'_{\ell-j+1} < \tau$, and therefore both a and b are selected by Q, and the strictness of the ℓ-threshold τ for P is the same. Hence both conditions are satisfied and $\delta = \text{strict}$.

- $\delta' =$ loose and $j = 2$.

  As $\delta' =$ loose, we have $\tau'_{\ell-j+1} = \tau$, and therefore just a be selected by Q and b is not selected. However, the gain $\tau'_{\ell-j+1}$ is selected by Q which violates the condition 2. We also note that there is no corresponding triple in $\Delta(\tau, a, b)$ for this case.

□

Lemma 6.8. *For a non-elementary subproblem $I = \langle n, k, \ell, \gamma, \delta \rangle$, we have*

$$X_{\max}(n,k,\ell,\gamma,\delta) = \max_{0\leqslant n'<n} \max_{0\leqslant a\leqslant n} \max_{0\leqslant b\leqslant n} \max_{(\delta',j,\delta)\in\Delta(\tau,a,b)} \\ \operatorname{span}(X_{\max}(n', k-1, \ell-j, \gamma-n+n'+a\oplus_j b, \delta'), n, a, b).$$

Proof. As stated earlier, when one of the conditions $\ell < 2(k+1)$ or $k \leqslant \|V_n\|$ is violated, the left hand side evaluates to $-\infty$. When this happens, then the corresponding condition for each subproblem $X_{\max}(n', k-1, \ell-j, \gamma-n+n'+a\oplus_j b, \delta')$ is also violated, i.e. $(\ell-j) < 2((k-1)+1)$ or $(k-1) \leqslant \|V_{n'}\|$. Hence, each of these subproblems and the right hand side evaluates to $-\infty$, too. When $\ell < 2(k+1)$ or $k \leqslant \|V_n\|$ are both satisfied, we prove the lemma as follows.

Let y be the solution to I and let this solution be realized by the strategy $P := P' \cup \{y\}$ for some $P' = \{p_1, \ldots, p_k\}$. By definition, P' realizes $X_{\max}(n', k-1, \ell-j, \gamma-n+n'+a\oplus_j b, \delta')$ for some $\delta' \in \{\text{loose}, \text{strict}\}$ and $j \in \{0,1,2\}$ where $a = \text{gain}_\alpha(V_n, p_k, y)$, $b = \text{gain}_\beta(V_n, p_k, y)$, and $n' = \arg\max_i v_i < p_k$. By the second and fifth conditions of a subproblem and Lemma 6.7, we have $(\delta', j, \delta) \in \Delta(\tau, a, b)$, which shows

$$X_{\max}(n,k,\ell,\gamma,\delta) \leqslant \max_{0\leqslant n'<n} \max_{0\leqslant a\leqslant n} \max_{0\leqslant b\leqslant a} \max_{(\delta',j,\delta)\in\Delta(\tau,a,b)} \\ \operatorname{span}(X_{\max}(n', k-1, \ell-j, \gamma-n+n'+a\oplus_j b, \delta'), n, a, b).$$

Inversely, consider a value $y = \operatorname{span}(y', n, a, b) > -\infty$, where $y' = X_{\max}(n', k-1, \ell-j, \gamma-n'+n''+a\oplus_j b, \delta')$ is realized by $P' = \{p_1, \ldots, p_k\}$ for some values n',a,b,δ', and j such that $(\delta', j, \delta) \in \Delta(\tau, a, b)$. Let $P := P' \cup \{y\}$. It is not hard to verify that the conditions for y to be a solution to I realized by P are satisfied by these values. Indeed, we have:

1. P wins at least γ voters from V_n.

   By the second and the fifth conditions and the fact that $\tau > 0$, we have $M_k = j$ which shows $\mathscr{P}$ can win at least γ voters.

2. P induces an ℓ-threshold τ with strictness δ on V_n.

   This condition is satisfied by the condition 1 of Lemma 6.7 and uniqueness of δ.

3. $\text{span}(p_k, n, a, b) = y$.

   This condition is satisfied by the assumption.

4. P' realizes $X_{\max}(n', k-1, \ell - j, \gamma - n + n' + a \oplus_j b, \delta')$.

   This condition is satisfied by the assumption.

5. Let $M(V_{n'}, P', \ell - j) = \langle m'_0, \dots, m'_k \rangle$ and $M(V_n, P, \ell) = \langle m_0, \dots, m_{k+1} \rangle$. Then $m'_i = m_i$ for all $0 \leqslant i < k$.

   This condition is satisfied by the condition 2 of Lemma 6.7.

As all the conditions are satisfied, we have

$$X_{\max}(n, k, \ell, \gamma, \delta) \geqslant \max_{0 \leqslant n' < n} \max_{0 \leqslant a \leqslant n} \max_{0 \leqslant b \leqslant a} \max_{(\delta', j, \delta) \in \Delta(\tau, a, b)} \text{span}(X_{\max}(n', k-1, \ell - j, \gamma - n + n' + a \oplus_j b, \delta'), n, a, b),$$

which completes the proof. □

If we can compute the span function efficiently, we can compute all the solutions by dynamic programming and solve the problem. However, a solution based on a trivial dynamic-programming algorithm will be of running time complexity $\lfloor n^*/\ell^* \rfloor \cdot O(k^* \ell^* (n^*)^2) \cdot O((n^*)^3 f(n^*)) = O(k^* (n^*)^6 f(n^*))$ where $\lfloor n^*/\ell^* \rfloor$ is the total number of choices for the threshold, $O(k^* \ell^* (n^*)^2)$ is the number of subproblems for each threshold, and $O((n^*)^3 f(n^*))$ is the time needed to solve each subproblem where $f(n)$ is the time needed to compute the $\text{span}(x, n, a, b)$ function. This algorithm is quite slow. More importantly it is not easy to compute the span function. In the following, we introduce some new concepts to compute the span function and also get a better running time.

6.1.4 Computing the span Function Using Gain Maps

Before we give the algorithm we introduce the *gain map*, which we need to compute the span function. Consider an arbitrary strategy P of $\mathscr{P}$ on V, and recall that such a strategy induces open intervals of the form (p_i, p_{i+1}) where $\mathscr{Q}$ can place her points. We can represent any possible interval (x, y) that may arise in this manner as a point (x, y) in the plane. Thus the locus of all possible intervals is the region $R := \{(x, y) : x < y\}$. We will define two subdivisions of this region, the A-map and the B-map, and the gain map will then be the overlay of the A-map and the B-map.

The A-map is the subdivision of R into regions A^t and B^t, for $0 \leqslant t \leqslant n^*$, defined as follows:

$A^t := \{(x, y) : \text{gain}_\alpha(V, x, y) = t\}$, and

$B^t := \{(x, y) : \text{gain}_\alpha(V, x, y) + \text{gain}_\beta(V, x, y) = t\}$.

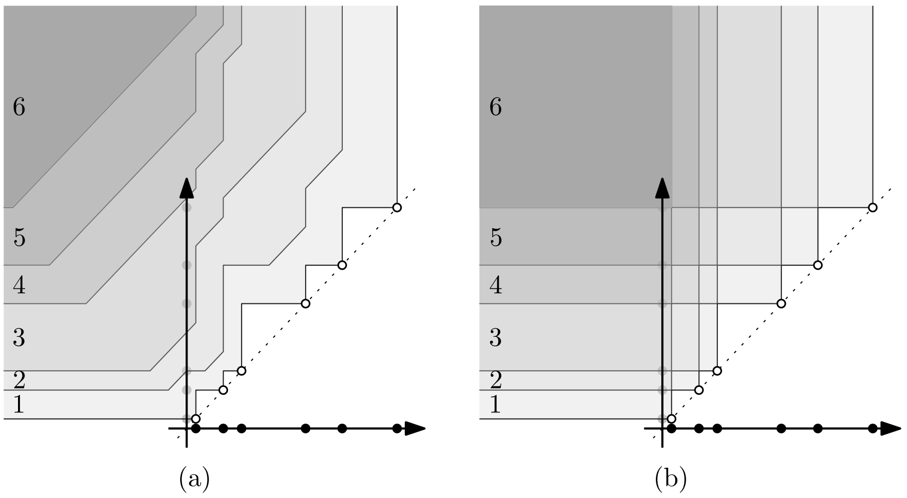


Figure 6.1: **a)** A-map of $V = \{1, 4, 6, 13, 17, 23\}$ with the corresponding α-gain for each region. **b)** B-map of V with the corresponding β-gain for each region.

In other words, A^t is the locus of all intervals (x, y) such that, if (x, y) is an interval induced by P, then $\mathcal{Q}$ can win t voters (but no more than t) from $V \cap (x, y)$ by placing a single point in (x, y). To construct the A-map, let $A^{\geqslant t}$ denote the locus of all intervals (x, y) such that $\text{gain}_\alpha(V, x, y) \geqslant t$. Note that $A^t = A^{\geqslant t} \setminus A^{\geqslant t+1}$. For $1 \leqslant i \leqslant n^* - t + 1$, let $V_i^t := \{v_i, \dots, v_{i+t-1}\}$ and define

$$A_i^{\geqslant t} := \{(x, y) : V_i^t \subset (x, y) \text{ and } \mathcal{Q} \text{ can win all voters in } V_i^t \text{ by placing a single point in } (x, y)\}.$$

Then we have $A_i^{\geqslant t} = \{(x, y) : x < v_i \text{ and } y > v_{i+t-1} \text{ and } y > x + 2(v_{i+t-1} - v_i)\}$. Here the conditions $x < v_i$ and $y > v_{i+t-1}$ are needed to guarantee that $V_i^t \subset (x, y)$. The condition $y > x + 2(v_{i+t-1} - v_i)$ implies that V_i^t can be covered with an interval of length $(y - x)/2$, which is necessary and sufficient for $\mathcal{Q}$ to be able to win all these voters. Note that each region $A_i^{\geqslant t}$ is the intersection of three halfplanes, bounded by a vertical, a horizontal and a diagonal line, respectively.

Since $\mathcal{Q}$ can win at least t voters in inside (x, y) with a single point if she can win at least t consecutive voters with a single point, we have $A^{\geqslant t} = \bigcup_{i=1}^{n^*-t+1} A_i^{\geqslant t}$. Thus $A^{\geqslant t}$ is a polygonal region, bounded from below and from the right by a a polyline consisting of horizontal, vertical, and diagonal segments, and the regions A^t are sandwiched between such polylines; see Figure 6.1a. We call the polylines that form the boundary between consecutive regions A^t *boundary polylines*.

The B-map can be constructed in a similar, but easier manner. Indeed, B^t is the locus of all intervals such that $\mathcal{Q}$ can win t voters (but no more) from $V \cap (x, y)$, and this is the case if and only if $|V \cap (x, y)| = t$. Hence, B^t is the union of the rectangular regions $[v_i, v_{i+1}) \times (v_{i+t}, v_{i+t+1}]$ (intersected with R), for $0 \leqslant i \leqslant n^* - t$, where $v_0 := -\infty$ and $v_{n^*+1} := \infty$, as shown in Figure 6.1b.

As mentioned, by overlaying the A- and B-map, we get the *gain map*. For any given region on this map, the intervals corresponding to points inside this region have equal α-gain and equal β-gain.

Lemma 6.9. *The complexity of the gain-map is* $O((n^*)^2)$.

Proof. The boundary polylines in the A-map are xy-monotone and comprised of vertical, horizontal, and diagonal lines. The B-map is essentially a grid of size $O((n^*)^2)$ defined by the lines $x = v_i$ and $y = v_i$, for $1 \leqslant i \leqslant n^*$. Since each of these lines intersects any xy-monotone polyline at most once—in a point or in a vertical segment—the complexity of the gain map is also $O((n^*)^2)$. □

Using the gain map, we can compute the values $\text{span}(x_0, n, a, b)$ for a given $x_0 \in \mathbb{R}$ and for all triples n, a, b satisfying $1 \leqslant n \leqslant n^*$, and $0 \leqslant a \leqslant n^*$ and $0 \leqslant b \leqslant n^*$, as follows. First, we compute the intersection points of the vertical line $x = x_0$ with (the edges of) the gain map, sorted by increasing y-coordinates. (If this line intersects the gain map in a vertical segment, we take the topmost endpoint of the segment.) Let $(x_0, y_1), \ldots, (x_0, y_z)$ denote this sorted sequence of intersection points, where $z \leqslant 2n^*$ denotes the number of intersections. Let a_i and b_i denote the α-gain and β-gain of the interval corresponding to the point (x_0, y_i), and let a_{z+1} and b_{z+1} denote the α-gain and β-gain of the unbounded region intersected by the line $x = x_0$. Define $n_i = \text{argmax}_n\, v_n < y_i$. Then we have

$$\text{span}(x_0, n, a, b) = \begin{cases} y_i & \text{if } a = a_i \text{ and } b = b_i \text{, and } n = n_i \text{, for some } 1 \leqslant i \leqslant z \\ +\infty & \text{if } a = a_{z+1} \text{ and } b = b_{z+1} \text{ and } n = n^* \\ -\infty & \text{for all other triples } n, a, b \end{cases} \tag{6.3}$$

Our algorithm presented below moves a sweep line from left to right over the gain map. During the sweep we maintain the intersections of the sweep line with the gain map. It will be convenient to maintain the intersections with the A-map and the B-map separately. We will do so using two sequences, $A(x_0)$ and $B(x_0)$.

- The sequence $A(x_0)$ is the sequence of all diagonal or horizontal edges in the A-map that are intersected by the line $x = x_0$, ordered from bottom to top along the line. (More precisely, the sequence contains (at most) one edge for any boundary polyline. When the sweep line reaches the endpoint of such an edge, the edge will be removed and it will be replaced by the next non-vertical edge of that boundary polyline, if it exists.)

- The sequence $B(x_0)$ is the sequence of the y-coordinates of the horizontal segments in the B-map intersected by the line $x = x_0$, ordered from bottom to top along the line.

The number of intersections of the A-map, and also of the B-map, with the line $x = x_0$ is equal to $n^* - n_0 + 1$, where $n_0 = \operatorname{argmin}_n v_n > x_0$. Hence, the sequences $A(x_0)$ and $B(x_0)$ have length $n^* - n_0 + 1 \leqslant n^* + 1$.

If we have the sequences $A(x_0)$ and $B(x_0)$ available then, using Equation (6.3), we can easily find all triples n, a, b such that $\text{span}(x_0, n, a, b) \neq -\infty$ (and the corresponding y-values) by iterating over the two sequences. We summarize the results of this section in the following observation.

Observation 6.10. *If we know the sequences $A(x_0)$ and $B(x_0)$ then we can compute all the values $\text{span}(x_0, n, a, b)$, with $1 \leqslant n \leqslant n^*$ and $0 \leqslant a, b \leqslant n$, that are not equal to $-\infty$ in $O(n^*)$ time in total.*

This observation, together with Lemma 6.8 forms the basis of our dynamic-programming algorithm.

6.1.5 The Sweep-Line Based Dynamic-Programming Algorithm

Usually in a dynamic-programming algorithm, the value of a subproblem is computed by looking up the values of certain smaller subproblems. In our case it is hard to determine which smaller subproblems we need, so we take the opposite approach: whenever we have computed the value of a subproblem we determine which other subproblems can use this value, and we update their solutions if necessary. To this end we will use a sweep-line approach, moving a vertical line from left to right over the gain map. We will maintain a table X, indexed by subproblems, such that when the sweep line is at position x_0, then $X[I]$ holds the best solution known so far for subproblem I, where the effect of all the subproblems with solution smaller than x_0 have been taken into account. When our sweep reaches a subproblem I', then we check which later subproblems I can use I' in their solution, and we update the solutions to these subproblems.

Recall the algorithm works with a fixed threshold value $\tau \in \{1, \ldots, \lfloor n^*/\ell^* \rfloor\}$ and that its goal is to compute the values $X_{\max}(n^*, k^*, \ell^*, \gamma, \delta)$ for all $0 \leqslant \gamma \leqslant n^*$ and $\delta \in \{\text{strict}, \text{loose}\}$. Our algorithm maintains the following data structures.

- $A[0..n^*]$ is an array that stores the sequence $A(x_0)$, where x_0 is the current position of the sweep line and $A[i]$ contains the i-th element in the sequence. When the i-th element does not exist then $A[i] = \text{NIL}$.
- Similarly, $B[0..n^*]$ is an array that stores the sequence $B(x_0)$.
- X: This is a table with an entry for each subproblem $I = \langle n, k, \ell, \gamma, \delta \rangle$ with $0 \leqslant n \leqslant n^*$, and $0 \leqslant k \leqslant k^*$ and $0 \leqslant \ell \leqslant \ell^*$, and $0 \leqslant \gamma \leqslant n^*$ and

$\delta \in \{\text{strict}, \text{loose}\}$. When the sweep line is at position x_0, then $X[I]$ holds the best solution known so far for subproblem I, where the effect of all the subproblems with solution smaller than x_0 have been taken into account. More precisely, in the right-hand side of the equation in Lemma 6.8 we have taken the maximum value over all subproblems $I' = \langle n', k-1, \ell - j, \gamma - n + n' + a \oplus_j b, \delta' \rangle$ with $X_{\max}(I') < x_0$. In the beginning of the algorithm the entries for elementary subproblems are computed using Lemma 6.6 and all other entries have value $-\infty$.

- E: This is the event queue, which will contain four types of events, as explained below.

The event queue E is a min-priority queue on the x-value of the events. There are four types of events, as listed next, and when events have the same x-value then the first event type (in the list below) has higher priority, that is, will be handled first. When two events of the same type have equal x-value then their order is arbitrary. Note that events with the same x-value are not degenerate cases—this is inherent to the structure of the algorithms, as many events take place at x-coordinates corresponding to voters.

A-map events, denoted by $e_A(a, s, s')$: At an A-map event, the edge s of the A-map ends—thus the x-value of an A-map event is the x-coordinate of the right endpoint of s—and the array A must be updated by replacing it with the edge s'. Here s' is the next non-vertical edge along the boundary polyline that s is part of, where $s' = \text{NIL}$ if s is the last non-vertical edge of the boundary polyline. The value a indicates that the edge s is on the boundary polyline between A^a and A^{a+1}. In other words s (and s', if it exist) are the a-th intersection point, $0 \leqslant a < n^*$, with the A-map along the current sweep line, and so we must update the entry $A[a]$ by setting $A[a] \leftarrow s'$.

B-map events, denoted by $e_B(v_n)$: At a B-map event, a horizontal edge of the B-map ends. This happens when the sweep line reaches a voter v_n—that is, when $x_0 = x_n$—and so the x-value of this event is v_n. The bottom-most intersection of he sweep line with the B-map now disappears (see Figure 6.1b), and so we must update B by shifting all other intersection points one position down in B and setting $B[n^* - n] \leftarrow \text{NIL}$.

Subproblem events, denoted by $e_X(n', k', \ell', \gamma', \delta')$: At a subproblem event the solution to the subproblem given by $I' = \langle n', k', \ell', \gamma', \delta' \rangle$ is known and the x-value of this event is equal to $X_{\max}(I')$. Handling the subproblem event for I' entails deciding which later subproblems I can use I' in their solution and how they can use it, using the sets $\Delta(\tau, a, b)$, and updating the solutions to these subproblems.

In the beginning of the algorithm all the events associated with elementary subproblems are known. The events associated with non-elementary subproblems are added to the event queue when handling an update event $e_E(v_n)$, as discussed next.

Update events, denoted by $e_E(v_n)$: At the update event happening at x-value v_n, all subproblem events of size n are added to the event queue E. These are simply the subproblems $\langle n, k, \ell, \gamma, \delta \rangle$ for all $k, \ell, \gamma \in \{0, \ldots, n\}$ and $\delta \in \{\text{strict}, \text{loose}\}$. The reason we could not add them at the start of the algorithm was that the x-value of such a subproblem I was now known yet. However, when we reach v_n then $X_{\max}(I)$ is determined, so we can add the event to E with $X_{\max}(I)$ as its x-value.

The pseudocode below summarizes the algorithm.

Algorithm 6: COMPUTESOLUTIONS(τ, V, k^*, ℓ^*)

```
for $i \leftarrow 0$ to $n^* - 1$ do
    $A[i] \leftarrow (v_i, v_{i+1}) - (v_{i+1}, v_{i+1})$;   $B[i] \leftarrow v_{i+1}$   ▷ *[r]define $v_0 := v_1 - 1$
$A[n^*] \leftarrow$ NIL;   $B[n^*] \leftarrow$ NIL
Initialize $X$ by the solutions to elementary subproblems
Initialize $E$ by all map events, update events, and elementary
  subproblem events
while $E$ is not empty do
    $e \leftarrow$ extractMin($E$);   $x_0 \leftarrow x$-value of $e$
    switch $e$ do
        case $e_A(a, s, s')$ do
            $A[a] \leftarrow s'$
        case $e_E(v_n)$ do
            $B[n^* - n] \leftarrow$ NIL
            for $i \leftarrow 0$ to $n^* - n - 1$ do
                $B[i] \leftarrow v_{n+i+1}$
        case $e_X(n', k', \ell', \gamma', \delta')$ do
            for all span($x_0, n, a, b$) = $y$ where $y \neq -\infty$ do
                for all $(\delta', j, \delta) \in \Delta(\tau, a, b)$ do
                    $I \leftarrow \langle n, k' + 1, \ell' + j, \gamma' + n - n' - \text{gain}^j(a, b), \delta \rangle$
                    $X(I) \leftarrow \max(X(I), y)$
        case $e_E(v_n)$ do
            Add all the events for subproblems of size $n$ to $E$, as
              explained above
```

Lemma 6.11. *Algorithm 6 correctly computes the solutions for subproblems $\langle n, k, \ell, \gamma, \delta \rangle$ for the given value τ, for all $n, k, \ell, \gamma, \delta$ with $0 \leqslant n \leqslant n^*$, and $0 \leqslant$*

$k, \ell, \gamma \leqslant n$, and $\delta \in$ {strict, loose}, and $\ell < 2(k+1)$. The running time of the algorithm is $O(k^* \ell^* (n^*)^3)$.

Proof. We handle the A-map and B-map events before a subproblem event so that A and B data structures are up-to-date when we want to compute the span function on handling a subproblem event. We also handle a subproblem event before an update event so that when we want to add a new subproblem event to the event queue on handling an update event, its entry in table X has the correct value. The correctness of the algorithm now follows from the discussion and lemmas above.

The running time is dominated by the handling of the subproblem events. By Observation 6.10, the algorithm handles each subproblem in $O(n^*)$ time, plus $O(\log n^*)$ for operations on the event queue, and there are $O(k^* \ell^* (n^*)^2)$ subproblems. Hence, the total running time is $O(k^* \ell^* (n^*)^3)$. □

By Lemmas 6.5 and 6.11, the algorithm described at the beginning of Section 6.1.2 computes $\Gamma_{k^*, \ell^*}(V)$ correctly. Since this algorithm calls the algorithm COMPUTESOLUTIONS $\lfloor n^*/\ell^* \rfloor$ times in Step 1, we obtain the following theorem.

Theorem 6.12. *There exists an algorithm that computes $\Gamma_{k^*, \ell^*}(V)$, and thus solves the one-dimensional case of the one-round discrete Voronoi game, in time $O(k^* (n^*)^4)$.*

Remark. We can also solve the one-dimensional case of the one-round discrete Voronoi game when voters are weighted, i.e. each voter $v \in V$ has an associated weight $\omega(v)$ and the players try to maximize the total weight of the voters they win. In this case, the α-gain and β-gain of an interval is defined as the total weight of voters the second player can win in that interval by placing one point and two points, respectively. The number of possible thresholds is not an integer in range $[0, n^*]$, but sum of any sequence of consecutive voters define a threshold, which makes a total of $O((n^*)^2)$ different thresholds. The gain map also become more complex and in the algorithm we need to spend $O((n^*)^2)$ time (instead of $O(n^*)$) to handle each subproblem event, which results in an algorithm with running time $O(k^* \ell^* (n^*)^5)$.

6.2 Σ_2^P-Hardness for $d \geqslant 2$

In this section we prove that the one-round discrete Voronoi game is Σ_2^P-hard in $\mathbb{R}^2$, which implies hardness for $d > 2$ as well. To prove this, it suffices to show that deciding if $\mathcal{Q}$ has a winning strategy against every possible strategy of $\mathcal{P}$ is Π_2^P-hard. Our proof will use a reduction from a special case of the quantified Boolean formula problem (QBF), as defined next. Let $S := \{s_1, \dots, s_{n_s}\}$ and $T := \{t_1, \dots, t_{n_t}\}$ be two sets of variables, and let $\bar{S} := \{\bar{s}_1, \dots, \bar{s}_{n_s}\}$ and $\bar{T} := \{\bar{t}_1, \dots, \bar{t}_{n_t}\}$

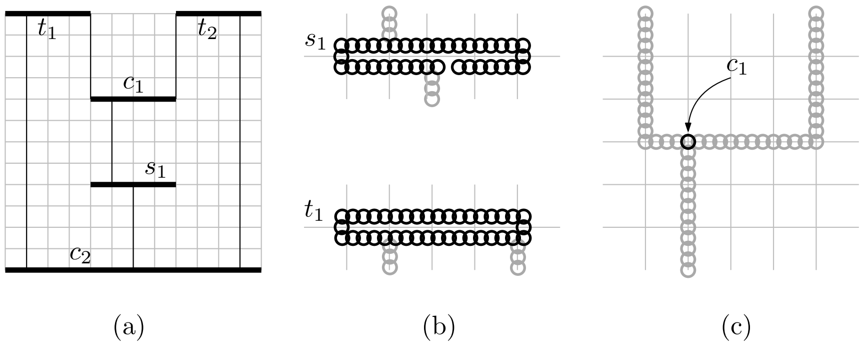


Figure 6.2: **a)** Bar graph of the boolean expression $\forall s_1 \exists t_1, t_2 : c_1 \wedge c_2$, where $c_1 := \bar{s}_1 \vee t_1 \vee t_2$ and $c_2 := s_1 \vee t_1 \vee \bar{t}_2$. **b)** Representation of variables in the transformed graph. **c)** Representation of a clause in the transformed graph.

denote their negated counterparts. We consider Boolean formulas B of the form

$$B \ := \ \forall s_1, \dots, s_{n_s} \exists t_1, \dots, t_{n_t} : c_1 \wedge \dots \wedge c_{n_c}$$

where each clause c_i in $C := \{c_1, \dots, c_{n_c}\}$ is a disjunctive combination of at most three literals from $S \cup \bar{S} \cup T \cup \bar{T}$. Deciding if a formula of this form is true is a Π_2^P-complete problem [67].

Consider the undirected graph $G_B := (N, A)$ representing B, where $N := S \cup T \cup C$ is the set of nodes of G_B and $A := \{(c_i, s_j) : s_j \in c_i \vee \bar{s}_j \in c_i\} \cup \{(c_i, t_j) : t_j \in c_i \vee \bar{t}_j \in c_i\}$ is the set of edges of G_B. Lichtenstein [55] showed how to transform an instance of QBF to an equivalent one whose corresponding graph is planar (and without increasing the size of the instance too much). We can use the same technique here. Hence, we may start our reduction from an Boolean formula B such that G_B is planar. We call the resulting problem PLANAR ∀∃3-CNF.

In the following, we transform an instance of PLANAR ∀∃3-CNF to an instance $\langle V, k, \ell \rangle$ of the Voronoi game problem in the plane such that B is true if and only if $\mathcal{Q}$ has a winning strategy.

The first (and standard) step in our reduction is to construct a specific embedding of the planar graph G_B. More precisely, we use a *bar graph*, where each node is represented by a horizontal segment and each edge is represented by a vertical segment; see Figure 6.2a. Rosenstiehl and Tarjan [63] showed that such a representation always exists. Before we describe the specific variable, clause, and edge gadgets that we use, it is useful to make the following observation about when $\mathcal{Q}$ wins a certain voter $v_i \in V$, when the strategy P of $\mathcal{P}$ is fixed. Define D_i, the *disk of* v_i with respect to the given set P, as the disk with center v_i and radius $\operatorname{dist}(v_i, P)$, that is, D_i is the largest disk centered at v_i that has no point from P in its interior. The key to our reduction is the following simple observation.

Observation 6.13. *Player $\mathscr{Q}$ wins a voter $v_i \in V$ against P if and only if she places a point q in the interior of D_i.*

Next we describe how to transform the bar graph G_B into a voter set V. Because of the above observation, it will be convenient to imagine that each voter is surrounded by a disk, and talk about placing disks. Thus, whenever we say we place a disk somewhere, in our construction we actually place a voter at the center of the disk. Later we will then put more voters that will force a certain strategy of $\mathscr{P}$ so that these disks become meaningful. The nodes of G_B are replaced by the following gadgets:

Clause gadgets. Each node $c_i \in C$ is transformed to a single disk as shown in Figure 6.2c. Recall that each such node is a horizontal segment, which has three incoming edges from its constituent literals. The position of the voter for c_i is (roughly) the point where the middle edge arrives at the segment.

Variable gadgets. Each node t_i of degree $\deg(t_i)$ in graph G_B is transformed to a ring of an even number of disks numbered in counterclockwise order starting from any arbitrary disk, which we call a *closed necklace*, containing at least $2\deg(t_i)$ disks as shown in Figure 6.2b.

Each node s_i of degree $\deg(s_i)$ in graph G_B is transformed to a ring of disks with one disk missing numbered in counterclockwise order starting from the disk after the missing disk, which we call an *open necklace*, as shown in Figure 6.2b. An open necklace has odd size and at least $2\deg(s_i)+1$ disks. We can assume the distance between the two disks in the place where the necklace is open is exactly 1. Let D_1 and D_2 be the two disks at distance 1, and let $z_1 \in \partial D_1$ and $z_2 \in \partial D_2$ be the closest pair of points on the boundaries of these two disks. (Thus $\operatorname{dist}(z_1, z_2) = 1$.) Then we place a cluster of w voters at z_1 and we place another cluster of w voters at z_2, where w is a suitable number specified later and a *cluster* of voters is simply a number of coinciding voters.

Edge gadgets. Each edge $\{c_i, s_j\}$ or $\{c_i, t_j\}$ is replaced by a chain of disks of even length that in one end intersects a pair of consecutive disks in a necklace corresponding to the node s_j or t_j, respectively, such that the first disk has an odd position and the next disk in clockwise order has an even position, and in the other end intersects the disk associated with the clause c_i

Similarly, each edge $\{c_i, \bar{s}_j\}$ or $\{c_i, \bar{t}_j\}$ is replaced by a chain of disks of even length that in one end intersects a pair of consecutive disks in a necklace corresponding to the node s_j or t_j, respectively, such that the first disk has an even position and the next disk in clockwise order has an odd position, and in the other end intersects the disk associated with the clause c_i.

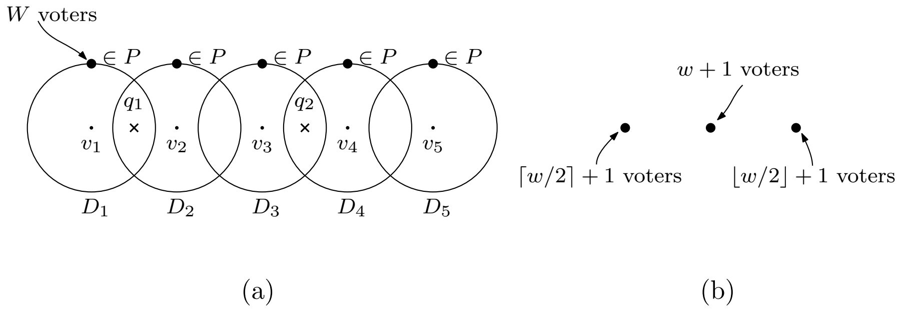


Figure 6.3: **a)** When all the heavy-weight clusters of W voters are chosen by $\mathscr{P}$, the best strategy fo $\mathscr{Q}$ to win the remaining single voters is to put his points in every other intersection of the disks of the voters. **b)** An example of a balancing gadget.

In our transformation of G_B to an instance of the Voronoi game, we will use that V can be a multiset, where we will use multiplicities 1, w, $w+1$, $\lfloor w/2 \rfloor + 1$, $\lceil w/2 \rceil + 1$, and W for the voters, for suitably chosen values w and W. We call a cluster of i voters an *i-cluster*. We denote the multisets containing clusters of $1, w$, and W voters by V_1, V_w, and V_W, respectively. The values w and W will be chosen such that they satisfy

$$w = |V_1| + 1,$$
$$W = |V_w| + |V_1| + 1.$$

We have already described the placement of voters in V_1 by describing the placement of the disks in defining the gadgets. In our construction, for each voter $v \in V_1$, we consider an imaginary disk $z(v,r)$ which is centered in the position of the voter v and has a radius r, which is a real number in range $10 \leqslant r \leqslant 20$. Each disk has a W-cluster inside it near its boundary which is denoted by $W(v)$. Two disks $z_1(v_1,r_1)$ and $z_2(v_2,r_2)$ either do not intersect or their interior have a non-empty intersection (see Figure 6.3a). Similarly when three disks intersect, their interior have a non-empty intersection. We also described the placement of voters in V_w when we defined the variable gadgets for the variables s_i.

We call this construction *the transformed graph*. Now we add $n' := |V_W| - |V_1| + 1$ gadgets, which we call *balancing gadgets*, each comprised of three clusters of sizes $\lfloor w/2 \rfloor + 1$, $\lceil w/2 \rceil + 1$, and $w+1$ far away from the graph of disks and from each other as shown in Figure 6.3b. Let V' be the multiset of all these voters.

We define $V := V_1 \cup V_w \cup V_W \cup V'$, $k := \|V_W\| + n' + n_s$, $\ell := (|V_1| - n_c - n_s)/2 + n_s + 2n'$, where $\|V_W\|$ denotes the number of distinct points in V_W.

Lemma 6.14. *$\mathscr{Q}$ has a winning strategy in $\langle V, k, \ell \rangle$ if and only if B is true.*

Proof. First we show that in an optimal strategy, $\mathscr{P}$ places his points on the most valuable positions, i.e. $\|V_W\|$ of his points on W-clusters, n' of his points on $(w+1)$-clusters, and the remaining n_s points on w-clusters where exactly one w-cluster is selected from each open necklace. Note that with such a strategy for $\mathscr{P}$, the number of remaining voters for $\mathscr{Q}$ is $n_s w + (w+2)n' + |V_1|$.

For a contradiction, assume $\mathscr{P}$ does not select some of the W-clusters. In that case $\mathscr{P}$ can either move some of his points from balancing gadgets in which he has two or more points or some of his points from the transformed graph which are not exactly on a W-cluster to the uncovered W-clusters and get a better gain; because each extra point in a balancing gadget has a gain of at most $2w+3 < W$ and all the voters of the transformed graph in V_1 and V_w together has a gain of at most $|V_w| + |V_1| < W$. Therefore, all the W-clusters will be covered by $\mathscr{P}$. Moreover, $\mathscr{P}$ must also select all the $(w+1)$ clusters; because by doing this, all the points of $\mathscr{Q}$ have a gain of at most $2w+2$ which is less that the gain of an unguarded balancing gadget, i.e. $2w+3$.

It is obvious that $\mathscr{P}$ does not place more than one point in each balancing gadget as the gain of each extra point in a balancing gadget is at most $\lceil w/2 \rceil + 1$, but the guaranteed gain of a point on a w-cluster is w. Now, we just need to show that $\mathscr{P}$ places all his remaining n_s points on w-clusters. This is also easy to see as if more that n_s w-clusters are uncovered, $\mathscr{Q}$ can easily win $(n_s+1)w + (w+2)n'$ voters which is more than the total number of remaining voters $n_s w + (w+2)n' + |V_1|$ for $\mathscr{Q}$ in the case where at most n_s w-clusters are uncovered by $\mathscr{P}$.

It is also easy to see that in an optimal strategy, $\mathscr{P}$ selects exactly one w-cluster in each open necklace. As otherwise, at least one pair of w-clusters remains unselected and $\mathscr{Q}$ can gain $2w$ voters using just one point, while if exactly one cluster from each pair of w-clusters is selected by $\mathscr{P}$, the maximum possible gain of each point of $\mathscr{Q}$ is at most $w+1$ voters.

Now, assuming $\mathscr{P}$ plays an optimal strategy, the gain of each point of $\mathscr{Q}$ is at most

- $w+1$ voters for a total of n_s points if he puts his point close to an uncovered w-cluster,
- $\lceil w/2 \rceil + 1$ or $\lfloor w/2 \rfloor + 1$ voters for a total of $2n'$ points if he puts his points in balancing clusters, and
- 2 or 3 voters if he puts his points in the intersection of disks in the transformed graph.

This shows that in her optimal strategy $\mathscr{Q}$ places all her $2n'$ points in the balancing clusters to win all the remaining voters there, and the remaining $(|V_1| - n_c - n_s)/2 + n_s$ points of $\mathscr{Q}$ will be placed on the transformed graph.

To win the remaining voters of a necklace efficiently, $\mathscr{Q}$ has two choices. She can either put her points in odd intersections or in even intersections of the necklace as shown in Figure 6.4a. In both cases the number points he spent is

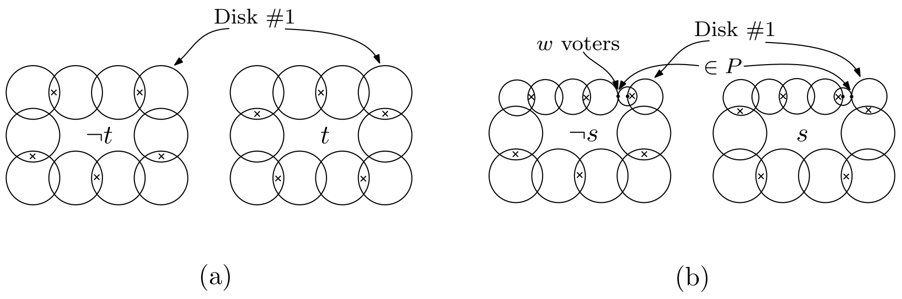


Figure 6.4: **a)** The two different ways of winning all voters for $\mathcal{Q}$. **b)** $\mathcal{P}$ can force $\mathcal{Q}$'s game.

equal to half the number of disks in the necklace. To win the remaining voters of an open necklace, $\mathcal{Q}$ has just one choice. We note that the w-cluster selected by $\mathcal{P}$ effectively changes the open necklace to a closed necklace with one more disk (see Figure 6.4b). Depending on the w-cluster chosen by $\mathcal{P}$, player $\mathcal{Q}$ must put her points in odd or even intersections as shown in Figure 6.4b. In either case $\mathcal{Q}$ spends $(m+1)/2$ points, where m is the number of disks in the open necklace. To win all the remaining voters in a chain of m disks, $\mathcal{Q}$ should place some points in every other intersection of the chain in odd positions and he spends $m/2$ points. Alternatively, she can place her $m/2$ points in even positions and also win the point in the clause on the one end of the chain only if the other end of the chain has already been covered by one of his points in the (open) necklace.

The placement of points by $\mathcal{Q}$ in odd intersections of a necklace corresponds to a true value for the associated t_j, and placement in even intersections corresponds to a false value for t_j. Similarly, selection of the starting w-cluster of an open necklace by $\mathcal{P}$ corresponds to a true value for the associated s_j, and selection of the ending w-cluster corresponds to a false value for s_j. When $\mathcal{Q}$ has a choice to put his points in even intersections of a chain and win the voter of the associated clause at the end of the chain, it corresponds to a true value for the associated literal of the (open) necklace at the other end of the chain which gives the clause a true value. Intuitively, it is clear that given an optimal strategy P, $\mathcal{Q}$ should use all his points to win the remaining voters and he can win the voter associated with each clause if and only if she can satisfy that clause.

More formally, given an optimal strategy P of $\mathcal{P}$, we set s_j to false if the starting w-cluster in clockwise order in the open necklace of the gadget for s_j is selected in P and set it to true otherwise; we call this set of values $\widehat{s}(P)$. By using the discussion in the two previous paragraphs, we can easily verify that if there exists an assignment $\widehat{t}$ for the variables $t_j (j = 1, \ldots, n_t)$ such that $\langle \widehat{s}(P), \widehat{t} \rangle$ satisfies $c_1 \wedge \cdots \wedge c_{n_c}$, then $\mathcal{Q}$ can win all the $n_s w + (w+2)n' + |V_1|$ remaining voters.

Conversely, it is not hard to see that if $\mathcal{Q}$ can win all the $n_s w + (w+2)n' + |V_1|$ remaining voters, then there exists an assignment $\hat{t}$ for t_js such that B is true.

Considering the fact that the number of remaining voters is exactly one more than the number of voters directly won by P, i.e.

$$\left(n_s w + (w+2)n' + |V_1|\right) - \left(\|V_W\| + n_s w + (w+1)n'\right) = 1,$$

$\mathcal{Q}$ has a winning strategy in $\langle V, k, \ell \rangle$ if and only if B is true. □

Lemma 6.14 is not enough to show that the Voronoi game is Σ_2^P-hard. We need to show that the reduction can be done in polynomial time and therefore the resulting Voronoi game has polynomial size.

We can easily generate balancing gadgets. Therefore we focus on drawing the transformed graph. As stated earlier, in order to define the exact position of clusters in the transformed graph we use a method devised by Rosenstiehl and Tarjan in [63] to draw a planar graph on a grid of size $O(n) \times O(n)$ where each node is represented by a horizontal line segment and each edge is represented by a vertical line segment (see Figure 6.2a). As the graph G_B is planar, we can draw it using this method. An (open) necklace associated with a variable t_j or s_j can be easily drawn as shown in Figure 6.2b. With a big enough cell-size for the grid, say 1000, we can adjust the disk sizes to get the desired properties for the intersections and also for the distance 1 between two ending disks in an open necklace. As a node associated with a clause has a degree of at most three, it can be drawn as in Figure 6.2c, and similarly with a big enough cell-size for the grid, we can adjust the size of the disks to get an even number of disks for each chain. The W-clusters can be placed at a distance of at most 0.1 at a rational point near the border of each disk, and placing w-clusters is also trivial.

As the total number of nodes in G_B is $O(n_s + n_t + n_c)$, we need a grid of size $O(n_s + n_t + n_c) \cdot O(n_s + n_t + n_c)$, and in worst case we have a constant number of disks on each edge of this grid. Therefore, the number of disk of $|V_1|$ is upperbounded by the size of the grid, i.e. $|V_1| \in O((n_s + n_t + n_c)^2)$, which means $|V| \in O(|V_1|W + n_s w + |V_1| + (2w+3)n') = O((n_s + n_t + n_c)^5)$. The following theorem is the result of this discussion.

Theorem 6.15. *The Voronoi game problem is Σ_2^P-hard for $d > 1$.*

We note that the final construction is rectilinear and therefore this reduction also works for all the L_p norms as disks in the final construction can be replaced by L_p-disks and all the required properties still hold. Moreover, as this reduction is done for the two-dimensional case, this hardness result is also true for the L_∞ norm, because in $\mathbb{R}^2$, the disks of the L_∞ norm are similar to those in the L_1 norm, just rotated by $\pi/4$.

6.3 $\exists\forall\mathbb{R}$ Containment and the Algorithm for $d \geqslant 2$

We now consider the one-round discrete Voronoi game in the L_p-norm, for some arbitrary p. Then a strategy $P = \{p_1, \dots, p_k\}$ can win a voter $v \in V$ against a strategy $Q = \{q_1, \dots, q_\ell\}$ if and only if the following Boolean expression is satisfied:

$$win(v) := \bigvee_{i \in [k]} \bigwedge_{j \in [\ell]} \big((\mathrm{dist}_p(p_i, v))^p \leqslant (\mathrm{dist}_p(q_j, v))^p\big),$$

where dist_p is the L_p-distance. This expression is of complexity $k\ell$ and degree p. The strategy P is winning if and only if majority of the expressions $win(v_1), \dots, win(v_n)$ are true. Having a majority function *Majority* that evaluates to true if at least half of its parameters evaluates to true, player $\mathscr{P}$ has a winning strategy if and only if

$$\begin{aligned}&\exists x_1(p_1), \dots, x_d(p_1), \dots, x_1(p_k), \dots, x_d(p_k)\\ &\forall x_1(q_1), \dots, x_d(q_1), \dots, x_1(q_\ell), \dots, x_d(q_\ell): \quad \mathit{Majority}(win(v_1), \dots, win(v_n))\end{aligned}$$

is true, where $x_i(\cdot)$ denotes the i-th coordinate of a point.

Ajtai *et al.* [8] show that it is possible to construct a sorting network, often called the AKS sorting network, composed of comparison units configured in $c \cdot \log n$ levels, where c is a constant and each level contains exactly $\lfloor n/2 \rfloor$ comparison units. Each comparison unit takes two numbers as input and outputs its input numbers in sorted order. Each output of a comparison unit (except those on the last level) feeds into exactly one input of a comparison unit in the next level, and the input numbers are fed to the inputs of the comparison units in the first level. The outputs of the comparison units in the last level, we call them the outputs of the network, output the input numbers in sorted order.

It is known that using AKS sorting networks we can construct a Boolean formula of size $O(n^c)$ that tests if the majority of its n inputs is true as follows. Assuming the boolean value *false* is smaller than the boolean value *true* value, we make an AKS sorting network that sorts n boolean values. This is possible using comparison units that get p and q as input, and output $p \wedge q$ and $p \vee q$. It is not hard to verify that the $\lceil n/2 \rceil$-th output of the network is equal to the value of the majority function on the input boolean values. By construction, we can write the Boolean formula representing the value of this output as *logical and* ($\wedge$) and *logical or* ($\vee$) combination of the input boolean values, and the size of the resulting formula is $O(n^c)$.

Thus we can write $\mathit{Majority}(win(v_1), \dots, win(v_n))$ as a Boolean combination of $O(n^c k\ell)$ polynomial inequalities of degree p, where each quantified block has kd and ℓd variables respectively. Basu *et al.* [18] gave an efficient algorithm for deciding the truth of quantified formulas. For our formula this gives an algorithm with $O((n^c k\ell)^{(kd+1)(\ell d+1)} p^{k\ell d^2})$ running time to decide if $\mathscr{P}$ has a winning strategy for a given instance $\langle V, k, \ell\rangle$ of the Voronoi game problem. Note that this is polynomial when k, ℓ and d are constants.

For the L_∞ norm, we can define $F(v)$ as follows

$$F(v) := \bigvee_{i\in[k]} \bigwedge_{j\in[\ell]} \bigvee_{s'\in[d]} \bigwedge_{s\in[d]} |x_s(p_i) - x_s(v)| \leqslant |x_{s'}(q_j) - x_{s'}(v)|,$$

By comparing the squared values instead of the absolute values, we have a formula which demonstrates that even with the L_∞ norm, the problem is contained in $\exists\forall\mathbb{R}$ and there exists an algorithm of complexity $O((n^c k\ell d^2)^{(kd+1)(\ell d+1)} 2^{k\ell d^2})$ to solve it.

Theorem 6.16. *The one-round discrete Voronoi game $\langle V, k, \ell\rangle$ in $\mathbb{R}^d$ with the L_p norm is contained in $\exists\forall\mathbb{R}$. Moreover, for fixed k, ℓ, d there exists an algorithm that solves it in polynomial time.*

De Berg *et al.* [29] introduced the notion of personalized preferences. More precisely, given a natural number p, assuming each axis defines an aspect of the subject voters are voting for, the voter v_i gives different weights to different axes, and v_i has a weighted L_p distance $(\sum_{j\in[d]} w_{ij}(x_j(p) - x_j(v_i))^p)^{1/p}$ from any point $p \in \mathbb{R}^d$. For the weighted L_∞ distance, v_i is at distance $\max_{j\in[d]}(w_{ij}|x_j(p) - x_j(v_i)|)$ from any point $p \in \mathbb{R}^d$. This approach also works when voters have personalized preferences.

6.4 Concluding Remarks

We presented the first polynomial-time algorithm for the one-round discrete Voronoi game in $\mathbb{R}^1$. The algorithm is quite intricate, and it would be interesting to see if a simpler (and possibly also faster) algorithm is possible. Finding a lower bound for the 1-dimensional case is another open problem.

We also showed that the problem is Σ_2^P-hard in $\mathbb{R}^2$. Fekete and Meijer [40] conjectured that finding an optimal strategy for the multi-round continuous version of the Voronoi game is PSPACE-complete. We conjecture that in the multi-round version of the discrete version, finding an optimal strategy is PSPACE-hard as well. Note that using the algebraic method presented in this chapter, it is easy to show that this problem is contained in PSPACE. While the algebraic method we used is considered a standard technique, it is, as far as we know, the first time this method is combined with polynomial-size boolean formulas for the majority function. We think it should be possible to apply this combination to other problems as well.

Chapter 7

Conclusions and Future Work

Chapters 2 and 3 are about the minimum perimeter-sum problem. In Chapter 2, we presented the first sub-quadratic algorithm for the bipartition case. The algorithm works in $O(n \log^4 n)$ time. It would be interesting to see if improving this result is possible. Finding a lower bound is another open problem. We also presented a linear-time $(1+\varepsilon)$-approximation algorithm for this case with running time $O(n+T(1/\varepsilon^2)) = O(n+1/\varepsilon^2 \cdot \log^4(1/\varepsilon))$, where $T(1/\varepsilon^2)$ is the running time of an exact algorithm on an instance of size $1/\varepsilon^2$. In Chapter 3, we presented the first polynomial-time algorithm to compute an optimal clustering (for the sum-of-perimeters as cost function) where the number of clusters is part of the input. This refutes the conjectured hardness by Arkin et al. [10]. Even though our algorithm runs in polynomial time, it is quite slow: the running time is $O(n^{27})$. One interesting question is if there exist an algorithm that solves this problem much faster. A (conditional) lower bound on the time complexity of such algorithm would also be interesting. A polynomial time algorithm already exists for the minimum radii-sum. However, the case for minimum diameter-sum is still open.

In Chapter 4, an algorithm was presented to compute an ε-coreset of size $O(k\left(f(k)/\varepsilon\right)^d)$ in linear time for clustering problems defined on a set of points in $\mathbb{R}^d$. Our method applies to a large class of clustering problems including the k-center problem in any L_p-metric, variants of the k-center problem where we want to minimize the sum (rather than maximum) of the cluster radii, and the 2-dimensional problem where we want to minimize the maximum or sum of the perimeters of the clusters. As mentioned in the introduction, Har-Peled and Mazumdar [47] also have a similar result for k-means and k-median problems. One interesting future work is to generalize our result for other cost functions.

Chapters 5 and 6 are about competitive facility-location problems. We studied plurality points in Chapter 5 and we provided the first $O(n \log n)$ time algorithm to verify if the first player can win the game. We also presented an exponential algorithm that solves the problem when the distance is measured in the L_1 norm and some generalized form of the L_1 norm we introduced which we call it the personalized L_1 norm. We also considered the problem when player $\mathscr{P}$ does not have a winning strategy. In this case, we considered two different approaches to enable player $\mathscr{P}$ win the game, that is either by removing some the voters or by restricting the location of player $\mathscr{Q}$ to be in certain fixed radius r from the location of player $\mathscr{P}$. First, we provided an algorithm that computes the minimum number of voters that needs to be ignored so that player $\mathscr{P}$ has a winning strategy that works in $O(n^4)$ time. We provide a faster algorithm for this problem when the number of voters that needs to be removed is small that works in $O(k^3 n \log n)$ time when $d = 2$ and in $O(k^5 \log k + k^3 n \log n)$ expected time when $d > 2$.

In Chapter 6, we presented the first polynomial-time algorithm for the one-round discrete Voronoi game in $\mathbb{R}^1$ that works in time $O(kn^4)$. The algorithm is quite intricate, and it would be interesting to see if a simpler (and possibly also faster) algorithm is possible. Finding a lower bound for the 1-dimensional case is another open problem.

We showed that for fixed k and ℓ this problem can be solved in polynomial time in higher dimensions. We also showed that the problem is Σ_2^P-hard in $\mathbb{R}^2$. Fekete and Meijer [40] conjectured that finding an optimal strategy for the multi-round continuous version of the Voronoi game is PSPACE-complete. We conjecture that in the multi-round version of the discrete version, finding an optimal strategy is PSPACE-hard as well. Note that using the algebraic method presented in this chapter, it is easy to show that this problem is contained in PSPACE. While the algebraic method we used is considered a standard technique, it is, as far as we know, the first time this method is combined with polynomial-size boolean formulas for the majority function. We think it should be possible to apply this combination to other problems as well.

Bibliography

[1] Mikkel Abrahamsen, Anna Adamaszek, Karl Bringmann, Vincent Cohen-Addad, Mehran Mehr, Eva Rotenberg, Alan Roytman, and Mikkel Thorup. Fast fencing. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing*, pages 564–573, 2018.

[2] Mikkel Abrahamsen, Mark de Berg, Kevin Buchin, Mehran Mehr, and Ali D. Mehrabi. A generic method for finding coresets for clustering problems. In *Abstracts of European Workshop on Computational Geometry*, pages 249–252, 2017.

[3] Mikkel Abrahamsen, Mark de Berg, Kevin Buchin, Mehran Mehr, and Ali D. Mehrabi. Minimum perimeter-sum partitions in the plane. In *Proceedings of the 33rd International Symposium on Computational Geometry*, pages 4:1–4:15, 2017.

[4] Mikkel Abrahamsen, Mark de Berg, Kevin Buchin, Mehran Mehr, and Ali D. Mehrabi. Range-clustering queries. In *Proceedings of the 33rd International Symposium on Computational Geometry*, pages 5:1–5:16, 2017.

[5] Pankaj K Agarwal, Rinat Ben Avraham, and Micha Sharir. The 2-center problem in three dimensions. *Computational Geometry*, 46(6):734–746, 2013.

[6] Pankaj K Agarwal and Micha Sharir. Efficient algorithms for geometric optimization. *ACM Computing Surveys*, 30(4):412–458, 1998.

[7] Hee-Kap Ahn, Siu-Wing Cheng, Otfried Cheong, Mordecai Golin, and Rene Van Oostrum. Competitive facility location: the voronoi game. *Theoretical Computer Science*, 310(1-3):457–467, 2004.

[8] Miklós Ajtai, János Komlós, and Endre Szemerédi. Sorting in $c \log n$ parallel steps. *Combinatorica*, 3(1):1–19, 1983.

[9] Nina Amenta. Helly-type theorems and generalized linear programming. *Discrete & Computational Geometry*, 12(3):241–261, 1994.

[10] Esther M. Arkin, Samir Khuller, and Joseph S. B. Mitchell. Geometric knapsack problems. *Algorithmica*, 10(5):399–427, 1993.

[11] Esther M Arkin, Samir Khuller, and Joseph SB Mitchell. Geometric knapsack problems. *Algorithmica*, 10(5):399–427, 1993.

[12] Tetsuo Asano, Binay Bhattacharya, Mark Keil, and Frances Yao. Clustering algorithms based on minimum and maximum spanning trees. In *Proceedings of the 4th Annual Symposium on Computational Geometry*, pages 252–257, 1988.

[13] Sang Won Bae, Hwan-Gue Cho, William Evans, Noushin Saeedi, and Chan-Su Shin. Covering points with convex sets of minimum size. *Theoretical Computer Science*, 718:14–23, 2018.

[14] Aritra Banik, Bhaswar B Bhattacharya, and Sandip Das. Optimal strategies for the one-round discrete voronoi game on a line. *Journal of Combinatorial Optimization*, 26(4):655–669, 2013.

[15] Aritra Banik, Bhaswar B Bhattacharya, Sandip Das, and Sreeja Das. Two-round discrete voronoi game along a line. In *Frontiers in Algorithmics and Algorithmic Aspects in Information and Management*, pages 210–220. Springer, 2013.

[16] Aritra Banik, Bhaswar B Bhattacharya, Sandip Das, and Satyaki Mukherjee. The discrete voronoi game in r2. *Computational Geometry*, 63:53–62, 2017.

[17] Aritra Banik, Jean-Lou De Carufel, Anil Maheshwari, and Michiel Smid. Discrete voronoi games and epsilon-nets, in two and three dimensions. *Computational Geometry*, 55:41–58, 2016.

[18] Saugata Basu, Richard Pollack, and Marie-Françoise Roy. On the combinatorial and algebraic complexity of quantifier elimination. *Journal of the ACM*, 43(6):1002–1045, 1996.

[19] Babak Behsaz and Mohammad R. Salavatipour. On minimum sum of radii and diameters clustering. *Algorithmica*, 73(1):143–165, 2015.

[20] Michael Ben-Or. Lower bounds for algebraic computation trees. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing*, pages 80–86, 1983.

[21] Peter Brass, Christian Knauer, Chan-Su Shin, Michiel Smid, and Ivo Vigan. Range-aggregate queries for geometric extent problems. In *Proceedings of*

19th Computing: The Australasian Theory Symposium-Volume 141, pages 3–10, 2013.

[22] Vasilis Capoyleas, Günter Rote, and Gerhard Woeginger. Geometric clusterings. *Journal of Algorithms*, 12(2):341–356, 1991.

[23] Timothy M Chan. More planar two-center algorithms. *Computational Geometry*, 13(3):189–198, 1999.

[24] Timothy M Chan. An optimal randomized algorithm for maximum tukey depth. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 430–436, 2004.

[25] Bernard Chazelle. On the convex layers of a planar set. *IEEE Transactions on Information Theory*, 31(4):509–517, 1985.

[26] Otfried Cheong, Sariel Har-Peled, Nathan Linial, and Jiří Matoušek. The one-round voronoi game. *Discrete & Computational Geometry*, 31(1):125–138, 2004.

[27] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

[28] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer Science & Business Media, 2008.

[29] Mark de Berg, Joachim Gudmundsson, and Mehran Mehr. Faster algorithms for computing plurality points. *ACM Transactions on Algorithms*, 14(3):36:1–36:23, 2018.

[30] Mark de Berg, Sándor Kisfaludi-Bak, and Mehran Mehr. On one-round discrete voronoi games. Manuscript, 2018.

[31] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Cheong. *Computational geometry: algorithms and applications (3rd edition)*. Springer-Verlag, 2008.

[32] Olivier Devillers and Matthew J Katz. Optimal line bipartitions of point sets. *International Journal of Computational Geometry & Applications*, 9(01):39–51, 1999.

[33] Michael G Dobbins, Linda Kleist, Tillmann Miltzow, and Paweł Rzążewski. $\forall\exists\mathbb{R}$-completeness and area-universality. In *Proceedings of the 44th International Graph-Theoretic Concepts in Computer Science (WG 2018)*, pages 164–175, 2018.

[34] Zvi Drezner. The planar two-center and two-median problems. *Transportation Science*, 18(4):351–361, 1984.

[35] Herbert Edelsbrunner. *Algorithms in combinatorial geometry*, volume 10. Springer Science & Business Media, 2012.

[36] Herbert Edelsbrunner, Joseph O'Rourke, and Raimund Seidel. Constructing arrangements of lines and hyperplanes with applications. *SIAM Journal on Computing*, 15(2):341–363, 1986.

[37] Horst A Eiselt and Gilbert Laporte. Sequential location problems. *European Journal of Operational Research*, 96(2):217–231, 1997.

[38] David Eppstein. Faster construction of planar two-centers. In *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms*, volume 97, pages 131–138, 1997.

[39] Jeff Erickson. New lower bounds for convex hull problems in odd dimensions. *SIAM Journal on Computing*, 28(4):1198–1214, 1999.

[40] Sándor P Fekete and Henk Meijer. The one-round voronoi game replayed. *Computational Geometry*, 30(2):81–94, 2005.

[41] Anka Gajentaan and Mark H Overmars. On a class of $o(n^2)$ problems in computational geometry. *Computational Geometry*, 5(3):165–185, 1995.

[42] Tibor Gallai. Solution to problem number 4065. *The American Mathematical Monthly*, 51(3):169–171, 1944.

[43] Matt Gibson, Gaurav Kanade, Erik Krohn, Imran A. Pirwani, and Kasturi Varadarajan. On clustering to minimize the sum of radii. *SIAM Journal on Computing*, 41(1):47–60, 2012.

[44] Pierre Hansen, J-F Thisse, and Richard E Wendell. Equivalence of solutions to network location problems. *Mathematics of Operations Research*, 11(4):672–678, 1986.

[45] Pierre Hansen and Jacques-Francois Thisse. Outcomes of voting and planning: Condorcet, weber and rawls locations. *Journal of Public Economics*, 16(1):1–15, 1981.

[46] Sariel Har-Peled. *Geometric approximation algorithms*, volume 173. American Mathematical Society, 2011.

[47] Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 291–300, 2004.

[48] John Hershberger. Minimizing the sum of diameters efficiently. *Computational Geometry*, 2(2):111–118, 1992.

[49] Michael Hoffmann. A simple linear algorithm for computing rectilinear 3-centers. *Computational Geometry*, 31(3):150–165, 2005.

[50] RZ Hwang, Ruei-Chuan Chang, and Richard C. T. Lee. The searching over separators strategy to solve some np-hard problems in subexponential time. *Algorithmica*, 9(4):398–423, 1993.

[51] Shreesh Jadhav, Asish Mukhopadhyay, and Binay Bhattacharya. An optimal algorithm for the intersection radius of a set of convex polygons. *Journal of Algorithms*, 20(2):244–267, 1996.

[52] Jerzy W Jaromczyk and Mirosław Kowaluk. An efficient algorithm for the euclidean two-center problem. In *Proceedings of the 10th Annual Symposium on Computational Geometry*, pages 303–311, 1994.

[53] David Kirkpatrick and Jack Snoeyink. Computing common tangents without a separating line. In *Workshop on Algorithms and Data Structures*, pages 183–193, 1995.

[54] Dominik Kress and Erwin Pesch. Sequential competitive location on networks. *European Journal of Operational Research*, 217(3):483–499, 2012.

[55] David Lichtenstein. Planar formulas and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982.

[56] Wei-Yin Lin, Yen-Wei Wu, Hung-Lung Wang, and Kun-Mao Chao. Forming plurality at minimum cost. In *Proceedings of the 9th International Workshop on Algorithms and Computation*, pages 77–88, 2015.

[57] Richard D McKelvey and Richard E Wendell. Voting equilibria in multidimensional choice spaces. *Mathematics of operations research*, 1(2):144–158, 1976.

[58] Joseph S. B. Mitchell. Private communication, 2017.

[59] Joseph SB Mitchell and Erik L Wynters. Finding optimal bipartitions of points and polygons. In *Workshop on Algorithms and Data Structures*, pages 202–213, 1991.

[60] Michael Paterson and Uri Zwick. Shallow circuits and concise formulae for multiple addition and multiplication. *Computational Complexity*, 3(3):262–291, 1993.

[61] Charles R Plott. A notion of equilibrium and its possibility under majority rule. *The American Economic Review*, 57(4):787–806, 1967.

[62] Jon Rokne, Shangzhi Wang, and Xiaolin Wu. Optimal bipartitions of point sets. In *Proceedings of the 4th Annual Canadian Conference on Computational Geometry*, page 11, 1992.

[63] Pierre Rosenstiehl and Robert E Tarjan. Rectilinear planar layouts and bipolar orientations of planar graphs. *Discrete & Computational Geometry*, 1(4):343–353, 1986.

[64] Michael Segal. Lower bounds for covering problems. *Journal of Mathematical Modelling and Algorithms*, 1(1):17–29, 2002.

[65] Micha Sharir. A near-linear algorithm for the planar 2-center problem. *Discrete & Computational Geometry*, 18(2):125–134, 1997.

[66] Joachim Spoerhase and H-C Wirth. (r, p)-centroid problems on paths and trees. *Theoretical Computer Science*, 410(47-49):5128–5137, 2009.

[67] Larry J Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1976.

[68] Sachio Teramoto, Erik D Demaine, and Ryuhei Uehara. The voronoi game on graphs and its complexity. *Journal of Graph Algorithms and Applications*, 15(4):485–501, 2011.

[69] John W Tukey. Mathematics and the picturing of data. In *Proceedings of the International Congress of Mathematicians*, volume 2, pages 523–531, 1975.

[70] Richard E Wendell and Richard D McKelvey. New perspectives in competitive location theory. *European Journal of Operational Research*, 6(2):174–182, 1981.

[71] Richard E Wendell and Stuart J Thorson. Some generalizations of social decisions under majority rule. *Econometrica*, 42(5):893–912, 1974.

[72] Yen-Wei Wu, Wei-Yin Lin, Hung-Lung Wang, and Kun-Mao Chao. Computing plurality points and condorcet points in euclidean space. In *International Symposium on Algorithms and Computation*, pages 688–698, 2013.

Summary

Faster Algorithms for Geometric Clustering and Competitive Facility-Location Problems

In this thesis we study algorithmic problems related to clustering and competitive facility location of point sets in 2- and higher-dimensional space.

In the geometric clustering problems we consider, we want to partition a given set of points in the plane into a number of clusters, so that a certain cost function is minimized. This cost function could be the sum of radii of the clusters or some other geometric measure.

One of the contributions of the thesis, which is described in Chapter 4, is an algorithm to compute an approximately optimal clustering that works for a large class of cost functions, which we call regular functions. Next we focus on a specific cost function, namely the sum of the perimeters of the clusters. We present the first sub-quadratic algorithm to compute an optimal clustering for this cost function, for the case where the number of clusters is two, described in Chapter 2. We also present an efficient approximation algorithm that is specifically optimized for this cost function and this number of clusters. Our final contribution on geometric clustering problems is the first polynomial-time algorithm to compute an optimal clustering (for the sum-of-perimeters as cost function) where the number of clusters is part of the input, which is described in Cahpter 3.

In the competitive facility-location problems we consider, we are given a set of voters, each represented by a point in a Euclidean space, two numbers k and ℓ, and two players $\mathscr{P}$ and $\mathscr{Q}$ who want to select k resp. ℓ locations in this Euclidean space in order to win as many voters as possible. More precisely, the game the two players play is as follows. First player $\mathscr{P}$ chooses k locations, then player $\mathscr{Q}$ chooses ℓ locations. Player $\mathscr{Q}$ wins a voter if he owns the closest location to the voter, and otherwise player $\mathscr{P}$ wins the voter. Player $\mathscr{P}$ wins the game if he can win at least the same number of voters as player $\mathscr{Q}$. We want to develop an algorithm to verify if player $\mathscr{P}$ has a winning strategy for the given point set.

In Chapter 6, we present the first polynomial-time algorithm for the one-

dimensional case of this problem. It was already known that this problem is NP-hard in higher dimensions when k and ℓ are part of the input. We narrow down the complexity class of the problem in higher dimensions by showing that it is Σ_2^P-hard and is contained in $\exists\forall\mathbb{R}$. The latter result also provides an explicit algorithm to solve the problem in polynomial time when k and ℓ are fixed. As a second contribution about competitive facility-location problems, we study the special case when players have exactly one point, that is where $k = \ell = 1$, with slightly modified tie-breaking rules, in Chapter 5. For this special case, we provide the first $O(n \log n)$ time algorithm to verify if the first player can win the game. We also present an exponential algorithm that solves the problem when the distance is measured in the L_1 norm.

For the special case, where $k = \ell = 1$, we also consider the problem when player $\mathscr{P}$ does not have a winning strategy. In this case, we consider two different approaches to enable player $\mathscr{P}$ win the game, that is either by removing some the voters or by restricting the location of player $\mathscr{Q}$ to be in certain fixed radius r from the location of player $\mathscr{P}$. First, we provide an algorithm that computes the minimum number of voters that needs to be ignored so that player $\mathscr{P}$ has a winning strategy that works in $O(n^4)$ time. We provide a faster algorithm for this problem when the number of voters that needs to be removed is small. Then, we present an algorithm that solves the problem with restricted locations for player $\mathscr{Q}$ efficiently.

Curriculum Vitae

Mehran Mehr was born on November 21, 1978 in Urmia, Iran. He studied Computer Engineering at Sharif University of Technology, where he graduated in July 2010. He continued his studies in the same university and he obtained his Master's degree in 2012 within the Algorithms group of Sharif University of Technology under the supervision of Mohammad Ghodsi.

Since September 2014, he has been a PhD student in the Algorithms group of Eindhoven University of technology under the supervision of Mark de Berg.

Titles in the IPA Dissertation Series since 2015

J.N. van Rijn. *Massively Collaborative Machine Learning*. Faculty of Mathematics and Natural Sciences, UL. 2016-14

M.J. Steindorfer. *Efficient Immutable Collections*. Faculty of Science, UvA. 2017-01

W. Ahmad. *Green Computing: Efficient Energy Management of Multiprocessor Streaming Applications via Model Checking*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2017-02

D. Guck. *Reliable Systems – Fault tree analysis via Markov reward automata*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2017-03

H.L. Salunkhe. *Modeling and Buffer Analysis of Real-time Streaming Radio Applications Scheduled on Heterogeneous Multiprocessors*. Faculty of Mathematics and Computer Science, TU/e. 2017-04

A. Krasnova. *Smart invaders of private matters: Privacy of communication on the Internet and in the Internet of Things (IoT)*. Faculty of Science, Mathematics and Computer Science, RU. 2017-05

A.D. Mehrabi. *Data Structures for Analyzing Geometric Data*. Faculty of Mathematics and Computer Science, TU/e. 2017-06

D. Landman. *Reverse Engineering Source Code: Empirical Studies of Limitations and Opportunities*. Faculty of Science, UvA. 2017-07

W. Lueks. *Security and Privacy via Cryptography – Having your cake and eating it too*. Faculty of Science, Mathematics and Computer Science, RU. 2017-08

A.M. Şutîi. *Modularity and Reuse of Domain-Specific Languages: an exploration with MetaMod*. Faculty of Mathematics and Computer Science, TU/e. 2017-09

U. Tikhonova. *Engineering the Dynamic Semantics of Domain Specific Languages*. Faculty of Mathematics and Computer Science, TU/e. 2017-10

Q.W. Bouts. *Geographic Graph Construction and Visualization*. Faculty of Mathematics and Computer Science, TU/e. 2017-11

A. Amighi. *Specification and Verification of Synchronisation Classes in Java: A Practical Approach*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2018-01

S. Darabi. *Verification of Program Parallelization*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2018-02

J.R. Salamanca Tellez. *Coequations and Eilenberg-type Correspondences*. Faculty of Science, Mathematics and Computer Science, RU. 2018-03

P. Fiterău-Broştean. *Active Model Learning for the Analysis of Network Protocols*. Faculty of Science, Mathematics and Computer Science, RU. 2018-04

D. Zhang. *From Concurrent State Machines to Reliable Multi-threaded Java Code*. Faculty of Mathematics and Computer Science, TU/e. 2018-05