Dilation, Transport, Visibility and Fault-Tolerant Algorithms

Dissertation

zur
Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Rainer Penninger
aus
Neuss

Bonn (2014)

Angefertigt mit der Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn

1. Gutachter: Prof. Dr. Rolf Klein, Universität Bonn
2. Gutachter: Prof. Dr. Heiko Röglin, Universität Bonn

Tag der Promotion: 07. Mai 2014

Erscheinungsjahr: 2014

Acknowledgments

First of all, I would like to thank my advisor, Prof. Rolf Klein, for introducing me to the field of computational geometry and giving me the opportunity to write this thesis. I am thankful to him and my other co-authors and colleagues. I want to mention Christian Sohler, Elmar Langetepe, David Woodruff, Alexander Gilbers, and Florian Berger. I also want to thank the other members of the committee, especially Prof. Heiko Röglin who agreed to be the co-referee of this thesis. Additionally, I would like to acknowledge with thanks Günter Rote for very helpful comments and Ansgar Grüne for his support. Also I would like to thank Antje Bertram and Mariele Knepper for assistance with administrative tasks. Many thanks go to my beloved wife Simone Lehmann, who helped me in so many ways. Last but not least my warmest thanks go to my parents, Brigitte and Stefan Penninger, for their constant encouragement and continuous support.

Abstract

Connecting some points in the plane by a road network is equivalent to constructing a *finite* planar graph G whose vertex set contains a predefined set of vertices (i. e., the possible destinations in the road network). The *dilation* between two vertices p and q of graph G is defined as the Euclidean length of a shortest path in G from p to q, divided by the Euclidean distance from p to q. That is, given a point set P, the goal is to place some additional *crossing vertices* C such that there exists a planar graph $G = (P \cup C, E)$ whose dilation is small. Here, the dilation of G is defined as the maximum dilation between two vertices in G. We show that, except for some special point sets P, there is a lower bound $\Delta(P) > 1$, depending on P, on the dilation of any finite graph containing P in its vertex set.

The transportation problem is the problem of finding a transportation plan that minimizes the total transport cost. We are given a set of suppliers, and each supplier produces a fixed amount of some commodity, say, bread. Furthermore, there is a set of customers, and each customer has some demand of bread, such that the total demand equals the amount of bread the suppliers produce. The task is to assign each unit of bread produced to some customer, such that the total transportation cost becomes a minimum. A first idea is to assign each unit of bread to the client to which the transport cost of this unit is minimal. Clearly, this gives rise to a transportation plan which minimizes the total transportation cost. However, it is likely that not every customer will obtain the required amount of bread. Therefore, we need to use a different algorithm for distributing the supplier's bread. We show that if the bread produced by the suppliers is given by a continuous probability density function and the set of customers is discrete, then every optimal transport plan can be characterized by a unique additively weighted Voronoi diagram for the customers.

When managing the construction process of a building by a digital model of the building, it is necessary to compute *essential* parts between walls of the building. Given two walls A and B, the essential part between A and B is the set of line segments s where one endpoint belongs to A, the other endpoint belongs to B, and s does not intersect A or B. We give an algorithm that computes, in linear time, the essential parts between A and B. Our algorithm is based on computing the visibility polygon of A and of B, and two shortest paths connecting points of A with points of B.

We conclude the thesis by giving fault-tolerant algorithms for some fundamental geometric problems. We assume that a basic primitive operation used by an algorithm fails with some small probability p. Depending on the results of the primitive operations, it is possible that the algorithm will not work correctly. For example, one faulty comparison when executing a sorting algorithm can result in some numbers being placed far away from their true positions. An algorithm is called *tolerant*, if with high probability a good answer is given, if the error probability p is small. We provide tolerant algorithms that find the maximum of n numbers, search for a key in a sorted sequence of n keys, sort a set of n numbers, and solve Linear Programming in $\mathbb{R}^2$.

Contents

Chapter 1

Introduction

Within the last few years, it has become more and more common to use navigation systems for planning trips, or even for every day navigation. This development is a consequence of the fact that navigation is an important problem for many people and companies, and also of the fact that using data services on mobile devices becomes more common. Since today's road networks are very large, recent attention has been turned to efficient computation of shortest paths in large road networks; see e. g. the Ninth DIMACS Implementation Challenge [15] for a variety of results. Perhaps the most outstanding result presented in [15] is an improvement in runtime by a factor of about $4 \cdot 10^6$ for shortest path queries, compared to using the classic Dijkstra algorithm [16].

Clearly, navigation requires good networks. It is said that the construction of a network of high-quality roads was one of the main strategic advantages of the Roman Empire. For example, their road network allowed for sending messages or soldiers quickly from one place to another. When constructing road networks, the following natural question occurs: how to design a road network such that *good connections* exist between each pair of destinations? This question is, for instance, the subject of a line of research called spanner construction. A wide range of algorithms is available for constructing spanners that have some desired properties, e. g., low dilation, linear size, bounded degree or small weight; see the book by Narasimhan and Smid [55].

In Chapter 2, we address a problem related to road network design. Our goal is to ensure that for each pair p, q of destinations the *dilation* is small, that is, the Euclidean length of a shortest path (using the road network) from p to q divided by the Euclidean distance from p to q. The question we are interested in is if networks of arbitrary low dilation can be constructed for arbitrary point sets, if only a finite number of crossing points may be used (two roads, each connecting a pair of destinations, are only allowed to cross at crossing points). For the special case where no additional crossing points may be introduced, the Delaunay triangulation of a point set P yields a network whose dilation is less than 2.42, see Keil and Gutwin [44]. Regarding lower bounds, we observe that more recently Bose et al. [9] have constructed a point set whose Delaunay triangulation has dilation > 1.5846. Proving lower dilation bounds in the general case is quite more involved, since for any given point set we allow an arbitrary finite number of crossing points, whose positions in the plane can be chosen arbitrarily. We prove the following theorem in Chapter 2.

Theorem 2.4. *If a finite point set S is not special, there exists a number $\Delta(S) > 1$ such that each finite plane graph P whose vertex set includes S is of dilation $\delta(P) \geq \Delta(S)$.*

Here, a point set P is called *special*, if P is not contained in the vertex set of a triangulation of dilation 1.

Besides travelling, planning the transportation of goods is important. For example, given a transportation network (or flow network), a fundamental question is how many goods can be transported from a given source to a given destination. Another interesting question is the *transportation problem*, i. e, how to move some identical goods from a set of suppliers to some customers in order to satisfy the customer's demand, at minimum cost. If the set of suppliers and customers is discrete, then this problem can be modelled as a minimum-weight matching problem. A more general form of this transportation problem was formulated by Monge [54] in the year 1781, which was finally solved in the 20th century by Kantorovich [42]; also see Gangbo and McCann [32] and Villani [64] regarding geometric aspects of optimal transport.

In Chapter 3, we consider the following variant of the transportation problem. While the set of customers is given by a discrete set $S = \{p_1, \ldots, p_n\}$ in $\mathbb{R}^d$, we assume that the goods offered by the suppliers are given by a continuous probability density function μ whose support is a bounded open set C in $\mathbb{R}^d$. For each point $p \in S$, the distance between p and a point z is given by a function $d_p(z)$. We show

Theorem 3.3. *Let $n \geq 2$ and let $d_{p_i}(\cdot)$, $1 \leq i \leq n$, be an admissible system as in Definition 3.1. Let C be a bounded open subset of $\mathbb{R}^d$. Suppose we are given n real numbers $\lambda_i > 0$ with $\lambda_1 + \lambda_2 + \cdots + \lambda_n = \mu(C)$. Then there is a weight vector $w = (w_1, w_2, \ldots, w_n)$ such that*

$$\mu(C \cap \mathrm{VR}_w(p_i, S)) = \lambda_i$$

holds for $1 \leq i \leq n$.

If, moreover, C is pathwise connected then w is unique up to addition of a constant to all w_i, and the parts $C_i = C \cap \mathrm{VR}_w(p_i, S)$ of this partition are unique.

Here, λ_i denotes the demand of customer $p_i \in S$ and $\mathrm{VR}_w(p_i, S)$ is the Voronoi region of p_i in the additively weighted Voronoi diagram of S based on the distance functions $d_{p_i}(\cdot)$, where w_i is the additive weight of p_i. A set of distance functions $d_{p_i}(\cdot)$ is called *admissible*, if, basically, the weighted bisector of every pair of points in S has measure 0, regardless of the additive weights assigned to the points. We also require that for any two points p_i and p_j in S there is a constant C_{ij}, such that $|d_{p_i}(c) - d_{p_j}(c)| \leq C_{ij}$ holds for all points $c \in C$.

In other words, we show that under quite liberal assumptions on the functions $d_p(\cdot)$, there is an additively weighted Voronoi diagram of the point set S which can be used to characterize *every* optimal solution to the given transportation problem, up to assignment of sets of measure zero. Whereas Gangbo and McCann [32] derive this fact from a more general measure-theoretic theorem, our proof uses the minimization of a quadratic objective function and geometric arguments. The purpose of Chapter 3 is to show that such a simple proof is possible.

If we want to match our current position to a map (without using GPS), or if we want to explore an unknown area, it is necessary to process visibility information. There is a wide range of different visibility problems, e. g., computing efficiently the the set of points visible from a given point in the plane, visibility graph recognition and polygon reconstruction from visibility information, just to mention a few examples. For an overview of efficient solutions for visibility problems in the plane we refer to the survey of Asano et al. [4] and the textbook of Ghosh [37]. Furthermore, one can consider the chapter covering visibility problems [56] in the Handbook of Computational Geometry by Goodman and O'Rourke [38].

In Chapter 4, we show how to compute the union of visibility segments where one endpoint lies on the boundary of one simple polygon P and the other endpoint lies on the boundary of another simple polygon Q. We refer to this set of segments as the *visibility area* between P and Q. Here, the technically most challenging case is where no boundary point of polygon P belongs to the convex hull of the union of P and Q, and P is not contained in Q. Our algorithm runs in optimal linear time, and our analysis shows that the resulting region is defined by (i) the visibility polygon of P and of Q and (ii) by two shortest paths, each connecting a point of Q with a point of P. The above is summarized in the following theorem, which we prove in Chapter 4.

Theorem 4.12. *The visibility area between two disjoint closed polygonal chains P and Q in the plane can be computed in optimal time $O(|P| + |Q|)$.*

In real life, nothing works without error. The final Chapter 5 is dedicated to the construction of fault-tolerant algorithms. Standard computation models (e. g. REAL-RAM, Decision tree) and the frequently used assumption of non-degeneracy of the input allow for platform-independent high-level algorithm design. A given problem can be analysed without worrying about special cases that have no influence on the general structure of the problem. This way, it is easier to describe the – from a theoretical point of view – relevant parts of the algorithm. However, when really implementing such an algorithm, the so called *degenerate cases* also have to be considered. The treatment of those degenerate cases is one possible source for implementation errors, if only due to a lot of case distinctions that have to be done correctly. Moreover, there are other reasons to assume that computation errors can occur. These range from hardware or software problems to errors due to floating point imprecision.

We present some basic toolbox algorithms that can tolerate errors occurring when performing primitive operations such as the comparison of two numbers. If the probability for such a primitive operation to fail is sufficiently small, then our algorithms will output a good result with high probability. For example, we study the searching problem and the sorting problem and obtain the following results.

Theorem 5.14. *Let $p \in (0, 1/4)$ be an upper bound on the error probability of comparisons and let $a_1 < a_2 < \ldots < a_n$ be a sequence of n numbers. Let the block* *size parameter be* $t = \left\lceil \frac{3(\ln(\log n) - \ln p)}{p \cdot \min\{\frac{1}{2p} - 1, (\frac{1}{2p} - 1)^2\}} \right\rceil$. *Given an arbitrary query key* $q \neq a_j$, $1 \leq j \leq n$, *Algorithm* BinarySearch *reports the rank of an element* q *with respect to* $a_1 < a_2 < \ldots < a_n$ *in time* $O(\log n \cdot \log \log n)$ *(*p *is assumed to be constant) and with probability at least* $1 - f(p)$ *for some function* $f(p)$ *that goes to 0 as* p *goes to 0.*

Theorem 5.10. *Let* $p \leq 1/20$. *There is a tolerant algorithm that given an input set* S *of* n *numbers computes in* $O(n^2)$ *time and with probability at least* $1 - 1/n^{26}$ *an output sequence such that the position of every element in this sequence deviates from its rank in* S *by at most* $O(\log n)$.

Our sorting algorithm can be seen as an alternative to an algorithm by Braverman and Mossel [10] who employed the same model. Their algorithm constructs the maximum likelihood permutation using only $O(n \log n)$ comparisons. We achieve similar accuracy using $\Theta(n^2)$ comparisons, but with a substantially faster running time.

We have given some related work to provide a first overview of the topics considered in this thesis. More related work will be mentioned later in the introduction of each chapter.

This thesis is typeset using LaTeX, and most figures have been created using the extensible drawing editor Ipe by Otfried Cheong.

Chapter 2

Most finite point sets in the plane have Dilation > 1

We prove the following fact for arbitrary finite point sets S in the plane. Either, S is a subset of one of the well-known sets of points whose unique triangulation has dilation 1. Or, there exists a number $\Delta(S) > 1$ such that each finite plane graph containing S among its vertices has dilation $\geq \Delta(S)$.[1]

Keywords: Computational geometry, geometric networks, dilation, detour, spanning ratio, stretch factor.

2.1 Introduction

Let S be a finite set of points in the plane. We are interested in finite, *plane* geometric graphs $T = (V, E)$ of minimum dilation whose vertex set V contains S. Here, the dilation of a graph T is defined by

$$\delta(T) := \max_{p \neq q \text{ vertices of } T} \delta_T(p, q),$$

where

$$\delta_T(p, q) := \frac{|\pi(p, q)|}{|pq|}$$

denotes the dilation between vertices p and q, that is, the Euclidean length of a shortest path $\pi(p, q)$ from p to q in T, divided by the Euclidean distance between p and q. The following simple fact will be quite useful.

Lemma 2.1. *Every shortest path $\pi(p, q)$ from p to q in T is contained in the ellipse with foci p and q whose boundary consists of all points e satisfying $|pe| + |eq| = \delta(T) \cdot |pq|$.*

Proof. For each point z on path $\pi(p, q)$ we have $|pz| + |zq| \leq |\pi(p, q)| = \delta_T(p, q) \cdot |pq| \leq \delta(T) \cdot |pq|$. □

[1] An extended abstract of this paper has appeared at SoCG'06 [46].

Definition 2.2. *Given an ellipse E, as in Lemma 2.1, with foci p and q whose boundary consists of all points e satisfying $|pe| + |eq| = \delta \cdot |pq|$, we denote by $\delta \cdot |pq|$ the* diameter *of E. Furthermore, we observe that Ellipse E is contained in the w-neighbourhood of the line segment pq, where $w = \frac{|pq|}{2} \cdot \sqrt{\delta^2 - 1}$; an example is shown in Figure 2.1. Finally, we denote by $2w = |pq| \cdot \sqrt{\delta^2 - 1}$ the* width *of E and observe that the two tangents to E parallel to the line segment pq are at distance $2w$.*

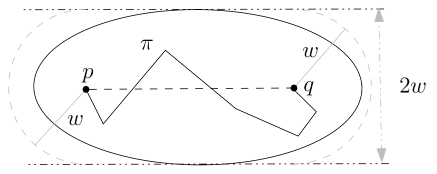

Figure 2.1: In a triangulation of dilation $\leq \delta$, the shortest path π between vertices p and q is contained in an ellipse of width $2w = |pq| \cdot \sqrt{\delta^2 - 1}$.

We observe that, for p and q fixed, these ellipses become arbitrarily thin as δ decreases to 1.

A motivation for the problem studied in this paper goes back to a scenario described by D. Eppstein in [26]. Let S be a finite set of important locations on a university campus (e. g. lecture halls, libraries, mensa, supermarket, dorms, etc.). These locations must be connected by a path system, such that the dilation between any two of them is small. Building bridges is impossible, due to budget limitations. Where two paths cross, a new point of interest (cafeteria, copy shop) appears, because these points are frequented by many people. Since students might want to get quickly from a cafeteria to a lecture hall, *such path crossings must also be considered in computing the dilation of the whole system.* If we treat these crossing points as vertices, just like the given points of S, the whole path system is a plane graph. So, the following question appears. What is the smallest dilation value that can be obtained by a finite plane graph whose vertex set contains a given finite point set, S?

Clearly, we may assume that all edges are straight, because pulling them taut cannot increase the dilation. Moreover, we can focus on triangulations, since adding

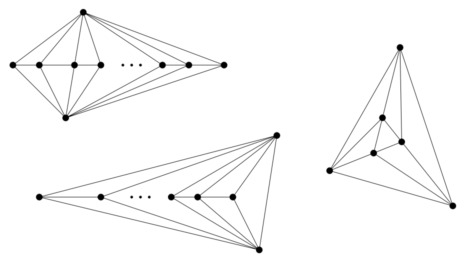

Figure 2.2: The triangulations of dilation 1.

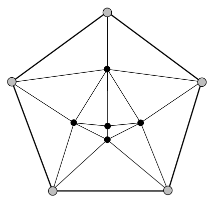

Figure 2.3: An embedding of five points in regular position into a triangulation of dilation 1.02046.

edges without introducing crossings never hurts.

Eppstein [26] has characterized all triangulations of dilation 1, see Figure 2.2. There are two infinite families and one special triangulation on six vertices. Exactly for the subsets S of their vertex sets can a dilation strictly equal to 1 be achieved.

Definition 2.3. *A finite point set S in the plane is called* special *if it is contained in the vertex set of a triangulation of dilation 1.*

The question we are interested in is the following. How small a dilation can be achieved for those point sets S that are *not* special? Even for the most simple case, five points evenly spaced on a circle, the answer is not known. The smallest value known to us is ≈ 1.02046, by a construction due to D. Lorenz [50]; see Figure 2.3. It is unknown if the dilation can be further lowered by more complicated graphs. The purpose of this paper is in proving that for each point set S that is not special there is a lower bound bigger than 1 that cannot be beaten by *any* finite plane graph whose vertex set contains S.

Theorem 2.4. *If a finite point set S is not special, there exists a number $\Delta(S) > 1$ such that each finite plane graph P whose vertex set includes S is of dilation $\delta(P) \geq \Delta(S)$.*

The difficulty in proving this result is as follows. Suppose we start with five points in regular position. We could attempt to enforce low dilation by forming their complete graph, that is, by connecting each point to any other by a straight line segment. While this results in dilation 1 connections between the five points given, the resulting crossing points form a new five-gon, which must again be dealt with, and so on. By iterating this argument, a dilation 1 embedding is not possible in this case (as we already know from the previous discussion). But if we settled for a dilation of $1 + \epsilon$, there would be some slack in this construction. One could perhaps move crossing points slightly away from their proper positions, such that, after some rounds, no further crossings are introduced by forming the complete graph.

As we shall show, this is not possible for arbitrary small values of ϵ. The proof will be given in two steps. In Section 2.4 we shall prove that repeating patterns cannot be avoided if we consider, instead of five points on a circle, 548 points on a square. Our analysis shows that for this point set P each finite plane graph containing P in its vertex set must have a dilation bigger than 1.0000014. Then, in Section 2.5, we combine this finding with a density result by A. Grüne and S. Kamali [39] in order to establish Theorem 2.4.

2.2 Preliminaries

First, we recall the definition of special point sets. A finite point set S is called *special* if it is contained in the vertex set of one of the triangulations of dilation 1 shown in Figure 2.2.

Definition 2.5. *Let S be a finite set of points in the plane. The* dilation*, $\Delta(S)$, of S is defined by*

$$\Delta(S) := \inf \left\{ \delta(T) \;\middle|\; T = (V, E) \text{ finite triangulation with } S \subseteq V \right\}.$$

Here, $\delta(T)$ denotes the dilation of T, as introduced in Section 2.1. We should remark that the dilation of a point set is invariant under scaling because the quotient in the definition of $\delta(T)$ is. The grand challenge is in computing $\Delta(S)$, given a point set S. Observe that we do not know if this value is actually achieved by some triangulation, or if it is just a limit. In this paper, we prove $\Delta(S) > 1$ for all sets S that are not special.

2.3 Related Work

The problem of how to connect a given point set by a network of small dilation has been extensively studied in the context of spanners, where the dilation of a graph is also called its stretch factor or spanning ratio. There exists a wealth of results on constructing spanners with a dilation arbitrarily close to 1, that have other nice properties like linear size, a weight not much exceeding that of the minimum spanning tree, or bounded degree; see, e. g., the monographs [27] and [55]. But these spanners can contain many edge crossings. A notable exception is the result by Aronov et al. [3] where the dilation of plane graphs with $n - 1 + k$ edges, $k < n$, over n given points was shown to be in $\Theta(n/(k + 1))$.

In his handbook chapter [27], Eppstein has posed the following open problems.

- (Problem 8) Can the minimum dilation triangulation of a given point set be computed in polynomial time?
- (Problem 9) How large a dilation can the minimum dilation triangulation have, in the worst case?

In Problem 9, an upper bound is the dilation of the Delaunay triangulation, which has been shown to be less than 2.42 by Keil and Gutwin [44]. More recently, Bose et al. [9] have constructed a point set whose Delaunay triangulation has dilation > 1.5846. A partial answer to Problem 8 was given by Eppstein and Wortman [28] who provided a randomized $O(n \log n)$ algorithm for computing an optimal star center for n points in d-space.

The problem setting we are studying in this paper can be seen as the "Steiner point" variant of Problem 9 above: Instead of looking for an optimal triangulation on just the given point set, we allow ourselves the introduction of a finite number of auxiliary "Steiner points" to help reduce the overall dilation. The crucial point here is that dilation amongst the new points, which we introduce to reduce detours for the given vertices, also needs to be controlled. Ebbers-Baumann et al. [24] have studied Problem 9 in this model. They proved $\Delta(S) < 1.1247$ for arbitrary finite point sets S, but did not address the question of a non-trivial lower bound for finite point sets. The present paper provides a first answer to this question.

A related measure called *geometric dilation* has been introduced and investigated in [1, 19, 18, 25, 23, 22], where *all* points of the graph, vertices and interior edge points alike, are considered in computing the dilation. The small difference in definition leads to rather different results. For example, plane graphs of minimum geometric dilation tend to have curved edges. Yet, it is interesting to observe that in this model non-trivial upper and lower bounds could be established. Each finite point set can be embedded into a plane graph of geometric dilation ≤ 1.678, and there are point sets for which each plane graph containing them has a geometric dilation $\geq (1 + 10^{-11})\pi/2$; see [19, 22]. These results are based on methods from convex geometry that do not apply here.

2.4 A Concrete Lower Bound

The goal of this section is to provide a concrete point set P whose dilation $\Delta(P)$ can effectively be bounded away from 1.

Definition 2.6. *A set $V \subseteq \mathbb{R}^2$ is an* ϵ-cover *of a set $B \subseteq \mathbb{R}^2$ if the closed ϵ-neighbourhood of every point of B contains a point of V.*

The main idea presented in this section is the following. Suppose we have a point set V that is the vertex set of a *finite* triangulation of dilation δ and, at the same time, an ϵ-cover of the boundary B of some square. We shall prove that V is also an $\alpha \cdot \epsilon$-cover of a smaller square boundary B', shrunk by the factor α. Since dilation is invariant under shrinking, this argument can be repeated *ad infinitum.* But V would have to be infinite in order to cover infinitely many square boundaries—a contradiction.

The proof of V's covering property is sketched in Figure 2.4. Given a point z' on B', we find points z_l, z_t, z_r, z_b on the four edges of B such that the line segments $z_l z_r$ and $z_t z_b$ cross at z'. Since, by assumption, V is an ϵ-cover of the boundary B, points z_l, z_t, z_r, z_b are ϵ-approximated by points v_l, v_t, v_r, v_b of V, respectively. Since V is vertex set of a low dilation triangulation T, there are paths from v_l to v_r, and from v_t to v_b, that are contained in thin ellipses; compare Lemma 2.1. Thus, the crossing point v' of these paths lies close to z' and belongs to the vertex set V of T.

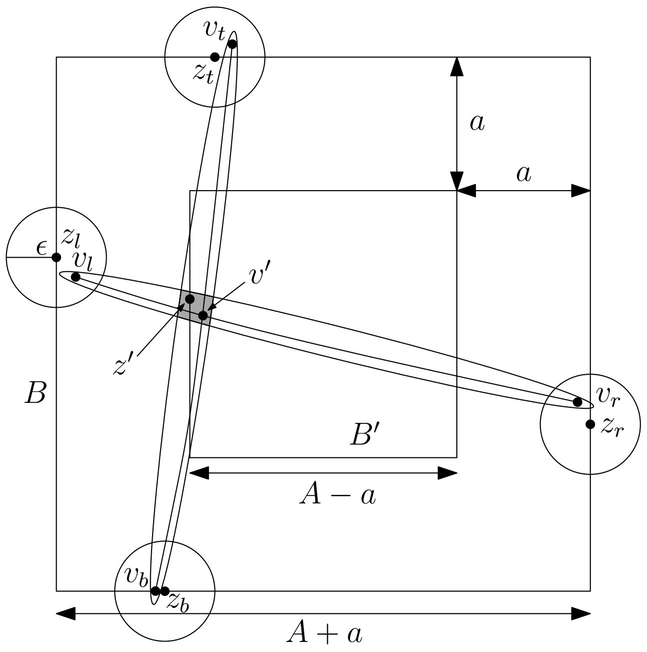

Figure 2.4: Approximating a point z' on B' by an intersection point v' of two shortest paths in triangulation T, one connecting v_l to v_r, and the other connecting point v_t to v_b. Point v' will be shown to lie within distance $\alpha \cdot \epsilon$ of z', where $\alpha = \frac{A-a}{A+a} < 1$.

In the remainder of this section we show how to make this idea precise.

Definition 2.7. *Constants* $0 < a < A$, $0 < \epsilon < a/2$ *and* $\delta > 1$ *are called* cover-preserving *if the following holds for arbitrary point sets* V *in the plane. If* V *is an* ϵ*-cover of the boundary,* B*, of a square of side length* $A + a$*, and if* V *is also vertex set of a triangulation* T *of dilation* δ*, then* V *is an* $\frac{A-a}{A+a} \cdot \epsilon$*-cover for the boundary* B' *of the smaller square of side length* $A - a$*, centred in* B*, see Figure 2.4.*

Lemma 2.8. *Every point set* V *as in Definition 2.7 must be infinite.*

Proof. First, we observe that the points of V covering B' are different from those covering B; in fact, no point can be within distance ϵ of B and, at the same time, be within distance $\epsilon\frac{A-a}{A+a}$ of B', because B and B' are at distance a and

$$a - \epsilon - \epsilon \cdot \frac{A-a}{A+a} \;>\; a - \frac{a}{2} - \frac{a}{2} \cdot \frac{A-a}{A+a} \;>\; 0$$

holds.

Second, we observe that, if $0 < a < A$, $0 < \epsilon < a/2$ and $\delta > 1$ are a set of cover-preserving constants, then $0 < a\frac{A-a}{A+a} < A\frac{A-a}{A+a}$, $0 < \epsilon\frac{A-a}{A+a} < \frac{a}{2} \cdot \frac{A-a}{A+a}$, $\delta > 1$ are also cover-preserving. Note that the value δ remains unchanged because the dilation of triangulation T, in Definition 2.7, is invariant under scaling.

Third, at least one point of V is necessary to ϵ-cover B, and at least 1 additional point of V is needed to $\frac{A-a}{A+a}\epsilon$-cover B'. Now the claim follows from iterating this argument. $\square$

Lemma 2.9. *Assume* a, A, ϵ, δ *are cover-preserving constants, see Definition 2.7. Let* P *be a finite point set that* ϵ*-covers the boundary* B *of a square of size* $A + a$*. Then,* $\Delta(P) \geq \delta > 1$*.*

Proof. Let $T = (V, E)$ be a *finite* triangulation such that $P \subseteq V$. Trivially, V, too, is an ϵ-cover for B. If the dilation of T were $\leq \delta$, Lemma 2.8 would imply that V is infinite. Hence, $\delta(T) > \delta$. By Definition 2.5, $\Delta(P) \geq \delta$ follows. $\square$

Since it is not *a priori* clear that cover-preserving constants exist, we need to provide concrete values for the parameters a, A, ϵ, δ before we can employ Lemma 2.9 to produce point sets P together with concrete lower dilation bounds. This will be done in the proof of Lemma 2.10, to which Subsection 2.4.1 is devoted.

Lemma 2.10. *The constants* $a = 1, A = 15, \epsilon = 0.0588$ *and* $\delta = 1.0000014$ *are cover-preserving, according to Definition 2.7.*

The rough idea of this proof is as follows. Let z' be a point of the boundary B' of the smaller square, as in Definition 2.7; we must show that it can be closely approximated by some point of V. We can describe point z' as the intersection point of two line segments, one vertical and one horizontal, with endpoints on B. Now if V is a point set as in Definition 2.7, these endpoints can be approximated by points of V, say v_t, v_b, v_l, v_r, that are situated near the top, bottom, left, and right edge of B, correspondingly. The intersection, z, of the segments v_tv_b and v_lv_r very closely approximates z'. Furthermore, V is also the vertex set of a triangulation T of dilation $\leq \delta$. Thus, there is a path π connecting v_t to v_b in T whose length does not exceed $\delta \cdot |v_tv_b|$. By Lemma 2.1, the whole path π is contained in the ellipse of diameter $\delta \cdot |v_tv_b|$ with foci v_t and v_b; compare Definition 2.2.

A similar path exists between v_l and v_r. The two paths cross in the intersection of their respective ellipses. The crossing point must be a vertex, v', of triangulation T, that is, v' is a point of V. The width of each ellipse equals the distance between its focal points, times $\sqrt{\delta^2 - 1}$. The latter factor tends to zero as δ gets close to 1. Therefore, v' is a good approximation of z, which approximates, in turn, the given point z' on B', compare Figure 2.4.

This idea will be made precise in the sequel. The main challenge is to achieve a better approximation for the points on B' than for the points on B. As the claim of Lemma 2.10 is of quantitative nature, careful analysis cannot be avoided. We shall, however, try to ease the reader's way through the calculations in the proof.

2.4.1 The Existence of Cover-Preserving Values

The main part of this section is devoted to showing the existence of cover-preserving values. We shall show that the values $a = 1, A = 15, \epsilon = 0.0588$, and $\delta = 1.0000014$ are cover-preserving, which proves Lemma 2.10. In order to prove this claim, let us assume that we are given a triangulation $T = (V, E)$ of dilation $\leq \delta$, such that V is an ϵ-cover for the boundary, B, of a square of side length $A + a$. As before, we denote with B' the boundary of the smaller square of side length $A - a$, centred in B, as shown in Figure 2.4. Note that in this section, for simplicity of illustration, the figures are not true to scale.

Let z' be a point on the bottom edge of B'. We define a coordinate system such that $z' = (0, 0)$ is the origin, the horizontal and vertical line through z' define the X- and Y-axis, respectively. Now let $w = (0, -1) = (0, -a)$ be the point below z' on B; see Figure 2.5.

We shall need the following technical fact.

Lemma 2.11. *There exists a point $p \in V$ near w, and points $q, r \in V$ near the upper horizontal edge of B, such that the following properties hold for the line segments pq, pr and their respective intersection points i, j with the X-axis, see Figure 2.5.*

1. *The distance from point p to point w is at most ϵ.*

2. *Points q, r have distance at most ϵ to some point t on the upper edge of B.*

3. *Points i and j lie on opposite sides of z'.*

4. *The distance, ℓ, between i and j is at most*

$$\ell_{max} := \frac{2\epsilon(a + 2\epsilon)}{A + a} = \frac{2\epsilon(1 + 0.1176)}{16} = 0.1397\epsilon < 0.14\epsilon.$$

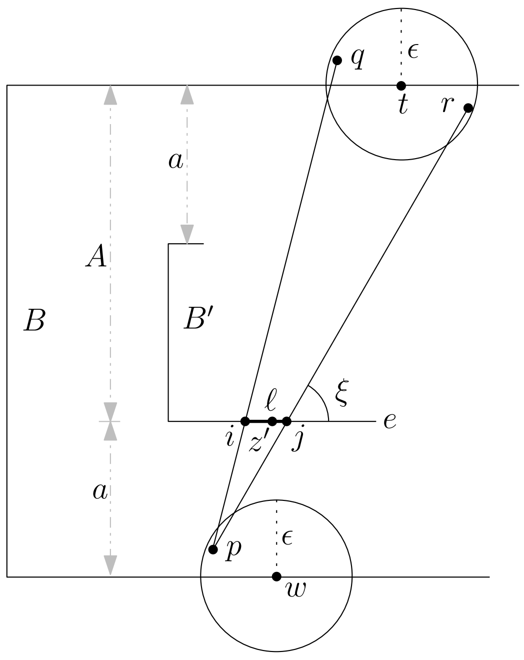

Figure 2.5: Including point z' in a wedge of segments that connect points of V.

5. *The minimum angle, ξ, between the segments pq, pr and the X-axis is at least*

$$\xi_{min} := \arctan \frac{(a-\epsilon)(A+a-2\epsilon)}{\epsilon(A+3a-2\epsilon)} = \arctan(14.216...) = 85.976...^\circ > 85.97^\circ.$$

Proof. Since V is an ϵ-cover of B by assumption, we can in fact find a point $p \in V$ in the ϵ-neighbourhood of the point $w = (0, -1)$.

To prove the existence of q and r we argue as follows. For each point u on the upper horizontal edge of B its ϵ-neighbourhood contains a point, denoted by $v(u)$, of V. Let us specifically consider the point $u_1 = (-1, 15)$ on B. Let R denote the right tangent to the bounding boxes of the ϵ-neighbourhoods of u_1 and w, respectively; see Figure 2.6. Since line R is passing through the points $u' = (-1+\epsilon, 15+\epsilon)$ and $w' = (\epsilon, -1+\epsilon)$, its equation is given by $Y = -16X - 1 + 17\epsilon$. Thus, the intersection point of R with the X-axis equals $v' = (\frac{17\epsilon-1}{16}, 0)$; it lies to the left of z', if $\epsilon < \frac{1}{17} = 0.058823...$ holds, which is true by our choice of ϵ. By construction, the line segment $pv(u_1)$ connecting p with $v(u_1)$ also intersects the X-axis to the left of z'.

By an analogous argument one can show that the ϵ-neighbourhood of $u_2 = (1, 15)$ contains a point $v(u_2) \in V$ such that the line segment $pv(u_2)$ with endpoints p and $v(u_2)$ passes the X-axis to the right of z'. As we walk along the upper horizontal edge of B from u_1 towards u_2 there must be a point t whose ϵ-neighbourhood contains two points, q and r, of V, whose connecting segments to p cross the X-axis to the left and to the right of z', as shown in Figure 2.5.

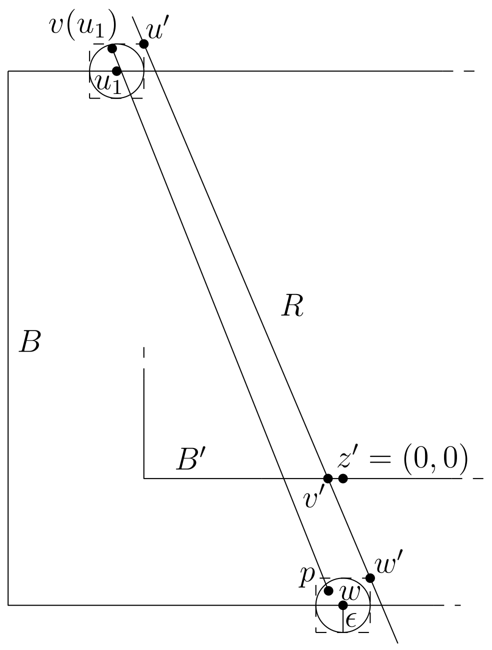

Figure 2.6: Existence of a segment connecting points in V, passing to the left of z'.

Now that Properties 1, 2, and 3 have been verified we turn to Properties 4 and 5, and prove an upper bound ℓ_{max} for the distance $|ij|$, and a lower bound, ξ_{min}, for the minimum angle between the segments pq, pr and the X-axis. First, we state a geometric fact whose proof is straightforward.

Lemma 2.12. *Let q, r be two points above two parallel, horizontal lines U and L in the plane. For a point p on the lower line, L, let i, j denote the intersection points of pq and pr, respectively, with the upper line U. Then the following holds for the distance $d(p) = |i - j|$, as a function of p.*

i) If q, r have identical y-coordinates, then $d(p)$ is constant.

ii) Otherwise, we have $d(p) = 0$ when p is co-linear with q and r. Moving p along L away from this position in either direction does strictly increase $d(p)$.

Since we do not know the exact location of the points p, q and r, we cannot give a straightforward lower bound for the minimum angle ξ between the segments pq, pr and the X-axis or an upper bound to the maximum distance ℓ between the intersection points i and j. We will thus move the points p, q and r inside the bounding boxes of the ϵ-neighbourhoods of w and t, as indicated in Figure 2.7, at the same time ensuring that the new resulting minimum angle ξ decreases and the resulting distance ℓ increases. Clearly, the bounds we obtain after moving the points also hold for the original positions of p, q and r, *i. e.*, the vertices $p, q, r \in V$.

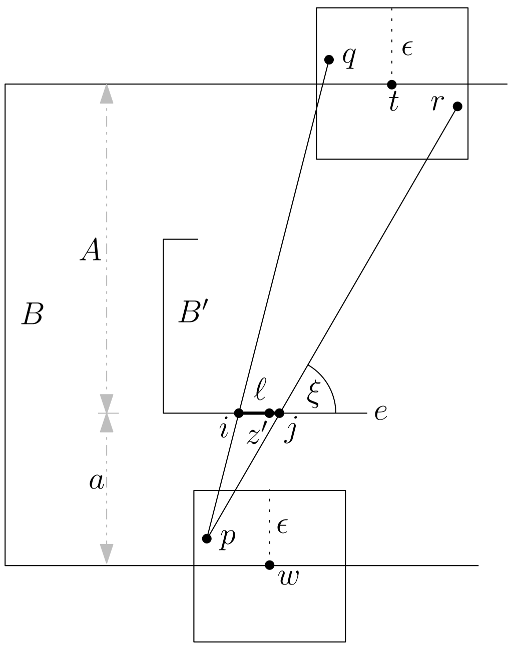

Figure 2.7: Maximizing ℓ and minimizing ξ.

In the following, we cannot avoid some case distinction. Remember that z' is the origin and we denote the coordinates of a point s by s_x and s_y. Recall that t denotes the point on B whose ϵ-neighbourhood contains q and r. W.l.o.g. we assume $t_x \geq 0$; the case $t_x < 0$ can be analysed by symmetric arguments. Now $p_x \leq \epsilon$ implies $p_x \leq \epsilon + t_x$, so we know that point p does not lie very far to the right of point t. There are two possible cases:

1. $p_x \in (t_x - \epsilon, t_x + \epsilon]$. As depicted in Figure 2.8, we move q to the lower left corner, and r to the lower right corner of the box centred at t, thus increasing $|ij|$ and decreasing ξ. While the increase of $|ij|$ is obvious, note that the angle formed by the X-axis and one segment, e. g. pq, may increase. But since now both segments, pq and pr, are tangents to the ϵ-box of t, one of the two segments must minimize the angle of all segments $\{px \,|\, x \textit{ belongs to the } \epsilon\textit{-box of } t\}$.

   Now the situation is as follows. If we moved p horizontally to the right until $p_x = t_x + \epsilon$, then this would decrease the angle of segment pq to some value ξ_q and increase the angle of segment pr to $\pi/2$. Moving p horizontally to the left until $p_x = t_x - \epsilon$, as shown in Figure 2.9, increases the angle of segment pq to $\pi/2$ and decreases the angle of segment pr to some value ξ_r. By symmetry, we have $\xi_q = \xi_r$. Therefore, after moving p horizontally to the left until $p_x = t_x - \epsilon$, the minimum angle ξ has decreased to ξ_r and it is now defined by segment pr. Furthermore, by Lemma 2.12, $|ij|$ remains unchanged while moving p, because q

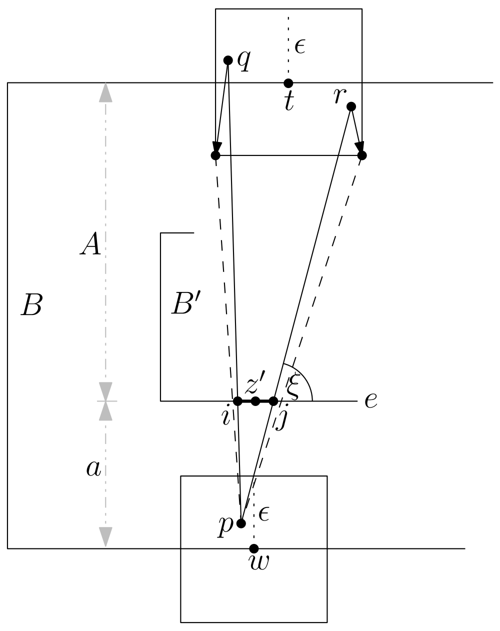

Figure 2.8: Moving q, r to the lower left and lower right corner of the box centred at t.

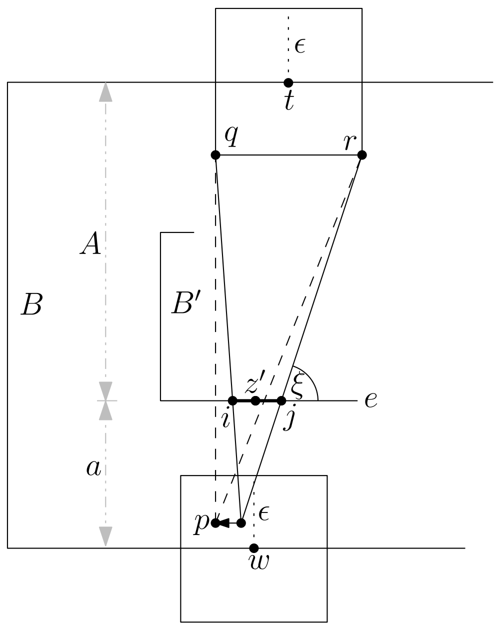

Figure 2.9: Moving p to the left until $p_x = t_x - \epsilon$.

and r have the same Y-coordinates. Instead of moving p to the left we may move t, q, r to the right, ensuring that z' remains between i and j. We have increased ℓ and decreased ξ, and, since $p_x = t_x - \epsilon$, now Case 2 applies.

2. $p_x \le t_x - \epsilon$. We move q to the upper left corner, and r to the lower right corner of their box; see Figure 2.10. From $p_x \le t_x - \epsilon$, it follows that we have increased $|ij|$

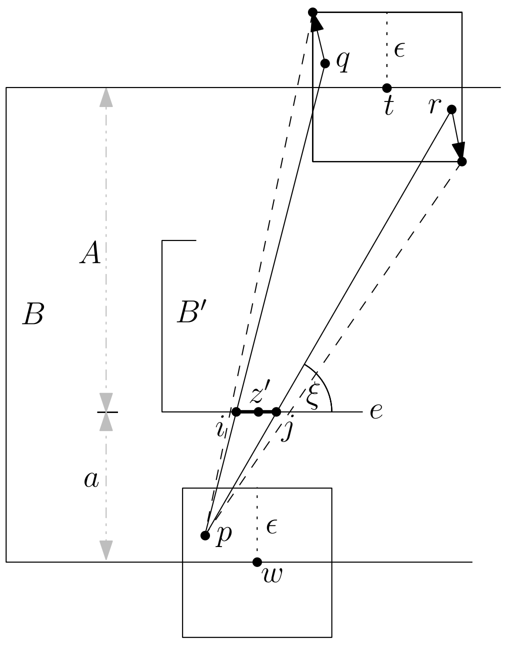

Figure 2.10: Moving q, r to the upper left and lower right corner of their box.

and decreased ξ. Now ξ is decreased as we move p, as shown in Figure 2.11, to the left edge of its box. Furthermore, by Lemma 2.12, this can only increase ℓ because p lies to the left of the line through q and r. Afterwards, we move t, q, r to the right until z' lies on segment pq. By the same arguments as above, this final step also decreases ξ and increases ℓ.

The resulting configuration is shown in Figure 2.12. The lines L_1, L_2 through pq, pr are given by the equations

$$\begin{aligned}
L_1(X) &= \frac{a-\eta}{\epsilon}X \\
&= \frac{1-\eta}{\epsilon}X, \text{ as follows from considering the coordinates of} \\
&\quad p = (-\epsilon, \eta - 1) \text{ and } z' = (0,0). \\
L_2(X) &= \frac{(a-\eta)(A+a-\epsilon-\eta)}{\epsilon(A+3a+\epsilon-3\eta)}X - \frac{2(a-\eta)(a+\epsilon-\eta)}{A+3a+\epsilon-3\eta} \\
&= \frac{(1-\eta)(16-\epsilon-\eta)}{\epsilon(18+\epsilon-3\eta)}X - \frac{2(1-\eta)(1+\epsilon-\eta)}{18+\epsilon-3\eta},
\end{aligned}$$

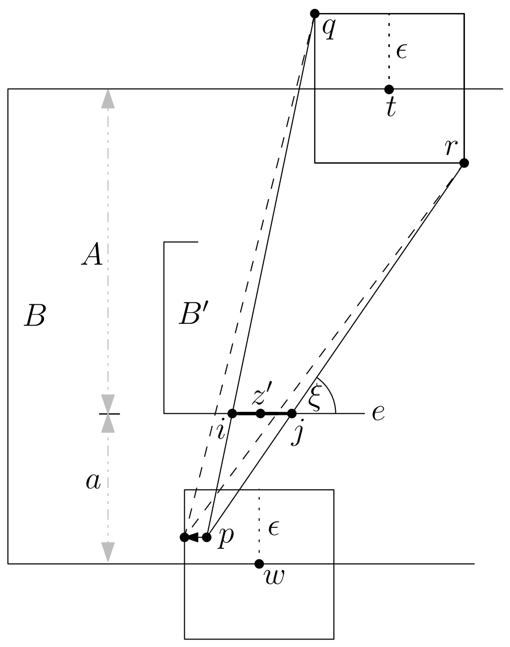

Figure 2.11: Moving p horizontally to the left edge of its box.

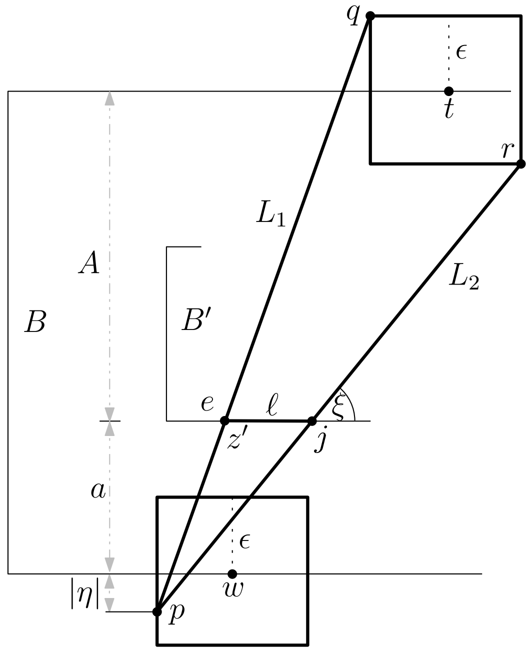

Figure 2.12: ℓ and ξ depending on the height η of p in its box.

where $\eta \in [-\epsilon, \epsilon]$ parametrizes the height of p in its box. The equation of L_2 can be verified by substituting the coordinates of points p and r. The latter can be obtained as follows. Since $q \in L_1$ we have $q = ((15+\epsilon)\frac{\epsilon}{1-\eta}, 15+\epsilon)$. Obviously, $r = q + (2\epsilon, -2\epsilon)$.

From the equation for L_2 we obtain

$$\ell = \frac{2\epsilon(1+\epsilon-\eta)}{16-\epsilon-\eta}$$

for the X-coordinate of the intersection point of line L_2 with the X-axis. Since the derivative by η is always negative this value is maximized for $\eta = -\epsilon$, which leads to

$$\ell_{max} \quad = \quad \frac{2\epsilon(1+2\epsilon)}{16}.$$

This proves Property 4. Angle ξ is minimized for p in the upper left corner of its box, that is, for $\eta = \epsilon$, which results in

$$\tan \xi_{min} = \frac{(1-\epsilon)(16-2\epsilon)}{\epsilon(18-2\epsilon)},$$

using the coefficient of X in the equation of L_2. This proves Property 5, and completes the proof of Lemma 2.11. □

Now let $p, r \in V$ be the points depicted in Figure 2.5. Because the dilation of triangulation T is at most δ, there is a path of length $\leq \delta \cdot |pr|$ in T that connects p and r. By Lemma 2.1, this path must be completely contained in the ellipse of diameter $\delta \cdot |pr|$ with foci p and r. The points p and r are at distance at most $(A+a+2\epsilon)/\sin \xi_{min}$, see Figure 2.13. Thus, the ellipse is of width at most

$$2 \cdot w_B \quad = \quad \frac{A+a+2\epsilon}{\sin \xi_{min}}\sqrt{\delta^2-1} < \frac{16+2\epsilon}{\sin 85.97^\circ}\sqrt{\delta^2-1} \tag{2.1}$$

$$\quad = \quad 0.02703... < 0.027048 = 0.46\epsilon; \tag{2.2}$$

using the values for a, A, ϵ, δ and the value for ξ_{min} established in Lemma 2.11.

After these preparations we shall now prove that V contains close approximations to the four corners of square B'. Let u denote the lower left corner of B'. We know that within distance ℓ_{max} to the right of u, an almost vertical segment connecting the points $v_b := p$ and $v_t := r$ of V is passing, whose angle with the lower edge of B' is at least ξ_{min}. Symmetrically, there is an almost horizontal segment connecting two points $v_l, v_r \in V$ that passes the left vertical edge of B' closely above u; see Figure 2.14. We now provide a condition sufficient for the existence of a vertex $v' \in V$ close to the intersection point, z, of the two segments.

Definition 2.13. *An ellipse $E_{b,t}$ of diameter $\delta \cdot |v_bv_t|$ with foci v_b, v_t and another ellipse $E_{l,r}$ of diameter $\delta \cdot |v_lv_r|$ with foci v_l, v_r* intersect properly*, if the segments v_bv_t and v_lv_r intersect at some point $z \notin \{v_b, v_t, v_l, v_r\}$, and furthermore $\{v_b, v_t\} \cap E_{l,r} = \emptyset$ and $\{v_l, v_r\} \cap E_{b,t} = \emptyset$ hold.*

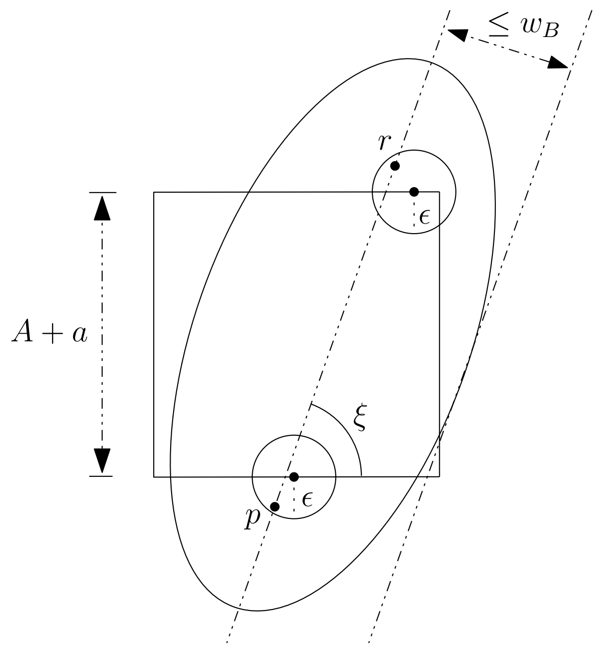

Figure 2.13: The ellipse with foci p and r and diameter $\delta \cdot |pr|$.

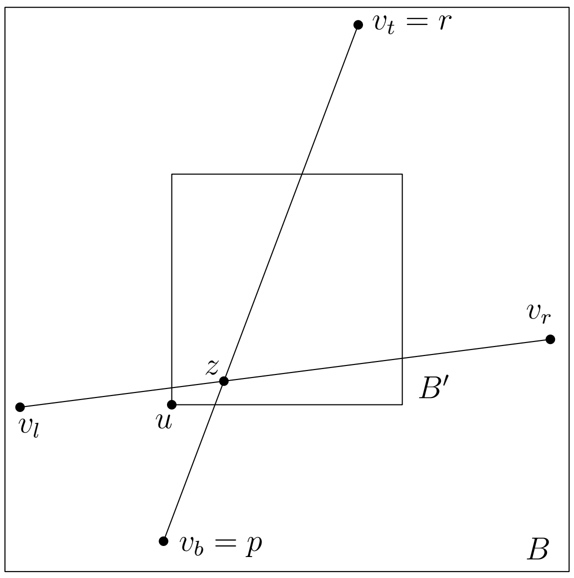

Figure 2.14: Two segments connecting points in V intersect at some point z close to u.

Suppose two ellipses $E_{b,t}$ and $E_{l,r}$ as in Definition 2.13 intersect properly, then any two paths $\pi_{b,t}$ and $\pi_{l,r}$ from v_b to v_t and from v_l to v_r, that are contained in $E_{b,t}$ and $E_{l,r}$, respectively, must intersect at some point $v' \in E_{b,t} \cap E_{l,r}$; compare Figure 2.15 for an example. In the remainder of this section we use Definition 2.13 in the following way.

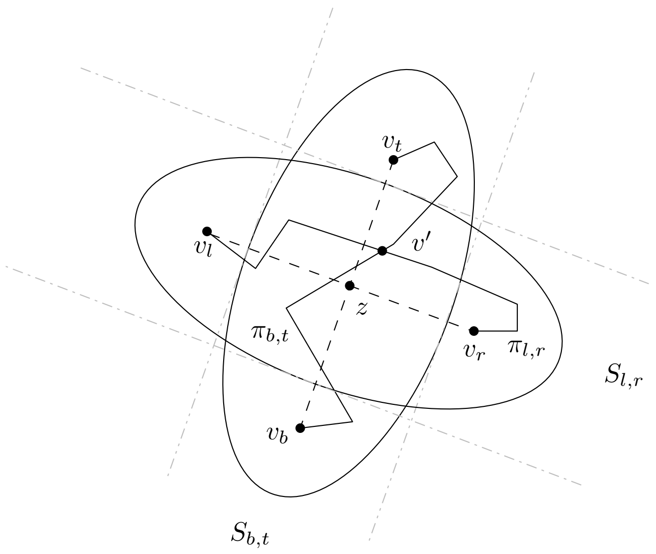

Figure 2.15: There must be an intersection point of the two paths connecting the foci of two properly intersecting ellipses, if the paths may not leave their ellipis.

Lemma 2.14. *Let v_b, v_t, v_l, v_r denote vertices of a triangulation T whose dilation is at most δ, where the segments $v_b v_t$ and $v_l v_r$ intersect at some point $z \notin \{v_b, v_t, v_l, v_r\}$. Given an ellipse $E_{b,t}$ of diameter $\delta \cdot |v_b v_t|$ with foci v_b, v_t and another ellipse $E_{l,r}$ of diameter $\delta \cdot |v_l v_r|$ with foci v_l, v_r, and two sets $S_{b,t} \supseteq E_{b,t}$ and $S_{l,r} \supseteq E_{l,r}$, then the following holds. If $\{v_b, v_t, v_l, v_r\} \cap \{S_{b,t} \cap S_{l,r}\} = \emptyset$, then there exists a vertex v' of T in $S_{b,t} \cap S_{l,r}$.*

Proof. Let $\pi_{b,t}$ and $\pi_{l,r}$ be two shortest paths in T that connect v_b with v_t and v_l with v_r, respectively. Since the dilation of T is at most δ, we have $\pi_{b,t} \subseteq E_{b,t}$ and $\pi_{l,r} \subseteq E_{l,r}$. According to Definition 2.13, ellipses $E_{b,t}$ and $E_{l,r}$ intersect properly. Hence $\pi_{b,t}$ and $\pi_{l,r}$ must intersect at some vertex v' of T in $E_{b,t} \cap E_{l,r} \subseteq S_{b,t} \cap S_{l,r}$. □

As described above, each point pair, v_b, v_t and v_l, v_r, defines an ellipse, $E_{b,t}$ and $E_{l,r}$, that is contained in a strip of width at most $2w_B$. Let $S_{b,t}$ and $S_{l,r}$ denote these strips. We first give an upper bound on the distance from u to any point in the intersection

$S_{b,t} \cap S_{l,r}$ of the two strips. Then, we use this upper bound and Lemma 2.14 to show that $S_{b,t} \cap S_{l,r}$ must contain a vertex v' of T; see Figure 2.15.

Because of properties 4 and 5 (compare Lemma 2.11) strip $S_{b,t}$ is contained in region $R_{b,t}$ depicted in Figure 2.16. Formally, region $R_{b,t}$ is defined as the region

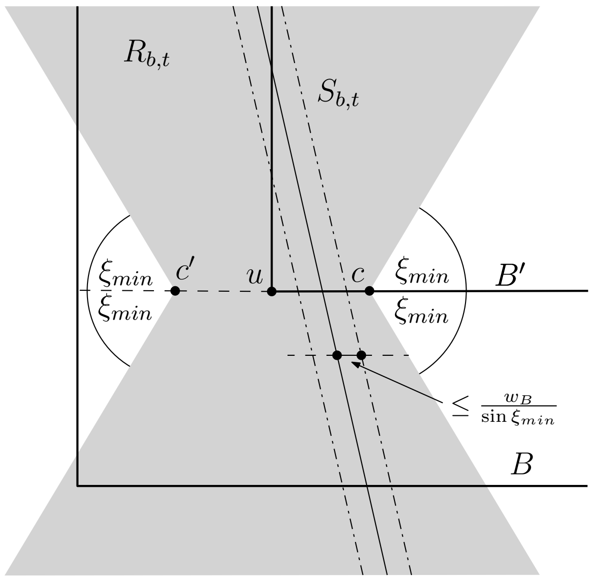

Figure 2.16: The unbounded region $R_{b,t}$ containing strip $S_{b,t}$. Recall that this figure is not true to scale.

between four half lines emanating from the points $c = (\ell_{max} + \frac{w_B}{\sin \xi_{min}}, 0)$ and $c' = (-(\ell_{max} + \frac{w_B}{\sin \xi_{min}}), 0)$. The slope of each half-line is either $\tan \xi_{min}$ or $-\tan \xi_{min}$. Rotating $R_{b,t}$ by 90 degrees, with u as centre of rotation, we obtain region $R_{l,r}$ containing $S_{l,r}$. Now any point $v' \in S_{b,t} \cap S_{l,r}$ must be contained in the intersection of $R_{b,t}$ with $R_{l,r}$, and hence cannot be further away from u than the intersection point s of two half-lines defining the boundary of $R_{b,t}$ and $R_{l,r}$, depicted in Figure 2.17.

To calculate the distance from u to s, let us consider the triangle whose vertices u, c, s are depicted in Figure 2.17. By symmetry, it has an angle of $\pi/4$ at vertex u, and its edge uc is of length $\ell_{max} + w_B / \sin \xi_{min}$. Applying the law of sines we obtain for $D := |us|$

$$\begin{aligned} \frac{D}{\sin \xi_{min}} &= \frac{\ell_{max} + \frac{w_B}{\sin \xi_{min}}}{\sin(\xi_{min} - \frac{\pi}{4})}, \text{ hence} \\ D &= \left(\ell_{max} + \frac{w_B}{\sin \xi_{min}}\right) \frac{\sqrt{2}}{1 - \cot \xi_{min}}. \end{aligned}$$

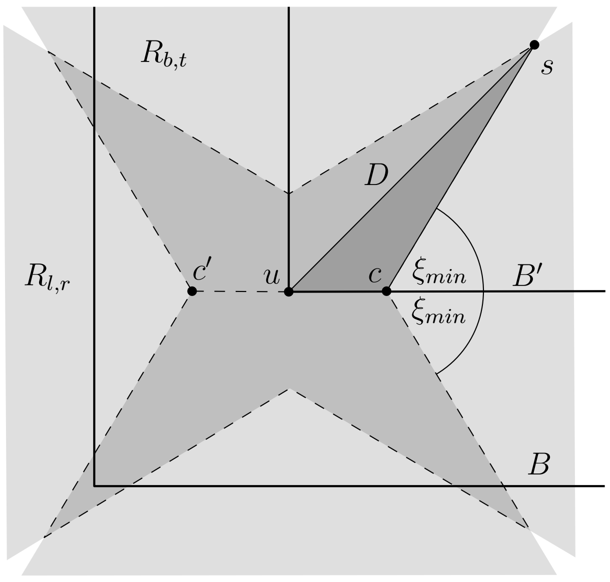

Figure 2.17: The intersection of region $R_{b,t}$ with region $R_{l,r}$. Recall that this figure is not true to scale.

Because of

$$1 > \sin \xi_{min} > \sin 85.97^\circ > 0 \text{ and } 1 > 1 - \cot \xi_{min} > 1 - \cot 85.97^\circ > 0$$

we obtain $D < 0.0331632 = 0.564\epsilon$, using $\ell_{max} < 0.14\epsilon, \xi_{min} > 85.97^\circ, w_B < 0.23\epsilon$ and $\epsilon = 0.0588$.

Now we use Lemma 2.14 to show that $S_{b,t} \cap S_{l,r}$ contains a vertex v' of T. The first step is showing that the segments $v_b v_t$ and $v_l v_r$ intersect. The two lines supporting $v_b v_t$ and $v_l v_r$, respectively, cannot be parallel. Hence, they must intersect at some point $z \in S_{b,t} \cap S_{l,r}$. The following estimate on the Y-coordinates $(v_b)_y, (v_t)_y$ and z_y of the points v_b, v_t and z

$$(v_b)_y \leq -(1-\epsilon) < -D \leq z_y \leq D < 15 - \epsilon \leq (v_t)_y$$

implies that z is interior point of the segment $v_b v_t$. By the same argument, applied to the X-coordinates of the points v_l, v_r and z, z is also interior point of the segment $v_l v_r$. The second step is showing that none of the points v_b, v_t, v_l, v_r is contained in $S_{b,t} \cap S_{l,r}$. Indeed, the distance of any of these four points to u is at least

$$a - \epsilon = 1 - \epsilon = 0.9412 > D \geq \max_{w \in S_{b,t} \cap S_{l,r}} |uw|.$$

This completes the proof that the lower left corner u of B' can be D-approximated by some vertex v' of T. Analogous arguments hold for the other three corners of B'.

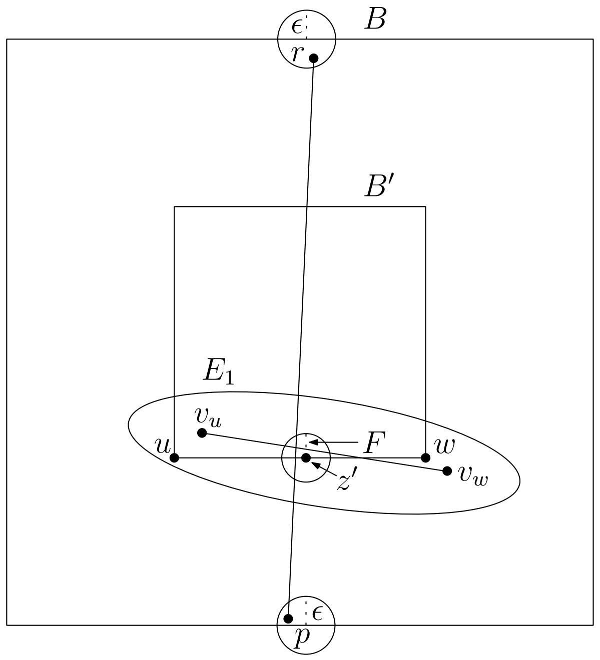

Figure 2.18: The ellipse E_1 of width $\delta \cdot |v_u v_w|$ with foci v_u, v_w will be used to show that z' can be approximated by a vertex in V, up to a distance of $F < \frac{A-a}{A+a}\epsilon$.

So far, we have shown that each corner of the boundary B' of the smaller square can be D-approximated by a point in V. Suppose $v_u, v_w \in V$ are such D-approximations to the endpoints u, w of the bottom edge e of B', correspondingly. With E_1 we denote the ellipse with foci v_u, v_w and diameter $\delta \cdot |v_u v_w|$, where δ is the dilation value defined at the beginning of this subsection.

Now let z' denote an arbitrary interior point on e, as shown in Figure 2.18. We shall show that z' can be approximated by a point of V up to a distance of

$$F < \frac{A-a}{A+a}\epsilon;$$

this will conclude the proof of Lemma 2.10.

By Lemma 2.11, there exist points $p, r \in V$ such that segment pr intersects e no more than a distance ℓ_{max} away from z', with an angle at least ξ_{min}, see Figure 2.5. From now on we shall use $\ell := \ell_{max}$ and $\xi := \xi_{min}$, for short. Let E_2 be the ellipse with foci p, r and diameter $\delta \cdot |pr|$. There are two cases. Either the two ellipses E_1 and E_2 intersect properly, compare Definition 2.13. Or, E_1 and E_2 do *not* intersect properly.

We start with the first case. We fix one path π_1 in T, connecting v_u with v_w, that is contained in E_1, and another path π_2 in T, connecting p with r, that is contained in E_2. These paths must exist because the dilation of T is at most δ. Because E_1 and E_2 intersect properly, paths π_1 and π_2 intersect at some point $v' \in V$. The width of ellipse E_1 is

$$2w_{B'} \leq (A - a + 2D)\sqrt{\delta^2 - 1} \leq 0.0235375... < 0.0235788 = 0.401\epsilon;$$

this upper bound follows from $|v_u v_w| \leq A - a + 2D$ and $D < 0.0331632$. We conclude that v' is contained in the horizontal strip $S_1 = \{(x, y) \mid -D - w_{B'} \leq y \leq D + w_{B'}\}$ because π_1 is contained in the $w_{B'}$-neighbourhood of segment $v_u v_w$. As before, we consider the region $R_{b,t}$, but centred in z instead of the left endpoint, u, of edge e. Thus, region $R_{b,t}$ is formally defined as the region between four half lines emanating from the points $c = (\ell + \frac{w_B}{\sin\xi}, 0)$ and $c' = (-(\ell + \frac{w_B}{\sin\xi}), 0)$, assuming w.l.o.g. that $z' = (0, 0)$. The slope of each half-line is either $\tan\xi$ or $-\tan\xi$; compare Figure 2.19. Again, from the construction of region $R_{b,t}$, it follows that ellipse E_2 is a subset of $R_{b,t}$. Hence, intersection point v' is contained in the intersection of S_1 with $R_{b,t}$. Therefore, v' cannot be further away from z' than the intersection point s of the horizontal line $\{(x, y) \mid y = D + w_{B'}\}$ with line $L = \{(x, y) \mid y = x\tan\xi - \left(\ell + \frac{w_B}{\sin\xi}\right) \cdot \tan\xi\}$. To upper-

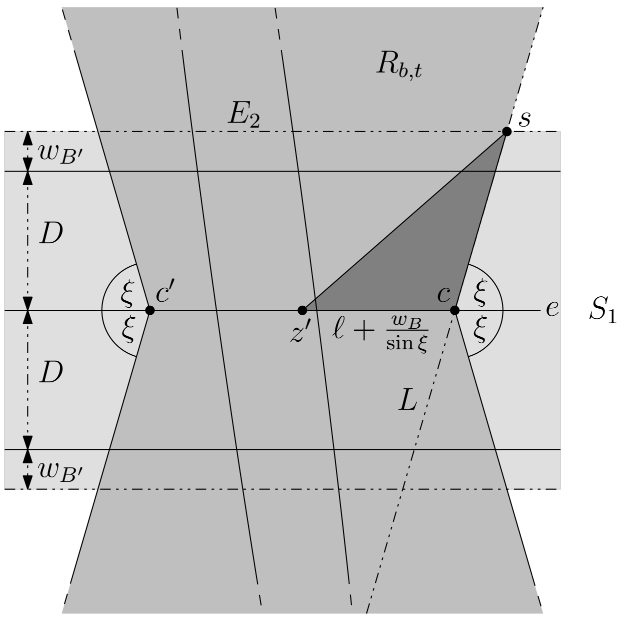

Figure 2.19: The distance from z' to the closest point in V is at most $|z's|$.

bound the maximum distance from z' to point v', we consider the triangle defined by the points z', c, s depicted in Figure 2.19, where $z' = (0, 0)$ and $c = (\ell + \frac{w_B}{\sin\xi}, 0)$. By the law of cosine we obtain

$$\begin{aligned}
|z'v'| \le |z's| =: F &= \sqrt{|z'c|^2 + |cs|^2 - 2\cdot|z'c|\cdot|cs|\cdot\cos(\pi-\xi)} \\
&= \sqrt{|z'c|^2 + |cs|^2 + 2\cdot|z'c|\cdot|cs|\cdot\cos\xi} \\
&= \sqrt{\left(\ell+\frac{w_B}{\sin\xi}\right)^2 + \left(\frac{D+w_{B'}}{\sin\xi}\right)^2 + 2\cdot\left(\ell+\frac{w_B}{\sin\xi}\right)\cdot\left(\frac{D+w_{B'}}{\sin\xi}\right)\cdot\cos\xi} \\
&< 0.05145 \\
&= 0.875\epsilon = \frac{A-a}{A+a}\epsilon.
\end{aligned}$$

The fourth line follows using $\xi > 85.97°$, $\frac{1}{\sin\xi} < \frac{1}{\sin(85.97°)}$, $\cos\xi < \cos(85.97°)$, $w_B < 0.23\epsilon$, $w_{B'} < 0.2005\epsilon$, $\ell < 0.14\epsilon$ and $D < 0.564\epsilon$.

We now continue with the second case, where ellipses E_1 and E_2 do not intersect properly (recall that the foci of E_1 are v_u, v_w and the foci of E_2 are p, r). In this case we show that $|v_u z'| \le F$ or $|v_w z'| \le F$ holds. We consider the rectangle $S_1 = [u_x - D, w_x + D] \times [-D, D]$, where, as before, u and w denote the lower left and lower right corner the boundary B' of the smaller square. Hence, the height of S_1 is $2D$ and its width is $A - a + 2D$. Furthermore, we consider the region $R_{b,t}$ centred at z' as defined above. Let HL_l denote the union of two half lines originating from point c', which lies at distance $\ell + \frac{w_B}{\sin\xi}$ to the left of $z' = (0,0)$; see Figure 2.20. The angles of these half lines with the (negative) X-axis are equal to ξ. In fact, HL_l is the left part of $R_{b,t}$'s boundary. Now point v_u, our D-approximation to the left endpoint u of edge e, can lie in a region labelled 1, 2 or 3 in Figure 2.20. If v_u lies in region 1, i. e. $(v_u)_x \ge 0$, we obtain $|z'v_u| \le |uv_u| \le D < F$, the latter by plugging in values. If v_u lies in region 2, i. e. $(v_u)_x < 0$ and v_u lies to the right of HL_l (including the case $v_u \in HL_l$), we conceptually move v_u horizontally to the left until HL_l is reached. This can only increase $|v_u z'|$. Now we have $|v_u z'| \le |sz'|$ where point s is the intersection point of HL_l with the horizontal line $\{(x,y) \mid y = D\}$, as depicted in Figure 2.20. Using the same estimate as in the first case (where E_1 and E_2 intersect properly), applied to $v' := v_u$, we obtain

$$\begin{aligned}
|z'v_u| \le |z's| &= \sqrt{|z'c'|^2 + |c's|^2 + 2\cdot|z'c'|\cdot|c's|\cdot\cos\xi} \\
&= \sqrt{\left(\ell+\frac{w_B}{\sin\xi}\right)^2 + \left(\frac{D}{\sin\xi}\right)^2 + 2\cdot\left(\ell+\frac{w_B}{\sin\xi}\right)\cdot\left(\frac{D}{\sin\xi}\right)\cdot\cos\xi} \\
&< 0.05145 \\
&= 0.875\epsilon = \frac{A-a}{A+a}\epsilon.
\end{aligned}$$

Should v_u be situated in region 3, that is, v_u lies strictly to the left of HL_l, we consider the D-approximation, v_w, of the opposite endpoint w of edge e; see Figure 2.21. The set HL_r of half-lines is defined analogously to HL_l. More precisely, it is the union

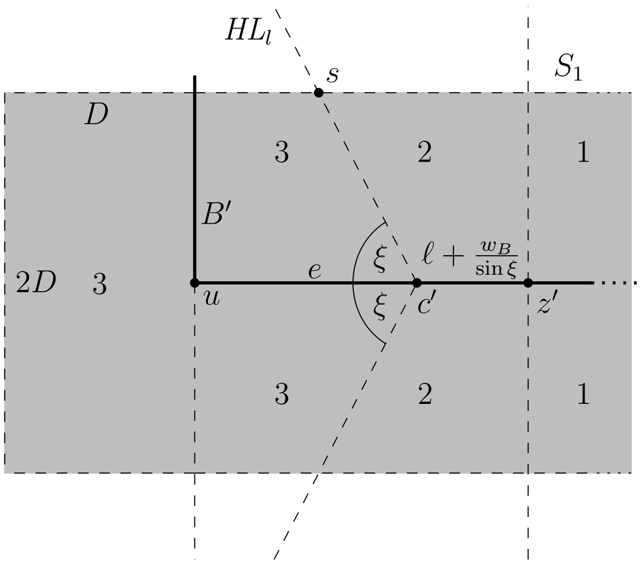

Figure 2.20: The possible regions for v_u.

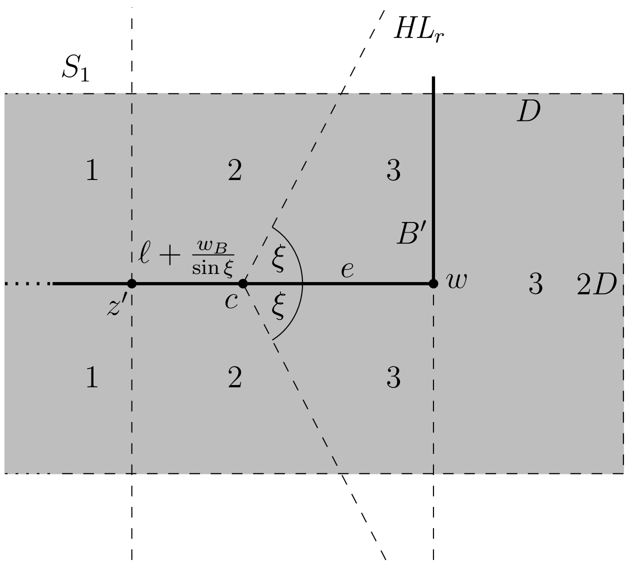

Figure 2.21: The possible regions for v_w.

of two half-lines originating from point c, which lies at distance $\ell + \frac{w_B}{\sin\xi}$ to the right of z'. The angles of these half lines with the X-axis are also equal to ξ, and HL_r is the right part of $R_{b,t}$'s boundary. If v_w lies in region 1 or 2, we are done, since z' is F-approximated by v_w. Otherwise, v_u and v_w lie on opposite sides of the region between HL_l and HL_r, as shown in Figure 2.22.

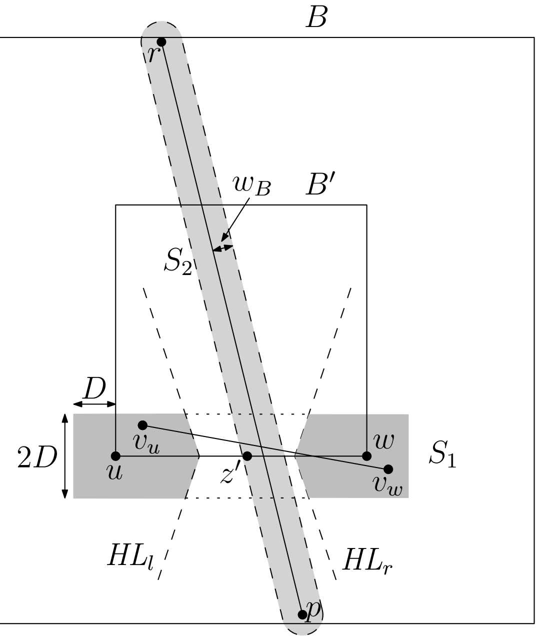

Figure 2.22: The segment pr passes between the two wedges containing v_u and v_w.

Let S_2 denote the set of points whose distance to segment pr is at most w_B (recall that $2w_B$ is the maximum width of E_2 and $2w_{B'}$ is the maximum width of E_1); see Figure 2.22. Note that S_2 contains ellipse E_2, and that E_1 is contained in the $w_{B'}$-neighbourhood of S_1. By construction of HL_l and HL_r, every point in S_2 lies between HL_l and HL_r. Therefore, we have $\{v_u, v_w\} \cap E_2 = \emptyset$. From the inequalities $p_y \leq -a + \epsilon < -(D + w_{B'})$ and $D + w_{B'} < A + a - \epsilon \leq r_y$ we can derive that the distances from r to S_1 and from p to S_1 are strictly greater than $w_{B'}$. Therefore, neither r nor p can be contained in the $w_{B'}$-neighbourhood of segment $v_u v_w$, which contains ellipse E_1. Hence, $\{p, r\} \cap E_1 = \emptyset$. Moreover, point r lies strictly above S_1 and point p lies strictly below S_1. Therefore, since v_u (v_w) lies strictly to the left (right) of HL_l (HL_r), and segment pr is contained in the region between HL_l and HL_r, the intersection of pr with $v_u v_w$ is some point z, and $z \notin \{v_u, v_w, p, r\}$. Now, according to Definition 2.13, using $E_{b,t} := E_2, v_b := p, v_t := r, E_{l,r} := E_1, v_l := v_u$ and $v_r := v_w$, we obtain that ellipses E_1 and E_2 intersect properly – a contradiction.

We have shown that, for the values of a, A, ϵ and δ chosen at the beginning of this subsection, the set V is an F-cover of the lower horizontal edge of B', for some number

$F < \frac{A-a}{A+a}\epsilon$. Analogous arguments hold for the other edges of B'. This completes the proof of Lemma 2.10.

2.4.2 A Lower Dilation Bound

As a direct consequence of Lemma 2.10 we obtain the main result of this section.

Theorem 2.15. *Let P be a set of 548 points that are evenly placed on the boundary of a square, such that each corner of the square receives a point. Then, the dilation $\Delta(P)$ is at least 1.0000014.*

Proof. Without affecting $\Delta(P)$, we can scale the square containing P such that its side length equals $A + a = 16$. Let $\epsilon := 0.0588$, and e be an arbitrary edge of the square. There are two points of P placed at the endpoints of e, and $(548 - 4)/4 = 136$ points placed in between.

The distance of two consecutive points of P on e is $16/137$, so P is an $8/137$-cover for e, and thus for the whole boundary of the square. Now the theorem follows from Lemma 2.9 and the fact that $8/137 < \epsilon$. □

Furthermore, we observe that Lemmas 2.8 and 2.10 and the fact that dilation is invariant under scaling imply the following corollary.

Corollary 2.16. *Let $\alpha > 0$ be a constant. The dilation $\Delta(P)$ of any point set P is least* 1.0000014*, if P is an* $0.0588 \cdot \alpha$*-cover for the boundary of a square whose side length is* $16 \cdot \alpha$.

2.5 The Existence of General Lower Bounds

In the previous section we have established a class of a finite point set P whose dilation $\Delta(P)$ is strictly greater than 1.

Now we want to generalize this result in the following way. Suppose that S is an arbitrary finite point set in the plane. Then one of two cases applies. Either, S is special, in that it happens to be contained in the vertex set of a triangulation of dilation 1; see Figure 2.2. Or, there exists a lower dilation bound > 1, depending on S, that cannot be beaten by any finite plane graph (V, E) where $P \subseteq V$.

The rest of this section is devoted to the proof of this fact. We need the following two definitions.

Definition 2.17. *Given a finite point set P in the plane, we obtain the* complete graph *on P by connecting each pair of points $p, q \in P$ by a straight edge iff the line segment pq contains no points in P, except p and q.*

Definition 2.18. *For a finite point set P, we define $P^0 := P$ and P^{k+1} as the union of P^k and all crossing points in the complete graph of P^k.*

In Section 2.4 we have made use of the fact that the vertices of a triangulation (V, E) of sufficiently low dilation approximate the set P^1 and P^2, for any $P \subseteq V$. The following lemma generalizes this fact to arbitrary sets P^k.

Lemma 2.19. *For every finite point set P, every natural number k, and every $\epsilon > 0$, there exists a bound $\delta > 1$ such that the following holds. If $T = (V, E)$ is a finite triangulation of dilation $\delta(T) \leq \delta$ such that V contains P, then V is an ϵ-cover for P^k.*

Proof. We prove the lemma for a fixed point set P, using induction on the parameter k.

Induction base: $k = 0$: For any $\epsilon > 0$, since the vertex set V contains P by assumption, V is an ϵ-cover for the set $P^0 = P$, regardless of the dilation of T.

Induction step: $k > 0$: We assume that the lemma holds for the values $0, \ldots, k-1$. Hence, for any fixed value $\epsilon > 0$, there exists a bound $\delta_{k-1} > 1$ such that if T is a finite triangulation whose dilation is at most δ_{k-1}, and T contains $P = P^0$ in its vertex set, then every point in the set P^{k-1} is ϵ-approximated by some vertex of T. Now if we find a constant $\delta_k > 1$ such that all the points in the set $P^k \setminus P^{k-1}$ are ϵ-approximated by the vertices of any finite triangulation T of dilation $\leq \delta_k$, given that T contains $P = P^0$ in its vertex set, then setting $\delta := \min\{\delta_{k-1}, \delta_k\} > 1$ proves the lemma.

We consider the set of lines $\mathcal{L}$ containing (at least) two points in the set P^{k-1}, and we denote with Π the set of pairs $\{L_1, L_2\} \subseteq \mathcal{L}$ of distinct lines whose intersection is a point $z \in P^k \setminus P^{k-1}$. We shall define δ_k using the following constants.

$$diam(P) := \max_{p,q \in P} |pq|$$

is the diameter of P,

$$\alpha := \min_{\{L_1, L_2\} \in \Pi} \alpha(L_1, L_2)$$

is the minimum angle $\alpha(L_1, L_2)$ formed by a pair of intersecting lines $L_1, L_2 \in \mathcal{L}$, and,

$$\rho := \min_{p \in P^{k-1}} \left\{ \min_{L \in \mathcal{L} \wedge p \notin L} d(p, L) \right\}$$

the smallest distance $d(p, L)$ from any point $p \in P^{k-1}$ to its closest line in $\mathcal{L}$ not containing p. Note that the constants α and ρ are both strictly greater than 0.

With the notation from above, we now define δ_k based on the values α, ρ and $diam(P)$ as follows. By the induction hypothesis there exists a constant $\delta^* > 1$ such that any given finite triangulation T containing P in its vertex set also ϵ^*-approximates every point in the set P^{k-1} by a vertex of T, where $\epsilon \geq \epsilon^* = \min\{\frac{\rho}{3}, \epsilon \cdot \frac{\sin(\alpha)}{3}\} > 0$. We will now show that setting

$$\delta_k := \min \left\{ \delta^*, \sqrt{\left(\frac{\epsilon^*}{2\epsilon^* + diam(P)} \right)^2 + 1} \right\} > 1.$$

proves the correctness of the lemma.

As before, let T be any finite triangulation of dilation δ_k that contains P in its vertex set. Because $\delta_k \leq \delta^*$ holds, every point in the set P^{k-1} is ϵ-approximated by some vertex of T. It remains to show that any point z in the set $P^k \setminus P^{k-1}$ is also ϵ-approximated by some vertex of T. To this end, we chose a pair of lines $\{L_1, L_2\} \in \Pi$ and four points $p_1, q_1, p_2, q_2 \in P^{k-1}$ such that point z is the (unique) intersection point of the two segments $s_1 = p_1q_1 \subset L_1$ and $s_2 = p_2q_2 \subset L_2$.

First, we observe that since the ϵ^*-approximations $v_{p_1}, v_{q_1} \in V$ of the points p_1, q_1 are at distance at most $\epsilon^* \leq \frac{\rho}{3}$ to p_1, q_1, respectively, the distances from v_{p_1} and v_{q_1} to line L_2 are at least $\rho - \frac{\rho}{3} \geq 2\epsilon^*$. Hence, the points v_{p_1} and v_{q_1} lie on different sides of the closed $\frac{3}{2}\epsilon^*$-neighbourhood S_2 of L_2. From the same argument, it follows that the ϵ^*-approximations $v_{p_2}, v_{q_2} \in V$ of the points p_2, q_2, respectively, lie on different sides of the closed $\frac{3}{2}\epsilon^*$-neighbourhood S_1 of line L_1. Together, this implies that segments $v_{p_1}v_{q_1}$ and $v_{p_2}v_{q_2}$ intersect at some point $z' \in S_1 \cap S_2$, and that $\{v_{p_1}, v_{q_1}, v_{p_2}, v_{q_2}\} \cap (S_1 \cap S_2) = \emptyset$ holds.

Next, we consider the ellipse E_1 with foci v_{p_1}, v_{q_1} of diameter $\delta_k \cdot |v_{p_1}v_{q_1}|$. The width of E_1 is at most

$$|v_{p_1}v_{q_1}| \cdot \sqrt{\delta_k^2 - 1} \leq |v_{p_1}v_{q_1}| \cdot \frac{\epsilon^*}{2\epsilon^* + diam(P)} \leq \epsilon^*.$$

Hence, the distance from any point $r \in E_1$ to segment $v_{p_1}v_{q_1}$ is at most $\frac{\epsilon^*}{2}$. Since the segment $v_{p_1}v_{p_2}$ is contained in the closed ϵ^*-neighbourhood of segment p_1q_1, we conclude that the distance from r to segment p_1q_1 is at most $\epsilon^* + \frac{\epsilon^*}{2} = \frac{3}{2}\epsilon^*$. Therefore, ellipse E_1 is contained in the set S_1. Analogously, if we consider the ellipse E_2 with foci v_{p_2}, v_{q_2} of diameter $\delta_k \cdot |v_{p_2}v_{q_2}|$, we obtain that E_2 is contained in the set S_2.

Now, from Lemma 2.14, using $v_b := p_1, v_t := q_1, v_l := p_2, v_r := q_2, \delta := \delta_k, E_{b,t} := E_1, E_{l,r} := E_2, S_{b,t} := S_1$, and $S_{l,r} := S_2$, we obtain that there exists a vertex v of T in the set $S_1 \cap S_2$. By definition of α, the lines L_1 and L_2 form, at their intersection point z, an angle $\alpha' \geq \alpha > 0$. Hence, the distance from z to v is at most $2 \cdot \frac{3}{2}\epsilon^* \cdot \frac{1}{\sin \alpha'} \leq \frac{3\epsilon^*}{\sin \alpha} \leq \epsilon$.

□

Grüne and Kamali [39] have studied topological properties of the sets P^k introduced in Definition 2.18, independent of dilation issues. They obtained the following result.

Theorem 2.20 (Grüne, Kamali [39]). *Let $S = S^0$ be a finite point set in the plane. If S is not special, then there exists a convex polygonal region $K(S)$, such that S^∞ is dense in $K(S)$.*

Lemma 2.21 (Grüne, Kamali [39]). *Let S be a finite point set in the plane. If S is not special, then the region $K(S)$ – compare Theorem 2.20 – contains a triangle whose interior is non-empty.*

A similar density result for the intersection of lines has been shown by Ismailescu and Radoičić [41].

Now we prove the main result of this section, i. e. Theorem 2.4, which states that the dilation $\Delta(S)$ of any point set S that is not special is strictly greater than 1.

Proof of Theorem 2.4. Suppose that S is not special. By Theorem 2.20 and Lemma 2.21 there exists a polygonal region $K(S)$ in which S^∞ lies dense, and $K(S)$ contains a triangle, $T(S)$, whose interior is non-empty. Let B be the boundary of a square contained in $T(S)$. Let (a, A, ϵ, δ) have the values provided in Subsection 2.4.1. We scale the values of (a, A, ϵ) by the same factor, such that B is of size $A + a$; the value of δ remains unchanged. Observe that the values of (a, A, ϵ, δ) remain cover-preserving upon rescaling the values of a, A and ϵ. Then we choose k so large that S^k is an $\epsilon/2$-cover for $K(S)$ and, thus, for B. This is possible, since S^∞ lies dense in $K(S)$. From Lemma 2.19 we obtain a value $\delta_1 > 1$ such that for each finite triangulation $T = (V, E)$ of dilation $\delta(T) < \delta_1$ the following holds. If $S \subseteq V$ then V is an $\epsilon/2$-cover of S^k. Now assume that there exists a finite triangulation $T = (V, E)$ such that $S \subseteq V$ and $\delta(T) < \min(\delta, \delta_1)$, where $\delta = 1.0000014$. By the above, V is an ϵ-cover for B. Thus, Lemma 2.8 implies that V is infinite—a contradiction. Therefore,

$$\Delta(S) \geq \min(\delta, \delta_1) > 1$$

holds. $\square$

Note that, given a finite point set S that is not special, we can compute the set $K(S)$ as described in [39], and scale the values of (a, A, ϵ) provided in Subsection 2.4.1 such that a square B of size $A + a$ fits into the set $K(S)$. Then, after finding a value k where S^k is an $\epsilon/2$-cover for the boundary of B, we can indeed use the construction used in the proof of Lemma 2.19 to find a lower bound $\delta > 1$ on the dilation, $\Delta(S)$, of S.

2.6 Conclusions

We have shown that all non-special point sets S have dilation $\Delta(S) > 1$, and we have provided a concrete lower bound to $\Delta(S)$ for one example set. One big challenge is in computing $\Delta(S)$ for a given point set S. Not even for five points evenly spaced on the unit circle is their dilation known. Another interesting question is if the value of $\Delta(S)$ is always attained by a finite triangulation, or if there are cases where $\Delta(S)$ is just the infimum of all values attainable. Can one prove that Steiner vertices outside the convex hull of S are unnecessary in triangulations of dilation equal to or near $\Delta(S)$? And, finally, how can we efficiently construct such near-to-optimal triangulations? Is it possible to upper bound the number (or, the total length) of edges used? We leave these questions to further research.

Chapter 3

Optimal Transport

We consider the following variant of the well-known Monge-Kantorovich transportation problem. Let S be a set of n point sites in $\mathbb{R}^d$. A bounded set $C \subset \mathbb{R}^d$ is to be distributed among the sites $p \in S$ such that (i) each p receives a subset C_p of prescribed volume and (ii) the average distance of all points z of C from their respective sites p is minimized. In our model, volume is quantified by a measure μ, and the distance between a site p and a point z is given by a function $d_p(z)$. Under quite liberal technical assumptions on C and on the functions $d_p(\cdot)$ we show that a solution of minimum total cost can be obtained by intersecting with C the Voronoi diagram of the sites in S, based on the functions $d_p(\cdot)$ equipped with suitable additive weights. Moreover, this optimum partition is unique, up to sets of measure zero. Unlike the deep analytic methods of classical transportation theory, our proof is based directly on simple geometric arguments. An extended abstract of this paper [35] has appeared at EuroCG'12 [33] and COCOON'12 [34].

3.1 Introduction

In 1781, Gaspard Monge [54] raised the following problem. Given two sets C and S of equal mass in $\mathbb{R}^d$, transport each mass unit of C to a mass unit of S at minimal cost. More formally, given two measures μ and ν, find a map f satisfying $\mu(f^{-1}(\cdot)) = \nu(\cdot)$ that minimizes

$$\int d(z, f(z))\, \mathrm{d}\mu(z),$$

where $d(z, z')$ describes the cost of moving z to z'.

Because of its obvious relevance to economics, and perhaps due to its mathematical challenge, this problem has received a lot of attention. Even with the Euclidean distance as cost function d it is not at all clear in which cases an optimal map f exists. Progress by Appell [2] was honored with a prize by the Academy of Paris in 1887. Kantorovich [42] achieved a breakthrough by solving a relaxed version of Monge's original problem. In 1975, he received a Nobel prize in Economics; see Gangbo and McCann [32] for mathematical and historical details.

While usually known as the *Monge-Kantorovich transportation problem*, the minimum cost of a transportation is sometimes called *Wasserstein metric* or, in computer science, *earth mover's distance* between the two measures μ and ν. It can be used to measure similarity in image retrieval; see Rubner et al. [59]. If both measures μ and ν have finite support, Monge's problem becomes the minimum weight matching problem for complete bipartite graphs, where edge weights represent transportation cost; see Rote [58], Vaidya [63] and Sharathkumar and Agarwal. [60].

We are interested in the case where only measure ν has finite support. More precisely, we assume that a set S of n point sites p_i is given, and numbers $\lambda_i > 0$ whose sum equals 1. A body C of volume 1 must be split into subsets C_i of volume λ_i in such a way that the total cost of transporting, for each i, all points of C_i to their site p_i becomes a minimum. In this setting, volume is measured by some measure μ, and transport cost $d(z, z')$ by some measure of distance.

Gangbo and McCann [32] report on the cases where either $d(z, z') = h(z - z')$ with a strictly convex function h, or $d(z, z') = l(|z - z'|)$ with a non-negative, strictly concave function l of the Euclidean norm. As a consequence of deep results on the general Monge-Kantorovich problem, they prove a surprising fact. The minimum cost partition of C is given by the additively weighted Voronoi diagram of the sites p_i, based on cost function d and some additive weights w_i. In this structure, the Voronoi region of p_i contains all points z satisfying

$$d(p_i, z) - w_i \ < \ d(p_j, z) - w_j \text{ for all } j \neq i.$$

Villani [64] has more recently observed that this holds even for more general cost functions, that need not be invariant under translations, and in the case where both distributions are continuous. The existence of the weights w_i is proved in a non-constructive way.

Figure 3.1 depicts how to obtain this structure in dimension $d = 2$. For each point site $p \in S$ we construct in $\mathbb{R}^3$ the cone $\{(z, d(p, z) - w_p) \mid z \in \mathbb{R}^2\}$. The lower envelope of the cones, projected onto the XY–plane, results in the Voronoi diagram. Independently, Aurenhammer et al. [5] have studied the case where $d(z, z') = |z - z'|^2$.

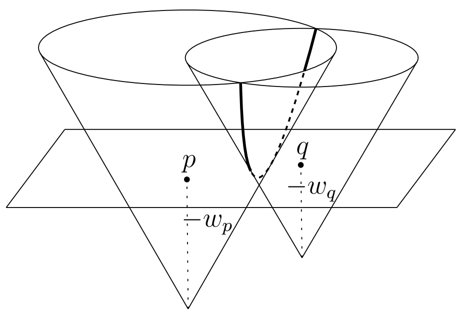

Figure 3.1: An additively weighted Voronoi diagram as the lower envelope of cones.

Here, a *power diagram* (see [6, 7]) gives an optimum splitting of C. In addition to structural results, they provide algorithms for computing the weights w_i. Regarding the case where the transportation cost from point z to site p_i is given by individual cost functions $d_{p_i}(z)$, it is still unknown how to compute the weights w_i.

In this paper we consider the situation where the cost $d(p_i, z)$ of transporting point z to site p_i is given by individual distance functions $d_{p_i}(z)$. We require that the weighted Voronoi diagram based on the functions $d_{p_i}(\cdot)$ is well-behaved, in the following sense. Bisectors are $(d-1)$–dimensional, and increasing weight w_i causes the bisectors of p_i to sweep d–space in a continuous way. These requirements are fulfilled for the Euclidean metric and, at least in dimension 2, if the distance function $d_p(z) = d(z-p)$ assigned to each site p is a translate of the same strictly convex distance function $d(\cdot)$.

We show that these assumptions are strong enough to obtain the results of [32] and [5]: The weighted Voronoi diagram based on the functions $d_{p_i}(\cdot)$ optimally solves the transportation problem, if weights are suitably chosen. Whereas [32] derives this fact from a more general measure-theoretic theorem, our proof uses the minimization of a quadratic objective function and geometric arguments. The purpose of our paper is to show that such a simple proof is possible.

After stating some definitions in Section 3.2 we generalize arguments from [5] to prove, in Section 3.3, that C can be split into parts of arbitrary given volumes by a weighted Voronoi diagram, for a suitable choice of weights. Then, in Section 3.4, we show that such a partition is optimal, and unique. In Section 3.5 we show that if C is connected, these weights are uniquely determined, up to addition of a constant.

3.2 Definitions

Let μ be a measure defined for all Lebesgue-measurable subsets of $\mathbb{R}^d$. We assume that μ and the d-dimensional Lebesgue measure are mutually continuous, that is, μ vanishes exactly on the sets of Lebesgue measure zero.

Let S denote a set of n point sites in $\mathbb{R}^d$. For each $p \in S$ we are given a continuous function

$$d_p \colon \mathbb{R}^d \to \mathbb{R}_{\geq 0}$$

that assigns to each point z of $\mathbb{R}^d$ a non-negative value $d_p(z)$ as the "distance" from site p to z.

For $p \neq q \in S$ and $\gamma \in \mathbb{R}$ we define the *bisector*

$$B_\gamma(p,q) := \{\, z \in \mathbb{R}^d \mid d_p(z) - d_q(z) = \gamma \,\}$$

and the *region*

$$R_\gamma(p,q) := \{\, z \in \mathbb{R}^d \mid d_p(z) - d_q(z) < \gamma \,\}.$$

The sets $R_\gamma(p,q)$ are open and increase with γ. Now let us assume that for each site $p \in S$ an additive weight w_p is given, and let

$$w = (w_1, w_2, \ldots, w_n)$$

denote the vector of all weights, according to some ordering $p_1, p_2, \ldots$ of S. Then, $B_{w_i - w_j}(p_i, p_j)$ is called the *additively weighted bisector* of p_i and p_j, and

$$\mathrm{VR}_w(p_i, S) := \bigcap_{j \neq i} R_{w_i - w_j}(p_i, p_j)$$

is the *additively weighted Voronoi region* of p_i with respect to S. It consists of all points z for which $d_{p_i}(z) - w_i$ is smaller than all values $d_{p_j}(z) - w_j$, by definition. As usual,

$$V_w(S) := \mathbb{R}^d \setminus \bigcup_i \mathrm{VR}_w(p_i, S)$$

is called the *additively weighted Voronoi diagram* of S; see [6]. Clearly, $\mathrm{VR}_w(p_i, S)$ and $V_w(S)$ do not change if the same constant is added to all weights in w. Increasing a single value w_i will increase the size of p_i's Voronoi region.

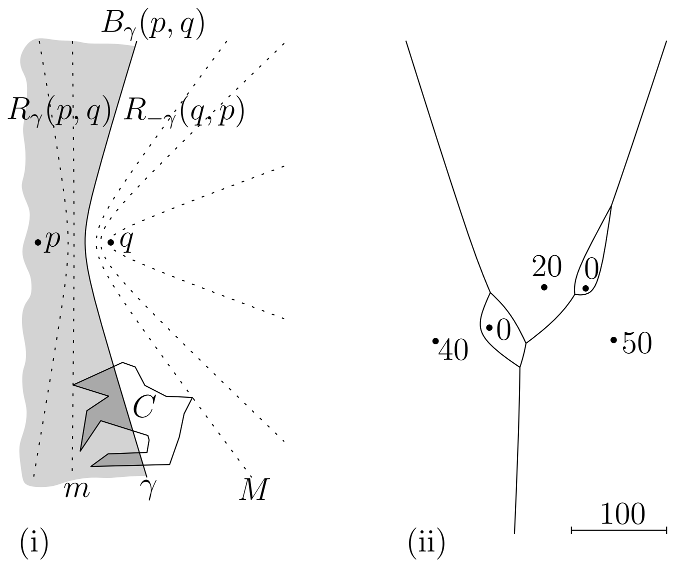

Figure 3.2: (i) Sets $R_\gamma(p, q)$ for increasing values of γ. (ii) An additively weighted Voronoi diagram.

Example. Let $d = 2$ and $d_p(z) = |p - z|$ the Euclidean distance. Given weights w_p, w_q, the bisector $B_{w_p - w_q}(p, q)$ is a line if $w_p = w_q$, and a branch of a hyperbola otherwise. Figure 3.2(i) shows how $B_\gamma(p, q)$ sweeps across the plane as γ grows from $-\infty$ to ∞. In fact, when $|\gamma| = |p - q|$, the bisector degenerates to a ray, and for larger $|\gamma|$, the bisectors are empty. The bisector $B_\gamma(p, q)$ forms the boundary of the set $R_\gamma(p, q)$, which increases with γ. Each bounded set C in the plane will be reached at some point. From then on the volume of $C \cap R_\gamma(p, q)$ is continuously growing until all of C is contained in $R_\gamma(p, q)$. Given n points p_j with additive weights w_j, raising the value of a single weight w_i will cause all sets $R_{w_i-w_j}(p_i,p_j)$ to grow until C is fully contained in the Voronoi region $V_w(p_i,S)$ of p_i.

Figure 3.2(ii) shows an additively weighted Voronoi diagram $V_w(S)$ based on the Euclidean distance. It partitions the plane into 5 two-dimensional cells (Voronoi regions), and consists of 9 cells of dimension 1 (Voronoi edges) and of 5 cells of dimension 0 (Voronoi vertices). Each cell is homeomorphic to an open ball of appropriate dimension.

The next definition generalizes the above properties to the setting used in this paper.

Definition 3.1. *A system of continuous distance functions $d_p(\cdot)$, where $p \in S$, is called* admissible *if for all $p \neq q \in S$, and for each bounded open set $C \subset \mathbb{R}^d$, there exist values m_{pq} and M_{pq}, $m_{pq} < M_{pq}$, such that $\gamma \mapsto \mu\left(C \cap R_\gamma(p,q)\right)$ is continuously increasing from* 0 *to $\mu(C)$ as γ grows from m_{pq} to M_{pq}. Furthermore, $C \cap R_\gamma(p,q) = \emptyset$ if $\gamma \leq m_{pq}$ and $C \subset R_\gamma(p,q)$ if $M_{pq} \leq \gamma$.*

By symmetry we can assume w.l.o.g. that $m_{qp} = -M_{pq}$ and $M_{qp} = -m_{pq}$; see Figure 3.2(i). We will need the following structural property, which essentially says that bisectors have measure 0.

Lemma 3.2. *For a system of admissible distance functions $d_p(\cdot)$, where $p \in S$, for any two points $p \neq q \in S$ and any $\gamma \in \mathbb{R}$, and for any bounded open set $C \subset \mathbb{R}^d$, we have $\mu\left(C \cap B_\gamma(p,q)\right) = 0$.*

Proof. By definition of $R_\gamma(p,q)$ and $B_\gamma(p,q)$, for all $\varepsilon > 0$ and $\gamma \in \mathbb{R}$ the following inequality holds:

$$\mu(C \cap R_{\gamma+\varepsilon}(p,q)) \geq \mu(C \cap R_\gamma(p,q)) + \mu(C \cap B_\gamma(p,q)).$$

We let ε decrease to 0. Since, according to Definition 3.1, the function $\gamma \longmapsto \mu\left(C \cap R_\gamma(p,q)\right)$ is continuous, we get

$$\lim_{\varepsilon \searrow 0} \left(\mu(C \cap R_{\gamma+\varepsilon}(p,q))\right) = \mu(C \cap R_\gamma(p,q)).$$

This implies $\mu(C \cap B_\gamma(p,q)) = 0$. $\square$

3.3 Partitions of Prescribed Size

The following theorem shows that we can use an additively weighted Voronoi diagram based on an admissible system of distance functions d_p to partition C into subsets of prescribed sizes.

Theorem 3.3. *Let $n \geq 2$ and let $d_{p_i}(\cdot)$, $1 \leq i \leq n$, be an admissible system as in Definition 3.1. Let C be a bounded open subset of $\mathbb{R}^d$. Suppose we are given n*

real numbers $\lambda_i > 0$ with $\lambda_1 + \lambda_2 + \cdots + \lambda_n = \mu(C)$. Then there is a weight vector $w = (w_1, w_2, \ldots, w_n)$ such that

$$\mu(C \cap \mathrm{VR}_w(p_i, S)) = \lambda_i$$

holds for $1 \le i \le n$.

If, moreover, C is pathwise connected then w is unique up to addition of a constant to all w_i, and the parts $C_i = C \cap \mathrm{VR}_w(p_i, S)$ of this partition are unique.

Proof. The function

$$\Phi(w) := \sum_{i=1}^{n} \big(\mu(C \cap \mathrm{VR}_w(p_i, S)) - \lambda_i\big)^2$$

measures how far w is from fulfilling the theorem. Since each function $\gamma \mapsto \mu(C \cap R_\gamma(p_i, p_j))$ is continuous by Definition 3.1, and because

$$|\mu(C \cap \mathrm{VR}_w(p_i, S)) - \mu(C \cap \mathrm{VR}_{w'}(p_i, S))| \le \\ \sum_{j \ne i} |\mu(C \cap R_{w_i - w_j}(p_i, p_j)) - \mu(C \cap R_{w'_i - w'_j}(p_i, p_j))|,$$

we conclude that the function Φ is continuous, too.

Let $D := \max\{\, M_{pq} \mid p \ne q \,\}$ with M_{pq} as in Definition 3.1. Note that $m_{pq} < M_{pq}$ and our assumption $m_{pq} = -M_{qp}$ together imply that $D > 0$ is a positive constant. On the compact set $[0, D]^n$, the function Φ attains its minimum value at some argument w. If $\Phi(w) = 0$ we are done. Suppose that $\Phi(w) > 0$. Since the volumes of the Voronoi regions inside C add up to $\mu(C)$, there must be some sites p_j whose Voronoi regions have too large an intersection with C, i. e., of volume strictly larger than λ_j, while other regions' intersections with C are too small.

We now show that by decreasing the weights of some points of S we can decrease the value of Φ. For any point $p_i \in S$ we consider the continuous "excess" function

$$\phi_{p_i}(w) := \mu(C \cap \mathrm{VR}_w(p_i, S)) - \lambda_i$$

that indicates by which amount of volume the region of p_i is too large or too small. We have

$$\sum_{i=1}^{n} \phi_{p_i}(w) = 0.$$

Let $T \subseteq S$ be the set of points p_i whose excess $\phi_{p_i}(w)$ equals $\tau := \max_{s \in S} \phi_s(w)$. By assumption, T is neither empty nor equal to S. We will reduce the weights of the points in T by some amount δ, which is the same for all points in T, obtaining a new weight vector w'. Note that, by construction, $\mathrm{VR}_w(t, S) \supseteq \mathrm{VR}_{w'}(t, S)$ for any site $t \in T$, and $\mathrm{VR}_w(s, S) \subseteq \mathrm{VR}_{w'}(s, S)$, for any site $s \in S \setminus T$.

Let $w'(\delta)$ denote the weight vector $(w'_1, \ldots, w'_n)$ where $w'_i := w_i - \delta$ if $p_i \in T$, and $w'_i := w_i$ otherwise. The volume of the set of points of C, that is assigned by $V_w(S)$ to some point in T and by $V_{w'(\delta)}(S)$ to some point in $S \setminus T$ is

$$\operatorname{Loss}(\delta) := \sum_{t \in T} \big(\phi_t(w) - \phi_t(w'(\delta))\big)$$

Let $\tau := \max_{p \in S} \phi_p(w) > 0$ and $\tau' := \max_{s \in S \setminus T} \phi_s(w) < \tau$. Our goal is to choose δ in such a way that

$$\operatorname{Loss}(\delta) = \frac{\tau - \tau'}{2n} \tag{3.1}$$

and to show that this choice decreases Φ strictly. Since $\operatorname{Loss}(0) = 0$ and $\operatorname{Loss}(\delta)$ is a monotone and continuous function, our aim is to solve (3.1) by the intermediate value theorem. For all $\delta > 2D$ we observe that $\operatorname{Loss}(\delta) = \sum_{p_i \in T} \mu(C \cap \mathrm{VR}_w(p_i, S)) = \sum_{p_i \in T} \phi_{p_i} + \lambda_i = \sum_{p_i \in T} \tau + \lambda_i > \tau$ holds, since in this case the Voronoi region $\mathrm{VR}_{w'(\delta)}(p_i, S)$ of every point $p_i \in T$ contains no point of C. Now $0 < \frac{\tau - \tau'}{2n} < \tau$ implies that we can indeed solve (3.1) by the intermediate value theorem. The latter inequality is evident if $\tau' \geq 0$. Otherwise, it follows from $-\tau' \leq -\sum_{s \in S \setminus T} \phi_s(w) = \sum_{s \in T} \phi_s(w) = |T| \cdot \tau \leq (n-1)\tau$.

In the following let $w' := w'(\delta)$, for short. In order to prove the theorem we use the following reformulation of the statement $\Phi(w) > \Phi(w')$:

$$\sum_{t \in T} \big(\phi_t(w)^2 - \phi_t(w')^2\big) > \sum_{s \in S \setminus T} \big(\phi_s(w')^2 - \phi_s(w)^2\big) \tag{3.2}$$

Now, in order to prove inequality (3.2), we give an upper bound on the right-hand side of the inequality, which will be compared with a lower bound on the left-hand side of the inequality. Let $\varepsilon_s \geq 0$ denote the volume increase, in C, of the region of site $s \in S \setminus T$. By the choice of δ, we have $\sum_{s \in S \setminus T} \varepsilon_s = \frac{\tau - \tau'}{2n}$, and

$$\begin{aligned}
\sum_{s \in S \setminus T} \big(\phi_s(w')^2 - \phi_s(w)^2\big) &= \sum_{s \in S \setminus T} \big((\phi_s(w) + \varepsilon_s)^2 - \phi_s(w)^2\big) \\
&= \sum_{s \in S \setminus T} 2\varepsilon_s \phi_s(w) + \sum_{s \in S \setminus T} \varepsilon_s^2 \\
&\leq 2\tau' \sum_{s \in S \setminus T} \varepsilon_s + \Big(\sum_{s \in S \setminus T} \varepsilon_s\Big)^2 \\
&= 2\tau' \frac{\tau - \tau'}{2n} + \left(\frac{\tau - \tau'}{2n}\right)^2 .
\end{aligned}$$

We now give a lower bound on the left-hand side of (3.2). If $\varepsilon_t \geq 0$ denotes the volume decrease, in C, of the region of site $t \in T$, then the corresponding calculation for the points in T yields:

$$\begin{aligned}\sum_{t\in T}\bigl(\phi_t(w)^2 - \phi_t(w')^2\bigr) &= \sum_{t\in T}\bigl(\phi_t(w)^2 - (\phi_t(w)-\varepsilon_t)^2\bigr)\\ &= \sum_{t\in T} 2\phi_t(w)\varepsilon_t - \sum_{t\in T}\varepsilon_t^2\\ &\geq 2\tau\sum_{t\in T}\varepsilon_t - \Bigl(\sum_{t\in T}\varepsilon_t\Bigr)^2\\ &= 2\tau\frac{\tau-\tau'}{2n} - \left(\frac{\tau-\tau'}{2n}\right)^2\end{aligned}$$

where we have used the same arguments as above and also that $\phi_t(w) = \tau$, by definition, for all $t \in T$. It is easy to verify the inequality

$$2\tau\frac{\tau-\tau'}{2n} - \left(\frac{\tau-\tau'}{2n}\right)^2 > 2\tau'\frac{\tau-\tau'}{2n} + \left(\frac{\tau-\tau'}{2n}\right)^2$$

for $\tau - \tau' \neq 0$ and $n \geq 2$. Hence inequality (3.2) holds.

We have now shown the existence of a weight vector w' with $\Phi(w') < \Phi(w)$. If w' belongs to $[0, D]^n$, we are done. Otherwise, suppose the maximum weight w'_m occurs for some point p'_m. We add $D - w'_m$ to all weights. This does not change the Voronoi diagram defined by w', but now the maximum weight w'_m is exactly D. Next we observe that if now the weight of some point p_j is negative, then its Voronoi region contains no point of C, because C is contained in the set $R_{w'_m - w'_j}(p_m, p_j)$, by definition of D. Assigning weight 0 to each of those points also results in empty Voronoi regions, in C, for those points. Altogether, we now have $0 \leq w'_i \leq D$ for all $p_i \in S$.

We have shown the existence of a weight vector $w' \in [0, D]^n$ that satisfies $\Phi(w') < \Phi(w)$. This contradicts the minimality of $\Phi(w)$. Hence the minimum value of Φ is 0.

The uniqueness of the partition will be discussed in Section 3.5. $\square$

3.4 Optimality

Again, let $d_p(\cdot)$, $p \in S = \{p_1, p_2, \ldots, p_n\}$, be an admissible system as in Definition 3.1, and let C denote a bounded and open subset of $\mathbb{R}^d$. Moreover, we are given real numbers $\lambda_i > 0$ whose sum equals $\mu(C)$.

By Theorem 3.3 there exists a weight vector $w = (w_1, w_2, \ldots, w_n)$ satisfying

$$\mu(C \cap \mathrm{VR}_w(p_i, S)) = \lambda_i \text{ for } i = 1, \ldots, n.$$

Now we prove that this subdivision of C minimizes the transportation cost, i. e., the average distance from each point to its site. It is convenient to describe partitions of C by μ-measurable maps $f\colon C \to S$ where, for each $p \in S$, $f^{-1}(p)$ denotes the region assigned to p. Let F_Λ denote the set of those maps f satisfying $\mu(f^{-1}(p_i)) = \lambda_i$ for $i = 1, \ldots, n$.

Theorem 3.4. *The partition of C into regions $C_i := C \cap \mathrm{VR}_w(p_i, S)$ minimizes*

$$\mathrm{cost}(f) := \int_C d_{f(z)}(z) \,\mathrm{d}\mu(z)$$

over all maps $f \in F_\Lambda$. Any other partition of minimal cost differs at most by sets of measure zero from $(C_i)_{1\le i\le n}$.

If the measure μ was a discrete measure concentrated on a finite set C of points, the optimum partition problem would turn into the classical transportation problem, a special type of network flow problem on a bipartite graph. The weights w_i would be obtained as dual variables. The following proof mimics the optimality proof for the discrete case.

Proof. Let f_w be defined by $f_w(C_i) = \{p_i\}$ for all i, and $f \in F_\Lambda$ be another map. The cost of map f is

$$\begin{aligned}
\mathrm{cost}(f) &= \int_C \left(d_{f(z)}(z) - w_{f(z)} + w_{f(z)}\right) \mathrm{d}\mu(z) \\
&= \int_C \left(d_{f(z)}(z) - w_{f(z)}\right) \mathrm{d}\mu(z) + \int_C w_{f(z)} \,\mathrm{d}\mu(z) && (3.3) \\
&\ge \int_C \left(d_{f_w(z)}(z) - w_{f_w(z)}\right) \mathrm{d}\mu(z) + \int_C w_{f(z)} \,\mathrm{d}\mu(z) && (3.4) \\
&= \mathrm{cost}(f_w) - \int_C w_{f_w(z)} \,\mathrm{d}\mu(z) + \int_C w_{f(z)} \,\mathrm{d}\mu(z) && (3.5)
\end{aligned}$$

where the inequality between lines (3.3) and (3.4) follows from the fact that

$$d_{f_w(z)}(z) - w_{f_w(z)} \le d_{f(z)}(z) - w_{f(z)}, \tag{3.6}$$

by the definition of additively weighted Voronoi diagrams. Here we have assumed for simplicity that the map f_w assigns every point $p \in V_w(S)$ to *some* weighted nearest neighbor of p in S, according to some tie-break rule. Since Lemma 3.2 implies $\mu(C \cap V_w(S)) = 0$, changing the assignment of those points has no influence on $\mathrm{cost}(f_w)$.

Both maps f and f_w partition C into regions of volume λ_i, $1 \le i \le n$, i. e., $\mu(f_w^{-1}(p_i)) = \mu(f^{-1}(p_i)) = \lambda_i$ for all $p_i \in S$. Therefore,

$$\int_C w_{f(z)} \,\mathrm{d}\mu(z) = \sum_{p_i \in S} \int_{f^{-1}(p_i)} w_{f(z)} \,\mathrm{d}\mu(z) = \sum_{p_i \in S} \lambda_i w_i,$$

and the same value is obtained for f_w:

$$\int_C w_{f_w(z)} \,\mathrm{d}\mu(z) = \sum_{p_i \in S} \int_{f_w^{-1}(p_i)} w_{f_w(z)} \,\mathrm{d}\mu(z) = \sum_{p_i \in S} \lambda_i w_i.$$

Now the optimality claim follows from (3.5).

We still have to prove uniqueness of the solution, up to a set of measure 0. Assume that f differs from f_w on some set G of positive measure. We are done if we show that the inequality between (3.3) and (3.4) holds as a strict inequality. We can remove the points of $V_w(S)$ from G, since they have measure 0. Then (3.6) holds as a strict inequality on G. The difference between (3.3) and (3.4) is then the integral of the positive function $g(z) := c(z, f(z)) - w_{f(z)} - \big(c(z, f_w(z)) - w_{f_w(z)}\big)$ over a set G of positive measure. Such an integral $\int_{z\in G} g(z)\,\mathrm{d}\mu(z)$ has always a positive value: the countably many sets $\{\, z \in G \mid \frac{1}{k} > g(z) \geq \frac{1}{k+1} \,\}$ and $\{\, z \in G \mid g(z) \geq 1 \,\}$ partition G, and thus their measures add up to $\mu(G)$. Therefore, at least one of these sets has positive measure, and it follows directly that the integral is positive. □

3.5 Uniqueness

In contradistinction to Theorem 3.4, the weight vector w and the resulting weighted Voronoi partition in Theorem 3.3 are not unique: if C is disconnected, changing w may move the bisectors between different connected components of C without affecting the partition, or only changing it in boundary points of C.

However, uniqueness can be obtained under the additional assumption that C is pathwise connected. Then in Theorem 3.3, the weight vector $w = (w_1, \ldots, w_n)$ that partitions C into regions of prescribed size is unique up to addition of the same constant to all w_i. Consequently, the parts in the weighted Voronoi partition outside of C are also uniquely determined.

The proof uses the following lemma.

Lemma 3.5. *Assume that C is path-connected, in addition to being an open bounded set. Consider a partition of the point set S into two nonempty sets T and $S \setminus T$. Suppose, that, for a given weight vector w, all regions $C \cap VR_w(p, S)$ have positive measure.*

If we increase the weight of every point in T by the same constant $\delta > 0$, then $\mu(C \cap VR_w(t, S))$ increases, for some $t \in T$.

Proof. In this proof, we will omit the reference to the set S from the Voronoi regions and simply write $\mathrm{VR}_w(p)$ instead of $\mathrm{VR}_w(p, S)$, since the point set S is fixed.

It is clear that the regions $\mathrm{VR}_w(t)$ for $t \in T$ can only grow, in the sense of gaining new points, but we have to show that this growth results in a strict increase of $\mu(C \cap \mathrm{VR}_w(t))$ for some $t \in T$.
For any weight vector w, consider the continuous function

$$d_{T,w}(z) := \min\{\, d_p(z) - w_p \mid p \in T \,\}$$

and the function $d_{S\setminus T,w}(z)$, which is defined analogously, and define

$$\mathrm{VR}_w(T) := \{\, z \in \mathbb{R}^d \mid d_{T,w}(z) < d_{S\setminus T,w}(z) \,\}$$
$$\mathrm{VR}_w(S \setminus T) := \{\, z \in \mathbb{R}^d \mid d_{T,w}(z) > d_{S\setminus T,w}(z) \,\}$$

These two sets partition $\mathbb{R}^d$, apart from some "generalized bisector" where equality holds.

Roughly, $\mathrm{VR}_w(T)$ is obtained by merging all Voronoi regions $\mathrm{VR}_w(t)$ for $t \in T$ into one set, together with some bisecting boundaries between them. Since these boundaries have zero measure inside C, we have

$$\mu(C \cap \mathrm{VR}_w(T)) = \sum_{t \in T} \mu(C \cap \mathrm{VR}_w(t)) \tag{3.7}$$

Let $w = (w_1, \ldots, w_n)$ and $w' = (w'_1, \ldots, w'_n)$, where $w'_i := w_i + \delta$ if $s_i \in T$ and $w'_i := w_i$ otherwise. It suffices to show that in going from w to w', $\mu(C \cap \mathrm{VR}_w(T))$ increases, since then, by (3.7), one of the constituent Voronoi regions $\mu(C \cap \mathrm{VR}_w(t))$ must increase. As mentioned, the set $C \cap \mathrm{VR}_w(T)$ can only grow, but we have to show that this growth results in a strict increase of μ. Note that by the definition of w and w', both equalities $d_{T,w}(z) - \delta = d_{T,w'}(z)$ and $d_{S \setminus T,w}(z) = d_{S \setminus T,w'}(z)$ are fulfilled.

If $C \cap \mathrm{VR}_{w'}(S \setminus T) = \emptyset$, we are done because $\mu(C \cap \mathrm{VR}_{w'}(T))$ has increased to $\mu(C)$. Otherwise, consider a path in C connecting some point $p_1 \in \mathrm{VR}_w(T)$ with some point $p_2 \in \mathrm{VR}_{w'}(S \setminus T)$. We have, by definition, $d_{T,w}(p_1) < d_{S \setminus T,w}(p_1)$ and $d_{T,w}(p_2) - \delta = d_{T,w'}(p_2) > d_{S \setminus T,w'}(p_2) = d_{S \setminus T,w}(p_2)$. Hence, by the intermediate value theorem, we can find a point p on the path with $d_{T,w}(p) - \delta/2 = d_{S \setminus T,w}(p)$. This point is an interior point of $C \cap \mathrm{VR}_w(S \setminus T)$ and $C \cap \mathrm{VR}_{w'}(T)$, and hence there is a neighborhood of p which adds a positive measure to $C \cap \mathrm{VR}_w(T)$ when going from w to w'. □

To finish the proof of uniqueness in Theorem 3.3, suppose that two weight vectors w, w' produce the desired measures λ_i for the regions, but $w' - w \neq (c, c, \ldots, c)$ for any constant c. We may assume, by adding a constant to all entries, that $w \leq w'$, and $w_i = w'_i$ for some i. We will now gradually change w into w' by modifying its entries. Let $T_w := \{\, s_i \in S \mid w'_i - w_i = \max_j (w'_j - w_j) \,\}$. By assumption, T_w is neither empty nor equal to S. We increase the weights of the points in T_w by the same amount δ which is the same for all points in T_w, leaving the remaining values fixed. The amount δ is chosen in such a way that $w'_i - (w_i + \delta)$ for $i \in T_w$ equals the maximum of the remaining differences $w'_j - w_j$ for $j \in S \setminus T_w$. By Lemma 3.5, the measure of the region assigned to some point $t_0 \in T_w$ strictly increases in this process. After increasing the weights of the points in T_w by δ, the set T_w of points where $w'_i - w_i$ achieves its maximum is enlarged, and the whole process is repeated until $w = w'$. During this process, t_0 will always be among the points whose weight is increased, and therefore, the measure of its region can never shrink back to the original value—a contradiction. □

3.6 Conclusion and Future Work

We have given a geometric proof for the fact that additively weighted Voronoi diagrams can optimally solve some cases of the Monge-Kantorovich transportation problem, where one measure has finite support. Surprisingly, the existence of an optimal solution—the main mathematical challenge in the general case—is an easy consequence of our proof. In remains to be seen to what extent our assumptions on the distance functions d_{p_i} can be further generalized.

Chapter 4

Visibility amongst two simple polygonal chains in the plane

Introduction

We consider two visibility problems, given two non-intersecting open or closed simple polygonal chains P and Q in the plane. The *visibility region* of P w. r. t. Q is the union of all points visible from P, where Q is considered an obstacle. We present an optimal linear time algorithm for computing this area. As has already been observed by Ghosh [36], the visibility of a polygon P contained in a polygon Q can be computed in linear time, using an algorithm that traverses shortest path trees. Lacking the knowledge of this algorithm's existence, we rediscovered this algorithm when designing an algorithm for computing the *visibility area* between two polygons, as described below. While Ghosh devised his algorithm in the context of visibility from a convex set, our approach is slightly different. Our first goal was to extend an algorithm by Guibas et al. [40] that computes the visibility region of a boundary edge, in order to compute the visibility region of a connected boundary chain. We then observed that using this algorithm we are also able to compute the visibility of a polygon P contained in a polygon Q – obtaining essentially the same algorithm as Ghosh. In addition, we show how to compute the (outer or inner) visibility of a polygon P w. r. t. an obstacle polygon Q, without restrictions on the position of P relative to Q (*i. e.*, P needs not be contained in Q).

The *visibility area* between P and Q is the union of all line segments pq where p lies on (the boundary of) P and q on (the boundary of) Q and pq does neither intersect with P nor with Q. We also present an optimal linear time algorithm for computing this area. The given problem arises in the context of a digital three dimensional building construction model. An efficient computation of the visibility area between two wall axes A and B in the digital model of a building is an important question for computer aided construction management.

Computing the visibility among geometric objects is one of the most natural subjects in the field of computational geometry. Furthermore, efficient solutions of visibility problems have application in many fields of computer science such as robotics [49], computer graphics [17] and computer vision [29]. For an overview of efficient solutions for visibility problems in the plane one can consider the survey of Asano et al. [4] or the textbook of Ghosh [37].

This paper can be divided into two parts. In the first part we discuss a visibility problem in the plane, given two non-intersecting open or closed simple polygonal chains P and Q. The *visibility region* (or *visibility polygon*) of Q is the set of points visible from Q, see Figure 4.1.

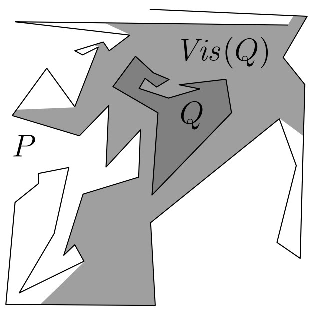

Figure 4.1: The (outer) visibility region $Vis(Q)$ of polygon Q w. r. t. polygonal chain P.

A point x is visible from Q iff there exists a point $q \in Q$ such that the line segment qx does not intersect P. If Q is (part of) a closed polygonal chain C, then we can also consider the *inner visibility region* $Vis_C(Q)$ of all points inside the region bounded by C, visible from Q.

In the second part we also discuss a visibility problem in the plane, given two non-intersecting closed or open simple polygonal chains P and Q. The problem stems from an application in construction industry. More precisely, the problem arises in the digital model of a building. For managing the construction process it is necessary to compute an *essential* part between two wall axes in the digital three dimensional representation of a building, see [62]. E.g. a construction schedule could contain an activity for constructing all columns in the second building storey which are positioned *between* two axes A and B. In addition, for large scale building elements which are constructed in several parts, axes are used to outline the construction sections. E.g. section II.2 of a floor slab may be defined as the part between two axes A and B, see Figure 4.2.

Furthermore, currently the building industry is shifting from the document based to a model based design approach where design data is managed within a digital three dimensional building information model and data processing is being automated as much as possible, see [20, 21]. In this context spatial queries based on sections would be an equivalent tool for linking and extracting design data as well as for sectioning the model. A first step in automatically evaluating spatial references based on axes is to calculate the polygon which describes the boundary of the area between two axes.

Altogether, given two closed or open simple polygonal chains P and Q it is necessary

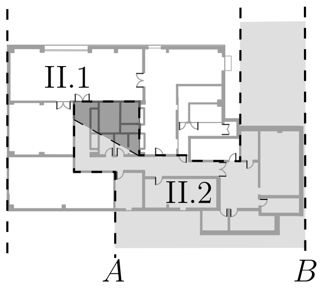

Figure 4.2: Section II.2 is given by two axes A and B in a digital building model.

to compute the visible part *between* P and Q efficiently. This means that if ∂P denotes the boundary of P, we would like to compute the union of all line segments pq with $p \in \partial P$ and $q \in \partial Q$, so that pq does neither intersect with P nor with Q. Note, that pq might of course *touch* P and/or Q. We can consider arbitrary non-intersecting open or closed simple polygonal chains P and Q and compute the visibility area in the sense described above, see Figure 4.3.

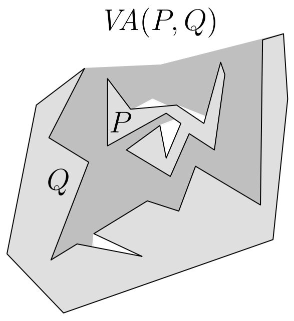

Figure 4.3: The *visibility area* $VA(P, Q)$ between polygon P and polygon Q.

The paper is organized as follows. In section 4.2 we show how to compute the inner visibility region of a connected boundary subchain Q of a simple polygon P in linear time, extending a known algorithm by Guibas et al. [40] that computes the visibility of a boundary edge of P. In section 4.3 we further generalize the algorithm, allowing two disjoint simple closed polygonal chains P and Q as input, while maintaining linear running time. In section 4.5 we describe how to compute, also in linear time, the visibility area between two simple polygons. The latter two algorithms can be modified to also accept open polygonal chains as input. An extended abstract of this work has appeared at EuroCG [48].

4.1 Definitions and Preliminaries

Throughout this paper we only consider simple polygonal chains in the plane, i.e. polygonal chains without self-intersections. A closed simple polygonal chain P is called a polygon, and we also denote with P the bounded region enclosed by the chain. In that case we may also refer to the polygonal chain as the boundary ∂P of P. The number of vertices of an open or closed polygonal chain P is denoted with $|P|$. Given two points p and q we denote with pq the line segment connecting p with q. We denote the *convex hull* of a point set X by $ch(X)$, and given a simple polygon P, each connected component of $ch(P) \setminus P$ is called a pocket P'. The edge e that belongs to the boundary of P', but not to the boundary of P is the edge *defining* P'. A subchain C of a polygonal chain P is a connected subset of P, unless we allow explicitly that C has a finite number of connected components. Given a polygonal boundary subchain $Q \subseteq \partial P$ of a polygon P, then the *inner* visibility region $Vis_P(Q)$ of Q in P is the set of points $\{p \in P \mid \exists q \in Q : pq \textit{ does not intersect with } \partial P\}$. In general, if P is a simple polygon and $Q \subset P$ is a set of points, then we denote with $Vis_P(Q)$ the visibility polygon of Q in P, i.e. the set of points

$$Vis_P(Q) = \{p \in P \,|\, \exists q \in Q : \, pq \in P\}.$$

We will also refer to a connected component of the points in polygon P that are *not* visible from Q as a pocket P'. If the boundary part of P' that does not belong to the boundary of P is a line segment c, then c is called a visibility cut.

Given two disjoint polygonal chains P and Q the visibility polygon $Vis(Q)$ of Q is the set of points $\{p \mid \exists q \in Q : \; pq \textit{ does not intersect with } P\}$. If the line segment pq intersects no polygonal (boundary) chain, then we say q sees p, and the segment pq is also called a *visibility segment.* Let X, Y be two point sets. Then a visibility segment uv is of *type* XY if one endpoint, u, belongs to X and the other, v, belongs to Y. In most cases we will consider $X = P$ and $Y = Q$ the two polygonal chains of the scene. A ray originating from a point v in direction of a point $u \neq v$ is denoted with $\rho_v(u)$. A ray passes tangential vertices or edges, but ends where it would intersect one of the polygonal chains. Given a simple polygon P, then the unique oriented shortest path in P from $u \in P$ to $v \in P$ is denoted with $\pi_{u,v}$. The *shortest path tree* rooted at point $u \in P$ is the union of all shortest paths from u to any vertex of P. Finally we assume general position of the vertices of the polygonal chains. That is, no three vertices are collinear.

The visibility area $VA(P, Q)$ between two disjoint simple polygonal chains P and Q is defined as the union of all segments of type PQ. If P and Q are both subchains of the boundary of a simple polygon R, then we restrict the visibility area to the points in R that lie on a segment of type PQ.

4.2 Inner Visibility of a Boundary Subchain

Given a simple polygon P, we want to compute the inner visibility polygon of a boundary subchain Q of P. Our first result on inner visibility of a polygonal chain is the following.

Theorem 4.1. *The inner visibility region of a connected boundary subchain Q of a simple polygon P can be computed in optimal time $O(|P|)$.*

To prove Theorem 4.1, we modify an algorithm by Guibas et al. [40] for computing the inner visibility region of a boundary edge of a simple polygon. A similar algorithm has been previously published by Ghosh. In his paper [36] he describes how to use the algorithm for computing the visibility polygon of a simple polygon Q which is contained in another polygon P.

Theorem 4.2 (Ghosh [36])**.** *The outer visibility region of a simple polygon Q inside a simple polygon P can be computed in optimal time $O(|P| + |Q|)$.*

We prove Theorem 4.1 in this section. In the following Section 4.3, we give a proof for Theorem 4.2 and also show how to compute the (inner or outer) visibility region of any simple polygon Q in the presence of one obstacle polygon P (inside or outside of Q) in optimal time $O(|P| + |Q|)$.

4.2.1 The Algorithm of Guibas et al.

We first briefly sketch the algorithm of Guibas et al. [40] for a better understanding of our extension to the algorithm. Given a boundary edge e of polygon P, the algorithm computes the visibility polygon of e in P. Any point $p \in P$ *not* visible from e lies *behind* a vertex v of P that is visible from e. It suffices to compute the set of vertices of P that are visible from e. At each such vertex v a simple test decides if some parts of P not visible from P lie behind v. This test can be performed, for each vertex v, in constant time using the information of the two shortest path trees rooted at the two endpoints e_l, e_r of edge e. Furthermore, connecting subsequent visible parts of ∂P with line segments, the boundary of the inner visibility region of edge e can be constructed in linear time $O(|P|)$.

If e_l is the vertex of ∂P adjacent to e_r, in clockwise direction along ∂P, then Vertex v is visible from e iff either, (1) v sees exactly one point of e, namely e_l *or* e_r. Or, (2) walking on the oriented shortest path $\pi_{e_l,v}$ from e_l to v we always turn left at the inner vertices *and* walking on the oriented shortest path $\pi_{e_r,v}$ from e_r to v we always turn right at the inner vertices. We denote such a pair $\pi_{e_l,v}$ and $\pi_{e_r,v}$ of paths an outwards convex pair of paths.

How to test if a vertex v of P meets condition (1) or (2) above? Clearly, condition (1) can be checked by testing if the father of v is e_l or e_r in either of the two shortest path trees. In order to test condition (2) for all vertices of P one traversal of the two shortest path trees suffices. In the shortest path tree rooted at e_l no right turns are allowed, whereas in the shortest path tree rooted at e_r no left turns are allowed.

A first observation is that conditions (1) and (2) together are equivalent to the following condition: A vertex v of P is visible from edge $e = (e_l, e_r)$ iff $\pi_{e_l,v}$ makes no right turn at any vertex $w \notin e$ of P, and $\pi_{e_r,v}$ makes no left turn at any vertex $w' \notin e$ of P. We will see that, given the clockwise order of the vertices $(q_l, \ldots, q_r)$ of a boundary chain Q of P, a vertex $v \notin Q$ of P is visible from Q iff the shortet path $\pi_{q'_l,v}$ in P makes no right turn at any vertex $w \notin Q$, and the shortest path $\pi_{q'_r,v}$ in P makes no left turn at any vertex $w' \notin Q$. Here, q'_l (q'_r) denotes the *last* vertex of the oriented shortest path $\pi_{q_l,v}$ ($\pi_{q_r,v}$) in P, that belongs to Q. Again, by a single traversal of the two shortest path trees rooted at q_l and q_r, respectively, we can decide which vertices of P are visible from Q. Connecting successive boundary parts of P visible from Q with a line segment we obtain the visibility polygon of Q in P.

4.2.2 Inner Visibility of an Outwards Bent Boundary Subchain

We now describe how the algorithm of Guibas et al. [40] can be applied to compute the visibility polygon of a boundary subchain Q (of polygon P) that is *bent outwards*, i.e. the interior angles at the the inner vertices of Q are $> \pi$. Visibility condition (2) defined above is no longer correct if we consider the inner visibility polygon of a polygonal boundary chain $Q = (e_l, \ldots, e_r)$ that is bent outwards, instead of a boundary edge e of P. Given a vertex p of P, $p \notin Q$, it is possible that the shortest paths $\pi_{e_l,p}$ and $\pi_{e_r,p}$ start with a part of Q, and the full paths are no longer outwards convex, although p is visible from some part of Q, see vertex x in Figure 4.4 for an example. In order to test if p is visible from Q we consider the two shortest paths $\pi_{e_l,p}$ and $\pi_{e_r,p}$.

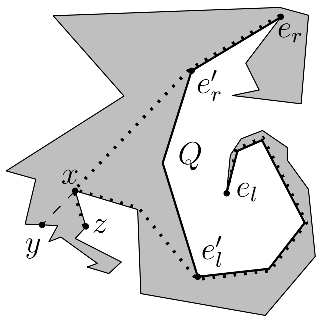

Figure 4.4: The parts behind the segment xy are cut off when reaching x in the shortest path tree rooted at e_r.

Let e'_l, e'_r denote the vertex of Q where $\pi_{e_l,p}$, $\pi_{e_r,p}$, leaves Q, respectively. Clearly, vertex p cannot be seen from any point q of Q between e_l and e'_l or between e_r and e'_r: If p were visible from q, then the shortest path $\pi_{r,q}$ would be a line segment. But $\pi_{r,q}$ either makes a right turn at vertex e'_r, or a left turn at vertex e'_l. We now show that p is visible from Q iff the two paths $\pi_{e'_l,p}$ and $\pi_{e'_r,p}$ are outwards convex.

Lemma 4.3. *With the notation from above, point p is visible from Q iff the two paths $\pi_{e'_l,p}$ and $\pi_{e'_r,p}$ are outwards convex.*

Proof. First, p is clearly visible from Q if the two paths $\pi_{e'_l,p}$ and $\pi_{e'_r,p}$ are outwards convex. Second, let q be a point of Q that sees p. The oriented ray $\rho_p(q)$ divides P in two simple subpolygons (if $\rho_p(q)$ lies tangential to a vertex v of P, one of the two subpolygons is a *weakly simple* polygon[1]). Now $\pi_{e'_l,p}$ is fully contained inside the subpolygon P_l to the left side of the oriented ray $\rho_q(p)$, and $\pi_{e'_r,p}$ is fully contained inside the subpolygon P_r to the right. It is well known that the vertices of shortest paths inside (weakly) simple polygons are reflex vertices of the polygon. The only reflex vertices of P_l (P_r) to the right (left) side of $\pi_{e'_l,p}$ ($\pi_{e'_r,p}$) belong to Q. This implies that the pair $\pi_{e'_l,p}, \pi_{e'_r,p}$ of shortest paths is outwards convex, because $\pi_{e_l,p}, \pi_{e_r,p}$ leaves Q at e'_l, e'_r, respectively, and never returns to Q. $\square$

Now we know how to test if a vertex p of P is not visible from Q: we have to test if $\pi_{e_l,p}$ ($\pi_{e_r,p}$) turns right (left) at some vertex $w \notin Q$. This information can be preprocessed for each vertex p of P by traversing the two shortest path trees, rooted at e_l and e_r, respectively, in P. Then, at each vertex p visible from Q the same test as used by the algorithm of Guibas et al. in [40] decides if some points not visible from Q lie behind p. Removing those points from P is also done as in [40], by connecting subsequent visible parts of ∂P with line segments. Note that the shortest path trees can be computed in linear time [40] after triangulating P, also in linear time [13]. Thus from the algorithm of Guibas et al. we obtain the following corollary.

Corollary 4.4. *The inner visibility region $Vis_P(Q)$ of an outwards bent boundary subchain Q of a polygon P can be computed in optimal time $O(|P|)$.*

We should remark that Lemma 4.3 is also one of the key observations in the paper of Ghosh [36] for computing the visibility from one polygon inside another polygon, and that the same proof has already been established by Ghosh.

4.2.3 Inner Visibility of Arbitrary Boundary Chains

Now we are able to prove our main result, Theorem 4.1, on inner visibility of arbitrary connected boundary subchains. Let Q be a boundary subchain of a simple polygon P. The shortest path Q_π in P between the endpoints of Q divides P into subpolygons, compare Figure 4.5. First, there are *right subpolygons* R_i, $1 \leq i \leq k$, in sequential order along Q_π, to the right of Q_π. The boundaries of the right subpolygons are defined by Q_π and Q. Second, there are *left subpolygons* L_j, $1 \leq j \leq m$, in sequential order along Q_π, to the left of Q_π, whose boundaries are defined by Q_π and $Q^c = \partial P \setminus Q$. The left and right subpolygons are simple polygons, except where an edge of Q_π is also an edge of ∂P, compare R_2 in Figure 4.5. By Lemma 4.5 below, every point $p \in R_i$ inside a right subpolygon R_i is visible from Q, and a point $q \in L_j$ inside a left subpolygon L_j is visible from Q iff q is visible from the outwards bent boundary chain $L_j(Q_\pi) := Q_\pi \cap \partial L_j$ of L_j that also belongs to Q_π. Before proving Lemma 4.5 we discuss its consequences.

[1]By general position of the vertices of P, there can be at most one tangential vertex v of P.

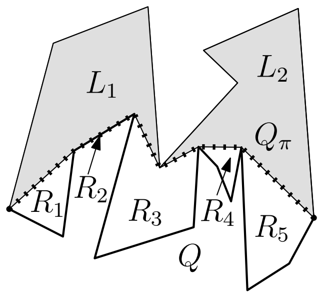

Figure 4.5: The polygon P is divided into subpolygons R_i and L_j by the shortest path Q_π, between the endpoints of the boundary chain Q.

After triangulating P in linear time [13], we compute the shortest path Q_π between the endpoints of Q in linear time [40]. Knowing which vertices of Q and of $Q^c = \partial P \setminus Q$ belong to Q_π we divide P into left and right subpolygons. The inner visibility region of $L_j(Q_\pi)$ in L_j can, by Corollary 4.4, be computed time $O(|L_j|)$. Each vertex of P belongs to at most two left subpolygons. Hence, computing the visibility polygons inside all left subpolygons is done in time $O(|L_1| + \cdots + |L_m|) \in O(|P|)$. We create a copy of Q_π and attach to it the above visibility polygons, e.g. in order of appearance along Q_π. Now replacing the copy of Q_π by a copy of Q we obtain, by Lemma 4.5, the visibility region of Q in P. This completes the proof of Theorem 4.1.

It remains to formulate and to prove Lemma 4.5. For each right subpolygon R_i let $R_i(Q_\pi)$, $R_i(Q)$ denote the part of Q_π, Q, respectively that is also part of ∂R_i. Boundary chains $L_j(Q_\pi)$ and $L_j(Q^c)$ are defined analogously for each left subpolygon L_j.

Lemma 4.5. *With the notation from above, let $P' = R_i$ or $P' = L_j$ be an arbitrary right or left subpolygon of P, then $P'(Q_\pi)$ is an outwards bent boundary chain of P'. Every point $r \in R_i$ is visible from $R_i(Q)$. Every point $l \in L_j$ is visible from Q iff l is visible from $L_j(Q_\pi)$.*

Proof. The following facts imply the correctness of the Lemma. Let R_i, L_j denote two subpolygons of P, and let $r \in R_i$, $l \in L_j$ be two points of P.

1. $P'(Q_\pi)$ is an outwards bent boundary chain of P':
   By symmetry, it suffices to consider the case where $P' = R_i$ is a right subpolygon of P. We orientate Q_π such that the vertices of Q lie to the right of Q_π and the vertices of Q^c lie to the left of Q_π. The fact that the inner vertics of Q_π are reflex vertices of P implies that Q_π turns left at every vertex of Q^c. The endpoints u, v of $R_i(Q_\pi)$ belong to Q, while the inner vertices of $R_i(Q_\pi)$, between u, v, belong to Q^c. Hence $R_i(Q_\pi)$ is an outwards bent boundary chain of $P' = R_i$.

2. r is visible from $R_i(Q)$:
   Let T be a triangulation of R_i, and t be a triangle in T that contains r. Since the boundary chain $R_i(Q_\pi)$ of R_i is bent outwards, at most two corners of triangle t

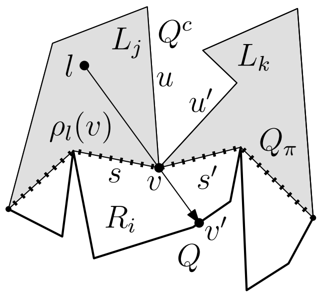

Figure 4.6: The four segments s, s', u, u' at vertex v.

belong to $R_i(Q_\pi)$, so at least one corner v of t belongs to $R_i(Q)$. Thus r is visible from $v \in R_i(Q)$.

3. If l is visible from $r \in R_i(Q)$ then l is also visible from $L_j(Q_\pi)$:
If $r \notin L_j$, then the visibility segment rl must intersect ∂L_j at some point $l' \in L_i(Q_\pi)$, which implies that l is visible from $L_j(Q_\pi)$. If otherwise $r \in L_j$, we have $r \in R_i \cap L_j \subseteq Q_\pi$, hence $r \in L_j(Q_\pi)$ sees point l.

4. If l is visible from $L_j(Q_\pi)$ then l is also visible from Q:
Suppose there is a path-segment $s \in L_j(Q_\pi)$ of Q_π whose endpoints are collinear with l, and l sees a point in s. Either s contains a vertex v of Q, or s is the only segment of $L_j(Q_\pi)$, which implies $l \in s$. In both cases l is visible from Q – either from vertex v or because $l \in s$ also belongs to some right subpolygon R_i, which implies that l is visible from $R_i(Q)$.

   Otherwise, if there is no segment $s \in L_j(Q_\pi)$ collinear to l where l sees a point of s, then we consider an arbitrary point v of some segment $s \in L_j(Q_\pi)$ where a point of s is visible from l. If $v \in Q$ we are done, so now we assume $v \notin Q$. Then, $v \notin Q$ implies that there is exactly one right subpolygon that contains v, which we denote with R_i. It is a well-known fact that the intersection of two shortest paths inside a simple polygon has at most one connected component. Booth Q_π and $\rho_l(v)$ are shortest paths in P. Thus if $\rho_l(v)$ intersects Q_π at v, after entering R_i, $\rho_l(v)$ must intersect the boundary of R_i at some point $v' \notin Q_\pi$, which implies that l is visible from $v' \in Q$. Since, by assumption, s and l are not collinear, if follows that if v is an interior point of segment s, then $\rho_l(v)$ intersects Q_π at v. Hence l is seen from some point $v' \in Q$. It now suffices to show that if v is a vertex of Q^c, then $\rho_l(v)$ intersects Q_π at v. This can be seen as follows. Because $v \in Q^c$ is not an endpoint of Q_π, there is another path-segment $s' \in Q_\pi$ adjacent to v. Since v is also a vertex of polygon P, there are two boundary edges u, u' of Q^c adjacent to v, see Figure 4.6.

   Let s, u denote the segments that belong to the boundary of L_j. Then, segments s', u' belong to the boundary of L_k, where $k \in \{j-1, j+1\}$. The segment lv is contained in the wedge formed by the segments s and u. Since the interior angle (between segments s and s') at vertex v, in subpolygon R_i, is $> \pi$, $\rho_l(v)$ must intersect Q_π at vertex v.

□

4.3 Visibility of a Polygon in Presence of One Obstacle

This section is devoted to the proof of Theorem 4.6 about the visibility region of a closed polygonal chain. Let P and Q be two disjoint simple closed polygonal chains. As before, $|C|$ denotes the number of vertices of a polygonal chain C.

Theorem 4.6. *With the notation from above, the (outer or inner) visibility region of Q w. r. t. P can be computed in optimal time $O(|Q| + |P|)$.*

We have to distinguish between different cases, depending on the relative position of P to Q. Note that we can compute the convex hull $ch(P)$, $ch(Q)$, $ch(P \cup Q)$ of P, Q and $P \cup Q$ in linear time [53, 61], and also check if one of the polygons is contained in the other polygon, also in linear time, assuming that the boundary chains do not intersect [45, p. 89]. Analogously, if one polygon is contained in a pocket of the other polygon, then the boundary of the pocket can be identified in linear time. In particular, we are able to identify in linear time which of the following cases applies.

4.3.1 Neither $P \subset ch(Q)$ nor $Q \subset ch(P)$

The two outer tangents to P and Q, which can be identified in linear time [53, 61], divide the region outside of P and Q into one unbounded region R and one bounded region $\overline{R}$. Once the tangents are known, computing the visibility region of Q in the unbounded region R is a trivial task. By Theorem 4.1 we are able to compute the visibility region of Q in polygon $\overline{R}$ in linear time. We conclude that we can compute $Vis(Q) = Vis_R(Q) \cup Vis_{\overline{R}}(Q)$ in linear time $O(|P| + |Q|)$.

4.3.2 P contained in $ch(Q)$

For the sake of coherent notation, in the following we always consider a polygon P that is contained in a polygon Q, compare Figures 4.7 and 4.8.

There are four cases, each of which is considered in a separate part in the remainder of this section:

1. The outer visibility region of P, where $P \subset Q$ and the boundary of polygon Q is considered an obstacle, compare Figure 4.7.

2. The outer visibility region of P, where Q is the pocket (containing P) of some polygon $\overline{Q}$, see Figure 4.8. This case is identical to case 1, except that the edge e defining Q, which belongs to the boundary of Q and the convex hull of $\overline{Q}$, is not considered an obstacle for P.

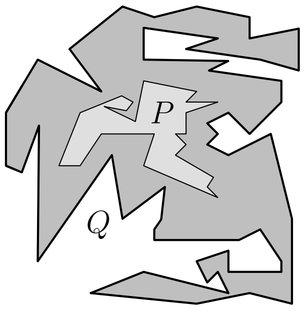

Figure 4.7: Polygon P contained in polygon Q.

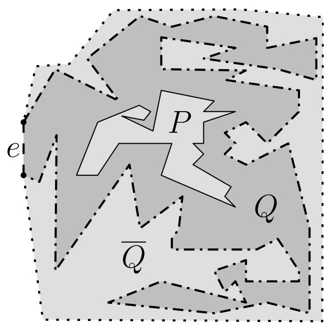

Figure 4.8: Polygon P contained in a pocket Q of polygon $\overline{Q}$. Dashed (dotted / dash dotted) boundary edges belong to Q ($\overline{Q}$ / the common boundary of Q and $\overline{Q}$).

3. The inner visibility region of polygon Q, where polygon $P \subset Q$ is considered an obstacle, see Figure 4.7.

4. The outer visibility region of $\overline{Q}$, where Q is the pocket (containing P) of some polygon $\overline{Q}$.
   This case is similar to case 2, but we are interested in the outer visibility polygon of polygon $\overline{Q}$. Therefore, the visibility from edge e (see 2.) in Q does not contribute to the visibility of $\overline{Q}$.

We now discuss cases 1 and 2 which cover the visibility region of polygon P. Thereafter, we analyze cases 3 and 4 which relate to the visibility region of polygon Q.

As mentioned before, Ghosh [36] has independently devised an algorithm similar to ours for computing the visibility polygon of a polygon P contained in a polygon Q (case 1 described above). We show that this approach can also be applied for

- the computation of the inner visibility region of a boundary chain Q of a polygon P (see Section 4.2).
- Cases 2, 3 and 4 described above.

The Visibility Region of Polygon P

Case 1: Polygon P contained in polygon Q

In order to compute the outer visibility region of polygon P contained in polygon Q, we consider the two vertical tangents t_1, t_2 to P. The tangents divide the polygon $R = Q \setminus P$ into subpolygons, see Figure 4.9.

Formally, the division of polygon R into subpolygons is as follows. Let s denote the segment between the highest point $p \in P$ on t_1 and the lowest point $q \in Q$ on t_1, above p. Similarly, t denotes the segment between the lowest point $p' \in P$ on t_1 and

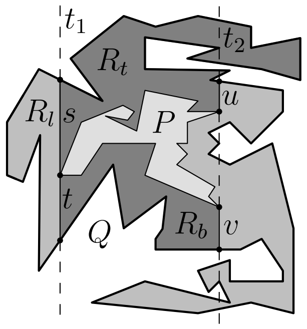

Figure 4.9: Polygon Q containing polygon P, and the division of $R = Q \setminus P$ into subpolygons.

the highest point $q' \in Q$ on t_1, below p'. Segments u, v on t_2 are defined analogously. The four segments s, t, u, v divide the polygon $R = Q \setminus P$ into a top, bottom, left and right polygon R_t, R_b, R_l and R_r, respectively. Finally, let $R_x(P), R_x(Q)$ denote the boundary parts of P, Q that belong to R_x, for any $x \in \{t, b, l, r\}$.

We now consider the polygon $\overline{R_l} = R_t \cup R_r \cup R_b$. Polygon $\overline{R_l}$ is obtained from R by connecting the boundaries of P and Q with the two segments s and t. The polygon $\overline{R_l}$ is a simple polygon (allowing that s and t have one endpoint in common). We compute the visibility polygon of boundary chain $R_t(P)R_r(P)R_b(P)$ in $\overline{R_l}$, by Theorem 4.1 in linear time. Clearly, no visibility segment in R between points $r \in \overline{R_l}$ and $p \in \partial P$ can intersect s or t, because in that case the segment pr must also intersect $R_l(Q)$. We conclude that we have correctly identified all points of $\overline{R_l}$ visible from P. Segments u and v are visible from P, so we can separate the points of R_r visible from P from those of R_t and R_b visible from P by adding u and v to the boundary of the visibility polygon $Vis_{\overline{R_l}}(P) = Vis_{\overline{R_l}}(R_t(P)R_r(P)R_b(P))$. Analogously, we can identify the points of R_l visible from P. It remains to connect the boundaries of the four visibility polygons of P in R_t, R_b, R_l and R_r at the endpoints of segments s, t, u and v. As result we obtain, in linear time $O(|P| + |Q|)$, the visibility region of P in Q.

Case 2: Polygon P contained in pocket Q of polygon $\overline{\mathbf{Q}}$

We want to compute the outer visibility region of polygon P, where P is contained in a pocket Q of a polygon $\overline{Q}$. There is one edge e of ∂Q that does not belong to $\partial\overline{Q}$, namely the edge e that defines Q. Let w. l. o. g. e be a vertical edge. Computing the visibility region of P w. r. t. $\overline{Q}$ is similar to the previous case $P \subset Q$. The only new aspect is that edge e, which also belongs to the boundary of $ch(Q)$, does not block the view of P. We start with computing the visibility polygon of P in Q, as described above. Only points in the half-plane $H^+(e)$, defined by the line supporting e, that does *not* contain P can possibly contain points outside Q visible from P. This set of visible points is either empty or bounded by (edge e and) two visibility cuts[2] of infinite length, originating from e. Suppose some part of edge e is visible from P. The two visibility cuts can be easily identified after computing the visibility polygon of P in Q. If an endpoint a of edge e is not visible from P then we have to extend the visibility cut separating a from P in Q to infinite length. Otherwise if a is visible from P, then we create a visibility cut starting at a, like we would at any other reflex vertex of Q that blocks the view of P on some other points. Proceeding analogously with the other endpoint, b, of edge e, all points between the new infinite visibility cuts are visible from P.

The Visibility Region of Polygon Q

Case 3: Polygon P contained in polygon Q

We now compute the inner visibility region of Q, assuming $P \subset Q$ holds. Remember the division of polygon $R = Q \setminus P$ into subpolygons by the two vertical tangents to P, as shown in Figure 4.9. We start with the visibility polygon of Q in R_r. If some point $r \in \overline{R_r} = R_t \cup R_l \cup R_b$ sees a point $p \in R_r$, then the segment pr intersects either segment u or segment v. Hence, ray $\rho_r(p)$ must reach a point q of $R_r(Q)$ behind p, so p is also visible from $R_r(Q)$. This implies that any point in R_r visible from some point $r \in R_t(Q) \cup R_l(Q) \cup R_b(Q)$ is also visible from $R_r(Q)$. Hence, the visibility polygon of Q in R_r equals the visibility polygon of $R_r(Q)$ in R_r. From the same argument, it follows that if some point $r \in R_t$ sees a point of segment s or segment u, then r is seen from Q. This implies that the visibility polygon of Q in R_t equals the visibility polygon of $sR_t(Q)u$ in R_t. Analogous arguments hold for the visibility polygons of Q in R_l and R_b.

Case 4: Polygon P contained in pocket Q of polygon $\overline{\mathbf{Q}}$

We now show how to compute the outer visibility region of polygon $\overline{Q}$, where the obstacle polygon P is contained in a pocket Q of $\overline{Q}$. Knowing which pocket of $\overline{Q}$ contains P the only non-trivial part is computing the set of points inside Q visible from $\overline{Q}$. This case is similar to computing the inner visibility polygon of Q; yet we have to keep in mind that now the visibility of edge e, that defines pocket Q, does not contribute to the outer visibility region of $\overline{Q}$. Remember that $e \in \partial Q$ is also part of the boundary of both polygons $ch(Q)$ and $ch(\overline{Q})$. Again, we assume that e is vertical, and we consider the division of polygon $R = Q \setminus P$ into subpolygons R_t, R_b, R_l and R_r, as before. We assume that R_r does *not* contain e on its boundary. Otherwise we start with R_l, and analogous arguments hold. We compute the visibility polygon of $\partial Q \setminus e$ in R_r by computing the visibility polygon of $R_r(Q)$ in R_r (compare Case 3: Polygon P contained in polygon Q). It remains to compute the visibility polygon of $\partial Q \setminus e$ in polygon $\overline{R_r} = R_t \cup R_l \cup R_b$. Removing edge $e \notin \{u, v\}$ from boundary

[2]A *visibility cut* is a line segment (or half-line) that separates the points visible from P and some points that are not visible from P.

chain $C_Q = uR_t(Q)R_l(Q)R_b(Q)v$ of $\overline{R_r}$, we obtain two polygonal chains Q_1 and Q_2. Let Q_1 be the chain containing v and Q_2 be the one that contains u, as depicted in Figure 4.10. Again, since a point $r \in \overline{R_r}$ is visible from $R_r(Q)$ iff r is visible from u or v, it suffices to unite the visibility polygons $Vis_{\overline{R_r}}(Q_1)$ and $Vis_{\overline{R_r}}(Q_2)$ of Q_1 and Q_2 respectively in $\overline{R_r}$, in order to compute the visibility polygon of $\partial Q \setminus e$ in $\overline{R_r}$. We will successively cut off from polygon $\overline{R_r}$ all points not visible from both Q_1 and Q_2.

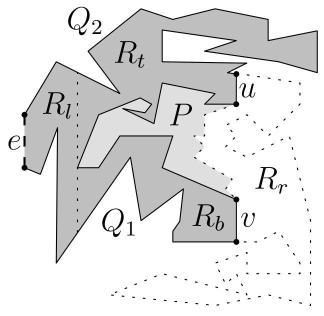

Figure 4.10: Removing edge e from the boundary of polygon $\overline{R_r}$, we obtain two boundary chains, Q_1 and Q_2, belonging to Q.

We now give a formal definition of the term *visibility cut*. As mentioned before, the inner visibility polygon of a boundary subchain C of a simple polygon P can be obtained by connecting consecutive visible boundary parts of P with line segments. If $C = (c_l, \ldots, c_r)$ and vertex $v \notin C$ is visible from C, then the two shortest paths $\pi_{c'_l,v}$ and $\pi_{c'_r,v}$ are outwards convex, where c'_l (c'_r) is the vertex of C on path $\pi_{c_l,v}$ ($\pi_{c_r,v}$) closest to v. Let the two boundary edges of P adjacent to v be denoted by p_1v and p_2v, and let the path-segment of $\pi_{c_l,v}$ and $\pi_{c_r,v}$ adjacent to v be s_lv and s_rv, respectively. If vertex v blocks the visibility of C on some points in P, then v must be a reflex vertex; see Figure 4.11. Furthermore, both segments vp_1 and vp_2 lie to the left (or to the

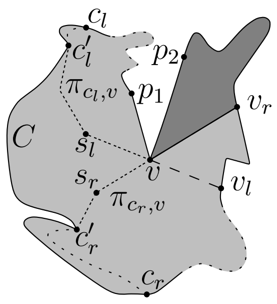

Figure 4.11: The visibility cut vv_r defined by boundary vertex v, and the pocket of cut vv_r (in dark grey colour). To simplify illustration, not all boundary parts are displayed explicitly – *e. g.* some vertices defining $\pi_{c_l,v}$ and $\pi_{c_r,v}$ were omitted.

right) of the two lines supporting s_lv and s_rv. Thus rays $\rho_{s_l}(v)$ and $\rho_{s_r}(v)$ intersect the boundary of P at some point v_l and v_r, respectively, behind v. If segments p_1v and p_2v lie to the left of the lines supporting s_lv and s_rv, then segment vv_r is called *visibility cut.* Otherwise (if the two segments lie to the right), segment vv_l is a *visibility cut.* The set of points *behind* vertex v not visible from C are called the *pocket* of the cut. If c is a visibility cut, then we denote the pocket of c by P_c.

We are now able to describe how to identify the points of polygon $\overline{R_r}$ that are not visible from both Q_1 and Q_2. The boundary of $Vis_{\overline{R_r}}(Q_1)$ is defined by parts of $\partial\overline{R_r}$ and a set C_1 of visibility cuts. Each point of $\overline{R_r}$ not visible from Q_1 lies behind a cut $c \in C_1$, and all points behind a cut c define the pocket P_c of c. We denote with C_2 the set of cuts defining the boundary of $Vis_{\overline{R_r}}(Q_2)$. Since edge e is visible from both Q_1 and Q_2, and no endpoint of edge e is a reflex vertex, there is no cut in the set $C_1 \cup C_2$ whose endpoint lies on edge e or whose pocket contains e on its boundary. Furthermore, if there were a cut $c \in C_1 \cup C_2$ of type Q_1Q_2, then either no point of e or no point of P were visible either from Q_1 or from Q_2 – a contradiction. Hence there are only cuts of type $Q_1Q_1, Q_1P, Q_2P, Q_2Q_2$, and of type PP. We now process the cuts depending on their type. We first remove from C_1 all cuts with both endpoints on Q_2, and we remove from C_2 all cuts with both endpoints on Q_1 (since all points behind those cuts are visible from $\overline{Q}$). Exactly one cut $c_1 = pq \in C_1$ exists where $p \in P$ and $q \in Q_2$ iff the endpoint $w = R_t(P) \cap u$ of Q_2 is not visible from Q_1. This is due to the fact that the endpoints of the cuts in C_1 appear in consecutive order along the polygonal chain $Q_2R_t(P)R_l(P)R_b(P)$. No cut of C_2 intersects c_1, because c_1 is visible from $q \in Q_2$. Hence the pocket of any cut in C_2 is either contained in pocket P_{c_1} of c_1 or has an empty intersection with P_{c_1}. We cut off from $\overline{R_r}$ all pockets P_c of cuts $c \in C_2$ whose endpoints lie on the boundary of P_{c_1}. Those points are seen neither from Q_1 nor from Q_2, since they lie behind a cut of C_1 and also behind a cut of C_2. Then we remove c_1 from C_1. If a corresponding cut $c_2 = p'q' \in C_2$ exists, where $p' \in P$ and $q' \in Q_1$, then c_2 is processed analogously. Now both endpoints of each remaining cut belong to the polygonal chain $C_P = R_t(P)R_l(P)R_b(P) \subseteq \partial P$. The cuts of C_1 appear consecutively on C_P, as do the cuts of C_2. We push the remaining cuts of C_1 and C_2 on a stack S_1 and S_2, respectively, in order of appearance on C_P. Simultaneously we identify, for each endpoint x of each cut, the distance $d(x)$ covered when walking on C_P from w (the endpoint of segment u) to x. From now on, whenever we pop a cut c from S_1 (S_2) we also remove c from C_1 (C_2).

We now traverse the boundary of P. Whenever two pockets P_a and P_b intersect, we cut off their intersection $P_a \cap P_b$ from $\overline{R_r}$. The correctness of the subroutine described below follows from the simple observation

$$P_{c_a} \cap P_{c_b} = \emptyset \Leftrightarrow d(a') \geq d(b),$$

where $c_a = aa'$ and $c_b = bb'$ are two cuts of $C_1 \cup C_2$, assuming w. l. o. g. that $d(a) > d(a')$, $d(b) > d(b')$ and $d(a) \geq d(b)$ holds. Because C_P is a simple polygonal chain, and c_a is a visibility segment, we know that if the intersection of P_{c_a} and P_{c_b} is non-empty, then at least one endpoint, b, of c_b lies, on C_P, between the endpoints of c_a. Note that the pocket of a cut in C_1 can only intersect the pocket of a cut in C_2, and vice versa.

On both stacks S_1 and S_2 the cuts are sorted, in ascending order, by their distance (measured by function $d(\cdot)$) to the endpoint, $w = R_t(P) \cap u$, of C_P. The cut at the bottom of S_1 (S_2) is the cut of C_1 (C_2) closest to point w. Let $\{c_a = aa', c_b = bb'\} = \{peek(S_1) \cup peek(S_2)\}$ denote the two cuts at the top of the two stacks S_1 and S_2. Again, we assume w. l. o. g. that $d(a) > d(a')$, $d(b) > d(b')$ and $d(a) \geq d(b)$ hold. If $d(a) = d(b)$, then we also assume w. l. o. g. that $d(b') \geq d(a')$ holds. There are three cases to consider:

1. $d(a) \geq d(b) > d(b') \geq d(a')$
   P_{c_b} is contained in P_{c_a}. We pop c_b from its stack and replace the part of C_P between b and b' by c_b.

2. $d(a) > d(a') \geq d(b) > d(b')$
   No pocket of a cut of $C_1 \cup C_2$ intersects the pocket P_{c_a} of c_a; we pop c_a from its stack.

3. $d(a) > d(b) > d(a') > d(b')$
   The two cuts c_a and c_b intersect at some point s. We replace the part of C_P between b and a' by the polygonal chain bsa'. Now we pop c_a from its stack.

If at any time either stack is empty, then there are no further points to subtract from $\overline{R_r}$, and the subroutine terminates.

A simple proof, analogous to Exercise 1.8 in [45, p. 22], shows that a connected subset of the boundary of a simple polygon $\overline{R_r}$ can, in $\overline{R_r}$, see at most one connected subset of a boundary edge of $\overline{R_r}$. Hence, at any time at most four endpoints of cuts in the set $C_1 \cup C_2$ lie on a single boundary edge of $\overline{R_r}$. Thus the endpoints of the cuts in the set $C_1 \cup C_2$, that lie on a given boundary edge of P, can be sorted in constant time according to their distance $d(\cdot)$ to point w. We conclude that the sequential order of the cut's endpoints along the boundary of $\overline{R_r}$ can be determined in linear time $O(|P| + |Q|)$, and thus

Lemma 4.7. *With the notation from above,*

1. *removing the cuts with both endpoints on Q_1 or on Q_2 is done in time $O(|Q|)$,*

2. *identifying the cuts whose pockets are contained in the same pocket as either of the two endpoints of C_P is done in time $O(|C_P|) \in O(|P|)$,*

3. *the order of cuts along C_P and the distances from endpoint w to the cut's endpoints can be determined in time $O(|C_P|) \in O(|P|)$.*

Proof. We store for each cut $c = vv'$ the reflex vertex v defining c and the boundary edge of $\overline{R_r}$ that contains the other endpoint v' of c. For each boundary edge of $\overline{R_r}$ we store a list of (the up to four) cuts of which an endpoint lies on that edge. The algorithm for computing the visibility polygon of Q_1 and of Q_2 can be easily modified to maintain this information.

1. In order to remove the cuts with both endpoints on Q_1 we traverse Q_1. Thus in time $O(|Q_1|)$ we have access to all cuts where one endpoint lies on Q_1. The number of those cuts is at most $4|Q_1|$, and thus in time $O(|Q_1|)$ we can identify which cuts to remove. The cuts with both endpoints on Q_2 can be removed analogously in time $O(|Q_2|)$. The total time needed is $O(|Q_1| + |Q_2|) = O(|Q|)$. If, during this process, we encounter a cut of type PQ_1 or of type PQ_2, we store those (up to two) cuts for using them later in step 2.

2. In step 1 above we have already identified any cut of type PQ_1 or of type PQ_2. If there exists a cut pq of type PQ_2 we traverse C_P, starting at its endpoint $w = u \cap R_t(P) = Q_2 \cap \partial P$, in direction of p. This way we can identify every cut whose endpoints lie between w and p, and therefore have to be removed (in total time $O(|P|)$). If there is a cut $p'q'$ of type PQ_1, we proceed analogously and identify the cuts whose endpoints lie on P between $w' = v \cap R_b(P) = Q_1 \cap P$ and p', also in time $O(|P|)$.

3. We traverse C_P starting at endpoint w in direction to the other endpoint w' and compute the distance $d(v)$ for every vertex v of C_P in the following way. Suppose we have just computed the distance $d(y)$ for some vertex y (initially, we have $y = w$ and $d(y) = 0$). Let x be the vertex following y on our way from w to w'. Then we have $d(x) = d(y) + |xy|$, where $|xy|$ denotes the length of the line segment connecting x with y.

   After the distance $d(v)$ from w to every vertex $v \in C_P$ has been determined, we traverse C_P a second time, again starting from w in direction to w'. Now, for each cut c where one endpoint z of c lies on edge xy, where $d(x) < d(y)$, we compute the distance $d(z) = d(x) + |xz|$. If z is the first endpoint of cut c whose distance $d(z)$ is determined, and c is contained in C_1 (C_2) we also push c on stack S_1 (S_2). Note that for each edge of C_P, in each of the sets C_1 and C_2 there exists at most one cut that is pushed on S_1 and S_2, respectively. Since for each edge e of C_P there are at most 4 cuts in the set $C_1 \cup C_2$ where one endpoint lies on e, each edge of C_P can be processed in constant time. When the second traversal of C_P is completed, the cuts in the set C_1 (C_2) are pushed on stack S_1 (S_2), in sorted order according to the distance $d(\cdot)$ of their endpoints.

□

Furthermore, we observe that the total number of *push*, *pop* and *peek* operations is also linear in the total number $O(|P| + |Q|)$ of cuts. Cutting off the appropriate parts of $\overline{R_r}$ is done in time $O(|\textit{parts cut off}|) \in O(|\overline{R_r}|) \in O(|P| + |Q|)$. Summing up, we obtain a total running time $O(|P| + |Q|)$ that is linear in the number of vertices of P and Q.

Summary

In order to compute the (inner or outer) visibility polygon of a given polygon Q w. r. t. another polygon P, we can decide in linear time whether we must apply the algorithm formulated in Section 4.3.1, or one of the four algorithms presented in Section 4.3.2. The runtime of any given algorithm is also linear. This completes the proof of Theorem 4.6. Furthermore, an open polygonal chain $C = (c_1, \ldots, c_n)$ can be treated as a closed polygonal chain $C' = (c_1, c_2, \ldots, c_n, \ldots, c_2, c_1)$. The visibility of C is identical to the visibility of C', and we can modify our algorithm to also accept C' as input.

Corollary 4.8. *Given two simple, non-intersecting open or closed polygonal chains P and Q in the plane, the visibility region of Q w. r. t. P can be computed in optimal time $O(|P| + |Q|)$.*

4.4 The Visibility Area Between Two Boundary Chains of a Polygon

We first restate some definitions. A segment pq between two mutually visible points p and q is of *type* PQ, iff $p \in P$ and $q \in Q$, where P and Q are sets of points. Given two polygonal chains P and Q, then the visibility area $VA(P, Q)$ consists of all points that lie on a segment of type PQ. If P and Q are subchains of the same closed polygonal chain C, then we restrict the visibility area to points in the bounded region defined by C.

We shall also need the following definition.

Given the inner visibility polygon $Vis_P(X)$ of a boundary subchain X of a polygon P, and a subchain C of ∂P, we define a polygonal chain $Vis_P(X)_{|C}$ – which is a boundary subchain of $Vis_P(X)$ – as the union of all points $c \in C$ visible from X, and all line segments (*i. e.*, visibility cuts) connecting the endpoints of subsequent parts of C visible from X.

We would like to compute the visibility area between two subchains A and C on the boundary of a common polygon P.

Let us consider a polygon $P = (A, B, C, D)$, where A, B, C, D denote four successive non-empty polygonal subchains of ∂P, see Figure 4.12; the endpoints of subchains A, B, C and D are vertices of P. We want to compute the visibility area between A and C inside P, that is, the union of all line segments $ac \in P$ with $a \in A$ and $c \in C$.

Our first observation is, that C sees only parts of A and vice versa. We can extract this information from the visibility polygons $Vis_P(A)$ and $Vis_P(C)$. More precisely, we consider $C^* := Vis_P(A)_{|C}$ and $A^* := Vis_P(C)_{|A}$. If no point of C is visible from A, then $VA(A, C) = \emptyset$. So from now on we assume that some point of A is visible from C.

Now let $C^* = (c_1, \ldots, c_m)$ and $A^* = (a_1, \ldots, a_n)$ be given in clockwise order, see Figure 4.12. Note that c_1, c_m, a_1 and a_n stem from visibility cuts between the chains C and A. Moreover, obviously every visibility cut that restricts the visibility from A to C

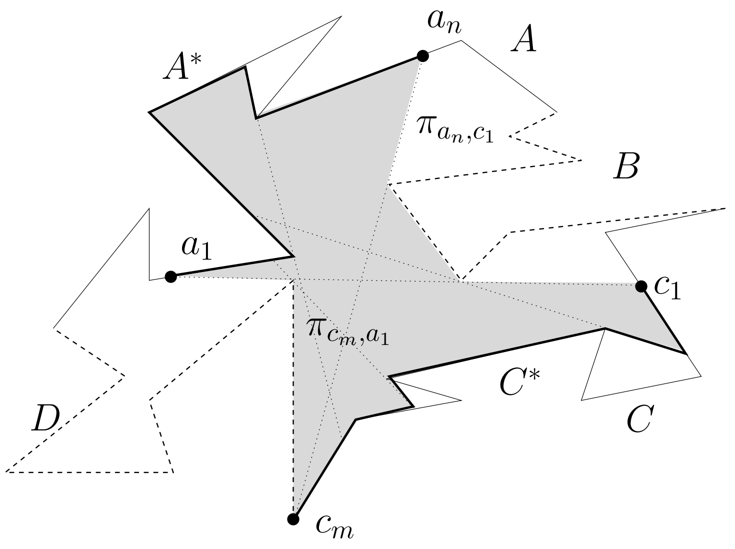

Figure 4.12: Four chains A, B, C and D define the polygon P. The chain A^* (C^*) is the part of A (C) which is seen from C (A). For the visibility area between A and C one has to combine A^* and C^* with the shortest path from a_n to c_1 and the shortest path from c_m to a_1.

(or vice versa) is given by two vertices of the polygon P. We consider the shortest path, π_{a_n,c_1}, from a_n to c_1 and the shortest path, π_{c_m,a_1}, from c_m to a_1 inside P. We would like to prove that the visibility area between A and C inside P is given by the closed chain $A^*\pi_{a_n,c_1}C^*\pi_{c_m,a_1}$.

In the following we assume general position, that is, no three vertices of the polygon P are collinear.

Lemma 4.9. *With the notation from above, $A^*\pi_{a_n,c_1}C^*\pi_{c_m,a_1}$ is the visibility area between A and C inside a polygon P.*

Proof. We first prove the equality $VA(A,C) = VA(A^*,C^*)$.

- $\subseteq$ Obviously, in P, every segment ac of type AC is also of type A^*C^*.
- $\supseteq$ Let $a^*c^* \in P$ be a segment of type A^*C^*. We now show that a^*c^* is also contained in a segment $ac \in P$ of type AC. We start with point a^*. If $a^* \in A$, we set $a := a^*$ and continue with c^*. Otherwise, by definition of A^*, point a^* must be an interior point of a visibility cut c_A of type AA. Now we know that ray $\rho_{c^*}(a^*)$ must intersect A at some point $a \in A$, behind a^*. Point a belongs to the boundary of the pocket P_{c_A} defined by cut c_A (this is also true if $\rho_{c^*}(a^*)$ is collinear with c_A). A similar point $c \in C$ exists for point c^*, and obviously the visibility segment ac contains segment a^*c^*.

It remains to show that every segment of type A^*C^* is contained in the region R bounded by chain $A^*\pi_{a_n,c_1}C^*\pi_{c_m,a_1}$, and every point in R lies on a segment of type A^*C^*. Let a^*c^* be a segment of type A^*C^*. We now show that $a^*c^* \in R$. We assume w. l. o. g. that $a^* \notin \{a_1, a_n\}$ and $c^* \notin \{c_1, c_m\}$. The case $a^* \in \{a_1, a_n\}$ and / or $c^* \in \{c_1, c_m\}$ follows with minor modifications. Because segment a^*c^* is a shortest path in P, and the two points a_n and c_1 (a_1 and c_m) lie in the region of P to the left (right) side of the oriented segment a^*c^*, we know that the shortest path π_{a_n,c_1} (π_{a_1,c_m}) cannot intersect the segment a^*c^* transversally. Moreover, segment a^*c^* also does not intersect A^* or C^* transversally. Now $a^*c^* \in P$ implies that a^*c^* is contained in region R. We continue by showing that every point $r \in R$ lies on a segment of type A^*C^*. We first show that every point $r \in R$ is visible from A^* and from C^*. Then, we use a geometric argument to show that r also lies on a segment of type A^*C^*.

Our first observation is that the shortest paths π_{a_n,c_1} and π_{a_1,c_m} are bent outwards (that is, the two boundary chains π_{a_n,c_1} and π_{a_1,c_m} of region R are bent outwards). We first prove this claim for π_{a_1,c_m}. We have to prove that when walking on π_{a_1,c_m} from a_1 to c_m, we always turn right at the inner vertices of π_{a_1,c_m}. If a_1 sees c_m we have $\pi_{a_1,c_m} = a_1c_m$ and we are done. Otherwise, since $a_1 \in A^*$ we know that a_1 sees a point $c \in C^*$, $c \neq c_m$, and point c_m is contained in the region P_r of P to the right of the oriented segment a_1c. Furthermore, no vertex of P_r belongs to A^*, because those vertices are contained in the region P_l of P to the left of a_1c. Now it follows, because c_m is visible from A^*, that π_{a_1,c_m} is bent outwards – a left turn at a vertex $v \notin A^*$ would imply that c_m is not visible from A^*. Using the same arguments we obtain that the shortest path π_{a_n,c_1} is bent outwards. We can now show that every point r in R is visible from A^*. It suffices to show that every point on the boundary of R is visible from A^*. By symmetry, from the same arguments it follows that every point in R is also visible from C^*.

We first show that every point $c \in C^*$ is (in polygon R) visible from A^*. Point c is, in polygon P, visible from some point $a \in A^*$. Neither of the two shortest paths π_{a_1,c_m} and π_{a_n,c_1} can intersect the visibility segment ac transversally. We conclude that $ac \in R$ and thus that point c is, in R, visible from A^*. We continue by showing that every point on both shortest paths π_{a_1,c_m} and π_{a_n,c_1} is visible from A^*. We first show that every point r on the shortest path π_{a_1,c_m} is visible from A^*. Because c_m is visible from A^*, we can assume $r \neq c_m$. We know that the shortest path π_{a_1,c_m} is bent outwards, *i. e.*, walking on π_{a_1,c_m} from a_1 to c_m we always turn right at the inner vertices. Let $a^* \in A^*$ denote a point that sees point c_m, as shown in Figure 4.13. Walking on π_{a_1,c_m} from a_1 to c_m, we denote by v the first vertex of π_{a_1,c_m} we encounter after reaching point r. Now the ray $\rho_v(r)$ must intersect A^* at some point a_r, between a_1 and a^*. Thus, r is visible from A^*. Using the same arguments we can show that every point of the shortest path π_{a_n,c_1} is also visible from A^*. Altogether, we have shown that every point in R is visible from A^*. By symmetry, the same arguments hold for boundary chain C^*, and therefore we have shown that every point in R is visible from A^* and from C^*. Now from Lemma 4.10, 1. below, using $A := A^*$, $B := \pi_{a_n,c_1}$, $C := C^*$

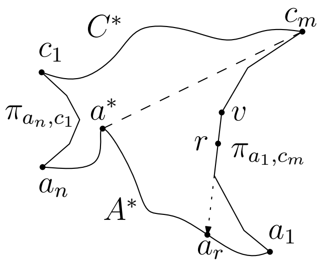

Figure 4.13: Ray $\rho_v(r)$ must intersect A^* at some point a_r between a_1 and a^*.

and $D := \pi_{a_1,c_m}$, it follows that every point $r \in R$ lies on a segment of type A^*C^*, because r is visible from A^* and from C^*. □

In the following lemma we state some structural properties of the visibility area between two subchains of the boundary of a polygon P.

Lemma 4.10. *Let $P = (A, B, C, D)$ be a polygon defined, as above, by four successive, non-empty boundary chains A, B, C, D (the endpoints of the four chains are also vertices of P).*

1. *If chains B and D are* bent outwards, *then any point $p \in P$ visible from both A and C belongs to the visibility area $VA(A, C)$ between A and C.*

2. *Given two mutually visible points $a_1, a_2 \in A$, and a point $p \in P$ contained in the region $R^A_{a_1a_2}$ bounded by the segment a_1a_2 and the part of A between a_1 and a_2, then the following holds*

   (a) *p belongs to $VA(A,C)$ iff p is visible from C.*

   (b) *p belongs to $VA(A, B \cup D)$ iff p is visible from $B \cup D$.*

   (c) *p belongs to $VA(A \cup C, B)$ iff p is visible from B.*

   (d) *If p does not lie on segment a_1a_2, then p does not lie on a segment of type BB, CC, DD, BC, BD or CD.*

Proof. 1. Given a point $p \in P$ visible from A and from C, we consider four intervals $I_B = [0, \alpha_1]$, $I_A = [\alpha_1, \alpha_2]$, $I_D = [\alpha_2, \alpha_3]$ and $I_C = [\alpha_3, 2\pi]$, given values $0 \leq \alpha_1 \leq \alpha_2 \leq \alpha_3 \leq 2\pi$. We assume that (i) $\alpha_1 \leq \pi$ and (ii) $\alpha_3 - \alpha_2 \leq \pi$. The following simple proof shows that there exists a value $p_1 \in I_A$ where $p_2 = p_1 + \pi \in I_C$. Let $p_1 = \min\{\alpha_2, \pi\}$ and $p_2 = p_1 + \pi$. Now $p_1 \in I_A$ holds because (i) and $\alpha_1 \leq \alpha_2$ imply $\alpha_1 \leq \min\{\pi, \alpha_2\} = p_1 \leq \alpha_2$. If $\alpha_2 > \pi$ we have, $p_1 = \pi$ and $p_2 = 2\pi$, which implies $p_2 \in I_C$. Otherwise $\alpha_2 \leq \pi$ holds, and thus $p_1 = \alpha_2$ and $p_2 = \alpha_2 + \pi$. Now from (ii) we obtain $\alpha_3 \leq \alpha_2 + \pi = p_2 \leq 2\pi$, which also implies $p_2 \in I_C$.

It now follows that p belongs to $VA(A,C)$. We denote with α_1, α_2 the angles defined by the extremal directions where p sees points of A and with $\alpha_3, 2\pi$ the angles defined by the extremal directions where p sees points of C, as shown in Figure 4.14. Furthermore, we observe that $\alpha_1 \leq \pi$ and $\alpha_3 - \alpha_2 \leq \pi$ holds

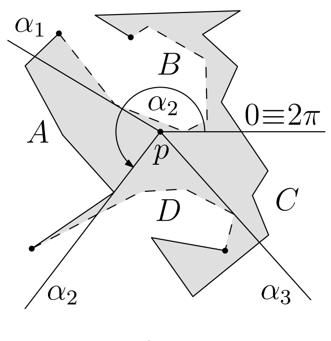

Figure 4.14: The angles α_1, α_2 and α_3 defined by the extremal directions where P sees points of A and C.

because the extremal directions where p sees points of B or D form an angle of at most π (since B and D are bent outwards). Note that the proof also holds if p is not seen from B (D); in this case we use $\alpha_1 = 0$ ($\alpha_2 = \alpha_3$).

2. If point p is seen from another point $x \in P \setminus R^A_{a_1a_2}$ or $x \in a_1a_2$, then p lies on a segment of type $\{x\}A$, compare Figure 4.15. This is due to the fact that the ray $\rho_x(p)$ must intersect the segment a_1a_2 transversally, or $x \in a_1a_2$ holds. In

both cases ray $\rho_x(p)$ must reach a point $a \in A$, behind p ($a = p$ is possible). Statements (a) to (d) are easy consequences of this observation.

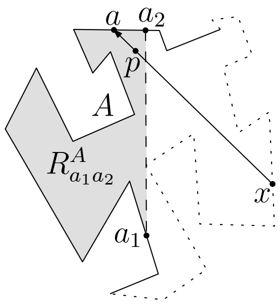

Figure 4.15: If p is visible from point $x \in (P \setminus R^A_{a_1a_2}) \cup a_1a_2$, then from x we must also see a point $a \in A$, behind p ($a = p$ is possible).

(a) According to the definition of $VA(A, C)$, if p is not visible from C, then p does not belong to $VA(A, C)$. Now if p is visible from some point $c \in C$, then using $x = c$ we obtain from the observation above that p lies on a segment ac of type AC.

(b) Using the same arguments as in the proof of (a), it follows that p belongs to $VA(A, B \cup D)$ iff p is visible from some point $x \in B \cup D$.

(c) If $p \in VA(A \cup C, B)$ holds, then p is visible from B. Using the same argument as in the proof of (a), if p is visible from some point $b \in B$, then p lies on a segment ab of type AB, and hence belongs to $VA(A, B)$. Now the claim follows from the fact $VA(A, B) \subseteq VA(A \cup C, B)$.

(d) Let $x \in B \cup C \cup D$ be a point that sees point p. The ray $\rho_x(p)$ intersects the segment a_1a_2 *before* reaching p. Since p does not lie on segment a_1a_2, after reaching p the ray $\rho_x(p)$ cannot reach further points of B, C or D, before it ends when intersecting the boundary of $R^A_{a_1a_2}$ at some point $a \in A$. This implies that given any segment $ax \in P$ containing p, where $x \in B \cup C \cup D$, it holds that $a \in A$. Therefore, segment ax cannot be of type BB, CC, DD, BC, BD or CD.

$\square$

Corollary 4.11. *The visibility area between two boundary subchains A and C of a simple polygon $P = (A, B, C, D)$ can be computed in linear time $O(|P|)$.*

Proof. We can compute $Vis_P(A)$ and $Vis_P(C)$ in linear time. Also in linear time, we traverse the boundaries of $Vis_P(A)$ and $Vis_P(C)$ obtaining the polygonal chains C^* and A^*. Connecting the endpoints of the chains with the shortest paths in P can also be done in linear time. $\square$

4.5 The Visibility Area Between Two Simple Polygons

The visibility area $VA(P,Q)$ between two simple polygonal chains P and Q in the plane is the union of all line segments pq, where $p \in P$ and $q \in Q$ holds, and p sees q. This section is devoted to the proof of the following Theorem 4.12 about the visibility area between two closed polygonal chains P and Q.

Theorem 4.12. *The visibility area between two disjoint closed polygonal chains P and Q in the plane can be computed in optimal time $O(|P| + |Q|)$.*

Let P and Q be two disjoint closed polygonal chains. We distinguish between different cases, depending on the position of P relative to Q. Note that we can compute the convex hull of one or two polygons in linear time [53, 61] and also check if a polygon is contained in another polygon, also in linear time, assuming that the boundary chains do not intersect [45]. Furthermore, we can identify the pocket of one polygon containing the other polygon in linear time. We first consider two simple cases in sections 4.5.1 and 4.5.2. The technically most challenging case is discussed in Section 4.5.3.

4.5.1 Neither $P \subset ch(Q)$ nor $Q \subset ch(P)$

There are two outer tangents to P and Q, which can be identified in linear time [53, 61]. Those tangents divide the outside of P and Q into one bounded region and one unbounded region, see Figure 4.16. The bounded region is a simple polygon R

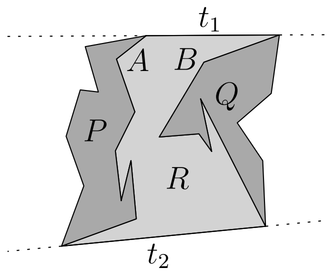

Figure 4.16: The visibility area between P and Q is contained in region R bounded by P, Q, and the two outer tangents to P and Q.

(allowing that the two tangents share a common tangent point of P or of Q). The boundary of R consists of two polygonal subchains A and B of ∂P and ∂Q, respectively, connected by two tangential segments t_1 and t_2. Each tangential segment is part of one of the two outer tangents. It is easy to see that the visibility area between P and Q equals the visibility area between A and B in R. Thus, we can employ the algorithm presented in Section 4.4 to compute, by Corollary 4.11 in linear time, the visibility area between P and Q.

4.5.2 P Contained in Q

Let $R = Q \setminus P$ then every point of the visibility area between P and Q must be contained in R. We compute the visibility polygon of Q and of P in R and cut off from R the points not visible from P or Q. This can, by Theorem 4.6, be done in linear time. We now verify that every remaining point $r \in R$ belongs to the visibility area between P and Q. If $r \in \partial P$ ($r \in \partial Q$) we are done, because r is visible from ∂Q (∂P), so now we assume $r \notin \partial P \cup \partial Q$. Let pq be *any* visibility segment in R, where p and q belong to the boundary of R, and pq contains r, see Figure 4.17. We are done

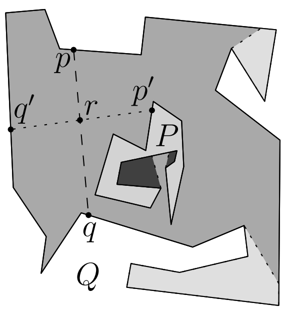

Figure 4.17: Any point $r \in Q \setminus P$ visible from P and Q belongs to the visibility area between P and Q. Points not visible from P (Q) are drawn in light grey (dark grey) colour.

if pq is of type PQ. Otherwise let pq be of type QQ. Because pq is of type QQ, pq divides polygon R in two parts – one of them containing P. By assumption, there exists a point p' of P that sees r. Now ray $\rho_{p'}(r)$ intersects segment pq at point r, and intersects the boundary of Q at some point q' behind r. Hence r lies on the visibility segment $p'q'$ of type PQ. Using the same argument, it follows that if the segment pq is of type PP, then r also lies on a segment $p'q'$ of type PQ.

4.5.3 Polygon P Contained in a Pocket P_e of Polygon $\overline{Q}$

The situation is as depicted in Figure 4.18, and is similar to the situation depicted in Figure 4.8 on page 63. Polygon P is contained in one pocket P_e of polygon $\overline{Q}$ that is defined by an edge e of $ch(\overline{Q})$. In other words, edge e is the unique edge that belongs to the boundary of polygons P_e and $ch(\overline{Q})$, but not to the boundary of polygon $\overline{Q}$.

In order to compute the visibility area between polygons $\overline{Q}$ and P, it suffices to compute the visibility area $VA(P, Q)$ between P and the common boundary, Q, of P_e and $\overline{Q}$, in polygon $R = P_e \setminus P$, because every segment of type PQ (or, equivalently, of type $P\overline{Q}$) is contained in R. Note that Q can be obtained by removing from ∂P_e the interior points of edge e. In contrast to the case where P is contained in a polygon Q, in this case it is not sufficient only to identify the set of points in R visible from both P and Q. It is possible that some points of R do not lie on a segment of type PQ, although they are visible from both P and Q. Thus in a first step we will show how to

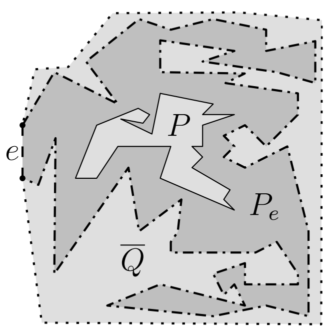

Figure 4.18: Polygon P contained in a pocket P_e of polygon $\overline{Q}$. Dashed (dotted / dash dotted) boundary edges belong to P_e ($\overline{Q}$ / the common boundary, Q, of P_e and $\overline{Q}$).

identify these points. We also compute the set of points visible from both P and Q in polygon R. Combining this information we obtain the visibility area between P and Q in R.

We now describe how to remove from polygon R the points visible from both P and Q, that do not lie on a segment of type PQ. It will turn out that we will, at the same time, remove from R some points not visible from P – which of course do not belong to the visibility area between P and Q. We will first decide if some point of P is visible from edge e. If no point of P is visible from e, then any segment of type PP_e is of type PQ. In this case we can compute the visibility area between P and Q as described in section 4.5.2, by computing $\mathit{VA}(P,Q) = \mathit{VA}(P,P_e)$. So from now on we assume that some point of P is visible from some point of e.

In the following we make use of arguments relying on properties of shortest paths. These properties only apply for shortest paths inside simple polygons, but polygon $R = P_e \setminus P$ is not simple. We overcome this structural barrier by dividing polygon R in two simple polygons and compute the visibility area in each polygon separately.

Let w. l. o. g. e be a horizontal edge, and P lies strictly below e, as do all points of $\overline{Q}$, compare Figure 4.19. We consider the lowest horizontal line, ℓ, that intersects P. Let p_l denote the leftmost point of P on ℓ, and q_l denote the rightmost point of Q on ℓ, to the left of p_l. Point p_r is the rightmost point of P on ℓ and q_r is the leftmost point of Q on ℓ, to the right of p_r. Adding the segments p_lq_l and p_rq_r to ∂R we divide polygon R in two subpolygons, the top polygon R_t and the bottom polygon R_b. Polygon R_t (R_b) is the subpolygon of R to the left (right) of the oriented segments q_lp_l and p_rq_r, as shown in Figure 4.19. In general (and, in fact, by general position of the vertices of R) there is exactly *one* vertex v of P on line ℓ (since ℓ is parallel to edge e and two distinct parallel lines contain at most three vertices, if the vertices are in general position). Then, we have $v = p_l = p_r$. However, there are two occurrences of vertex v on the boundary of R_t; the occurrence of vertex v adjacent to q_l is denoted by p_l and the occurrence adjacent to q_r by p_r. Further note that e belongs to the boundary of R_t, if some point of P is visible from e.

Each point $r_b \in R_b$ visible from P and Q belongs to the visibility area between P

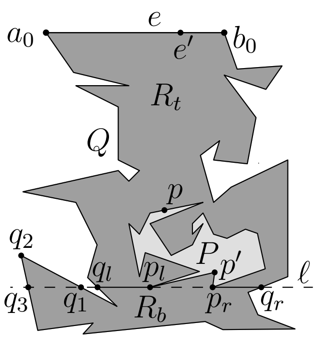

Figure 4.19: The division of polygon R into subpolygons R_b and R_t.

and Q. If r_b lies (on or) below line ℓ (compare Figure 4.19) this is obvious, since the ray $\rho_p(r_b)$ originating from some point $p \in \partial P$ that sees r_b must reach some boundary point $q \in Q$ of R_b, behind r_b. If otherwise r_b lies strictly above line ℓ, then r_b is either not seen from P (see the points above ℓ contained in the triangle with endpoints q_1, q_2, and q_3, in Figure 4.19), or r_b lies in the region belonging to R_b bounded by segment p_lp_r and the boundary of P (compare the points above ℓ contained in the triangle with endpoints p_l, p_r, and p', in Figure 4.19). In the latter case, if r_b is visible from some point $q \in Q$, then the ray $\rho_q(r_b)$ must intersect segment p_lp_r and reach a point of ∂P, behind r_b.

We have already shown, in Section 4.3, how to compute the set of points in R_b neither visible from P nor from Q (compare the part of "Case 1", where the visibility of P in subpolygon R_r is discussed and the part of "Case 3", where the visibility of Q in subpolygon R_r is considered, on pages 64 and 65). Since we know how to identify the points in subpolygon R_b that belong to $VA(P, Q)$, it remains to consider the subpolygon R_t of R.

Let a_0 and b_0 denote the endpoints of edge $e = (a_0, b_0)$, in clockwise order, on the boundary of R_t. By assumption, there exists a point p of P that is visible from some point e' of edge e, therefore Q cannot intersect the segment $e'p$. Hence, walking in clockwise direction on ∂R_t, starting at e', we first reach b_0 before reaching $q_r, p_r, p, p_l, q_l, a_0, e'$, in that order; compare Figure 4.19. Now our goal is to identify all points that lie on a segment of type PQ in R_t. We can treat the segments p_lq_l and p_rq_r of ∂R_t as parts of Q: consider any segment rr' of type $P(p_lq_l)$ (or, analogously, of type $P(p_rq_r)$). The ray $\rho_r(r')$ must intersect the segment p_lq_l at r', and thus $\rho_r(r')$ must reach a point q of Q, behind r'. Hence any point belonging to the segment rr' also lies on segment rq of type PQ. We conclude that every point in R_t that belongs to $VA(P, Q)$ lies on a visibility segment pq between a point $p \in P$ on the common boundary of polygons P and R_t, and a point q that lies on one of the two two boundary chains of R_t connecting the endpoints a_0 and b_0 of edge e with p_l and p_r, respectively. Hence, for convenience, in the following we denote with P the common boundary of polygons P and R_t, and by Q we denote the two boundary chains of R_t connecting the endpoints of edge e with the vertices p_l and p_r of P.

Next, we define a subpolygon $\overline{R}$ of R_t as follows. Suppose we walk, along boundary chain P, from p_l in direction of p_r. At some time we reach the first point p'_l of P that is visible from e. Then, we reach point p before we finally reach the last point p'_r of P that is visible from e; see Figure 4.20. We obtain polygon $\overline{R}$ from R_t by replacing

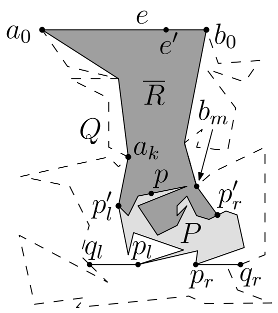

Figure 4.20: Region $\overline{R} \subseteq R_t$ defined by edge e, the shortest paths π_{a_0,p'_l} and π_{b_0,p'_r}, and the part of ∂P between p'_l and p'_r.

the boundary parts $(a_0, \ldots, q_l, p_l, \ldots, p'_l)$ and $(b_0, \ldots, q_r, p_r, \ldots, p'_r)$ of ∂R_t with the shortest paths π_{a_0,p'_l} and π_{b_0,p'_r} in R_t, respectively. Note that the two pairs of paths $\pi_{a_0,p'_l}, \pi_{b_0,p'_l}$ and $\pi_{a_0,p'_r}, \pi_{b_0,p'_r}$ are outwards convex. In particular, walking on the shortest path from a_0 to p'_l we always turn right at the inner vertices, and walking on the the shortest path from b_0 to p'_r we always turn left. Further note that polygons R_t, R_b and $\overline{R}$ can be obtained in linear time $O(|R|)$ from polygon R.

Our previous observations and the following lemma imply that for every point r in R visible from P and from Q, it holds that if r that does not belong to $VA(P,Q)$, then r is contained in region $\overline{R}$.

Lemma 4.13. *With the notation from above, if a point $r \in R_t$ is visible from both P and Q, and r does not lie on a segment of type PQ, then $r \in \overline{R}$ holds.*

Proof. From the same arguments as used in Section 4.5.2, it follows that if r is visible from P and Q, and r is not visible from edge e, then r lies on a segment of type PQ. Thus it remains to consider the points in $R_t \setminus \overline{R}$, visible from e, P, and Q. To prove the lemma we have to show that those points belong to $VA(P,Q)$. Let point $e' \in e$ be a point of edge e that sees a point $r \in R_t \setminus \overline{R}$, and r is also visible from P and Q. Clearly, the visibility segment re' must intersect either π_{a_0,p'_l} or π_{b_0,p'_r}. We now prove that if the segment re' intersects any segment of the shortest path $\pi_{a_0,p'_l} = (a_0, a_1, \ldots, a_k, p'_l)$, then r lies on a segment of type PQ. Analogous arguments apply if the segment re' intersects a segment of the shortest path $\pi_{b_0,p'_r} = (b_0, b_1, \ldots, b_m, p'_r)$.

We first define a division of the boundary of R_t into four successive boundary chains A, B, C and D as follows. We traverse the boundary of R_t in counterclockwise direction, starting at point a_0, and obtain $A = (a_0, \ldots, a_k, \ldots, q_l, p_l)$, $B = P = (p_l, \ldots, p_r)$, $C = (p_r, q_r, \ldots, b_m, \ldots b_0)$ and $D = e$. Hence $P = B$ and $Q = A \cup C$. Next, we observe that every vertex of $\pi_{a_0,p_l'}$ is visible from edge e, because p_l' is visible from e. Hence, from the definition of point p_l', it follows that p_l' is the only vertex of $\pi_{a_0,p_l'}$ that belongs to P; the other vertices must belong to $A \subseteq Q$. Now if segment re' intersects a segment a_ia_{i+1}, $i \in \{0,\dots,k-1\}$, then we know that segment a_ia_{i+1} is of type AA. Now we can apply Lemma 4.10, 2c, which implies $r \in \mathit{VA}(Q,P)$, using $A \cup C = Q$ and $B = P$. It remains to consider the case where segment re' intersects the segment a_kp_l'. Adding segment a_kp_l' to ∂R_t subdivides polygon R_t in two subpolygons R_1 and R_2, see Figure 4.21. We know that r is contained in the subpolygon R_1 of R_t that does

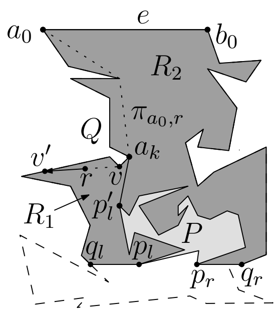

Figure 4.21: Segment a_kp_l' divides polygon R_t in two subpolygons R_1 and R_2.

not contain edge e on its boundary. Now since the shortest path $\pi_{a_0,r}$ must intersect the segment a_kp_l' at some point $z \in \pi_{a_0,p_l'}$, we know that all points of $\pi_{a_0,p_l'}$ between a_0 and z also lie on the shortest path $\pi_{a_0,r}$. Hence $\pi_{a_0,r}$ contains vertex a_k. Now let v be the vertex of $\pi_{a_0,r}$ adjacent to r, as shown in Figure 4.21. The ray $\rho_v(r)$ must intersect the boundary of R_t at some point v', that is visible from v. Both points v and v' belong to the boundary of subpolygon R_1. If segment vv' is of type PQ, then r belongs to $\mathit{VA}(P,Q)$, by definition of the visibility area. Otherwise, if vv' is of type PP (QQ), from Lemma 4.10, 2b (2c), it follows that $r \in \mathit{VA}(P,Q)$, using $A = P$ and $B \cup D = Q$ ($A \cup C = Q$ and $B = P$), because r is visible from Q (P). □

We have shown that, in order to remove from R_t every point r visible from both P and Q, that does not lie on a segment of type PQ, we only have to consider the points in $\overline{R}$. More precisely, we only have to consider points in $\overline{R}$ visible from e, P and Q. We shall use the following fact which follows from the general position of the vertices of polygon R.

Lemma 4.14. *If a point $p \in P$ is, in polygon R, visible from some point $e' \in e$, then p is not the only point of P visible from e.*

Proof. We show that p lies on a boundary edge $\overline{e} \in \partial P$ of which edge e sees at least some connected segment pp', where $p' \in \overline{e}$, $p \neq p'$. There are four different cases.

1. Both p and e' are vertices of R.
   The segment pe' cannot contain another vertex of R, so there is one edge $\overline{e}$

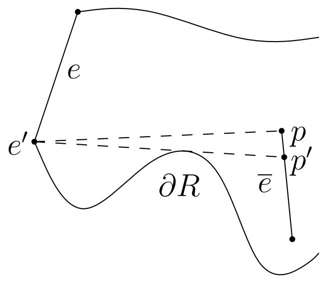

Figure 4.22: Some point p' near p on edge $\overline{e}$ is visible from e'.

adjacent to vertex p where the neighbourhood of p on $\overline{e}$ is visible from e', compare Figure 4.22.

2. Point p is a vertex of P and e' is an interior point of e.
   There is at most one tangential vertex v contained in the segment pe'. If there is no such vertex v, we argue as in case 1. If a tangential vertex v exists we can rotate the segment pe', with v as centre of rotation, in a direction where the visibility between e and P is maintained; see Figure 4.23.

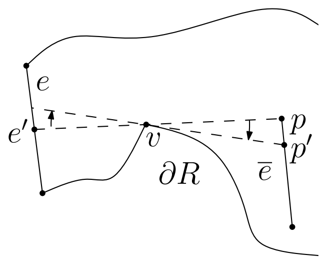

Figure 4.23: Rotating segment pe' while maintaining the visibility between P and edge e.

3. Point p is not a vertex of P, but e' is a vertex of e.
   Again, we can assume that there is exactly one tangential vertex v contained in segment pe'; see Figure 4.24. This time we can rotate the segment ep' with centre of rotation e' to prove the existence of a point $p' \neq p$ on edge $\overline{e}$ containing p, where p' is visible from e'.

4. Neither p nor e' is a vertex of R.
   There are at most 2 tangential vertices v, v' contained in the segment pe'; see Figure 4.25 for an example. Independently of the positions of vertices v, v' (and on which side of the segment pe' the boundary edges adjacent to v, v' are on) we can rotate the segment pe' while maintaining the visibility between e and edge $\overline{e}$. This proves the existence of a point $p' \in \overline{e}$, $p' \neq p$, visible from edge e.

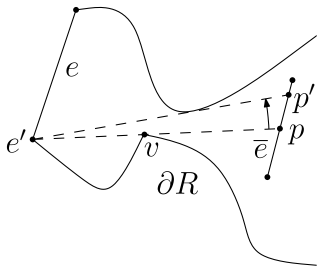

Figure 4.24: Some point $p' \neq p$ on edge $\overline{e}$ is visible from e'.

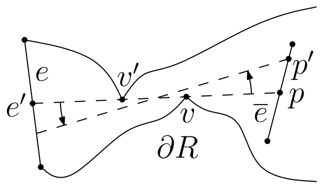

Figure 4.25: Rotating the segment pe' while maintaining the visibility between edge e and edge $\overline{e}$.

$\square$

We will now have a closer look at those points in $\overline{R}$ that do not lie on a segment of type PQ.

Lemma 4.15. *There exists a simple polygonal chain C in polygon $\overline{R}$ that divides R_t in two regions A and B, such that any point $r \in R_t$ belongs to the visibility area between P and Q iff $r \notin B \setminus C$ and r is visible from both P and Q. The polygonal chain C can be computed in linear time $O(|P| + |Q|)$.*

Proof. Using the notation from above, we consider the two shortest paths π_{a_0,p'_r} and π_{b_0,p'_l} in $\overline{R}$, see Figure 4.26. Because p'_r is visible from e, walking on π_{a_0,p'_r} from a_0 to p'_r, we always turn right at the inner vertices. Furthermore the first vertices $(a_0, \ldots, a_i)$ of $\pi_{a_0,p'_r} = (a_0, \ldots, a_i, p_a, \ldots, p'_r)$ belong to Q and the last vertices $(p_a, \ldots, p'_r)$ of π_{a_0,p'_r} belong to P. Analogously we have $\pi_{b_0,p'_l} = (b_0, \ldots, b_j, p_b, \ldots, p'_l)$, where the vertices $(b_0, \ldots, b_j)$ of π_{b_0,p'_l} belong to Q, and the vertices $(p_b, \ldots, p'_l)$ of π_{b_0,p'_l} belong to P. The shortest path π_{b_0,p'_l} must intersect the unique segment $a_i p_a$ of π_{a_0,p'_r} that connects a vertex of Q with a vertex of P, because, by Lemma 4.14, b_0 and p'_l lie, in $\overline{R}$, on different sides of π_{a_0,p'_r} and – more precisely – on different sides of the segment $a_i p_a$. By symmetry, π_{a_0,p'_r} intersects the corresponding segment $b_j p_b$ on π_{b_0,p'_l}. Hence, the two paths intersect at point $p' = a_i p_a \cap b_j p_b$ (it is impossible that π_{b_0,p'_l} intersects the visibility segment $a_i p_a$ in more than one point). After triangulating $\overline{R}$

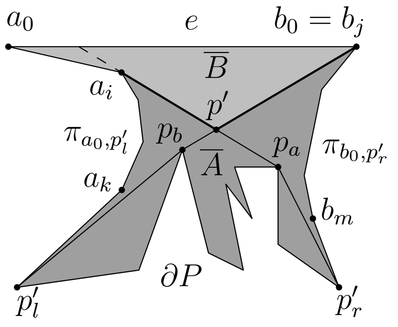

Figure 4.26: The two shortest paths π_{a_0,p'_r} and π_{b_0,p'_l} divide $\overline{R}$ in two regions $\overline{A}$ and $\overline{B}$.

in linear time [13], we can compute the two shortest paths in linear time [40], and the polygonal chain $C = (a_i, p', b_j)$ also in linear time.

Connecting the boundary vertices a_i and b_j of $\overline{R}$ with the polygonal chain $C = (a_i, p', b_j) \in \overline{R}$ divides $\overline{R}$ in two subpolygons $\overline{A}$ and $\overline{B}$; see Figure 4.26. Edge e belongs to the boundary of subpolygon $\overline{B}$, and P is part of the boundary of subpolygon $\overline{A}$. Furthermore, chain C divides polygon R_t in two subpolygons A and B, where $A \supseteq \overline{A}$ and $B \supseteq \overline{B}$. More precisely, we have $\overline{A} = A \cap \overline{R}$ and $\overline{B} = B \cap \overline{R}$, compare Figure 4.27.

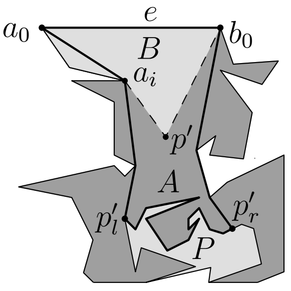

Figure 4.27: The polygonal chain $C = (a_i, p', b_0)$ divides polygon R_t in two polygons A and B. The bold polygonal chain indicates the boundary of polygon $\overline{R}$.

Now, we first show that any point of P visible from $\overline{B}$ lies between p'_l and p'_r, and that no point of $B \setminus \overline{B}$ is visible from P. Then, we use this fact to show that a point $r \in R_t$ belongs to $VA(P, Q)$, iff $r \notin B \setminus C$ and r is visible from P and Q. We shall first state a technical fact based on the general position of the vertices of R (that is, no three vertices of R are collinear).

Lemma 4.16. *With the notation from above, point p' does not belong to Q.*

Proof. Point p' is the intersection point of two visibility segments between vertices of R. We will show in the proof of Lemma 4.17 that the first segment s_1 is either $b_j a_k$

or $b_j p_b$, and that the second one, s_2, is either segment $a_i b_m$ or segment $a_i p_a$; in any case the endpoints of both segments s_1 and s_2 are vertices of R. We can assume $p_a \neq p_b$, because in that case we have $p_a = p_b = p' \in P$. Hence the endpoints of the visibility segments s_1 and s_2 are four *distinct* vertices of R. None of those four vertices can be the intersection point p', by general position of the vertices of polygon R. Now if p' belongs to Q, then point p' can only be a vertex of Q which implies that there are more than two vertices of R collinear – a contradiction. □

Lemma 4.17. *With the notation from above, let $p \in \partial P$ be a point that lies between p_l and p'_l or between p_r and p'_r, where $p \notin \{p'_l, p'_r\}$. Now for any given point $r \in \overline{B}$, it holds that p is not visible from r.*

Proof. By symmetry, we can assume w. l. o. g. that $p \neq p'_l$ lies between p_l and p'_l. There are two cases.

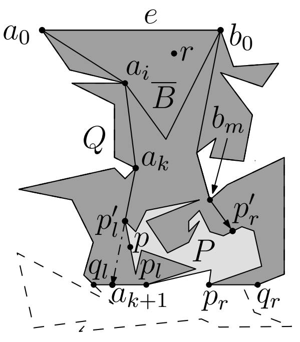

Figure 4.28: No point p, $p \neq p'_l$, between p_l and p'_l is visible from a point $r \in \overline{B}$.

1. Vertex p'_l lies tangentially to ray $\rho_{a_k}(p'_l)$, as depicted in Figure 4.28.
   Ray $\rho_{a_k}(p'_l)$ ends where it intersects the boundary of R_t at some point a_{k+1}, behind p'_l. Point a_{k+1} is visible from e, because the two shortest paths $\pi_{a_0,a_{k+1}}$ and $\pi_{b_0,a_{k+1}}$ are outwards convex. Therefore, $a_{k+1} \notin P$ holds and $p \notin \{p'_l, a_{k+1}\}$ lies between p'_l and a_{k+1}. It follows that the shortest path $\pi_{r,p}$ makes a left turn at its inner vertex p'_l. Hence, p is not visible from r.

2. Ray $\rho_{a_k}(p'_l)$ intersects the boundary of P at p'_l (this situation occurs at point p'_r depicted in Figure 4.28, compare the ray $\rho_{b_m}(p'_r)$).
   We first observe that (i) region $\overline{B}$ is contained in the region bounded by edge e and the two outwards convex shortest paths π_{a_0,p'_l} and π_{b_0,p'_l}, and that (ii) $\overline{B}$ is also contained in the region bounded by edge e and the two outwards convex shortest paths π_{a_0,p'_r} and π_{b_0,p'_r}, see Figure 4.29. Furthermore, because the segment $a_k p'_l$ is a visibility cut defined by a vertex $v \in \pi_{b_0,p'_l}$ and by vertex $a_k \in \pi_{a_0,p'_l}$, the three points v, a_k and p'_l are collinear; point a_k is an interior point of the segment $v p'_l$. Hence the segment $a_k p'_l$ is part of the last segment of both shortest paths π_{a_0,p'_l} and π_{b_0,p'_l}. Now because point $p \in P$ cannot be

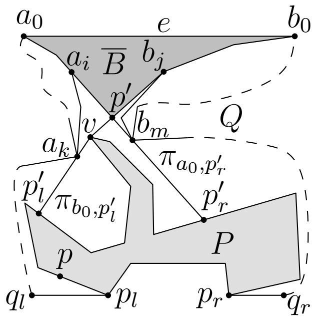

Figure 4.29: Region $\overline{B}$ is contained in the region bounded by edge e and the two shortest paths π_{a_0,p'_l} and π_{b_0,p'_l}. For convenience, most parts of Q are not displayed explicitly, but are indicated as dashed lines.

contained in segment $a_kp'_l$, (i) implies that it if $a_k \notin \overline{B}$, then the oriented shortest path $\pi_{r,p}$ must turn right at its inner vertex a_k, which implies that p is not visible from r. Thus in the following we will show that indeed $a_k \notin \overline{B}$ holds. If $v \in P$ (as shown in Figure 4.29), then the shortest path π_{a_0,p'_r} intersects π_{b_0,p'_l} at some point p' between v and b_j. Hence point a_k, which is interior point of the segment vp'_l, cannot be contained in the region bounded by edge e and the two shortest paths π_{a_0,p'_r} and π_{b_0,p'_r}. Now from (ii), it follows that $a_k \notin \overline{B}$. It remains to consider the case where $v \in Q$. Then, we have $v = b_j$ and $p'_l = p_b$ because segment vp'_l is the first segment of type QP on the shortest path π_{b_0,p'_l}; compare Figure 4.30. Furthermore, the segment vp'_l is a tangent to vertex a_k, and π_{a_0,p'_r} intersects the segment va_k. Assuming $a_k \in \overline{B}$ for a contradiction, (ii) implies

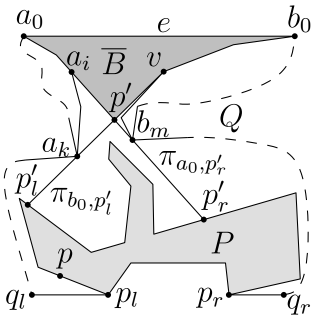

Figure 4.30: If the vertex v defining the visibility cut $a_kp'_l$ belongs to Q, then we have $v = b_j$ and $p'_l = p_b$.

that π_{a_0,p'_r} intersects segment va_k at point $p' = a_k \in Q$. This is a contradiction to Lemma 4.16. Altogether, we have shown $a_k \notin \overline{B}$ which implies that p is not visible from r. Using the same arguments we can show that if $p \neq p'_r$ lies between p_r and p'_r, then p is also not visible from r.

□

From Lemma 4.17 we obtain the following corollary.

Corollary 4.18. *If $r \in B$ and $p \in P$ are two mutually visible points of R_t, then p lies between p'_l and p'_r.*

Proof. Segment pr must intersect chain C at some point $r' \in \overline{B}$, and r' sees P. Now the correctness of the corollary follows from Lemma 4.17. □

Corollary 4.19. *No point $q \in Q$, that lies between a_0 and a_i or between b_0 and b_j, where $q \notin \{a_i, b_j\}$, is visible from P.*

Proof. We assume w.l.o.g. that q lies between a_0 and a_i and we assume for a contradiction that q is visible from some point $p \in P$. Now our assumption $q \notin \{a_i, b_j\}$ implies $i > 0$, $q \in B$, and that q lies in the region of R_t to the right of the oriented shortest path π_{a_0,a_i}. From Corollary 4.18, it follows that p lies between p'_l and p'_r and thus that p lies in the region of R_t to the right of the oriented shortest path π_{a_i,p'_r}; compare Figure 4.31. Since the oriented shortest path π_{a_0,p'_r} is bent outwards (and

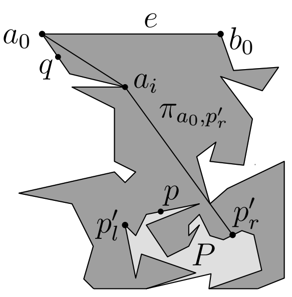

Figure 4.31: Point q (p) lies in the region of R_t to the right of the oriented shortest path π_{a_0,a_i} (π_{a_i,p'_r}).

the vertices of R are in general position), it makes a right turn at its inner vertex a_i. Hence, the oriented shortest path $\pi_{q,p}$ also makes a right turn at its inner vertex a_i, because $\pi_{q,p}$ cannot intersect π_{a_0,p'_r} transversally. Therefore, q is not visible from p – a contradiction. Using analogous arguments we can also show that if q lies between b_0 and b_j, then q is not visible from P. □

Now we are finally able to complete the proof of Lemma 4.15. Let pq be a segment in R_t of type PR_t that contains a point r, where $r \in B \setminus C$. Since $p \in P$ is visible from point $r \in B$, Corollary 4.18 implies that point p lies (on the boundary of R_t) between p'_l and p'_r. Hence before reaching point $r \in B \setminus C$, ray $\rho_p(q)$ must intersect both shortest paths π_{a_0,p'_r} and π_{b_0,p'_l}, and thus the intersection with chain C is transversal. Now Corollary 4.19 implies that after reaching point r, ray $\rho_p(q)$ can leave region $\overline{B}$ – and thus region B – only by intersecting edge e at some point e' (more precisely, the segment re' cannot contain any point of Q); otherwise a point of Q between a_0 and a_i

or between b_0 and b_j were visible from P. Hence no point $r \in B \setminus C$ belongs to the visibility area $VA(P, Q)$ between P and Q.

The final step in the proof of Lemma 4.15 is to show that a point $r \in A$ visible from P and from Q belongs to $VA(P, Q)$. By Lemma 4.13, this holds for the points in $A \setminus \overline{A}$, and, trivially, this also holds for the points $r \in P \cup Q$. So in the following we assume $r \in \overline{A} \setminus (P \cup Q)$, where $\overline{A} = A \cap \overline{R}$. Point r must lie, in $\overline{R}$, either to the right of the oriented shortest path π_{a_i,p'_r} or to the left of the oriented shortest path π_{b_j,p'_l}, compare Figure 4.32. Suppose w. l. o. g. that r lies to the right of $\pi_{a_i,p'_r} = (a_i, p_a, \ldots, p'_r)$

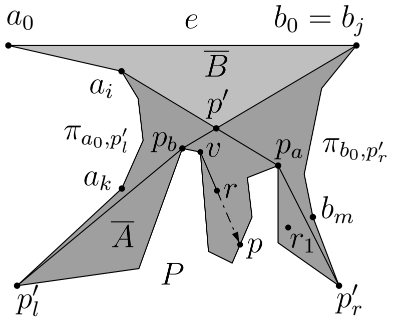

Figure 4.32: Point $r \in \overline{A}$ visible from P and from Q belongs to $VA(P, Q)$.

(analogous arguments hold if r lies to the left of $\pi_{b_j,p'_l} = (b_j, p_b, \ldots, p'_l)$). If r lies inside the region to the right of a segment of π_{p_a,p'_r} (which is of type PP) – compare point r_1 in Figure 4.32 – we obtain from Lemma 4.10, 2b (using $A = P$ and $B \cup D = Q$), that r belongs to $VA(P, Q)$. Otherwise, r lies in the region to the right of segment $a_i p_a$. As in the proof of Lemma 4.13, we consider the shortest path $\pi_{a_i,r}$ in $\overline{R}$ and denote by $v \in P \cup Q$ the inner vertex of $\pi_{a_i,r}$ adjacent to r, see Figure 4.32. Ray $\rho_v(r)$ must reach a point p on the boundary of R_t, behind r. Since the shortest path π_{a_i,p'_l} is a boundary subchain of $\overline{R}$ that is bent outwards, and $\pi_{a_i,p_a} = a_i p_a$ is a line segment, it follows that if $r \in \pi_{a_i,p'_l}$ or $r \in \pi_{a_i p_a}$ holds, then p lies on P between p'_l and p_a. Otherwise, if $r \notin \pi_{a_i,p'_l} \cup \pi_{a_i p_a}$, then the oriented shortest path $\pi_{a_i,p}$ cannot intersect π_{a_i,p'_l} or $\pi_{a_i p_a}$ after reaching r. Hence, p also lies on P between p'_l and p_a. Now if $v \in Q$ holds, then r belongs to $VA(P, Q)$ since r lies on segment vp of type QP. Otherwise, if $v \in P$, then by Lemma 4.10, 2b, point r also belongs to $VA(P, Q)$, because r is visible from Q. We have shown that if $r \in \overline{A}$ lies to the right of π_{a_i,p'_r} and is visible from P and Q, then r belongs to the visibility area between P and Q. Analogous arguments hold for the points of $\overline{R}$ to the left of π_{b_j,p'_l}.

We have shown how to compute, in linear time $O(|P| + |Q|)$, a polygonal chain C that divides R_t in two subpolygons A and B, such that no point in $B \setminus C$ belongs to $VA(P, Q)$ and every point in A visible from P and from Q belongs to $VA(P, Q)$. This completes the proof of the lemma. □

We recapitulate that in order to compute the visibility area between P and Q we first divide polygon R into the two polygons R_b and R_t. If edge e belongs to the boundary of R_b, or if $e \in \partial R_t$ and no point of P is, in R_t, visible from edge e, we use the algorithm presented in Section 4.5.2 to compute $\mathit{VA}(P,Q) = \mathit{VA}(P,P_e)$. It remains to consider the case where $e \in \partial R_t$ and some point of P is visible from e. After computing the visibility area between P and Q in R_b, we compute the visibility polygon of Q and of P in R_t. Now we compute chain C, see Lemma 4.15, and cut off from R_t the region B, keeping C as new boundary chain of R_t. Then we cut off all remaining points of R_t not visible from P or Q, using the visibility polygons of P and Q computed previously. Note that every point of chain C lies on a segment of type PQ, so C does not intersect with the boundary of the visibility polygon of P or Q in R_t – hence we can cut off region B and the points not visible from P and Q independently. In total, our algorithm runs in linear time $O(|P|+|Q|)$. The correctness of the algorithm follows from Lemma 4.15.

Summary

In sections 4.5.1, 4.5.2 and 4.5.3 we have shown how to compute the visibility area between two closed polygonal chains P and Q in linear time $O(|P| + |Q|)$. Also, deciding which of the given algorithms has to be used is done in linear time. This completes the proof of Theorem 4.12.

An open polygonal chain $C = (c_1, \ldots, c_n)$ can be treated as a closed polygonal chain $C' = (c_1, c_2, \ldots, c_n, \ldots, c_2, c_1)$. The visibility of C and C' is identical and we can modify our algorithm to also accept C' as input.

Corollary 4.20. *The visibility area between two disjoint open or closed polygonal chains P and Q in the plane can be computed in optimal time $O(|P| + |Q|)$.*

4.5.4 The Visibility Area in Presence of an Obstacle

Shanni Cai [12] showed how to extend the previous algorithms to compute the visibility area between two simple polygons P and Q whose boundaries are disjoint, in the presence of an additional obstacle polygon H. Polygon H is also simple, and its boundary is disjoint from the boundaries of polygons P and Q. The different cases considered are that either all three polygons are disjoint, or, any two of the three polygons are contained in the third one.

Theorem 4.21 (Cai [12])**.** *Given three simple polygons P, Q and H whose boundaries are pairwise disjoint, the visibility area $\mathit{VA}(P,Q)$ between P and Q in presence of obstacle polygon H can be computed in optimal linear time $O(|P| + |Q| + |H|)$.*

Chapter 5

Tolerant Algorithms

Assume we are interested in solving a computational task, e.g., sorting n numbers, and we only have access to an unreliable primitive operation, for example, comparison between two numbers. Suppose that each primitive operation fails with probability at most p and that repeating it is not helpful, as it will result in the same outcome. Can we still approximately solve our task with probability $1 - f(p)$ for a function f that goes to 0 as p goes to 0? While previous work studied sorting in this model, we believe this model is also relevant for other problems. We

- find the maximum of n numbers in $O(n)$ time,
- solve 2D linear programming in $O(n \log n)$ time,
- approximately sort n numbers in $O(n^2)$ time such that each number's position deviates from its true rank by at most $O(\log n)$ positions,
- find an element in a sorted array in $O(\log n \log \log n)$ time.

Our sorting result can be seen as an alternative to a previous result of Braverman and Mossel [10] who employed the same model. While we do not construct the maximum likelihood permutation, we achieve similar accuracy with a substantially faster running time. An extended abstract of this paper has appeared at ESA'11 [47].

5.1 Introduction

Many algorithms can be designed in such a way that the input data is accessed only by means of certain primitive queries. For example, in comparison-based sorting an algorithm might specify two key indices, i and j, and check whether the relation $q_i < q_j$ holds. Except for such yes/no replies, no other type of information on the key set $\{q_1, \ldots, q_n\}$ is necessary, or this information might not even be available. Similarly, in solving a linear program, given a point and a line, the primitive may return which side of the line the point is on.

Given an oracle that can answer all possible primitive queries on the input data, one can develop an algorithm without worrying about the input data type or numerical issues. This allows for more modular and platform-independent algorithm design. A natural question is what happens if the oracle errs?

There are several ways to model such errors. One natural model is that each time the oracle is queried it returns the wrong answer with a small error probability bounded by some $p > 0$, independent of past queries. In this case, repeating a query can be used to boost the success probability at the cost of additional work. In this paper we shall employ a different model introduced by Braverman and Mossel [10] in the context of sorting. The oracle errs on each query independently with probability at most p, but now each possible primitive query on the input is answered by the oracle *only once*.

There are good reasons to study this model. First, it may not be possible to repeat a query, as observed in [10]. For example, in ranking soccer clubs, or even just determining the best soccer club, the outcome of an individual game may differ from the "true" ordering of the two teams and it is impossible to repeat this game. A different example is ranking items by experts [10], which provides an inherently noisy view of the "true" ranking. Another reason to study this model is that the oracle may err on a particular query due to a technical problem whose consequences are deterministic, so that we would obtain the same wrong answer in any repetition. This is common in computational geometry algorithms due to floating point errors. Geometric algorithms based on primitive queries that additionally work with faulty primitives are more modular and tolerant to errors.

In this model, two natural questions arise: (1) Given the answers to all possible primitive queries, what is the best possible solution one can find if the computational time is unlimited? (2) How good of a solution can be found efficiently, i. e., by using only a subset of the set of oracle answers?

Previous work. Braverman and Mossel [10] consider the sorting problem and provide the following answers to the questions above. With high probability, they construct the maximum likelihood permutation σ with respect to the set of oracle answers. They prove that permutation σ does not place any key more than $O(\log n)$ positions away from its true position. This fact allows the computation of σ using only $O(n \log n)$ key comparisons. The algorithm employs dynamic programming and has running time in $O(n^{3+24c_3})$ for some $c_3 > 0$ that depends on the error probability p. In a subsequent paper [11] they also consider the problem of constructing a maximum likelihood permutation π for generating a given set of r randomly generated permutations $\{\pi_1, \ldots, \pi_r\}$ (cf. [51]), where for any two permutations π and π_j the probability of π generating π_j decreases exponentially with the number of inversions between π and π_j.

Feige *et al.* [30] and Karp and Kleinberg [43] study the model where repeated queries are always independent. Searching games between questioner and a lying responder have been extensively studied in the past; see Pelc [57]. In his terminology, our model allows the responder random lies in response to nonrepetitive comparison questions in an adaptive game. Another related model is studied by Blum, Luby and Rubinfeld [8]. They consider self-testing and self-correction under the assumption that one is given a program P which computes a function f very quickly but possibly not very reliably. The difference with our approach is that we need to work with unreliable *primitives.*

In a series of papers culminating in work by Finocchi *et al.* [31], a thorough study of resilient sorting, searching, and dictionaries in a faulty memory RAM model was performed. Here, at most δ memory words can be corrupted, while there is a small set of $O(1)$ memory words that are guaranteed not to be corrupted. The main difference is that only non-corrupted keys need to be sorted correctly. In contrast to this approach, we need to ensure that with high probability every key appears near its true position.

Our results. In this paper we assume the same model as in [10]. That is, primitive queries may fail independently with some probability at most p and noisy answers to all possible queries are pre-computed and available to the algorithm. While [10] only considers the problem of sorting, we initiate the study of a much wider class of problems in this model.

In Section 5.2 we show how to compute, with probability $1 - f(p)$, the maximum of n elements in $O(n)$ time. Here $f(p)$ is a function independent of n that tends to 0 as p does. There is only a constant factor overhead in the running time for this problem in this model. In Section 5.3 we show how obtain running time guarantees for (randomized or deterministic) tolerant algorithms, at the cost of success probability. In Section 5.4 a sorting algorithm is discussed. Like the noisy sorting algorithm presented by Braverman and Mossel [10], ours guarantees that each key is placed within distance $O(\log n)$ of its true position, even though we do not necessarily obtain the maximum likelihood permutation. Also, we need $\Theta(n^2)$ rather than $O(n \log n)$ key comparisons. However, our algorithm is faster than the algorithm of [10], as the running time is $O(n^2)$, providing a considerable improvement over the $O(n^{3+24c_3})$ time of [10]. Finally, in Section 5.5, we briefly discuss the noisy search problem. By a random walk argument we show that the true position of a search key can be determined in a correctly sorted list with probability $(1 - 2p)^2$ if all n comparison queries are made. With $O(\log n \log \log n)$ comparisons, we can find the correct position with probability $1 - f(p)$. As an application of our max-finding result, we present in Section 5.6 an $O(n \log n)$ time algorithm for linear programming in two dimensions. Linear programming is a fundamental geometric problem which could suffer from errors in primitives due, e.g., to floating point errors. Our algorithm is modular, being built from simple point/line side comparisons, and robust, given that it can tolerate faulty primitives. It is based on an old technique of Megiddo [52]. We do not know how to modify more "modern" algorithms for linear programming to fit our error model. The correctness and running time hold with probability $1 - f(p)$.

5.2 Finding the Maximum of n Numbers

We are given an unordered set $a_1, \ldots, a_n$ of n distinct numbers and we would like to output the maximum using only comparison queries. Each comparison between input elements fails independently with probability p. If the same comparison is made twice, the same answer is given.

Our algorithm LinearMax consists of two phases and each phase consists of several stages. In each stage the set of input numbers is pruned by a constant fraction such that with sufficiently high probability the maximum remains in the set. The pruning is done by sampling a set S and comparing each number outside of S with each number inside of S. During the first phase the size of the sample set increases with each stage. The second phase begins once the size of the set of numbers is successfully reduced to below some critical value. Then, we sample fewer elements in each stage so that the probability to accidently sample the maximum remains small. We will say that a stage has an error if either the maximum is removed during the stage or the input is not pruned by a constant fraction. If during a stage the input set is not sufficiently pruned, the algorithm stops and outputs "error". We remark that the purpose of this stopping rule is to simplify the analysis (this way, we make sure that any stage is executed only once and allows us directly to sum up error probabilities of different stages).

If the maximum is removed this will not be detected by the algorithm. In our analysis we derive a bound for the error probability of each stage and then use a union bound to bound the overall error probability.

Theorem 5.1. *For any constant $1/2 > \lambda > 0$ there exists a constant $C = C(\lambda)$ such that for any $p \leq 1/64$, algorithm* LinearMax *succeeds with probability at least $1 - C \cdot p^{\frac{1}{2}-\lambda}$.*

```
LinearMax($M, p$)
1.   $n = |M|$
2.   while $|M| \geq \max\{n^{1-\lambda}, \frac{100}{\sqrt{p}}\}$ do
3.      $j = 0$
4.      Select $i \in \mathbb{N}$ such that $n \cdot \left(\frac{15}{16}\right)^{i-1} \geq |M| > n \cdot \left(\frac{15}{16}\right)^{i}$
5.      while $j < 100 \cdot i \cdot \log(1/p)$ and $|M| > n \cdot \left(\frac{15}{16}\right)^{i}$ do
6.         Select a set $S$ of $s_i = 4i$ elements from $M$ uniformly at random
7.         Remove all elements in $S$ from $M$
8.         Compare all elements from $S$ with $M$
9.         Let $M$ be the list of elements larger than at least $\frac{3}{4} \cdot s_i$ elements of $S$
10.        $j = j + 1$
11.     if $|M| > n \cdot \left(\frac{15}{16}\right)^{i}$ then output "error" and exit
12.  while $|M| \geq \frac{100}{\sqrt{p}}$ do
13.     $j = 0$
14.     Select $i'$ such that $\left(\frac{16}{15}\right)^{i'} \geq |M| > \left(\frac{16}{15}\right)^{i'-1}$ and $\left(i' - \log_{16/15} \frac{100}{\sqrt{p}} + 1\right) \in \mathbb{N}$
```

```
15.       $i = i' - \log_{16/15} \frac{100}{\sqrt{p}} + 1$
16.       **while** $j < 100 \cdot i \cdot \log(1/p)$ and $|M| > \left(\frac{16}{15}\right)^{i'-1}$ **do**
17.           Select a set $S$ of $s_i = 4i$ elements from $M$ uniformly at random
18.           Remove all elements in $S$ from $M$
19.           Compare all elements from $S$ with $M$
20.           Let $M$ be the list of elements larger than at least $\frac{3}{4} \cdot s_i$ elements of $S$
21.           $j = j + 1$
22.       **if** $|M| > \left(\frac{16}{15}\right)^{i'-1}$ **then output** "error" and **exit**
23.   Let *currentMax* be any element $x$ from $M$
24.   Remove $x$ from $M$
25.   **while** $M \neq \emptyset$
26.       Remove an arbitrary element $x$ from $M$
27.       **if** $x >$ *currentMax* **then** *currentMax* $= x$
28.   **return** *currentMax*
```

Proof of Theorem 5.1. Phase 1 of the algorithm begins at line 2 and Phase 2 begins at line 12. Each phase consists of several stages, which correspond to the value of i. We refer to the loops starting in line 5 and in line 16 as to the *inner loop* respectively of Phase 1 and of Phase 2. Lines 11 and 22 ensure that in each stage i of Phase 1 and of Phase 2 the inner loop is entered at most once. Therefore, in Phase 1 and in Phase 2 the body of the inner loop is, in any given stage i, executed at most $100 \cdot i \cdot \log(1/p)$ times. More precisely, in Phase 1, i starts with value 1 and increases only while $i \leq \log_{16/15}\left(n^{\lambda}\right) + 2$ and $i \leq \log_{16/15}\left(\frac{n\sqrt{p}}{100}\right) + 2$ hold. In Phase 2, $i = \left\lceil \log_{16/15}\left(\frac{|M|\sqrt{p}}{100}\right)\right\rceil + 1$ decreases in each stage while $i \geq 1$ holds, because $\left(\frac{16}{15}\right)^{i'} \geq |M| \geq 100/\sqrt{p}$ implies $i \geq 1$, and $i \leq 0$ implies $|M| < 100/\sqrt{p}$.

We first analyse the expected running time of the algorithm. We observe that the running time is dominated by the number of comparisons the algorithm performs. In the first phase in stage i in each iteration of the loop the algorithm samples $4i$ elements uniformly at random and compares them to at most $n \cdot \left(\frac{15}{16}\right)^{i-1}$ elements of M. We will show that the expected number of loop iterations in each stage is $O(1)$. We need the following lemma.

Lemma 5.2. *The probability that in a fixed stage i of Phase 1, after more than $100k$ iterations of the body of the inner loop, $|M| > n \cdot \left(\frac{15}{16}\right)^{i}$ holds is at most $\left(\frac{1}{2}\right)^{k}$. Similarly, the probability that in a fixed stage i of Phase 2, after more than $100k$ iterations of the body of the inner loop, $|M| > \left(\frac{16}{15}\right)^{i'-1}$ holds is at most $\left(\frac{1}{2}\right)^{k}$.*

Proof. We show that in a single iteration of the loop, with probability at least 1/15 the size of the list shrinks by a factor of 15/16. Let n be the current size of M in the current stage i. With probability 1/2, S contains at least $|S|/2$ items in the top $\lceil n/2 \rceil$ ranked elements of M. Call this event $\mathcal{E}$. Suppose $\mathcal{E}$ occurs, and let S^H be this subset of items of S. Consider an arbitrary element $x \in (M \setminus S)$ among the least

$\lfloor n/2 \rfloor$ elements of M. Since we are in Phase 1 or 2, we know that $n \geq 100/\sqrt{p} \geq 100$ and so we can say,

$$\frac{\lfloor n/2 \rfloor}{n} \geq \frac{50}{101}.$$

In order for x to be included in M after processing line 9 (or line 20, if we are in Phase 2), it must be reported larger than at least half of the elements of S^H. The probability this happens is at most

$$\begin{aligned} p^{|S^H|/2}\binom{|S^H|}{|S^H|/2} &\leq p^{|S^H|/2} 2^{|S^H|} \\ &\leq \left(\frac{1}{8^2}\right)^{|S^H|/2} 2^{|S^H|} \\ &\leq \left(\frac{1}{8}\right)^{|S^H|} 2^{|S^H|} \\ &= \left(\frac{1}{4}\right)^{|S^H|} \\ &\leq \frac{1}{2^{|S|}}, \end{aligned}$$

where we have used that $p \leq \frac{1}{64}$ and $|S^H| \geq |S|/2$. For each element x among the least $\lfloor n/2 \rfloor \geq 50n/101$ elements of M we have that either $x \in S$ and therefore x is later removed from M, or x stays in M with probability at most $1/2^{|S|}$. Let $\tilde{n}$ be the number of elements in M after the loop iteration, i. e., after in line 9 (or in line 20, if we are in Phase 2) the elements have been removed from M. It follows that

$$\mathbf{E}[\tilde{n}] \leq \frac{1}{2} \cdot n + \frac{1}{2} \cdot \left(\frac{50n}{101} \cdot \frac{1}{2^{|S|}} + \frac{51n}{101}\right) \leq \frac{7n}{8},$$

where the last inequality follows because $|S| \geq 4$. We say that an iteration is successful, if $\tilde{n} \leq \frac{15}{16} \cdot n$. By Markov's inequality, with probability at least 1/15, an iteration is successful:

$$\begin{aligned} \mathbf{Pr}\,[\text{iteration } \textit{not} \text{ successful}] &= \mathbf{Pr}\left[\tilde{n} > \frac{15}{16} \cdot n\right] \\ &\leq \frac{\mathbf{E}\,[\tilde{n}]}{\frac{15}{16} \cdot n} \\ &\leq \frac{7n}{8} \cdot \frac{16}{15n} \\ &= \frac{14}{15} \end{aligned}$$

If an iteration is successful the algorithm proceeds with a different stage. The probability that no iteration out of $100k$ is successful is at most

$$\left(\frac{14}{15}\right)^{100k} \leq \left(\frac{1}{2}\right)^{k}$$

$\square$

In the remainder of this section, by n we denote (again) the number of elements in M when algorithm LinearMax is invoked, i. e., the number computed in the first line of the algorithm. From the lemma, it follows immediately that the expected number of loops in a fixed stage is $O(1)$. By linearity of expectation, the overall expected running time for Phase 1 is

$$O\left(n \cdot \sum_{i=1}^{\infty} \mathbf{E}\left[\text{iterations in Stage } i\right] \cdot i \cdot \left(\frac{15}{16}\right)^{i-1}\right) = O(n),$$

since, in stage i, in each iteration at most $|M| \cdot 4i \leq n \cdot 4i \cdot \left(\frac{15}{16}\right)^{i-1}$ comparisons are made.

In order to analyse the second phase, let $i_0 = O(\log n)$ be the maximal value of i, *i. e.* the first value of i computed in line 15 when the algorithm enters Phase 2. We observe that for any stage $i \leq i_0$ we have $\left(\frac{16}{15}\right)^i \leq n^{1-\lambda}$. It follows that the expected running time of the second phase is

$$\begin{aligned} O\left(\sum_{i=1}^{i_0} \mathbf{E}\left[\text{iterations in Stage } i\right] \cdot i \cdot \left(\frac{16}{15}\right)^{i}\right) &\leq O\left(\sum_{i=1}^{i_0} i \cdot n^{1-\lambda}\right) \\ &= O\left(n^{1-\lambda} \cdot (\log n)^2\right) \\ &= O\left(n \cdot \frac{(\log n)^2}{n^{\lambda}}\right) \\ &= O(n). \end{aligned}$$

The running time of the standard maximum search is also $O(n)$. Hence, the overall expected running time is $O(n)$. We continue by analysing the error probability of the algorithm. An error happens at any stage i if

(a) the maximum is contained in the sample set S,

(b) the maximum is reported to be smaller than $\frac{1}{4} \cdot s_i$ of the elements of S, or

(c) after the body of an inner loop has been executed $100i \cdot \log(1/p)$ times in this stage, the algorithm stops (in line 11 or 22) with output "error".

We start by analysing (a) during the first phase. The error probability at each loop iteration is $|S|/|M|$. The number of items in M in stage i of Phase 1 is at least $n \cdot \left(\frac{15}{16}\right)^i \geq n^{1-\lambda}$. We also have $i = O(\log n)$. Hence, $|S|/|M| \leq O(\log n)/n^{1-\lambda}$ and summing up over the at most $O(i \log(1/p)) \leq O(\log^2 n)$ loop iterations, we obtain that the overall error probability of item (a) in Phase 1 is at most $O(\log^3 n)/n^{1-\lambda} \leq O(1)/n^{1-2\lambda}$. Since we are in Phase 1, $n \geq 100/\sqrt{p}$ holds which implies $n^{1-2\lambda} \geq (100/p^{1/2})^{1-2\lambda} \geq 1/(p^{1/2-\lambda})$. Altogether, we obtain that the overall error probability of item (a) in Phase 1 is at most $\frac{C}{5} \cdot p^{1/2-\lambda}$ for a sufficiently large constant $C > 0$. In the second phase, the error probability of item (a) is at most

$$\begin{aligned}
\sum_{i=1}^{\infty} \frac{4 \cdot i}{\left(\frac{16}{15}\right)^{i'-1}} &= \sum_{i=1}^{\infty} \frac{4 \cdot i}{\left(\frac{16}{15}\right)^{i+\log_{16/15} \frac{100}{\sqrt{p}}-2}} \\
&= \sum_{i=1}^{\infty} \frac{4 \cdot i}{\frac{100}{\sqrt{p}} \cdot \frac{15}{16} \cdot \left(\frac{16}{15}\right)^{i-1}} \\
&= \frac{64\sqrt{p}}{1500} \cdot \sum_{i=1}^{\infty} i \cdot \left(\frac{15}{16}\right)^{i-1} \\
&\leq \frac{C}{5} \cdot \sqrt{p},
\end{aligned}$$

for a sufficiently large constant $C > 0$, using the definition $i = i' - \log_{16/15} \frac{100}{\sqrt{p}} + 1$ (see line 15 of the algorithm).

We continue by analysing (b). Here, an error occurs in stage i if at least $\frac{1}{4} \cdot s_i$ comparisons fail. By the following lemma, this probability is small.

Lemma 5.3. *Let $1 \geq p \geq 0$ be the failure probability of a comparison. Let $k > 0$ be a multiple of 4. The probability that at least $k/4$ out of k comparisons fail is at most $(4p)^{k/4}$.*

Proof. The probability for at least $k/4$ failures is at most

$$p^{k/4} \cdot \binom{k}{k/4} .$$

Plugging in

$$\begin{aligned}
\binom{4n}{n} &= \prod_{j=0}^{n-1} \frac{4n-j}{n-j} = \prod_{j=0}^{n-1} \left(\frac{3n}{n-j} + 1\right) \\
&\leq \prod_{j=0}^{n-1} \left(\frac{3n}{n} + 1\right) = 4^n
\end{aligned}$$

using $n = k/4$ we obtain

$$p^{k/4} \cdot \binom{k}{k/4} \leq p^{k/4} \cdot 4^{k/4} = (4p)^{k/4} .$$

□

Since, $p \leq 1/64$, it follows by a union bound that for $C > 0$ sufficiently large, the probability of failure in Phase 1 is at most $\sum_{i=1}^{O(\log n)} 100i \cdot \log(1/p) \cdot (4p)^i \leq \frac{C}{10} \cdot \sqrt{p}$, because in stage i the maximum is compared with $k = 4i$ elements, up to $100i \cdot \log(1/p)$

times. The same estimate holds for the error probability of item (b) in Phase 2. Therefore, the total error probability for item (b) is bounded by $\frac{C}{5} \cdot \sqrt{p}$. Next, we analyse (c). By Lemma 5.2 and since $p \leq 1/64$, we have that in both Phase 1 and Phase 2 respectively the error probability for this item is bounded by

$$\sum_{i=1}^{\infty} \left(\frac{1}{2}\right)^{i \cdot \log \frac{1}{p}} = \sum_{i=1}^{\infty} p^i = p \sum_{i=0}^{\infty} p^i \leq \frac{C}{10} \cdot \sqrt{p} \ ,$$

for $C > 0$ a sufficiently large constant. Finally, we consider the error probability of the standard maximum search. A maximum search over a set of n items uses $n-1$ comparisons. If $n \leq 100/\sqrt{p}$ then the expected number of errors is at most $100p/\sqrt{p} = 100\sqrt{p}$. Let X be the random variable for the number of errors. Since the number of errors is integral, by Markov's inequality we get that the probability of error is

$$\mathbf{Pr}\left[\text{at least one error occurs}\right] \leq \mathbf{Pr}\left[X \geq \frac{1}{100\sqrt{p}} \cdot \mathbf{E}\left[X\right]\right] \leq 100\sqrt{p} \leq \frac{C}{5} \cdot \sqrt{p}$$

for a sufficiently large constant C. Summing up all errors yields an overall error probability of at most $\frac{C}{5} \cdot p^{\frac{1}{2}-\lambda} + \frac{4C}{5} \cdot \sqrt{p} \leq C \cdot p^{\frac{1}{2}-\lambda}$. □

5.3 Expected Running Time vs. Worst-Case Running Time

Given that primitive operations fail with a certain probability, we observe that the runtime of even a deterministic algorithm depends on the randomness due to the probability of error for each primitive operation. However, sometimes we are not only interested in tolerant algorithms with good expected running time, but also in algorithms with the same asymptotic running time in the worst case. We now show how to construct such algorithms at the cost of success probability. Note that the construction works for deterministic and randomized algorithms alike. At the end of this section, we will show how to construct a modified tolerant algorithm LINEARMAX* which finds the maximum of n given numbers, and whose runtime is $O(n)$ in the worst case.

Let ALG denote a tolerant algorithm that solves a computational task. That is, ALG computes the correct result with probability $\geq 1 - f(n,p)$ where p is an upper bound on the error probability for the primitive operations used by ALG, and n denotes the size of the input. Moreover, function f goes to 0 as p goes to 0 and / or n goes to ∞. If $T(n)$ is (an upper bound on) the *expected* running time of ALG, then the following holds.

Theorem 5.4. *Given a tolerant algorithm* ALG *that computes the correct result with probability* $1 - f(n,p)$ *and a positive function* $g(n,p)$*, we can construct a tolerant algorithm* ALG* *that solves the same computational task as algorithm* ALG*, with probability* $1 - f(n,p) - g(n,p)$*. The worst-case running time of* ALG* *is* $T(n)/g(n,p)$*.*

Proof. We modify algorithm ALG to obtain ALG* as follows. Algorithm ALG* performs the same calculations as algorithm ALG. If the algorithm has not terminated after $T(n)/g(n,p)$ time, the algorithm terminates with output "error" (or the algorithm returns the currently most promising solution). Since the claimed runtime bound obviously holds, it remains to prove that algorithm ALG* computes the correct result with probability $1-f(n,p)-g(n,p)$. For the analysis, we run the unmodified algorithm ALG on the same input as ALG* and consider the following two events.

- $\mathcal{E}_1$ = ALG does not compute the correct result.
- $\mathcal{E}_2$ = ALG needs more than $T(n)/g(n,p)$ computation time.

By assumption, we have $\mathbf{Pr}[\mathcal{E}_1] \leq f(n,p)$. If we denote the running time of algorithm ALG with T, then by Markov's inequality we have

$$\begin{aligned}
\mathbf{Pr}[\mathcal{E}_2] &\leq \mathbf{Pr}[T \geq T(n)/g(n,p)] \\
&\leq \mathbf{Pr}[T \geq \mathbf{E}[T]/g(n,p)] \\
&\leq \frac{\mathbf{E}[T]}{\mathbf{E}[T]/g(n,p)} \\
&= g(n,p).
\end{aligned}$$

The theorem now follows by a union bound. □

Corollary 5.5. *Given a tolerant algorithm* Alg *that computes the correct result with probability* $1 - f(n,p)$*, where for any constant* $p > 0$ *the function* $f(\cdot,p)$ *is bounded from below by some positive constant* c_p*, there is a tolerant algorithm* Alg* *that solves the same computational task as algorithm* Alg*, with probability* $1 - 2f(n,p)$*. The worst-case running time of* Alg* *is* $O(T(n))$*.*

Proof. We construct ALG* according to Theorem 5.4, using $g(n,p) := f(n,p)$. By construction, the correctness probability of Alg* is at least $1 - f(n,p) - g(n,p) = 1 - 2f(n,p)$. Moreover, since p is constant, we have by assumption that $f(n,p) \geq c_p$ holds. Therefore the running time of Alg* is at most $T(n)/g(n,p) = T(n)/f(n,p) \leq T(n)/c_p = O(T(n))$. □

We now give a modification of our LinearMax algorithm, with worst-case running time $O(n)$ (where again $n = |M|$ is the size of the input). Note that "line x" always refers to line x of the algorithm LinearMax.

LinearMax$^*(M,p)$

- Run algorithm LinearMax(M,p).
- Stop the execution of the algorithm after $T(n)/(C \cdot p^{\frac{1}{2}-\lambda})$ time, where $C = C(\lambda)$ is the same constant as in Theorem 5.1, and $T(n) = O(n)$ is the expected running time of algorithm LinearMax.
  - If the algorithm has been stopped while processing line 8 or 19, jump immediately to line 23.
  - Replace line 8 and line 19 by an unconditional jump to line 23.
- Continue executing the LinearMax algorithm, but with the above modification to lines 8 and 19.

Our stopping rule guarantees that the running time of algorithm LinearMax* is $O(n)/(C \cdot p^{\frac{1}{2}-\lambda}) + O(n) = O(n)$. Using the same arguments as in the proofs of Theorem 5.4 and Corollary 5.5, we have the following.

Corollary 5.6. *Algorithm* LinearMax$^*(M,p)$ *succeeds with probability at least* $1 - C \cdot p^{\frac{1}{2}-\lambda}$*, for any* $p \leq 1/64$ *and* $1/2 > \lambda > 0$ *where* $C = 2C(\lambda)$ *is a large enough constant, in worst-case time* $O(|M|) = O(n)$*.*

5.4 Sorting

In the following section we are given a set S of n distinct keys and the comparison relation $<_E$ with errors. We present an algorithm SortWithBuckets that computes an output sequence such that the position of every key in this sequence differs from its rank by at most an additive error of $O(\log n)$. In the following, we use $\mathrm{rank}(x,R)$ to refer to the (true) rank of input key x within $R \subseteq S$, i. e., $1 + |\{y \in R : y < x\}|$. We also use $\mathrm{rank}_E(x,R)$ to refer to the *virtual* rank of x with respect to a set R, i. e.,

$1 + |\{y \in R : y <_E x\}|$. Note that there may be more than one key having the same virtual rank.

The algorithm. In the following, we will assume that n is a power of 2. If this is not the case, we can add additional special items which will be assumed to be larger than any input key and run our algorithm on the modified input. Here we may assume no errors in the comparisons as the algorithm can keep track of these items. The algorithm will partition the set $\{1, \dots, n\}$ into buckets each corresponding to a set of 2^i consecutive numbers for certain i. We call this set the *associated range* of the bucket. At the beginning we will assume that there is a single bucket with associated range $\{1, .., n\}$. In the next step, we will subdivide this bucket into two buckets, with associated ranges $\{1, \dots, n/2\}$ and $\{n/2 + 1, \dots, n\}$, respectively. Then the two resulting buckets are further subdivided into four buckets, and so on. The algorithm stops, if the associated ranges contain $O(\log n)$ numbers. Ideally, we would like our algorithm to maintain the invariant that each bucket contains all input numbers whose ranks are in its associated range, e.g., the bucket corresponding to numbers $\{1, \dots, 2^i\}$ is supposed to contain the 2^i smallest input numbers. Due to the comparison errors, the algorithm cannot exactly maintain this invariant. However, we can almost maintain it in the sense that an item with rank k is either in the bucket whose associated range contains k or it is in one of the neighbouring buckets.

This is done as follows. Let us consider a set of buckets B_j with associated ranges of 2^i numbers such that bucket B_j has associated range$\{(j-1)2^i + 1, \dots, j2^i\}$. Now assume that the input numbers have been inserted into these buckets in such a way that our relaxed invariant is satisfied. For each bucket B_j let us use S_j to denote the set of input keys inserted into B_j. Now we would like to refine our buckets, i. e., insert the input numbers in buckets B'_j with associated ranges $\{(j-1)2^{i-1} + 1, \dots, j2^{i-1}\}$. This is simply done by inserting an item x that is previously in bucket B_j into bucket B'_r, where

$$r = \left\lceil \frac{\mathrm{rank}_E\left(x, \bigcup_{j-2 \le k \le j+2} S_k\right) + \left|\bigcup_{k<j-2} S_k\right|}{2^{i-1}} \right\rceil .$$

Thus, the algorithm computes the cardinality of the buckets up to B_{j-3} and adds to it the virtual rank of x with respect to buckets $B_{j-2}, \dots, B_{j+2}$. The idea behind this approach is that $\mathrm{rank}(x, S)$ can be written as $\left|\bigcup_{k<j-3} S_k\right| + \mathrm{rank}(x, \bigcup_{j-2 \le k \le j+2} S_k)$ since, by our relaxed invariant, the keys in buckets $1, \dots, j-3$ are smaller than x and the keys in $j+3, \dots$ are larger than x. Now, since we do not have access to $\mathrm{rank}(x, \bigcup_{j-2 \le k \le j+2} S_j)$ we approximate it by $\mathrm{rank}_E(x, \bigcup_{j-2 \le k \le j+2} S_j)$. Since the rank involves fewer elements when the ranges of the buckets decrease, this estimate becomes more and more accurate.

Analysis. Thus, it remains to prove that the relaxed invariant is maintained. The difficulty in analysing this (and other) algorithms is that there are many dependencies between different stages of the algorithm. In our case, the bucket of an element x is highly dependent on the randomness of earlier iterations. Since analysing such algorithmic processes is often close to impossible, we use a different approach. We first show that certain properties of the comparison relation $<_E$ hold for certain sets of elements with high probability. Then we show that these properties already suffice to prove that the algorithm sorts with additive error $O(\log n)$.

Lemma 5.7. *Let $p \leq 1/20$. Let $R \subseteq S$ be a set of $|R| = 9{\cdot}2^i \geq 100000 \log n$ keys and let $x \in R$. Let $X = |\{y \in R : x < y$ and $y <_E x\}| + |\{y \in R : x > y$ and $y >_E x\}|$ be the number of false comparisons of x with elements from R. Then $\mathbf{Pr}\left[X \geq 2^{i-1}\right] \leq n^{-29}$.*

Proof. For simplicity of calculations, we allow an additional 'dummy error' corresponding to a self-comparison of x. Then we have $\mathbf{E}[X] = p \cdot |R|$. We will assume that $p = 1/20$ as this maximizes the expected number of errors. Hence, using $X = \sum X_i$ with X_i being independent $0-1-$random variables with $\mathbf{Pr}[X_i = 1] = p$ we can apply Chernoff bounds to obtain

$$\begin{aligned}
\mathbf{Pr}\left[X \geq 2^{i-1}\right] &= \mathbf{Pr}\left[X \geq \frac{\mathbf{E}[X]}{18p}\right] \\
&= \mathbf{Pr}\left[X \geq \left(1 + \frac{1}{9}\right) \cdot \mathbf{E}[X]\right] \\
&\leq e^{-\frac{1}{3}\cdot\left(\frac{1}{9}\right)^2 \cdot \mathbf{E}[X]} \\
&= e^{-\frac{1}{243}\cdot\mathbf{E}[X]} \ .
\end{aligned}$$

The correctness of the lemma follows using

$$e^{-\frac{1}{243}\cdot\mathbf{E}[X]} = e^{-\frac{|R|}{4860}} \leq e^{-\frac{100000}{4860}\cdot\log n} = e^{-\frac{100000}{4860 \ln 2}\cdot\ln n} \leq n^{-29}.$$

□

Using the same arguments as in the proof of Lemma 5.7, we obtain the following.

Lemma 5.8. *Let $p \leq 1/20$, S be a set of $|S| \geq 100000 \log n$ keys and $x \in S$. Let X be the number of false comparisons of x with elements from S. Then $\mathbf{Pr}[X \geq |S|/16] \leq n^{-29}$.*

Proof. Again, we allow an additional 'dummy error' corresponding to a self-comparison of x, and we assume $p = 1/20$ as this maximizes the number of errors. As in the proof of Lemma 5.7, using $X = \sum X_i$ with X_i being independent $0-1-$random variables with $\mathbf{Pr}[X_i = 1] = p$, we can apply Chernoff bounds to obtain

$$\begin{aligned}
\mathbf{Pr}\left[X \geq \frac{|S|}{16}\right] &= \mathbf{Pr}\left[X \geq \frac{\mathbf{E}[X]}{16p}\right] \\
&\leq \mathbf{Pr}\left[X \geq \frac{\mathbf{E}[X]}{18p}\right] \\
&\leq e^{-\frac{1}{243}\cdot\mathbf{E}[X]} \ .
\end{aligned}$$

The correctness of the lemma follows using

$$e^{-\frac{1}{243}\cdot\mathbf{E}[X]} = e^{-\frac{|S|}{4860}} \leq e^{-\frac{100000}{4860}\cdot\log n} = e^{-\frac{100000}{4860 \ln 2}\cdot\ln n} \leq n^{-29}.$$

□

Corollary 5.9. *For* $\log(100000 \log n) \leq i \leq \log n$ *and* $1 \leq j \leq \max\{n/2^i - 8, 1\}$ *let* $S_{ij} = \{x \in S : (j-1) \cdot 2^i + 1 \leq \mathrm{rank}(x) \leq (j+8) \cdot 2^i\}$. *With probability at least* $1 - 1/n^{26}$ *we have that every* $x \in S_{ij}$ *has less than* 2^{i-1} *comparison errors with elements from* S_{ij}.

Proof. For all values of i where $\log(100000 \log n) \leq i \leq \log n - 4$ we can apply Lemma 5.7 for each S_{ij} since the corresponding number of buckets is $n/2^i \geq 9$, and therefore $|S_{ij}| = 9 \cdot 2^i \geq 100000 \log n$ holds for any value of j, where $1 \leq j \leq n/2^i - 8$.

For any value of $i \in \{\log(n) - 3, \ldots, \log(n)\}$, we have $j = 1$ and $S_{ij} = S$, so we can apply Lemma 5.8 to bound the probability that some key $x \in S$ has at least $\frac{n}{16}$ comparison errors with elements in S. Observe that $\frac{n}{16} = 2^{\log(n) - 4} \leq 2^{i-1}$ holds for any value of $i \in \{\log(n) - 3, \ldots, \log(n)\}$.

The number of choices for indices i, j and $x \in S_{ij}$ is bounded by $\log n \cdot n \cdot n$. Hence by a union bound, the probability that there are at least 2^{i-1} false comparisons with any element $x \in S_{ij}$ in any of the sets S_{ij} is at most $n^2 \log(n)/n^{29} \leq n^{-26}$. □

We now claim that if the comparison relation $<_E$ satisfies Corollary 5.9 then our algorithm computes a sequence such that any element deviates from its true rank by at most $O(\log n)$. In order to prove this claim, let us assume that our relaxed invariant is maintained for a set of buckets B_j with associated ranges of size $2^i \geq 100000 \log n$. Let x be an element and j^* be the bucket whose associated range contains x. By our relaxed invariant, x is either in bucket B_{j^*-1}, B_{j^*} or B_{j^*+1}. Hence, to sort x into the next finer bucket, the algorithm inspects (a subset of) buckets $B_{j^*-3}, \ldots, B_{j^*+3}$. By our relaxed invariant, these buckets only contain elements x with $\mathrm{rank}(x) \in S_{i\ell}$ for some value $1 \leq \ell \leq \max\{n/2^i - 8, 1\}$. In general we have $\ell = j^* - 4$. One exception is the case $n/2^i - 8 \leq 1$ where we use $\ell = 1$ (for the analysis, not in the algorithm). The other exception is where $n/2^i - 8 > 1$ and (1) $j^* - 4 < 1$ or (2) $j^* - 4 > n/2^i - 8$. Then, we use (1) $\ell = 1$ or (2) $\ell = n/2^i - 8$. Now, in order to sort x into a finer bucket, the algorithm compares x with a subset of elements from $S_{i\ell}$. Therefore, the number of errors in this comparison is certainly bounded by the number of errors within $S_{i\ell}$, which is less than 2^{i-1}. Since the next finer buckets have associated ranges of size 2^{i-1} and the virtual rank of x differs from its true rank at most by the number of comparison errors, this implies that our relaxed invariant will be maintained. We summarize our results in the following theorem.

Theorem 5.10. *Let* $p \leq 1/20$. *There is a tolerant algorithm that given an input set* S *of* n *numbers computes in* $O(n^2)$ *time and with probability at least* $1 - 1/n^{26}$ *an output sequence such that the position of every element in this sequence deviates from its rank in* S *by at most* $O(\log n)$.

5.5 Searching

In this section we assume that we are given a sorted sequence of keys $a_1 < a_2 < \ldots < a_n < a_{n+1} = \infty$, and a query key q. We know that the sorting is accurate, and that q

is different from all a_i. Our task is to determine rank(q), the smallest index i such that $q < a_i$ holds. A noisy oracle provides us with a table containing answers $q <_E a_i$ or $q >_E a_i$ to all possible key comparisons between q and the numbers a_i. Each of them is wrong independently with probability $p < 1/2$.

Let conf(j) denote the number of table entries that would be in conflict with rank(q) $= j$. Our first algorithm, Search, takes $O(n)$ time to read the whole table and to output a position j that minimizes conf(j); ties are broken arbitrarily. By the same argument as in Braverman and Mossel [10], Search reports a maximum likelihood position for q, given the answer table. As opposed to the sorting problem, we can prove a lower bound to the probability that Search correctly computes the rank of q.

Theorem 5.11. Search *reports* rank(q) *with probability at least* $(1-2p)^2$.

Proof. We want to argue that positions j to the left or to the right of rank(q) are less likely to get reported because they have higher conflict numbers. By definition,

$$\operatorname{conf}(j) = \begin{cases} \operatorname{conf}(j-1)+1, & \text{if } q <_E a_{j-1} \\ \operatorname{conf}(j-1)-1, & \text{if } q >_E a_{j-1} \end{cases}$$

holds, as can be quickly verified. Let us first consider the indices $j = \operatorname{rank}(q), \ldots, n$ to the right of rank(q). The oracle's process of producing these answers, for indices j increasing from rank(q) to n, corresponds to a *random walk* of the value of conf(j) through the integers, starting from conf(rank(q)). With probability $1-p$, the value of conf(j) will increase by 1 (since $q < a_j$ holds, by assumption), and with probability p decrease. Using Lemma 5.12 with values $r = \operatorname{conf}(\operatorname{rank}(q))+1$, $m = 1$ and $s = n$ we conclude that with probability $\geq (1-p)\cdot(1-\frac{p}{1-p}) = 1-2p$, conf($j$) will increase in the first step *and* never sink below this value again. Thus, with probability $\geq 1-2p$, all j to the right of rank(q) will have values conf(j) higher than conf(rank(q)) and, therefore, not get reported. A symmetric claim holds for the indices j to the left of rank(q). Consequently, Search does with probability $\geq (1-2p)^2$ report rank(q). □

We consider a random walk through the integers with starting position conf(0) $= r$. For any integer $t > 0$, if conf(t) denotes the state of the random walk at time t, we have

$$\operatorname{conf}(t) := \begin{cases} \operatorname{conf}(t-1)+1 & \text{with probability } 1-p \\ \operatorname{conf}(t-1)-1 & \text{with probability } p \end{cases}$$

where $p < 1/2$ is the probability that the oracle gives a wrong answer.

Lemma 5.12. *Let r denote the current state of a random walk, as defined above. Then, given any integer values $m \geq 1, s \geq 0$, the probability that the random walk will reach state $r-m$ within the next s steps is at most $\left(\frac{p}{1-p}\right)^m$.*

Proof. Let $P(s,m)$ denote the probability that the the random walk, with current state r, reaches state $r-m$ within the next s steps. We show by induction over s

that $P(s,m)$ is at most $\left(\frac{p}{1-p}\right)^m$.

Base case $s = 0$:

$$P(0,m) = 0 \leq \left(\frac{p}{1-p}\right)^m$$

Induction step $s \to s+1$:

There are two cases, $m = 1$ and $m > 1$:

1. $m = 1$

$$\begin{aligned} P(s+1,1) &= p \cdot P(s,0) + (1-p) \cdot P(s,2) \\ &= p \cdot 1 + (1-p) \cdot P(s,2) \\ &\leq p + (1-p) \cdot \left(\frac{p}{1-p}\right)^2 \\ &= \left(\frac{p}{1-p}\right)^1 \\ &= \left(\frac{p}{1-p}\right)^m \end{aligned}$$

2. $m > 1$

$$\begin{aligned} P(s+1,m) &= p \cdot P(s,m-1) + (1-p) \cdot P(s,m+1) \\ &\leq p \cdot \left(\frac{p}{1-p}\right)^{m-1} + (1-p) \cdot \left(\frac{p}{1-p}\right)^{m+1} \\ &= \left(\frac{p}{1-p}\right)^m \end{aligned}$$

This completes the proof of the lemma. □

Theorem 5.11 casts some light on what can be achieved utilizing all information available. If sequence $a_1, \ldots, a_n$ is given as a linear list, the $O(n)$ time algorithm Search is of practical interest, too. For a sorted sequence stored in an array, we have a more efficient tolerant binary search algorithm based on Search.

Our algorithm is a tolerant version of binary search. We pick a block of pivot elements uniformly at random from the "middle" elements of the current array and compare q with this block. If q is smaller than the majority of the elements, we continue our search to the left, if q is larger than the majority of the elements, we continue on the right. After the block elements are used for key comparisons, they are removed from the set of elements considered. The algorithm requires a block size parameter t whose value will be determined below.

```
BinarySearch($A, l, r, q$)
1.  if $r - l \le t/p$ then return $l - 1$ plus the result of algorithm Search
    on the sequence $a_l, \ldots, a_r$
2.  pick $k \in \{l + \lceil (r-l)/4 \rceil, \ldots, l + \lfloor \frac{3}{4}(r-l) \rfloor\}$ uniformly at random
3.  Compare $q$ with keys $a_k, a_{k+1}, ..., a_{k+t}$
4.  if the majority of the comparisons says $q <_E a_j$ then return
    BinarySearch(A,l,k-1,q)
5.  else return BinarySearch(A,k+t+1,r,q)
```

For the analysis we use the fact that $r - l$ shrinks by a factor of at least $3/4$ at every recursive call of the algorithm. Remember that $p < 1/4$ and $k \le l + \frac{3}{4}(r-l)$ hold. Together with $t < (r-l)p$ (which holds if the algorithm executes lines 2-5), this implies $k + t < r$, so in each iteration t items are compared with q. The algorithm cannot compute the rank of q if in line 2 some value k is chosen where $a_k < q < a_{k+t}$ holds, *i. e.*, where $\text{rank}(q) \in \{k+1, \ldots, k+t\}$. The probability that $\text{rank}(q) \in \{k+1, \ldots, k+t\}$ is at most

$$\begin{aligned}
\frac{t}{\#\text{choices for } k} &= \frac{t}{l + \lfloor \frac{3}{4}(r-l) \rfloor - \left(l + \lceil \frac{1}{4}(r-l) \rceil\right) + 1} \\
&= \frac{t}{\lfloor \frac{3}{4}(r-l) \rfloor - \lceil \frac{1}{4}(r-l) \rceil + 1} \\
&\le \frac{t}{\frac{3}{4}(r-l) - 1 - \frac{1}{4}(r-l) + 1} \\
&= \frac{2t}{r-l}
\end{aligned}$$

Now assume that $\text{rank}(q) \notin \{k+1, \ldots, k+t\}$. Then either the whole block $a_k, \ldots, a_{k+t}$ is to the left of q or the whole block is to the right of q. An upper bound for the probability that the majority of the elements used for comparison is faulty is given in the next lemma. It also contains the definition of block size t. We use n to denote the number of elements when BinarySearch is invoked for the first time.

Lemma 5.13. *Let X_j be independent $0-1-$random variables with $\mathbf{Pr}[X_j = 1] = p$ and let $t \ge \frac{3(\ln(\log n) - \ln p)}{p \cdot \min\{\frac{1}{2p}-1, (\frac{1}{2p}-1)^2\}}$. Then $\mathbf{Pr}\left[\sum_{j=1}^t X_j \ge \frac{t}{2}\right] \le p/\log n$.*

Proof. By a Chernoff bound we obtain

$$\mathbf{Pr}\left[\sum_{j=1}^{t} X_j \ge t/2\right] = \mathbf{Pr}\left[\sum_{j=1}^{t} X_j \ge \left(1 + \left(\frac{1}{2p} - 1\right)\right) \cdot pt\right] \le e^{-\frac{1}{3} \cdot \min\left\{\frac{1}{2p}-1, (\frac{1}{2p}-1)^2\right\} \cdot pt}$$

as $\mathbf{E}\left[\sum_{j=1}^t X_j\right] = pt$. Setting $t \ge \frac{3(\ln(\log n) - \ln p)}{p \cdot \min\left\{\frac{1}{2p}-1, (\frac{1}{2p}-1)^2\right\}}$ proves the lemma. □

In the following, we will assume that t satisfies the requirement of Lemma 5.13. It remains to find an upper bound for the error probability. We distinguish between errors that are caused by the fact that the majority of the comparisons are faulty and errors caused by the fact that during one recursion of the algorithm some value for k is chosen such that $\operatorname{rank}(q) \in \{k+1, \ldots, k+t\}$ holds. Since the algorithm recurs at most $\log_{\frac{4}{3}} n$ times, by the previous lemma the error caused by false comparisons is at most $\log_{\frac{4}{3}} n \cdot \frac{p}{\log n} = O(p)$. The total error caused by a bad choice for the value of k during any recursion of the algorithm is at most

$$\sum_{i=0}^{\log_{\frac{4}{3}} n - \log_{\frac{4}{3}} (t/p)} \frac{8}{3} \cdot \frac{t}{n} \cdot \left(\frac{4}{3}\right)^i = \frac{8}{3} \cdot \frac{t}{n} \cdot \sum_{i=0}^{\log_{\frac{4}{3}} \frac{np}{t}} \left(\frac{4}{3}\right)^i \leq \frac{8}{3} \cdot \frac{t}{n} \cdot 3 \cdot \left(\frac{4}{3}\right)^{\log_{\frac{4}{3}} \frac{np}{t} + 1} = O(p).$$

This is due to the fact that in every recursion the value of $r - l$ shrinks by a factor of at least 3/4. Hence for any $i \in \left\{0, 1, \ldots, \log_{\frac{4}{3}} n - \log_{\frac{4}{3}}(t/p)\right\}$ there is at most one invocation of BinarySearch(A, l, r, q) where $n \cdot \left(\frac{3}{4}\right)^i \geq r - l > n \cdot \left(\frac{3}{4}\right)^{i+1}$, and the error probability for this invocation is at most $\frac{2t}{n} \cdot \left(\frac{4}{3}\right)^{i+1} = \frac{8}{3} \cdot \frac{t}{n} \cdot \left(\frac{4}{3}\right)^i$.

Finally, the error probability in line 1 of algorithm BinarySearch is also in $O(p)$, by Theorem 5.11. Hence, the algorithm returns $\operatorname{rank}(q)$ with probability $1 - f(p)$ for some function $f(p)$ that goes to 0 as p goes to 0.

Theorem 5.14. *Let $p \in (0, 1/4)$ be an upper bound on the error probability of comparisons and let $a_1 < a_2 < \ldots < a_n$ be a sequence of n numbers. Let the block size parameter be $t = \left\lceil \frac{3(\ln(\log n) - \ln p)}{p \cdot \min\{\frac{1}{2p} - 1, (\frac{1}{2p} - 1)^2\}} \right\rceil$. Given an arbitrary query key $q \neq a_j$, $1 \leq j \leq n$, Algorithm* BinarySearch *reports the rank of an element q with respect to $a_1 < a_2 < \ldots < a_n$ in time $O(\log n \cdot \log\log n)$ (p is assumed to be constant) and with probability at least $1 - f(p)$ for some function $f(p)$ that goes to 0 as p goes to 0.*

5.6 Linear Programming in 2 Dimensions

As an application of our LinearMax algorithm, we consider the linear programming problem in two dimensions, namely, the problem of minimizing a linear objective function subject to a family $\mathcal{F}$ of half-plane constraints. We assume our problem is in standard form [14], namely that the problem is to find the lowest (finite) point in the non-empty feasible region, defined by the intersection of a non-degenerate family $\mathcal{F}$ of n half-planes. Define a *floor* to be a half-plane including all points in the plane above a line, whereas a *ceiling* is a half-plane including all points in the plane below a line. We say a half-plane is *vertical* if the line defining it is vertical. In the standard setting, we can assume that vertical half-planes have been preprocessed and replaced with the constraint $L \leq x \leq R$ for reals L and R, and that no two ceilings and no two floors are parallel. The half-planes are given in a sorted list according to their slope. Our algorithm is based on several basic geometric primitives which can make mistakes.

Error Model: We are given a few black boxes that perform side comparisons.

The first box is SideComparator, which is given four lines ℓ_A, ℓ_B, ℓ_C, and ℓ_D, and decides if the the intersection of ℓ_A and ℓ_B is to the left of the intersection of ℓ_C

and ℓ_D. This description can be simplified to the following: "given two points, is one to the left of the other"? (however, since the input does not contain explicit points, we have chosen to describe the test in this more abstract way).

The second box is VerticalComparator, which is given four lines ℓ_A, ℓ_B, ℓ_C, and ℓ_D, and returns the line in the set $\{\ell_C, \ell_D\}$ whose signed vertical distance is larger from the intersection point of ℓ_A and ℓ_B. This description can be simplified to the following: "given a point and two lines, which line is closer in vertical distance"? (again, since the input does not contain explicit points, we choose to describe the test in this more abstract way).

More precisely, if the intersection of ℓ_A and ℓ_B is the point (α, β), and we draw the line $x = \alpha$, then we look at the signed distance from (α, β) to the intersection point of ℓ_C and $x = \alpha$ as well as the intersection point of ℓ_D and $x = \alpha$. Here, by signed, we mean that if (α, β) is above one of these intersection points, then its distance to that point is negative, otherwise it is non-negative. If the signed distances are the same, i. e., the lines ℓ_C and ℓ_D meet $x = \alpha$ at the same point, VerticalComparator reports this.

Such primitives are basic, and play an essential role in geometric algorithms.

We assume the primitives have a small error probability p of failing. In the case of SideComparator, the box reports the opposite side with probability p. In the case of VerticalComparator, the box reports the further line (in signed vertical distance), or fails to detect if two lines have the same signed distance, with probability p. Multiple queries to the tester give the same answer.

The output of our algorithm is not given explicitly, but rather is specified as the intersection of two half-planes in $\mathcal{F}$ (since this is all that can be determined given abstract access to the input lines).

Theorem 5.15. *There is an algorithm* LP *that with probability* $1 - f(p)$ *terminates in* $O(n \log n)$ *time and solves the linear programming problem in 2 dimensions, where* $f(p)$ *approaches 0 as* p *approaches 0.*

We now review Megiddo's algorithm and then describe our main new ideas.

Megiddo's Algorithm: Megiddo's algorithm defines: $g(x) = \max\{a_i x + b_i \mid y \geq a_i x + b_i \text{ is a floor}\}$, and $h(x) = \min\{a_i x + b_i \mid y \leq a_i x + b_i \text{ is a ceiling}\}$. A point $p = (x, y)$ is feasible iff $g(x) \leq h(x)$, that is, if the intersection points of all floors with a vertical line ℓ through p lie below the intersection points of all ceilings with ℓ. The algorithm has $O(\log n)$ stages. The number of remaining constraints in $\mathcal{F}$ in the i-th stage is at most $(7/8)^i \cdot n$. If at any time there is only a single floor g in $\mathcal{F}$, then the algorithm outputs the lowest point on g that is feasible. Otherwise it arbitrarily groups the floors into pairs and the ceilings into pairs, and computes the pair (ℓ_A, ℓ_B) whose intersection point has the median x-coordinate of all intersection points of all pairs.

The algorithm checks if the intersection point (α, β) of ℓ_A and ℓ_B is feasible, i. e., if $g(\alpha) \leq h(\alpha)$. The algorithm then attempts to determine if the optimum is to the right or the left of (α, β). It is guaranteed there is a feasible solution - an invariant maintained throughout the algorithm - and only one side of (α, β) contains a feasible

point if (α, β) is infeasible. Megiddo defines the following:
$s_g = \min\{a_i \mid y \geq a_i x + b_i$ is a floor and $g(\alpha) = a_i\alpha + b_i\}$,
$S_g = \max\{a_i \mid y \geq a_i x + b_i$ is a floor and $g(\alpha) = a_i\alpha + b_i\}$,
$s_h = \min\{a_i \mid y \leq a_i x + b_i$ is a ceiling and $h(\alpha) = a_i\alpha + b_i\}$,
$S_h = \max\{a_i \mid y \leq a_i x + b_i$ is a ceiling and $h(\alpha) = a_i\alpha + b_i\}$.

Suppose first that (α, β) is infeasible. This means that $g(\alpha) > h(\alpha)$. Then if $s_g > S_h$, any feasible x satisfies $x < \alpha$. Also, if $S_g < s_h$, then any feasible x satisfies $x > \alpha$. The last case is that $s_g - S_h \leq 0 \leq S_g - s_h$, but this implies the LP is infeasible, contradicting the above invariant.

Now suppose that (α, β) is feasible. As Megiddo argues, if $g(\alpha) < h(\alpha)$ then if $s_g > 0$, then the optimal solution is to the left of α. Also, if $S_g < 0$ then the optimal solution is to the right of α. Otherwise $s_g \leq 0 \leq S_g$, and (α, β) is the optimal solution. Finally, if $g(\alpha) = h(\alpha)$, then if (1) $s_g > 0$ and $s_g \geq S_h$, then the optimum is to the left of α, or if (2) $S_g < 0$ and $S_g \leq s_h$, then the optimum is to the right of α. Otherwise (α, β) is the optimum.

Hence, in $O(n)$ time, the algorithm finds the optimum or reduces the solution to the left or right of (α, β). In this case, in each of the pairs of constraints on the other side of α, one of the two constraints can be removed since it cannot participate in defining the optimum. When there are a constant number of constraints left, the algorithm solves the resulting instance by brute force.

Intuition of Our Algorithm: Let Π be a partition of the set $\mathcal{F}$ of input floors and ceilings into pairs. Inspired by our maximum-finding algorithm, instead of computing the median of pairs of intersection points, we randomly sample a set S of $\Theta(\log|\mathcal{F}|)$ pairs from Π. For each pair in S, we find if the optimum is to the left or right of the intersection point. If (α, β) is the intersection point of a pair of constraints in S, we use VerticalComparator to find the *lower floor* $\{y \geq a_i x + b_i$ is a floor and $g(\alpha) = a_i\alpha + b_i\}$ as well as the *upper ceiling* $\{y \leq a_i x + b_i$ is a ceiling and $h(\alpha) = a_i\alpha + b_i\}$. By non-degeneracy, each of these sets has size at most 2. We modify our earlier LinearMax algorithm to return the maximum two items (i. e., constraints) instead of just the maximum, using VerticalComparator to perform the comparisons. By a union bound, we have the lower floor and upper ceiling with large probability. Using the slope ordering of s_g, S_g, s_h, and S_h we know which side of (α, β) the optimum is on.

To avoid performing the same comparison twice, in any phase each primitive invocation has as input at least one of the lines in our sample set. Since we discard the sample set after a phase, comparisons in different phases are independent. To ensure that comparisons in the same phase are independent, when computing the upper ceiling and lower floor of a sampled intersection point (α, β), we do not include the other sampled pairs of constraints in the comparisons. This does not introduce errors, with high probability, since we have only $O(\log|\mathcal{F}|)$ randomly sampled constraints, while the union of the upper ceiling and lower floor of an intersection point has at most four constraints, and so it is likely the upper ceilings and lower floors of all sampled pairs are disjoint from the set of sampled constraints.

Since the sample size is $\Theta(\log|\mathcal{F}|)$, we can show that with high probability we throw away a constant fraction of constraints in each phase, and so after $O(\log n)$ recursive calls the number of remaining constraints is bounded as a function of p alone. The total time is $O(n \log n)$. Our main algorithm is described below.

```
LP($\mathcal{F}, p$)
1.  If there are at most $C(p)$ constraints in $\mathcal{F}$ solve the problem by brute force.
2.  If there is at most one floor constraint $f \in \mathcal{F}$, if the slope of $f$ is positive,
    output the intersection of $f$ and the line $x = L$. If the slope of $f$ is negative,
    output the intersection of $f$ and the line $x = R$.
3.  Otherwise, randomly partition the floors into pairs, as well as the ceilings
    into pairs (possibly with one unpaired floor and one unpaired ceiling).
    Let the set of pairs be denoted $\Pi$.
    Draw a set $S$ of $1000 \log|\mathcal{F}|$ pairs of constraints from the pairs in $\Pi$
    uniformly at random, without replacement.
4.  Let $\Phi_F$ be the set of floors in $\mathcal{F}$, excluding those in $S$.
    Let $\Phi_C$ be the set of ceilings in $\mathcal{F}$, excluding those in $S$.
5.  For each pair $(\ell_A, \ell_B)$ of constraints in $S$,
a.      Let $U(\ell_A, \ell_B) = \textsc{Lowest}(\ell_A, \ell_B, \Phi_C, p)$.
b.      Let $L(\ell_A, \ell_B) = \textsc{Highest}(\ell_A, \ell_B, \Phi_F, p)$.
c.      Compute $\textsc{Tester}(\ell_A, \ell_B, U(\ell_A, \ell_B), L(\ell_A, \ell_B))$.
6.  Let $T \subseteq S$ be the pairs for which Tester does not output "fail", and compute
    the majority output direction $dir$ of the result of Tester on the pairs in $T$.
7.  For each pair $(\ell_A, \ell_B)$ of constraints in $\Pi \setminus S$,
a.      For each pair $(\ell_C, \ell_D) \in S$, compute $\textsc{SideComparator}(\ell_A, \ell_B, \ell_C, \ell_D)$.
b.      If for at least a 2/3-fraction of pairs in $S$, the pair $(\ell_A, \ell_B)$ is to the right
        (resp. to the left), and if $dir$ is to the left (resp. to the right), then
        remove the constraint in the pair $(\ell_A, \ell_B)$ from $\mathcal{F}$ that cannot
        participate in the optimum assuming the optimum is really
        to the left (resp. to the right) of the pair $(\ell_A, \ell_B)$.
8.  Return $LP(\mathcal{F} \setminus S, p)$.
```

Our analysis will show that Theorem 5.15 holds if we use $C(p) = 160000/p^{1/5}$ in line 1 of the algorithm LP.

Intuitively, Lowest finds the upper envelope[1] of a point (that is, the lowest ceilings), and Highest finds the lower envelope (the highest floors). Tester tests which side of the optimum the point is on based on the slope information in the union of upper and lower envelopes. Both Highest and Lowest employ the following subroutine.

```
2Max($M, p$)
1.  Let $x = \textsc{LinearMax}^*(M, p)$.
```

[1]Sometimes envelope refers to all boundary lines that touch the convex hull. Here we use it to simply refer to the extremal two constraints of a point.

```
2.   Let $y$ =LinearMax$^*(M \setminus \{x\}, p)$.
3.   Output $(x, y)$.
```

Theorem 5.16. 2Max *returns the largest two elements of M in time $O(n)$ with probability at least $1 - 2\tilde{f}(p)$, where $\tilde{f}(p)$ is the error probability of* LinearMax* *with comparison error probability p.*

Proof. Let r be the random string used by the LinearMax* algorithm (see Theorem 5.1 and Corollary 5.6), and let s be the random characteristic vector of comparisons. Let LinearMax$^*_{r,s}$ be the (deterministic) output of LinearMax* given r and s. Let $\sigma = \max(x \in M)$ and let $\tau = \max(x \in M \setminus \{\sigma\})$. Then our prior analysis shows both

$$\Pr[\textsc{LinearMax}^*_{r,s}(M, p) = \sigma, \text{ in time } O(n)] \geq 1 - \tilde{f}(p),$$

and

$$\Pr[\textsc{LinearMax}^*_{r,s}(M \setminus \{\sigma\}, p) = \tau, \text{ in time } O(n)] \geq 1 - \tilde{f}(p),$$

where the probability is over the joint distribution of r and s. It follows by a union bound that 2Max succeeds in time $O(n)$ with probability at least $1 - 2\tilde{f}(p)$. □

In our application of 2Max, we set the universe M to be either Φ_F, a subset of floors in the constraint set $\mathcal{F}$, or Φ_C, a subset of ceilings in the constraint set $\mathcal{F}$. We then fix an intersection point (α, β) of two lines ℓ_A and ℓ_B. Our universe is the set of intersection points of the lines in Φ_F (or Φ_C) with the vertical line $x = \alpha$. The line $\ell \in \Phi_F$ whose intersection point is the highest point on the line $x = \alpha$ is considered the maximum of Φ_F, while the line $\ell \in \Phi_C$ whose intersection point is the lowest on the line $x = \alpha$ is considered the maximum of Φ_C. To do the comparison between two lines $\ell_C, \ell_D \in \Phi_F$, we invoke VerticalComparator$(\ell_A, \ell_B, \ell_C, \ell_D)$, which reports the line whose intersection point has the larger signed vertical distance (if instead considering Φ_C, we use the line with the smaller signed vertical distance).

By non-degeneracy, the upper envelope of (α, β) contains at most two lines, each of which will be found by 2Max with probability at least $1 - 2\tilde{f}(p)$, while the lower envelope of (α, β) contains at most 2 lines, each of which will be found by 2Max with probability at least $1 - 2\tilde{f}(p)$. Hence, with probability $1 - 4\tilde{f}(p)$, both invocations of 2Max will succeed. Finally, given two lines ℓ and ℓ', the purported maximum and second maximum in Φ_F, we need a way of checking if ℓ and ℓ' intersect the line $x = \alpha$ in the same point. This can be done by invoking VerticalComparator$(\ell_A, \ell_B, \ell, \ell')$, which succeeds with probability $1 - p$. We can similarly compare the purported maximum and second maximum in Φ_C. Notice that these additional two applications of VerticalComparator may have already been applied (as subroutines in 2Max). Nevertheless, by a union bound, all these algorithms jointly succeed with probability at least $1 - 4\tilde{f}(p) - 2p$. Summarizing the algorithms above:

1. Let Lowest$(\ell_A, \ell_B, \Phi_C, p)$ be the above algorithm for finding the maximum and second maximum of Φ_C (where maximum, as defined with respect to Φ_C, actually means the lowest points on the line (α, β)). Further, given the purported

maximum ℓ and second maximum ℓ', it determines if the two lines intersect the line $x = \alpha$ at the same point.

2. Let HIGHEST$(\ell_A, \ell_B, \Phi_F, p)$ be the above algorithm for finding the maximum and second maximum of Φ_F . Given the purported maximum ℓ and second maximum ℓ', it determines if the two lines intersect the line $x = \alpha$ at the same point.

The above analysis shows,

Theorem 5.17. *Given ℓ_A, ℓ_B, Φ_F, and Φ_C, with probability at least $1 - 4\tilde{f}(p) - 2p$, algorithms* LOWEST$(\ell_A, \ell_B, \Phi_C, p)$ *and* HIGHEST$(\ell_A, \ell_B, \Phi_F, p)$ *succeed.*

We also need the following subroutine in Figure 5.1, which takes two lines ℓ_A and ℓ_B, two sets *Up* and *Low* of size at most two, the purported upper and lower envelopes of the intersection point of ℓ_A and ℓ_B, and determines if the optimum is to the left, the right, or equals the intersection of ℓ_A and ℓ_B. It does so by analysing slopes as in Megiddo's algorithm [52]. Recall that after preprocessing the vertical constraints, we know that $L \leq x \leq R$. Here *Up* contains only ceilings, while *Low* contains only floors.

The assumption of the algorithm is that *Low* $\neq \emptyset$, yet *Up* may equal $\emptyset$. Note that this assumption holds *e. g.* if HIGHEST$(\ell_A, \ell_B, \Phi_F, p)$ succeeds and the (up to) two constraints defining the optimum are contained in $\mathcal{F}$.

Theorem 5.18. *Conditioned on U and L being the upper and lower envelopes of the intersection point q of ℓ_A and ℓ_B, and on q not being the optimum point,* TESTER *correctly locates which side of the optimum q is on with probability at least $1 - 4p$.*

Proof. By a union bound, with probability at least $1 - 4p$, both SIDECOMPARATOR tests and both VERTICALCOMPARATOR tests succeed. We condition on this event in the rest of the proof. It follows that if TESTER halts in step 2 or 3, then it succeeds. It also follows that q is correctly classified as feasible or infeasible in step 5. If TESTER halts in step 7, correctness follows from the correctness of Megiddo's algorithm [52], as the algorithm performs the same comparisons. Notice that the algorithm cannot return "fail", since conditioned on U and L being the upper and lower envelopes of q, this would mean (as shown in Megiddo's analysis) that the LP is infeasible, a contradiction.

The last case is that TESTER halts in step 8. If the upper and lower envelopes are not equal, then since we condition on q not being the optimum point, our test coincides with that of Megiddo's algorithm. Since we also condition on U and L being the upper and lower envelopes of q, we cannot output "fail", as this would mean (as shown in Megiddo's analysis) that q is the optimum point.

Finally, it cannot be that the upper and lower envelopes are equal since the point q is the intersection of two ceilings or two floors. Since q is also feasible, this means if it is the intersection of two ceilings, a floor must also intersect it, while if it is the intersection of two floors, a ceiling must also intersect it. This contradicts the non-degeneracy of the input. Hence, our test coincides with Megiddo's and q is correctly located. □

```
TESTER($\ell_A, \ell_B, Up, Low$)
  1. Let $Low_1$ be the first line in $Low$, and if $|Low| = 2$, let $Low_2$ be the second.
     If $Up \neq \emptyset$, let $Up_1$ be the first line in $Up$, and if there is a second, denote it
     by $Up_2$. Let point $q = (\alpha, \beta)$ refer to the intersection of $\ell_A$ and $\ell_B$.
  2. If SIDECOMPARATOR($\ell_A, \ell_B, x = L, y = 0$) reports that $\alpha < L$, return "to the
     right".
  3. If SIDECOMPARATOR($\ell_A, \ell_B, x = R, y = 0$) reports that $\alpha > R$, return "to the
     left".
  4. Run VERTICALCOMPARATOR($\ell_A, \ell_B, \ell_A, Low_1$) to determine if $Low_1$ is above
     or below $q$. If $Up \neq \emptyset$, run VERTICALCOMPARATOR($\ell_A, \ell_B, \ell_A, Up_1$) to deter-
     mine if $Up_1$ is above or below $q$.
  5. If $Low_1$ is below $q$ and $Up_1$, if it exists, is above $q$, then $q$ is declared feasible.
     Otherwise it is declared infeasible.
  6. Let $s_g$ be the smaller of the two slopes of lines in $Low$, and let $S_g$ be the larger
     (if there is only one line, then $s_g = S_g$.) Let $s_h$ be the smaller of the two slopes
     of lines in $Up$, and let $S_h$ be the larger. If $Up$ is empty, let $s_h = S_h = 0$.
  7. If $q$ is declared infeasible,
       (a) if $s_g > S_h$, then return "to the left".
       (b) if $S_g < s_h$, then return "to the right".
       (c) otherwise return "fail".
  8. Otherwise, $q$ is declared feasible, and
       (a) if $s_g > 0$, then return "to the left".
       (b) if $S_g < 0$, then return "to the right".
       (c) otherwise return "fail".
```

Figure 5.1: TESTER pseudocode.

Theorem 5.19. *Given any constant $1 > \varepsilon > 0$, there is a constant $\tilde{C}(\varepsilon)$ such that if $C(p) \geq \tilde{C}(\varepsilon)$, then with probability $1-\varepsilon$ Algorithm* LP$(\mathcal{F}, p)$ *terminates in $O(n \log n)$ time with $O(\log n)$ recursive calls.*

Proof. Note that step 1 can be done in constant time for constant p. Also, since LinearMax* takes $O(n)$ time, so does 2Max and so do Lowest and Highest. Note that Tester can be done in $O(1)$ time. Notice that step 5 takes $O(n \log n)$ time. To prove the theorem we will provide a function $\tilde{C}$ where for every constant ε with probability $1-\varepsilon$ in *every* iteration at least $|\mathcal{F}|/200$ constraints are removed, if $C(p) \geq \tilde{C}(\varepsilon)$. It will follow that there are $O(\log n)$ recursive calls and the overall time complexity is dominated by that of the first recursive call, or $O(n \log n)$.

Consider any call to LP with some value of $|\mathcal{F}| > C(p)$. We need a property of $\mathcal{F}$ that follows from the lines being in general position. Namely, consider any given x-coordinate x_0. We claim that the number of pairs of lines in $\mathcal{F}$ whose intersection point has x-coordinate equal to x_0 is at most $|\mathcal{F}|$. We know that since no two floors and no two ceilings are parallel, if there are r floors, there can be at most $r/2$ intersection points of pairs of floors which have x-coordinate equal to x_0. There are also $|\mathcal{F}| - r$ ceilings, and there can be at most $(|\mathcal{F}| - r)/2$ pairs of ceilings which have intersection point with x-coordinate equal to x_0. Hence, there are at most $|\mathcal{F}|/2$ pairs of the same type (floor or ceiling) which have the same x-coordinate.

In what follows we shall assume that $|\mathcal{F}|$ is even for simplicity (the case of odd cardinality follows with minor modifications). Recall that Π is randomly chosen. Let us consider the process of choosing the pairs in Π one at a time. Let Z_i be an indicator variable if the x-coordinate of the intersection of the i-th pair in Π does not equal the x-coordinate of the intersection of the j-th pair in Π, for any $1 \leq j < i$. For any choice of the first $i-1$ pairs of Π, there are $s = |\mathcal{F}| - 2(i-1)$ remaining constraints, and so at least $2\binom{s/2}{2} = (s/2)(s/2-1)$ possible pairs of the same type. Moreover, the number of pairs of the same type with intersection point realizing any fixed x-coordinate is at most $s/2$, by the previous paragraph. Hence,

$$\mathbf{Pr}\left[Z_i = 1\right] \geq 1 - \frac{s/2}{(s/2)(s/2-1)} = 1 - \frac{1}{s/2-1}$$

For $i \leq 99|\mathcal{F}|/200$, $s \geq |\mathcal{F}|/100$, and so

$$\mathbf{Pr}\left[Z_i = 1\right] \geq 1 - \frac{201}{|\mathcal{F}|} \geq \frac{99}{100},$$

assuming $|\mathcal{F}|$ is larger than a large enough constant (as otherwise step 1 will solve the LP). It follows that $Z = \sum_{i=1}^{99|\mathcal{F}|/200} Z_i$ stochastically dominates $Z' = \sum_{i=1}^{99|\mathcal{F}|/200} Z'_i$, where the Z'_i are i. i. d. Bernoulli(99/100), and so

$$\mathbf{Pr}\left[Z < (9/20)|\mathcal{F}|\right] < e^{-\Omega(|\mathcal{F}|)},$$

by a Chernoff bound. Hence, with probability at least $1 - e^{-\Omega(|\mathcal{F}|)}$, there are at least $9|\mathcal{F}|/20$ distinct x-coordinates among the intersection points formed by pairs in Π.

More precisely, we have

$$\begin{aligned}
\mathbf{Pr}\left[Z < (9/20)|\mathcal{F}|\right] &= \mathbf{Pr}\left[Z < \left(1 - \frac{89}{1089}\right) \cdot \frac{99|\mathcal{F}|}{200} \cdot \frac{99}{100}\right] && (5.1)\\
&< e^{-\frac{1}{2}\cdot\left(\frac{89}{1089}\right)^2 \cdot \frac{99|\mathcal{F}|}{200} \cdot \frac{99}{100}} && (5.2)\\
&= e^{-\frac{7921}{4840000}\cdot|\mathcal{F}|} && (5.3)\\
&= |\mathcal{F}|^{-\frac{7921|\mathcal{F}|}{4840000 \ln |\mathcal{F}|}} && (5.4)\\
&\leq |\mathcal{F}|^{-6} && (5.5)\\
&\leq \frac{81}{100} \cdot |\mathcal{F}|^{-5} && (5.6)
\end{aligned}$$

where inequalities (5.5) and (5.6) follow from $|\mathcal{F}|$ being larger than a sufficiently large constant.
Let $\mathcal{L}$ be the list of intersection points of the pairs in Π, that is,

$$\mathcal{L} = \{x \mid \exists\, \pi = (\ell_A, \ell_B) \in \Pi, \exists\, y \in \mathbb{R} : \ell_A \cap \ell_B = (x, y)\}.$$

Consider the set W of the smallest $|\mathcal{F}|/100$ intersection points in this list. Then, conditioned on there being at least $9|\mathcal{F}|/20$ distinct x-coordinates among the intersection points formed by pairs in Π, it follows that every item in the set W is strictly less than at least $|\mathcal{F}|(9/20 - 1/100) = (11/25)|\mathcal{F}|$ intersection points (in terms of x-coordinate). On the other hand, Π contains at most $|\mathcal{F}|/2$ distinct intersection points, and we draw a set of $1000 \log |\mathcal{F}|$ pairs without replacement. Hence, with probability at least

$$\frac{\frac{11}{25}|\mathcal{F}| - 1000 \log |\mathcal{F}|}{\frac{|\mathcal{F}|}{2}} = \frac{22}{25} - 2000 \frac{\log |\mathcal{F}|}{|\mathcal{F}|} \geq \frac{21}{25},$$

an intersection point in the set W has smaller x-coordinate than that of a random pair in Π, where the last inequality follows from $|\mathcal{F}|$ being larger than a large enough constant. Using a Chernoff bound we obtain that, with probability $1 - e^{-(135/28)\ln(|\mathcal{F}|)/\ln(2)} \geq 1 - |\mathcal{F}|^{-6}$, an intersection point in the set W is smaller than at least a 3/4-fraction of the points in S. The number of pairs in Π whose intersection point's x-coordinate belongs to W is at most $|\mathcal{F}|/2 - (9/20 - 1/100)|\mathcal{F}| = (6/100)|\mathcal{F}|$. Hence, by a union bound, this property holds of all points in W with probability at least $1 - 6/(100|\mathcal{F}|^5)$. Analogously, letting W' be the set of largest $|\mathcal{F}|/100$ intersection points in this list, with probability at least $1 - 6/(100|\mathcal{F}|^5)$, every point in W' is larger than at least a 3/4-fraction of the points in S. Hence, with probability at least $1 - 12/(100|\mathcal{F}|^5)$, both of these events (for W and W' occur), which we condition on.

It follows that since the SideComparator comparisons in step 7a are independent, for small enough p, by a Chernoff and union bound, all points in W are declared to be to the left of at least a 2/3-fraction of the points in S, and all points in W' are declared to be to the right of at least a 2/3-fraction of the points in S. For a fixed point w in W let Z be number of points s in S declaring w to the left of s. Since $\mathbf{E}[Z]$

is at least $3/4 \cdot |S| \cdot (1-p)$ we obtain

$$\begin{aligned}
\mathbf{Pr}\left[Z < (2/3)|S|\right] &= \mathbf{Pr}\left[Z < \frac{3}{4} \cdot (1-p) \cdot \left(1 - \frac{9p-1}{9p-9}\right) \cdot |S|\right] && (5.7)\\
&\leq \mathbf{Pr}\left[Z < \left(1 - \frac{9p-1}{9p-9}\right) \cdot \mathbf{E}\left[Z\right]\right] && (5.8)\\
&< e^{-\frac{1}{2}\left(\frac{9p-1}{9p-9}\right)^2 \cdot \mathbf{E}[Z]} && (5.9)\\
&\leq e^{-\frac{1}{2}\left(\frac{9p-1}{9p-9}\right)^2 \cdot \frac{3}{4} \cdot |S| \cdot (1-p)}. && (5.10)
\end{aligned}$$

Using $|S| = 1000 \log |\mathcal{F}|$ we obtain $e^{-\frac{1}{2}\left(\frac{9p-1}{9p-9}\right)^2 \cdot \frac{3}{4} \cdot |S| \cdot (1-p)} = |\mathcal{F}|^{\frac{125(81p^2-18p+1)}{27(p-1)\ln(2)}}$. A simple calculation shows that $\frac{125(81p^2-18p+1)}{27(p-1)\ln(2)} \leq -6$ holds for all $p \leq 0.00612$. Thus, with probability at least $1 - |\mathcal{F}|^{-6}$, w is reported to be smaller than at least a 2/3-fraction of the points in S. The same estimate holds for any point in W'. Since the number of pairs whose intersection point's x-coordinate is in $W \cup W'$ is at most

$$|\mathcal{F}|/2 - |\mathcal{F}| \cdot (9/20 - 2/100) = 7|\mathcal{F}|/100,$$

by a union bound, with probability at least $1 - 7/(100|\mathcal{F}|^5)$, all the points in W and W' are reported to be to the left and right respectively of at least a 2/3-fraction of the points in S.

Now it follows that no matter which direction *dir* is equal to, in step 7b either all pairs of constraints in W or all those in W' will have at least one of their constraints removed. Hence, at least $|\mathcal{F}|/200$ constraints are removed. In summary, we have shown that with probability at least $1 - 1/|\mathcal{F}|^5$, at least $|\mathcal{F}|/200$ constraints are removed.

Now, by construction the comparisons in different recursive calls are independent. Notice that the recursion stops when $|\mathcal{F}|$ is at most a large enough constant $C(p)$ (depending on p). We will now provide a function $\tilde{C}$ such that after $O(\log n)$ recursive calls, $|\mathcal{F}|$ will be at most $C(p)$ with probability at least $1 - \varepsilon$, if $C(p) \geq \tilde{C}(\varepsilon)$.

Let $a = 200/199$. We say a recursive call $\mathrm{LP}(\mathcal{F}, p)$ happens in *phase* i, $i \in \mathbb{N}$, if $a^i \leq |\mathcal{F}| < a^{i+1}$ holds. Our goal is to define a function $\tilde{C}$ such that if the recursion stops when the size of $\mathcal{F}$ is smaller than $C(p)$, and $C(p) \geq \tilde{C}(\varepsilon)$, then with probability at least $1 - \varepsilon$ no two consecutive recursive calls happen in the same phase. This property clearly holds if in each recursion of the algorithm the number of constraints is reduced by at least a 1/200-fraction.

We say an error occurs during a recursive call if the number of constraints is not reduced by the required amount of a 1/200-fraction. Let Z_i denote the indicator variable for the event that an error occurs during the *first* recursive call in phase i. The preceding analysis shows that $Z_i = 1$ holds with probability at most $1/|\mathcal{F}|^5 \leq 1/a^{5i}$. Consequently, we have $Z_i = 0$ with probability at least $1 - 1/a^{5i}$. From the definition of the Z_i it follows that if all the Z_i are 0, then in each recursion the amount of constraints is reduced sufficiently.

We define $\tilde{C}(\varepsilon) = a^k$ where $k = \frac{5\ln a - \ln(\varepsilon(a^5-1))}{5\ln a}$. Now for any value of $C(p) \geq \tilde{C}(\varepsilon)$ the total error probability is at most

$$\begin{aligned}
\mathbf{Pr}\left[\exists i : \left(a^i \geq C(p)\right) \wedge (Z_i = 1)\right] &\leq \mathbf{Pr}\left[\exists i \geq k : Z_i = 1\right] && (5.11)\\
&\leq \sum_{j=k}^{\infty} \left(a^{-5}\right)^j && (5.12)\\
&= \frac{1}{a^{5(k-1)} \cdot (a^5 - 1)} && (5.13)\\
&\leq \varepsilon, && (5.14)
\end{aligned}$$

where the last inequality can be verified by plugging in the value of k. Note that, by definition of k and a, $\tilde{C}(\varepsilon) = a^k$ equals $\frac{a}{(a^5-1)^{1/5}} \cdot \frac{1}{\varepsilon^{1/5}} = \frac{200}{(7920399001\varepsilon)^{1/5}}$.

The above analysis shows that with probability at least $1 - \varepsilon$ the total running time of our algorithm LP is

$$O\left(\sum_{i=0}^{\infty} n(199/200)^i \log\left(n(199/200)^i\right)\right) = O\left(\sum_{i=0}^{\infty} n(199/200)^i \log n\right) = O(n \log n).$$

This completes the proof. □

Now, the following theorem implies the correctness of Theorem 5.15.

Theorem 5.20. *With probability* $1 - f(p)$*,* $\mathrm{LP}(\mathcal{F}, p)$ *correctly finds the optimum. Here* $f(p)$ *goes to 0 as* p *goes to 0.*

Proof. In Theorem 5.19 above, we showed that with probability at least $1 - p$, the algorithm terminates in $O(\log n)$ rounds, if $C(p) \geq \frac{200}{(7920399001p)^{1/5}}$. Now, the only way for the algorithm to terminate is in step 1 or step 2. It is clear that if the two constraints defining the optimum are in $\mathcal{F}$, then with probability $1 - f'(p)$, for some positive function $f'(p)$ that goes to 0 as p goes to 0, step 1 succeeds, since with this probability all calls to the comparators are correct, as we have only a constant number $C(p)$ of remaining constraints. Step 2 is correct with probability 1. Hence, we just need to bound the probability that in each of $O(\log n)$ rounds, neither of the two constraints defining the optimum is discarded from $\mathcal{F}$. We will first give upper bounds on different error probabilities before constructing a constant $C(p)$, depending on p, where those error probabilities are small enough. Afterwards we give an upper bound on the error probability when the algorithm terminates in step 1.
We now define events $\mathcal{E}_1$, $\mathcal{E}_2$, $\mathcal{E}_3$, $\mathcal{E}_4$ and $\mathcal{E}_5$, and bound the probability of each occurring:

- $\mathcal{E}_1$ is the event that one of the constraints of the optimum is in the set S.
- $\mathcal{E}_2$ is the event that one of the constraints in the set S is part of the upper or lower envelope of another constraint in S.

- $\mathcal{E}_3$ is the event that for at most a 2/5-fraction of the points in S, the optimum is in the direction *dir* from these points (where *dir* is the direction computed in step 6).

- $\mathcal{E}_4$ is the event that if a pair of constraints $(\ell_A, \ell_B) \in \Pi \setminus S$ has intersection point in direction dir' from at least a 3/4-fraction of the intersection points in S, then of the $|S|$ invocations of SideComparator with first two arguments ℓ_A and ℓ_B, less than a 2/3-fraction of them will return the answer dir'.

- $\mathcal{E}_5$ is the event that one of the constraints defining the optimum is removed from set $\mathcal{F}$ in step 7b.

To bound $\mathbf{Pr}[\mathcal{E}_1]$, let Π_{opt} denote the set of (up to two) pairs which contain a constraint defining the optimum. We consider the process of randomly drawing the set S from Π. Now with probability at least

$$\neg\mathbf{Pr}[\mathcal{E}_1] \geq \prod_{i=1}^{|S|} \frac{|\Pi| - |\Pi_{opt}| - i + 1}{|\Pi| - i + 1}$$

none of the constraints in Π_{opt} is contained in S, since before drawing the i-th pair there are $|\Pi| - i + 1$ pairs in Π of which at least $|\Pi| - |\Pi_{opt}| - i + 1$ do not contain a constraint defining the optimum. As there are at least $(|\mathcal{F}| - 2)/2 \geq |\mathcal{F}|/3$ pairs in Π, we have

$$\prod_{i=1}^{|S|} \frac{|\Pi| - |\Pi_{opt}| - i + 1}{|\Pi| - i + 1} \geq \left(\frac{|\mathcal{F}|/3 - 2 - |S|}{|\mathcal{F}|/3 - |S|}\right)^{|S|} = \left(1 - \frac{2}{|\mathcal{F}|/3 - |S|}\right)^{|S|} .$$

Using the definition $|S| = 1000 \cdot \log|\mathcal{F}|$ and the fact that $|\mathcal{F}|$ is larger than a large enough constant, we obtain

$$\left(1 - \frac{2}{|\mathcal{F}|/3 - |S|}\right)^{|S|} \geq \left(1 - \frac{8}{|\mathcal{F}|}\right)^{|S|} .$$

Now

$$\left(1 - \frac{8}{|\mathcal{F}|}\right)^{|S|} = 1 - \left(\frac{|\mathcal{F}|\left(1 - |\mathcal{F}|^{1000(\log(|\mathcal{F}|-8) - \log(|\mathcal{F}|))}\right)}{8000\log(|\mathcal{F}|)}\right) \cdot \left(\frac{8|S|}{|\mathcal{F}|}\right) \geq 1 - \frac{8|S|}{|\mathcal{F}|},$$

implies that

$$\mathbf{Pr}[\mathcal{E}_1] \leq \frac{8|S|}{|\mathcal{F}|} = \frac{8000\log|\mathcal{F}|}{|\mathcal{F}|}.$$

To bound $\mathbf{Pr}[\mathcal{E}_2]$, fix two sampled pairs $s_1, s_2 \in S$, and let us look at the event $X(s_1, s_2)$ that one of the constraints in s_1 is in the upper or lower envelope of the intersection point of the constraints in s_2. The union of upper and lower envelopes of the intersection point for s_2 contains at most 4 constraints. In the worst case, these 4 constraints lie in 4 distinct pairs of Π. Hence,

$$\mathbf{Pr}\left[X(s_1, s_2)\right] \le \frac{4}{|\mathcal{F}|/3 - 1} \le \frac{16}{|\mathcal{F}|}.$$

Then,

$$\mathbf{Pr}\left[\mathcal{E}_2\right] \le \sum_{i,j} \mathbf{Pr}\left[X(s_i, s_j)\right] \le 16\frac{|S|^2}{|\mathcal{F}|} = 16000000\frac{(\log|\mathcal{F}|)^2}{|\mathcal{F}|}.$$

We now bound $\mathbf{Pr}\left[\mathcal{E}_3\right]$. Suppose at most a 2/5-fraction of points in S go in direction *dir* to reach the optimum. Since the majority vote is *dir*, it means that when TESTER did not return fail in step 5c, it returned the value *dir* at least half of the time. Now, conditioned on $\neg\mathcal{E}_2$, for each pair (ℓ_A, ℓ_B) of constraints in S, the corresponding Φ_F and Φ_C contain the upper and lower envelope of the intersection point of the pair. Then, by Theorem 5.17, for a fixed (ℓ_A, ℓ_B), with probability $1 - 4\tilde{f}(p) - 2p$, we have that $U(\ell_A, \ell_B)$ and $L(\ell_A, \ell_B)$ are the upper and lower envelopes of the intersection point z of ℓ_A and ℓ_B. It follows by conditioning on $\neg\mathcal{E}_1$ and applying Theorem 5.18, TESTER correctly locates which side of the optimum z is on, with probability $1 - 4p$. So, conditioned on $\neg\mathcal{E}_1 \wedge \neg\mathcal{E}_2$, with probability at least $1 - 4\tilde{f}(p) - 6p$, any given pair in S has its intersection point correctly located with respect to the optimum. This probability is at least 999/1000 for p sufficiently small.

By assumption at most a 2/5-fraction of the points in S go in direction *dir* to reach the optimum, so at least a 3/5-fraction of the points in S goes in the opposite direction *dir'* to reach the optimum. Now in order for *dir* to be reported in step 6 it is necessary that at least a 1/10-fraction of the points in S is not correctly located with respect to the optimum. Since the comparisons performed are independent for the different iterations of step 5, if Z denotes the number of points in S that are correctly located, it follows by a Chernoff bound that

$$\mathbf{Pr}\left[\mathcal{E}_3\right] \le \mathbf{Pr}\left[Z < \left(\frac{9}{10}|S| + 1\right)\right] = \mathbf{Pr}\left[Z < \left(1 - \left(\frac{11}{111} - \frac{1000}{999|S|}\right)\right) \cdot |S| \cdot \frac{999}{1000}\right].$$

Now using $|S| \ge 1000$ which implies $1 - \left(\frac{11}{111} - \frac{1000}{999|S|}\right) \le 1 - \frac{10}{111}$ we obtain that

$$\mathbf{Pr}\left[\mathcal{E}_3\right] \le \mathbf{Pr}\left[Z < \left(1 - \frac{10}{111}\right) \cdot |S| \cdot \frac{999}{1000}\right] < e^{-\frac{1}{2}\cdot\left(\frac{10}{111}\right)^2 \cdot |S| \cdot \frac{999}{1000}} < |\mathcal{F}|^{-5}.$$

We can now bound $\mathbf{Pr}\left[\mathcal{E}_4\right]$ by a Chernoff and union bound because the tests for different constraints (ℓ_A, ℓ_B) are independent of each other. Given a fixed pair (ℓ_A, ℓ_B) of constraints whose intersection point is in direction *dir'* from at least a 3/4-fraction of the intersection points in S, let Z denote the number of invocations of SIDECOMPARATOR that return *dir'*. Now using the same estimate as in inequalities (5.7) – (5.10) shows that $\mathcal{E}_4$ occurs for pair (ℓ_A, ℓ_B) with probability at most $|\mathcal{F}|^{-6}$. By a union bound over $(\ell_A, \ell_B) \in \Pi \setminus S$,

$$\mathbf{Pr}\left[\mathcal{E}_4\right] \le \frac{1}{|\mathcal{F}|^5}.$$

Finally, we can bound $\mathbf{Pr}[\mathcal{E}_5]$. In the following, we condition on the event $\neg E_1 \wedge \neg E_2 \wedge \neg E_3 \wedge \neg E_4$. Since we conditioned on the event $\neg\mathcal{E}_1$, a constraint defining the optimum could only be discarded in step 7b. But this is only possible if a pair (ℓ_A, ℓ_B) of constraints in $\Pi \setminus S$ contains a constraint defining the optimum, and for at least a 2/3-fraction of pairs in S, (ℓ_A, ℓ_B) is to the right (resp. to the left) and *dir* is to the left (resp. to the right). Let *dir*′ denote the direction opposite to direction *dir*. Now if $(\ell_A, \ell_B) \in \Pi \setminus S$ is the pair defining the optimum, and either ℓ_A or ℓ_B is discarded in step 7b, then (ℓ_A, ℓ_B) lies in direction *dir*′ of at least a 2/3-fraction of the pairs in S. Since we conditioned on $\neg\mathcal{E}_3$, (ℓ_A, ℓ_B) also lies in direction *dir* of at least $2/5 \cdot |S|$ pairs of constraints, which is a contradiction because $2/3 + 2/5 > 1$. Therefore, if the pair defining the optimum is in $\Pi \setminus S$, we have $\mathbf{Pr}[\mathcal{E}_5] = 0$. Otherwise, we consider a pair (ℓ_A, ℓ_B) where ℓ_A is a constraint defining the optimum, whereas ℓ_B does not define the optimum. Remember that (ℓ_A, ℓ_B) is either a pair of floors or a pair of ceilings. Constraint ℓ_A is discarded in step 7b if (ℓ_A, ℓ_B) goes in direction *dir*′ to reach the optimum. Otherwise, ℓ_B is discarded, according to the slope ordering of ℓ_A and ℓ_B, so in the following we assume that (ℓ_A, ℓ_B) goes in direction *dir*′ to reach the optimum. Conditioned on $\neg\mathcal{E}_3$, there are at least $2/5 \cdot |S|$ pairs in S that go in direction *dir* to reach the optimum. Since (ℓ_A, ℓ_B) goes in direction *dir*′ to reach the optimum, there are at most $3/5 \cdot |S|$ pairs $(\ell_C, \ell_D) \in S$ where (ℓ_A, ℓ_B) lies in direction *dir*′ of (ℓ_C, ℓ_D). Thus in order for ℓ_A to be removed in step 7b it is required that at least a 1/15-fraction of the $|S|$ invocations of SideComparator$(\ell_A, \ell_B, \ell_C, \ell_D)$, where $(\ell_C, \ell_D) \in S$, fail. Now let Z denote the number of pairs $(\ell_C, \ell_D) \in S$ where in line 7a the invocation of SideComparator$(\ell_A, \ell_B, \ell_C, \ell_D)$ fails. Since the invocations of SideComparator are independent, we can apply a Chernoff bound to obtain

$$\begin{aligned}
\mathbf{Pr}\left[Z \geq \frac{1}{15}|S|\right] &= \mathbf{Pr}\left[Z \geq \left(1 + \frac{1}{15p} - 1\right) \cdot p \cdot |S|\right] \\
&= \mathbf{Pr}\left[Z \geq \left(1 + \frac{1}{15p} - 1\right) \cdot \mathbf{E}[Z]\right] \\
&\leq e^{-\frac{1}{3}\cdot\left(\frac{1}{15p}-1\right)^2 \cdot 1000 \cdot p \cdot \log|\mathcal{F}|} \\
&= |\mathcal{F}|^{-\frac{40(15p-1)^2}{27p\ln(2)}} \\
&\leq |\mathcal{F}|^{-6}
\end{aligned}$$

where we have used that $-\frac{40(15p-1)^2}{27p\ln(2)} \leq -6$ for all $p \leq 0.043$. Since there are at most two pairs of constraints containing a constraint defining the optimum, by a union bound, we have shown in the previous paragraph that

$$\mathbf{Pr}[\mathcal{E}_5] \leq 2 \cdot \frac{1}{|\mathcal{F}|^6} \leq \frac{1}{|\mathcal{F}|^5} \ .$$

Hence,

$$\begin{aligned}
\mathbf{Pr}[\neg\mathcal{E}_1 \wedge \neg\mathcal{E}_2 \wedge \neg\mathcal{E}_3 \wedge \neg\mathcal{E}_4 \wedge \neg\mathcal{E}_5] &\geq 1 - 8\frac{|S|}{|\mathcal{F}|} - 16\frac{|S|^2}{|\mathcal{F}|} - \frac{1}{|\mathcal{F}|^5} - \frac{1}{|\mathcal{F}|^5} - \frac{1}{|\mathcal{F}|^5} \\
&\geq 1 - 8\frac{|S|}{|\mathcal{F}|} - 16\frac{|S|^2}{|\mathcal{F}|} - \frac{3}{|\mathcal{F}|} \\
&\geq 1 - 9\frac{|S|}{|\mathcal{F}|} - 16\frac{|S|^2}{|\mathcal{F}|} \\
&= 1 - 9000\frac{\log|\mathcal{F}|}{|\mathcal{F}|} - 16000000\frac{(\log|\mathcal{F}|)^2}{|\mathcal{F}|} \\
&\geq 1 - 16010000\frac{(\log|\mathcal{F}|)^2}{|\mathcal{F}|} \quad .
\end{aligned}$$

By Theorem 5.19, our choice of constant $C(p) \geq \frac{200}{(7920399001p)^{1/5}}$ ensures that with probability at least $1-p$ in each round at least a 1/200-fraction of constraints is removed from $\mathcal{F}$. We now condition on this event and provide an upper bound on the probability that during *any* round any of the events $\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4$ or $\mathcal{E}_5$ occurs.

Let $a = 200/199$ and $A = 16010000$, for short. With probability $1-A(\log|\mathcal{F}|)^2/|\mathcal{F}| \geq 1 - A/|\mathcal{F}|^{1/2}$, no constraint of the optimum is discarded in any given round. By construction, the comparisons performed in different rounds are independent. Because in each round the algorithm discards at least a 1/200-fraction of the constraints in $\mathcal{F}$, for any fixed integer i, there is at most one invocation of the algorithm where $a^i \leq |\mathcal{F}| < a^{i+1}$ holds. Now, if $C(p) \geq a^k \geq \frac{200}{(7920399001p)^{1/5}}$, then with probability at most

$$\sum_{i=k}^{\infty} A \cdot \frac{1}{a^{i/2}} = \frac{A}{a^{(k-1)/2}(a^{1/2}-1)}$$

a constraint defining the optimum is removed from the set $\mathcal{F}$. If we choose $C(p) \geq a^k$ where $k = \frac{\ln a - 2\ln(p^{1/10}(a^{1/2}-1))}{\ln a}$, then this probability is less than $A \cdot p^{1/10}$. Note that by definition of a and k, a^k equals $\frac{a}{(a^{1/2}-1)^2 p^{1/5}} = \frac{200}{199(\frac{10}{100}\sqrt{398}-1)^2 p^{1/5}} \leq \frac{160000}{p^{1/5}}$. Finally, we define $C(p) = \frac{160000}{p^{1/5}}$.

For the analysis we have used the assumption that $p > 0$ is smaller than some constant $\tilde{p}$, and algorithm LP solves the problem by brute force in step 1 unless $|\mathcal{F}|$ is larger than a large enough constant $\tilde{\mathcal{F}}$ (in fact, it is sufficient to use $\tilde{\mathcal{F}} := 10^6$). These two assumptions can be summarized by the condition

$$p \leq \min\left\{\left(\frac{160000}{\tilde{\mathcal{F}}}\right)^5, \tilde{p}\right\},$$

since $p \leq \left(\frac{160000}{\tilde{\mathcal{F}}}\right)^5$ implies $\tilde{\mathcal{F}} \leq \frac{160000}{p^{1/5}} = C(p)$.

It remains to give an upper bound on the error probability for when the algorithm terminates in step 1. Let $B = 160000$ for short, then $C(p) = B/p^{1/5}$ by the definition of $C(p)$ above. Remember that for each primitive test (i. e., VerticalComparator or HorizontalComparator) the input consists of two pairs of constraints in $\mathcal{F}$. There are up to $\binom{B/p^{1/5}}{2}$ possible choices for the first pair (ℓ_A, ℓ_B), and for each such pair there are at most $\binom{B/p^{1/5}-2}{2}$ choices for the second pair (ℓ_C, ℓ_D). Hence the number of possible inputs is bounded by $\binom{B/p^{1/5}}{2} \cdot \binom{B/p^{1/5}-2}{2} \leq \frac{1}{4} \cdot B^4/p^{4/5}$, and thus the total number of different primitive queries in step 1 is less than $\frac{1}{2} \cdot B^4/p^{4/5}$. Now Markov's inequality implies that the probability for at least one comparison error is at most the expected number of errors, which is less than $p \cdot \frac{1}{2} \cdot B^4/p^{4/5} = \frac{1}{2} \cdot p^{1/5} \cdot B^4$. The proof of the theorem now follows by a union bound. $\square$

List of Figures

Bibliography

[1] P. K. Agarwal, R. Klein, C. Knauer, S. Langerman, P. Morin, M. Sharir, and M. A. Soss. Computing the detour and spanning ratio of paths, trees, and cycles in 2D and 3D. *Discrete & Computational Geometry*, 39(1-3):17–37, 2008.

[2] P. Appell. *Mémoire sur les déblais et les remblais des systèmes continus ou discontinus: présenté à l'Académie des sciences pour le concours du Prix Bordin pour 1884*. Imprimerie nationale, 1886.

[3] B. Aronov, M. de Berg, O. Cheong, J. Gudmundsson, H. J. Haverkort, M. H. M. Smid, and A. Vigneron. Sparse geometric graphs with small dilation. *Comput. Geom*, 40(3):207–219, 2008.

[4] T. Asano, S. K. Ghosh, and T. C. Shermer. Visibility in the plane. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 829–876. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000.

[5] F. Aurenhammer, F. Hoffmann, and B. Aronov. Minkowski-type theorems and least-squares clustering. *Algorithmica*, 20(1):61–76, 1998.

[6] F. Aurenhammer and R. Klein. Voronoi diagrams. In and J. R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 201–290. Elsevier, 1999.

[7] F. Aurenhammer, R. Klein, and D.-T. Lee. *Voronoi Diagrams and Delaunay Triangulations*. World Scientific Publishing Company, 2013.

[8] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *JCSS*, 47(3):549–595, Dec. 1993.

[9] P. Bose, L. Devroye, M. Löffler, J. Snoeyink, and V. Verma. The spanning ratio of the Delaunay triangulation is greater than pi/2. In *CCCG*, pages 165–167, 2009.

[10] M. Braverman and E. Mossel. Noisy sorting without resampling. In S.-H. Teng, editor, *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, San Francisco, California, USA, January 20-22, 2008*, pages 268–276. SIAM, 2008.

[11] M. Braverman and E. Mossel. Sorting from noisy information. *CoRR*, abs/0910.1191, 2009.

[12] S. Cai. Berechnung der Sichtbarkeitsregion zwischen zwei polygonalen Ketten mit einem Hindernis (Diploma thesis). Master's thesis, Universität Bonn, Institut für Informatik I, 2011.

[13] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete Comput. Geom.*, 6(5):485–524, 1991.

[14] K. L. Clarkson. Las vegas algorithms for linear and integer programming when the dimension is small. *J. ACM*, 42(2):488–499, 1995.

[15] C. Demetrescu, A. V. Goldberg, and D. S. Johnson. *The Shortest Path Problem: Ninth DIMACS Implementation Challenge*, volume 74. American Mathematical Soc., 2009.

[16] Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[17] D. P. Dobkin and S. Teller. Computer graphics. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 42, pages 779–796. CRC Press LLC, Boca Raton, FL, 1997.

[18] A. Dumitrescu, A. Ebbers-Baumann, A. Grüne, R. Klein, and G. Rote. On geometric dilation and halving chords. In F. K. H. A. Dehne, A. López-Ortiz, and J.-R. Sack, editors, *Algorithms and Data Structures, 9th International Workshop, WADS 2005, Waterloo, Canada, August 15-17, 2005, Proceedings*, volume 3608 of *Lecture Notes in Computer Science*, pages 244–255. Springer, 2005.

[19] A. Dumitrescu, A. Ebbers-Baumann, A. Grüne, R. Klein, and G. Rote. On the geometric dilation of closed curves, graphs, and point sets. *Computational Geometry: Theory and Applications*, 36(1):16–38, 2007.

[20] C. M. Eastman. *Building Product Models: Computer Environments Supporting Design and Construction.* CRC Press, 1999.

[21] C. M. Eastman, P. Teicholz, R. Sacks, and K. Liston. *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors.* John Wiley and Sons, 2008.

[22] A. Ebbers-Baumann, A. Grüne, and R. Klein. The geometric dilation of finite point sets. *Algorithmica*, 44(2):137–149, 2006.

[23] A. Ebbers-Baumann, A. Grüne, and R. Klein. Geometric dilation of closed planar curves: New lower bounds. *CGTA: Computational Geometry: Theory and Applications*, 37(3):188–208, 2007.

[24] A. Ebbers-Baumann, A. Grüne, R. Klein, M. Karpinski, C. Knauer, and A. Lingas. Embedding point sets into plane graphs of small dilation. *International Journal of Computational Geometry and Applications (IJCGA)*, 17(3):201–230, June 2007.

[25] A. Ebbers-Baumann, R. Klein, E. Langetepe, and A. Lingas. A fast algorithm for approximating the detour of a polygonal chain. *CGTA: Computational Geometry: Theory and Applications*, 27:123–134, 2004.

[26] D. Eppstein. The Geometry Junkyard.

[27] D. Eppstein. Spanning trees and spanners. In J. R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry.* Elsevier, 2000.

[28] D. Eppstein and K. A. Wortman. Minimum dilation stars. *CGTA: Computational Geometry: Theory and Applications*, 37(1):27–37, 2007.

[29] O. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint.* MIT Press, Cambridge, MA, 1993.

[30] U. Feige, D. Peleg, P. Raghavan, and E. Upfal. Computing with unreliable information (preliminary version). In H. Ortiz, editor, *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 128–137. ACM, 1990.

[31] I. Finocchi, F. Grandoni, and G. F. Italiano. Resilient dictionaries. *ACM Transactions on Algorithms*, 6(1):1:1–1:19, 2009.

[32] W. Gangbo and R. J. Mccann. The geometry of optimal transportation. *Acta Math*, 177:113–161, 1996.

[33] D. Geiß, R. Klein, and R. Penninger. Optimally solving a general transportation problem using Voronoi diagrams. In W. Didimo and G. Liotta, editors, *Abstracts of the 28th European Workshop on Computational Geometry. (EuroCG'12), Assisi, Italy, (March 2012): 241—244*, 2012.

[34] D. Geiß, R. Klein, and R. Penninger. Optimally solving a general transportation problem using Voronoi diagrams. In J. Gudmundsson, J. Mestre, and A. Viglas, editors, *Proceedings of 18th Annual International Computing and Combinatorics Conference (COCOON'12), Sydney, Australia, (August 2012): 264—274*, volume 7434 of *Lecture Notes in Computer Science.* Springer, 2012.

[35] D. Geiß, R. Klein, R. Penninger, and G. Rote. Optimally solving a transportation problem using Voronoi diagrams. *Comput. Geom.*, 46(8):1009–1016, 2013.

[36] S. K. Ghosh. Computing the visibility polygon from a convex set and related problems. *J. Algorithms*, 12(1):75–95, 1991.

[37] S. K. Ghosh. *Visisbility Algorithms in the plane.* Cambridge University Press, 2006.

[38] J. Goodman and J. O'Rourke. *Handbook of Discrete and Computational Geometry, Second Edition.* Discrete Mathematics and Its Applications. CRC Press, Inc., 2004.

[39] A. Grüne and S. Kamali. On the density of iterated line segment intersections. *Computational Geometry: Theory and Applications*, 40(1):23–36, 2008.

[40] L. J. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. E. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2:209–233, 1987.

[41] D. Ismailescu and R. Radoičić. A dense planar point set from iterated line intersections. *Computational Geometry: Theory and Applications*, 27(3):257 – 267, 2004.

[42] L. V. Kantorovich. On a problem of Monge (in russian). *Uspekhi Math. Nauk.*, 3:225–226, 1948.

[43] R. M. Karp and R. Kleinberg. Noisy binary search and its applications. In N. Bansal, K. Pruhs, and C. Stein, editors, *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, pages 881–890. SIAM, 2007.

[44] J. M. Keil and C. A. Gutwin. Classes of graphs which approximate the complete euclidean graph. *Discrete & Computational Geometry*, 7:13–28, 1992.

[45] R. Klein. *Algorithmische Geometrie.* Springer, Bonn, 2005.

[46] R. Klein and M. Kutz. The density of iterated crossing points and a gap result for triangulations of finite point sets. In N. Amenta and O. Cheong, editors, *Proceedings of the 22nd ACM Symposium on Computational Geometry, Sedona, Arizona, USA, June 5-7, 2006*, pages 264–272. ACM, 2006.

[47] R. Klein, R. Penninger, C. Sohler, and D. P. Woodruff. Tolerant algorithms. In C. Demetrescu and M. M. Halldórsson, editors, *ESA*, volume 6942 of *Lecture Notes in Computer Science*, pages 736–747. Springer, 2011.

[48] E. Langetepe, R. Penninger, and J. Tulke. Computing the visibility area between two simple polygons in linear time. In *Proceedings of the 26th European Workshop Computational Geometry*, pages 237–240, 2010.

[49] J.-C. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Boston, 1991.

[50] D. Lorenz. On the dilation of finite point sets (Diploma thesis), 2005.

[51] C. L. Mallows. Non-null ranking models. I. *Biometrika*, 44(1-2):114–130, June 1957.

[52] N. Megiddo. Linear programming in linear time when the dimension is fixed. *J. ACM*, 31(1):114–127, 1984.

[53] A. A. Melkman. On-line construction of the convex hull of a simple polyline. *Inf. Process. Lett.*, 25:11–12, April 1987.

[54] G. Monge. Mémoire sur la théorie des déblais et des remblais. *Histoire de l'Académie Royale des Sciences*, pages 666–704, 1781.

[55] G. Narasimhan and M. H. M. Smid. *Geometric Spanner Networks*. Cambridge University Press, 2007.

[56] J. O'Rourke. Visibility. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry, Second Edition*, chapter 28, pages 643–663. CRC Press, Inc., Boca Raton, FL, USA, 2004.

[57] A. Pelc. Searching games with errors - fifty years of coping with liars. *Theor. Comput. Sci*, 270(1-2):71–109, 2002.

[58] G. Rote. Two applications of point matching. In P. K. Agarwal, H. Alt, and M. Teillaud, editors, *Computational Geometry*, number 09111 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2009. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany.

[59] Y. Rubner, C. Tomasi, and L. J. Guibas. A metric for distributions with applications to image databases. In S. Chandran and U. Desai, editors, *Procedings of the Sixth International Conference on Computer Vision (ICCV-98)*, pages 59–66, New Delhi, Jan. 4–7 1998. Narosa Publishing House.

[60] R. Sharathkumar and P. K. Agarwal. Algorithms for the transportation problem in geometric settings. In Y. Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 306–317. SIAM, 2012.

[61] G. T. Toussaint. Solving geometric problems with the rotating calipers. In *Proc. IEEE MELECON '83*, pages A10.02/1–4, 1983.

[62] J. Tulke, M. Nour, and K. Beucke. Decomposition of BIM objects for scheduling and 4D simulation. In R. S. A. Zarli, editor, *Proceedings of the ECPPM 2008 e-Work and e-Business in Architecture Engineering and Construction*, 2008.

[63] P. M. Vaidya. Geometry helps in matching. *SIAM J. Comput*, 18(6):1201–1225, 1989.

[64] C. Villani. *Optimal Transport: Old and New.* Grundlehren der mathematischen Wissenschaften. Springer, 2009.