Geometry: Combinatorics & Algorithms
Lecture Notes HS 2025

Bernd Gärtner <gaertner@inf.ethz.ch>
Michael Hoffmann <hoffmann@inf.ethz.ch>
Patrick Schnider <schnpatr@inf.ethz.ch>
Emo Welzl <emo@inf.ethz.ch>
Manuel Wettstein <manuelwe@inf.ethz.ch>

Friday 24th July, 2026

Preface

These lecture notes are designed to accompany a course on "Geometry: Combinatorics & Algorithms" that we teach at the Department of Computer Science, ETH Zürich, since 2005. The topics covered have changed over the years, as has the name of the course. The current version is a synthesis of topics from computational geometry, combinatorial geometry, and graph drawing that are loosely centered around triangulations, that is, geometric representations of maximal plane graphs. The selection of topics is guided by the following criteria.

Importance. What are the most essential concepts and techniques that we want our students to know? (for instance, if they plan to write a thesis in the area)

Overlap. What is covered in other courses of our curriculum, and to which extent?

Coherence. How closely is something related to the focal topic of triangulations, and how well does it fit with the other topics selected?

Our main focus is on low-dimensional Euclidean space (mostly 2D), although we sometimes discuss possible extensions and/or remarkable differences when going to higher dimensions. At the end of each chapter there is a list of questions that we expect our students to be able to answer in the oral exam.

In the current setting, the course runs over 14 weeks, with three hours of lectures and two hours of exercises each week. In addition, two sets of graded homework are distributed over the course. The target audience are third-year Bachelor or Master students of Mathematics or Computer Science.

This year's course covers the material in Chapters 1–11. The appendices A–H contain material that appeared in previous editions of the course.

Most parts of these notes have gone through several iterations of proofreading over the years. But experience tells that there may always be some mistakes that escape detection. So in case you notice some problem, please let us know, regardless of whether it is a minor typo or punctuation error, a glitch in formulation, a hole in an argument, or other material (for instance, related to recent developments or improvements) that should be mentioned. This way the issue can be addressed for the next edition and future readers profit from your findings.

We thank Luis Barba, Kateřina Böhmová, Sergio Cabello, Tobias Christ, Cyril Frei, Anna Gundert, Vincent Kusters, Tillmann Miltzow, Gabriel Nivasch, Johannes Obenaus, Júlia Pap, Kalina Petrova, Alexander Pilz, Günter Rote, Marek Sulovský, May Szedlák, Hemant Tyagi, Paula Vondrlik, Alexandra Wesolek, and Joel Widmer for their helpful contributions. Special thanks go to Yanheng Wang for making a full pass over the material taught in 2022.

Bernd Gärtner, Michael Hoffmann, Patrick Schnider, Emo Welzl, and Manuel Wettstein
Department of Computer Science, ETH Zürich, OAT Z
Andreasstr. 5, CH-8092 Zürich, Switzerland
E-mail: {gaertner,hoffmann,schnpatr,emo,manuelwe}@inf.ethz.ch

Contents

Chapter 1

Fundamentals

1.1 Models of Computation

When designing algorithms, one has to agree on a model of computation according to which the algorithms are executed. There are various models to choose from, but when it comes to geometry, a Turing machine type model, for instance, would be rather inconvenient to represent and manipulate the frequent encounters of real numbers. Remember that even elementary geometric operations—such as taking the center of a circle defined by three points or computing the length of a circular arc—could quickly leave the realms of rational and even algebraic numbers.

Therefore, other models of computation are more prominent in the area of geometric algorithms and data structures. In this course we will be mostly concerned with two models: the *Real RAM* and the *algebraic computation/decision tree* model. The former is rather convenient when designing algorithms, as it abstracts away the aforementioned representation issues by simply *assuming* that it can be done. The latter typically appears in the context of lower bounds, that is, in proofs that solving a given problem requires at least certain amount of resource (as a function of the input size and possibly other parameters).

Let us look into these models in more detail.

Real RAM Model. RAM stands for random access machine, that is a machine whose memory cells are indexed by integers, and any specified cell can be accessed in constant time. "Real" means that each cell can store a real number. Any single arithmetic operation (addition, subtraction, multiplication, division, and k-th root, for small constant k) or comparison can be computed in constant time. [1]

This is a quite powerful model of computation, as a single real number in principle can encode an arbitrary amount of information. On the positive side, it allows to abstract

[1] In addition, sometimes also logarithms, other analytic functions, indirect addressing (integral), or floor and ceiling are used. As adding some of these operations makes the model more powerful, it is usually specified and emphasized explicitly when an algorithm uses them.

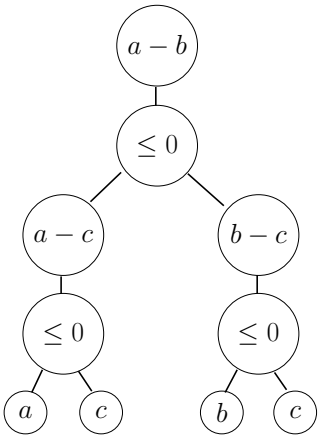
from the lowlands of numeric and algebraic computation and to concentrate on the algorithmic core from a combinatorial point of view.

But there are also downsides. First, the model is somewhat unrealistic, and it poses a challenge to efficiently implement an algorithm designed for it on an actual computer. With bounded memory there is no way to represent general real numbers explicitly, and operations using a symbolic representation can hardly be considered constant time. Therefore we have to ensure that we do not abuse the power of this model. For instance, we may want to restrict the numbers that are manipulated by any single arithmetic operation to be some fixed polynomial in the numbers that appear in the input.

Second, it is difficult if not impossible to derive reasonable lower bounds in the real RAM model. So when interested in lower bounds, it is convenient to use a different, less powerful model of computation. One such model is the computation tree model, which encompasses and explicitly represents all possible execution paths of an algorithm.

Algebraic Computation Trees (Ben-Or [1]). A model is as a rooted binary tree, where each node has at most two children. The computation starts at the root and proceeds down to leaves.

- Every node v with one child has an associated operation in $+, -, *, /, \sqrt{}, \ldots$. The operands of this operation are constant input values, or among v's ancestors in the tree.
- Every node v with two children is associated with a branching of the form > 0, $\geqslant 0$, or $= 0$. The branch is with respect to the result of v's parent node. If the expression yields `true`, the computation continues with the left child of v; otherwise, it continues with the right child of v.
- Every leaf contains the result of the computation.



The term *decision tree* is used if all of the final results (leaves) are either `true` or `false`. If every branch is based on a linear function in the input values, we face a *linear decision tree*. Analogously one can define, say, quadratic decision trees.

The complexity of a computation or decision tree is the maximum number of nodes among all root-to-leaf paths. It is well known that $\Omega(n \log n)$ comparisons are required to sort n numbers. But also for some problems that appear easier than sorting at first glance, the same lower bound holds. Consider, for instance, the following problem.

Element Uniqueness

Input: $\{x_1, \ldots, x_n\} \subset \mathbb{R}$, $n \in \mathbb{N}$.

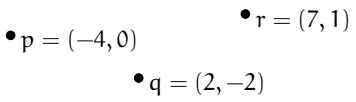
Output: Is $x_i = x_j$, for some $i, j \in \{1, \ldots, n\}$ with $i \neq j$?

Ben-Or [1] has shown that any algebraic decision tree to solve Element Uniqueness for n elements has complexity $\Omega(n \log n)$.

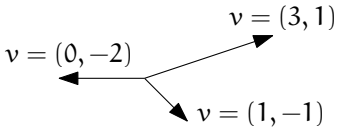
1.2 Basic Geometric Objects

We will mostly be concerned with the d-dimensional Euclidean space $\mathbb{R}^d$ for small $d \in \mathbb{N}$; typically $d = 2$ or $d = 3$. The basic objects of interest in $\mathbb{R}^d$ are the following.

Points. A point $p \in \mathbb{R}^d$ is typically described by its d Cartesian coordinates $p = (x_1, \ldots, x_d)$.

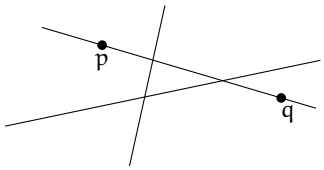


Vectors. A vector $v \in \mathbb{R}^d$ is typically described by its d Cartesian coordinates $v = (x_1, \ldots, x_d)$. Its length (in Euclidean metric) is denoted as $\|v\| := \sqrt{\sum_{i=1}^{d} {x_i}^2}$. If v has unit length, we also call it a direction.



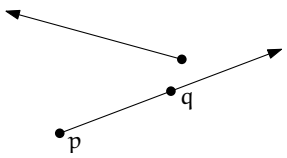
What is the difference between a point and a vector? Mathematically they are both elements of Euclidean space $\mathbb{R}^d$, hence they are the same. The different terms are used to indicate how we think of such an element in a given context. We think of a point as a location in space (a "dot"), while we think of a vector as a translation in space (an "arrow" starting from the origin). The point view is dominant in geometry. But since every point can be understood as a vector (the arrow from the origin to the dot), we can seamlessly apply vector space operations (addition, scalar multiplication) to points; and the resulting vector can be cast as a point again (the dot at the arrowhead). There are a number of sources that insist differentiating points and vectors, and they are right when it comes to how we interpret them. But when it comes to what they actually are, there is no need to make a difference.

Lines. A line is a one-dimensional affine subspace in $\mathbb{R}^d$. It can be described by two distinct points p and q as the point set $\{p + \lambda(q - p) : \lambda \in \mathbb{R}\}$.



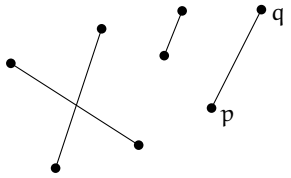
While any pair of distinct points defines a unique line, a line in $\mathbb{R}^2$ contains infinitely many points and so it may happen that a collection of three or more points lie on a line. Such a collection of points is termed *collinear*[2].

Rays. If we split a line at a point and only look into one direction from the point, then we obtain a ray. It can be described by two distinct points p and q as the point set $\{p + \lambda(q - p) : \lambda \geqslant 0\}$. The *orientation* of a ray is the vector $q - p$.



[2]Not *colinear*, which refers to a notion in the theory of coalgebras.

Line segments. A line segment is, as its name suggests, the segment between two points p, q on a line. It can be described as the point set $\{p + \lambda(q - p) : 0 \leqslant \lambda \leqslant 1\}$. We will also denote this line segment by $\overline{pq}$. Depending on the context we may allow or disallow *degenerate* line segments consisting of a single point only ($p = q$ in the above equation).



Hyperplanes and halfspaces. A hyperplane h is a $(d-1)$-dimensional affine subspace in $\mathbb{R}^d$. It can be described algebraically by $d+1$ coefficients $h_1, \ldots, h_{d+1} \in \mathbb{R}$ as the point set $\{(x_1, \ldots, x_d) \in \mathbb{R}^d : \sum_{i=1}^{d} h_i x_i = h_{d+1}\}$. Usually, we require at least one of $h_1, \ldots, h_d$ to be nonzero. Otherwise, the equation is satisfied by either all points (if $h_{d+1} = 0$) or no point (if $h_{d+1} \neq 0$), which we call a *degenerate* hyperplane. Degeneracy is useful in some contexts, and we will explicitly say so where we allow them. If we change "$=$" in the definition to "$\geqslant$", the obtained object is called a halfspace (or halfplane in $\mathbb{R}^2$).

Spheres and balls. A sphere is the set of all points that are equidistant to a fixed point. It can be described by its center $c \in \mathbb{R}^d$ and radius $r \in \mathbb{R}$ as the point set $\{x \in \mathbb{R}^d : \|x-c\| = r\}$. Likewise, the ball of radius r around c is the point set $\{x \in \mathbb{R}^d : \|x-c\| \leqslant r\}$. In $\mathbb{R}^2$, spheres and balls are called circles and disks, respectively.

1.3 Topology

In this section we review some basic concepts and notation from set-theoretic topology, on a level of what you also encounter in courses on real analysis typically. These concepts arise here and there, for instance, when we formalize intuitive objects such as "curves" and "polygons". They are also indispensible when we study certain abstract objects such as convex sets.

A set $P \subseteq \mathbb{R}^d$ is *bounded*, if it is contained in some ball $B_r := \{x \in \mathbb{R}^d : \|x\| \leqslant r\}$ of radius $r > 0$ around the origin.

A point $p \in \mathbb{R}^d$ is *interior* to $P \subseteq \mathbb{R}^d$, if for some $\varepsilon > 0$, there exists a ball $B_\varepsilon(p) = \{x \in \mathbb{R}^d : \|x - p\| \leqslant \varepsilon\}$ around it that is completely contained in P. A set is *open* if all of its points are interior; and it is *closed* if its complement is open. Beware that a set can be both open and closed (e.g. $\mathbb{R}^d$), or neither open nor closed (e.g. the interval $(0, 1]$ in one-dimension).

Finally, a set is *compact* if it is both bounded and closed. An important fact from analysis states that every continuous function from a compact set $P \subseteq \mathbb{R}^d$ to $\mathbb{R}$ attains its minimum/maximum at some point $p \in P$.

Exercise 1.1. *Determine for each of the following sets whether they are open or closed in* $\mathbb{R}^2$. *a)* $B_1(0)$ *b)* $\{(1, 0)\}$ *c)* $\mathbb{R}^2$ *d)* $\mathbb{R}^2 \backslash \mathbb{Z}^2$ *e)* $\mathbb{R}^2 \backslash \mathbb{Q}^2$ *f)* $\{(x, y) : x \in \mathbb{R}, y \geqslant 0\}$

Exercise 1.2. *Show that the union of countably many open sets in $\mathbb{R}^d$ is open. Show that the union of a finite number of closed sets in $\mathbb{R}^d$ is closed. (For the curious reader: These are two of the axioms of an abstract topology. So here we show that the Euclidean space is a topology.) What follows for intersections of open and closed sets? Finally, show that the union of countably many closed sets in $\mathbb{R}^d$ is not necessarily closed.*

The *boundary* ∂P of $P \subseteq \mathbb{R}^d$ consists of all points in $\mathbb{R}^d$ (not necessarily in P) that are neither interior to P nor to $\mathbb{R}^d \setminus P$. In other words, $p \in \partial P$ if every ball $B_\varepsilon(p)$ intersects both P and $\mathbb{R}^d \setminus P$.

Sometimes one wants to approximate a set $P \subseteq \mathbb{R}^d$ by an open/closed set. In the former scenario we can resort to its *interior* P° formed by all points interior to P. It is not hard to see that $P^\circ = P \setminus \partial P$. Similarly, in the latter scenario one can use its *closure* $\overline{P} = P \cup \partial P$.

Exercise 1.3. *Show that for any $P \subseteq \mathbb{R}^d$ the interior P° is open, and the closure $\overline{P}$ is closed. (Why is there something to show to begin with?)*

What is the interior of a lower-dimensional object living in a higher dimensional space, such as a line segment in $\mathbb{R}^2$ or a triangle in $\mathbb{R}^3$? The answer is $\emptyset$, because the balls considered in the definition are higher-dimensional creatures which will always contain points from the "outer space". To overcome this undesirable artefact, we can use the notion of a *relative interior*, denoted by $\operatorname{relint}(S)$. It refers to the interior of S where all the "balls" in the definition are restricted to live in the smallest affine subspace that contains S.

For instance, the smallest affine subspace that contains the line segment $\overline{pq}$ in $\mathbb{R}^2$ is the line through p, q. So all the "balls" considered by the relative interior will be intervals of that line, thus $\operatorname{relint}(\overline{pq}) = \overline{pq} \setminus \{p, q\}$. Similarly, the smallest affine subspace that contains a triangle in $\mathbb{R}^3$ is a plane. Hence the relative interior is just the interior of the triangle, considered as a two-dimensional object.

1.4 Graphs

Next we review some basic definitions and properties of graphs. For more details and proofs, refer to any standard textbook on graph theory [2, 3, 5].

A (simple undirected) graph $G = (V, E)$ is defined on a set V of *vertices* whose pairwise relations are captured by the set $E \subseteq \binom{V}{2}$ of edges. Unless stated otherwise, V is always finite. Two vertices u, v are *adjacent* if $\{u, v\} \in E$, in which case both vertices are *incident* to the edge $\{u, v\}$. To avoid clutter we often omit brackets and write uv for edge $\{u, v\}$.

For a vertex $v \in V$, its *neighborhood* in G, denoted $N_G(v)$, consists of all vertices from G that are adjacent to v. Similarly, for a set $W \subset V$ of vertices its neighborhood $N_G(W)$ is defined as $\bigcup_{w \in W} N_G(w)$. The *degree* $\deg_G(v)$ of a vertex $v \in V$ is the size of its

neighborhood, that is, the number of edges from E incident to v. The subscript is often omitted if the graph under consideration is clear from the context.

Lemma 1.4 (Handshaking Lemma). *In any graph* $G = (V, E)$ *we have*

$$\sum_{v \in V} \deg(v) = 2|E|.$$

Two graphs $G = (V, E)$ and $H = (U, F)$ are *isomorphic*, denoted $G \simeq H$, if there is a bijection $\phi : V \to U$ such that $\{u, v\} \in E \iff \{\phi(u), \phi(v)\} \in F$. Such a bijection ϕ is called an *isomorphism* between G and H. The structure of isomorphic graphs is identical and often we do not distinguish between them when looking at them as graphs.

For a graph G denote by $V(G)$ the set of vertices and by $E(G)$ the set of edges. A graph $H = (U, F)$ is a *subgraph* of G if $U \subseteq V$ and $F \subseteq E$. In case that $U = V$ the graph H is a *spanning* subgraph of G. For a set $U \subseteq V$ of vertices denote by $G[U]$ the *induced subgraph* of G on U, that is, the graph $(U, E \cap \binom{U}{2})$. For $F \subseteq E$ denote $G \setminus F := (V, E \setminus F)$. Similarly, for $U \subseteq V$ denote $G \setminus U := G[V \setminus U]$. In particular, for a vertex or edge $x \in V \cup E$ we write $G \setminus x$ for $G \setminus \{x\}$. The *union* of two graphs $G = (V, E)$ and $H = (U, F)$ is the graph $G \cup H := (V \cup U, E \cup F)$.

For an edge $e = uv \in E$ the graph G/e is obtained from $G \setminus \{u, v\}$ by adding a new vertex w with $N_{G/e}(w) := (N_G(u) \cup N_G(v)) \setminus \{u, v\}$. This process is called *contraction* of e in G. Similarly, for a set $F \subseteq E$ of edges the graph G/F is obtained from G by contracting all edges from F (the order in which the edges from F are contracted does not matter).

Graph traversals. A *walk* in G is a sequence $W = (v_1, \ldots, v_k)$, $k \in \mathbb{N}$, of vertices such that v_i and v_{i+1} are adjacent in G, for all $1 \leqslant i < k$. The vertices v_1 and v_k are referred to as the walk's *endpoints*, and the other vertices its *interior*. A walk with endpoints v_1 and v_k is sometimes called a walk *between* v_1 and v_k. If the endpoints coincide (namely $v_1 = v_k$), then the walk is *closed*; otherwise it is *open*. For a walk W denote by $V(W)$ its set of vertices and by $E(W)$ its set of edges (that is, pairs of consecutive vertices along W). We say that W *visits* its vertices and edges.

A walk that uses each edge of G at most once is called a *trail*. A closed walk that visits each edge (hence also each vertex) at least once is called a *tour* of G. An *Euler tour* is both a trail and a tour of G, that is, it visits each edge of G exactly once. A graph that contains an Euler tour is termed *Eulerian*.

If the vertices $v_1, \ldots, v_k$ of a closed walk W are pairwise distinct except for $v_1 = v_k$, then W is a *cycle* of size $k - 1$. If the vertices $v_1, \ldots, v_k$ of a walk W are pairwise distinct, then W is a *path* of size k. A *Hamilton cycle (path)* is a cycle (path) that visits every vertex of G. A graph that contains a Hamilton cycle is *Hamiltonian*.

Two trails are *edge-disjoint* if they do not share any edge. Two paths are called *internally vertex-disjoint* if they do not share any vertices (except for potential common endpoints). For two vertices $s, t \in V$ any path with endpoints s and t is called an (s, t)-*path* or a path *between* s and t.

Connectivity. Define an equivalence relation "$\sim$" on V by setting $a \sim b$ if and only if there is a path between a and b in G. The equivalence classes with respect to "$\sim$" are called *components* of G. A graph G is *connected* if it has only one component, and *disconnected* otherwise.

A set $C \subset V$ of vertices in a connected graph $G = (V, E)$ is a *cut-set* of G if $G \setminus C$ is disconnected. A graph is *k-connected*, for a positive integer k, if $|V| \geqslant k + 1$ and every cut-set has at least k vertices. Similarly a graph $G = (V, E)$ is *k-edge-connected*, if $G \setminus F$ is connected, for any set $F \subseteq E$ of at most $k - 1$ edges. Connectivity and cut-sets are related via the following well-known theorem.

Theorem 1.5 (Menger [4]). *For any two nonadjacent vertices u, v of a graph $G = (V, E)$, the minimum size of a cut-set that disconnects u and v is the same as the maximum number of pairwise internally vertex-disjoint paths between u and v.*

Specific families of graphs. A graph with all potential edges present, that is $(V, \binom{V}{2})$, is called a *clique*. Up to isomorphism there is only one clique on n vertices; it is referred to as the *complete graph* K_n, for $n \in \mathbb{N}$. At the other extreme, the *empty graph* $\overline{K_n}$ consists of n isolated vertices, so no edge is present. A set U of vertices in a graph G is *independent* if $G[U]$ is an empty graph. A graph whose vertex set can be partitioned into two independent sets is *bipartite*. An equivalent characterization states that a graph is bipartite if and only if it does not contain any odd cycle. The bipartite graphs with a maximum number of edges (unique up to isomorphism) are the *complete bipartite graphs* $K_{m,n}$, for $m, n \in \mathbb{N}$. They consist of two disjoint independent sets of size m and n, respectively, and all mn edges in between.

A *forest* is a graph that is *acyclic*, that is, it does not contain any cycle. A connected forest is called *tree* and its *leaves* are the vertices of degree one. Every connected graph contains a spanning subgraph which is a tree—a so-called *spanning tree*. Beyond the definition given above, there are several equivalent characterizations of trees.

Theorem 1.6. *The following statements for a graph G are equivalent.*

(1) G is a tree (that is, it is connected and acyclic).

(2) G is a connected graph with n vertices and $n - 1$ edges.

(3) G is an acyclic graph with n vertices and $n - 1$ edges.

(4) Any two vertices in G are connected by a unique path.

(5) G is minimally connected, that is, G is connected but removal of any single edge yields a disconnected graph.

(6) G is maximally acyclic, that is, G is acyclic but adding any single edge creates a cycle.

Directed graphs. In a directed graph or, short, *digraph* $D = (V, E)$ the set E consists of ordered pairs of vertices, that is, $E \subseteq V^2$. The elements of E are referred to as *arcs*. To avoid clutter we often omit brackets and write uv for an arc (u, v). An arc $uv \in E$ is said to be directed from its *source* u to its *target* v. For $uv \in E$ we also say "there is an arc from u to v in D". Usually, we consider *loop-free* graphs, that is, arcs of the type vv, for some $v \in V$, are not allowed.

The *in-degree* $\deg^-_D(v) := |\{(u, v) | uv \in E\}|$ of a vertex $v \in V$ is the number of *incoming* arcs at v. Similarly, the *out-degree* $\deg^+_D(v) := |\{(v, u) \mid vu \in E\}|$ of a vertex $v \in V$ is the number of *outgoing* arcs at v. Again the subscript is often omitted when the graph under consideration is clear from the context.

From any undirected graph G one can obtain a digraph on the same vertex set by specifying a direction for each edge of G. Each of these $2^{|E(G)|}$ different digraphs is called an *orientation* of G. Similarly every digraph $D = (V, E)$ has an *underlying* undirected graph $G = (V, \{\, \{u, v\} \mid (u, v) \in E \text{ or } (v, u) \in E\})$. Hence most of the terminology for undirected graphs carries over to digraphs.

A *directed walk* in a digraph D is a sequence $W = (v_1, \ldots, v_k)$, for some $k \in \mathbb{N}$, of vertices such that there is an arc from v_i to v_{i+1} in D, for all $1 \leqslant i < k$. In the same way we define *directed trails*, *directed tours*, *directed paths*, and *directed cycles*.

Multigraphs. Sometimes we also consider *multigraphs*, where each edge may have multiple copies. Unless forbidden explictly, a multigraph may contain loops. Just as simple graphs/digraphs, multigraphs may be undirected or directed, and also most of the other basic notions for graphs discussed above naturally generalize to multigraphs.

References

[1] Michael Ben-Or, Lower bounds for algebraic computation trees. In *Proc. 15th Annu. ACM Sympos. Theory Comput.*, pp. 80–86, 1983.

[2] John Adrian Bondy and U. S. R. Murty, *Graph Theory*, vol. 244 of *Graduate texts in Mathematics*, Springer, London, 2008.

[3] Reinhard Diestel, *Graph Theory*, vol. 173 of *Graduate texts in Mathematics*, Springer, Heidelberg, 5th edn., 2016.

[4] Karl Menger, Zur allgemeinen Kurventheorie. *Fund. Math.*, 10/1, (1927), 96–115.

[5] Douglas B. West, *Introduction to Graph Theory*, Prentice Hall, Upper Saddle River, NJ, 2nd edn., 2001.

Chapter 2

Plane Embeddings

Graphs can be represented in various ways, for instance, as an adjacency matrix or using adjacency lists. In this chapter we explore another class of representations that are quite different in nature, namely *geometric* representations. In a geometric representation, vertices and edges are represented by geometric objects, for example points and curves. This approach is appealing because it succinctly visualizes a graph along with its many properties. We have many degrees of freedom in selecting the geometric objects and the details of their geometry. This freedom allows us to tailor the representation to meet specific goals, such as emphasizing certain structural aspects of the graph at hand or reducing the complexity of the obtained representation.

The most common geometric graph representation is a *drawing*, where vertices are mapped to points and edges to curves in $\mathbb{R}^2$. It is desirable to make such a map injective by avoiding edge crossings, both from a mathematically aesthetic viewpoint and for the sake of the practical readability. Those graphs that allow such an *embedding* into the Euclidean plane are known as *planar*. Our goal is to study the interplay between abstract planar graphs and their plane embeddings. Specifically, we want to answer the following questions:

- What is the combinatorial complexity (that is, the number of edges and faces) of planar graphs?
- Under which conditions are plane embeddings unique (up to a certain sense of equivalence)?
- How can we represent plane embeddings in a data structure?
- What is the geometric complexity (that is, the encoding size of the geometric objects used to represent vertices and edges) of plane embeddings?

Most definitions we use directly extend to multigraphs. But for simplicity, we use the term “graph” throughout.

2.1 Drawings, Embeddings and Planarity

A **curve** is a set $C \subset \mathbb{R}^2$ of the form $\{\gamma(t) : 0 \leqslant t \leqslant 1\}$, where $\gamma : [0,1] \to \mathbb{R}^2$ is a continuous function. The function γ is called a *parameterization* of C. The points $\gamma(0)$ and $\gamma(1)$ are the *endpoints* of the curve. A curve is *closed* if $\gamma(0) = \gamma(1)$. A curve is *simple* if it admits a parameterization γ that is injective on $[0,1]$; for a closed simple curve we allow as an exception that $\gamma(0) = \gamma(1)$. The following famous theorem describes an important property of the plane. A proof can, for instance, be found in the book of Mohar and Thomassen [24].

Theorem 2.1 (Jordan). *Any simple closed curve C partitions the plane into exactly two regions (connected open sets), each bounded by C.*

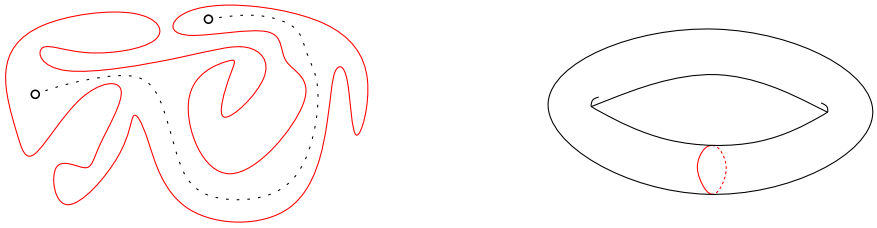


Figure 2.1: *Left: a simple closed curve in the plane and two points in one of its faces. Right: a simple closed curve that does not disconnect the torus.*

Observe that, for instance, on the torus there are simple closed curves that do not disconnect the surface, and thus the theorem does not hold there.

Drawings. As a first criterion for a reasonable geometric representation of a graph, we would like to have a clear separation between different vertices and also between a vertex and nonincident edges. Formally, a *drawing* of a graph $G = (V, E)$ in the plane is a function f that assigns

- a point $f(v) \in \mathbb{R}^2$ to every vertex $v \in V$ and
- a simple curve $f(uv)$ with endpoints $f(u)$ and $f(v)$ to every edge $uv \in E$,

such that

(1) f is injective on V and

(2) $f(uv) \cap f(V) = \{f(u), f(v)\}$, for every edge $uv \in E$.

A common point $f(e) \cap f(e')$ between two curves that represent distinct edges $e, e' \in E$ is called a *crossing* if it is not a common endpoint of e and e'.

Commonly, when discussing a drawing of a graph $G = (V, E)$, we do not differentiate a vertex/an edge from its geometric realization. That is, a vertex $v \in V$ is identified with the point $f(v)$, and an edge $e \in E$ is identified with the curve $f(e)$. For instance, the last

sentence in the previous paragraph may be phrased as "A common point of two edges is called a crossing if it is not their common endpoint."

Often it is convenient to make additional assumptions about edge intersections in a drawing. For example, we may demand *nondegeneracy* in the sense that no three edges can meet at a single crossing, or that any two edges can intersect at only finitely many points.

Planar vs. plane. A graph is *planar* if it admits a drawing in the plane without crossings. Such a drawing is also called a *crossing-free* drawing or a (plane) *embedding* of the graph. A planar graph together with a particular plane embedding is called a *plane* graph. Note the distinction between "planar" and "plane": the former refers to an abstract graph and indicates the possibility of an embedding, whereas the latter refers to a concrete embedding (Figure 2.2).

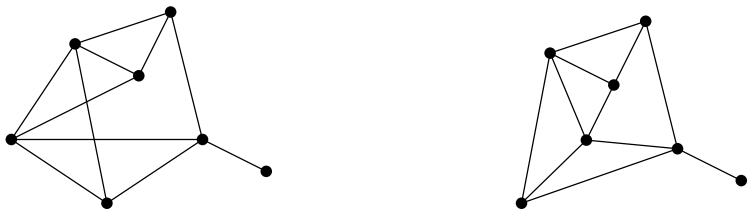


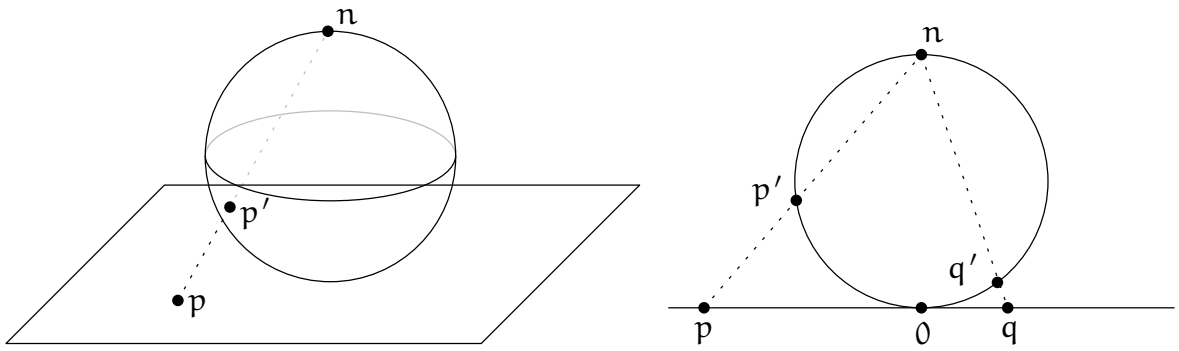
Figure 2.2: *A planar graph (left) and a plane embedding of it (right).*

A *geometric graph* is a graph together with a drawing in which all edges are straight-line segments. Note that such a drawing is fully determined by the vertex positions. A plane graph which is also geometric is called a *plane straight-line graph* (PSLG). On the other hand, a plane graph whose edges are arbitrary simple curves is emphasized as *topological plane graph*.

The *faces* of a plane graph G are the maximally connected regions of $\mathbb{R}^2 \setminus G$, that is, the plane without the points occupied by the embedding (as the image of a vertex or an edge). Each embedding of a finite graph has exactly one *unbounded face*, also called *outer* or *infinite* face. Using stereographic projection, we could show that any face can be swapped out to serve as the unbounded face:

Theorem 2.2. *If a graph G has a plane embedding in which some face is bounded by a cycle $(v_1, \dots, v_k)$, then G also has a plane embedding in which the unbounded face is bounded by the cycle $(v_1, \dots, v_k)$.*

Proof Sketch. Take a plane embedding Γ of G and map it to the sphere using *stereographic projection*: Imagine $\mathbb{R}^2$ being the x/y-plane in $\mathbb{R}^3$ and place a unit sphere S whose south pole touches the origin. We establish a bijection between $\mathbb{R}^2$ and $S \setminus \{n\}$, where $n := (0, 0, 2)$ is the north pole position: A point $p \in \mathbb{R}^2$ is mapped to the intersection p' of the segment $\overline{pn}$ and S, see Figure 2.3. The map is continuous, so it preserves incidence between vertices, edges and faces.

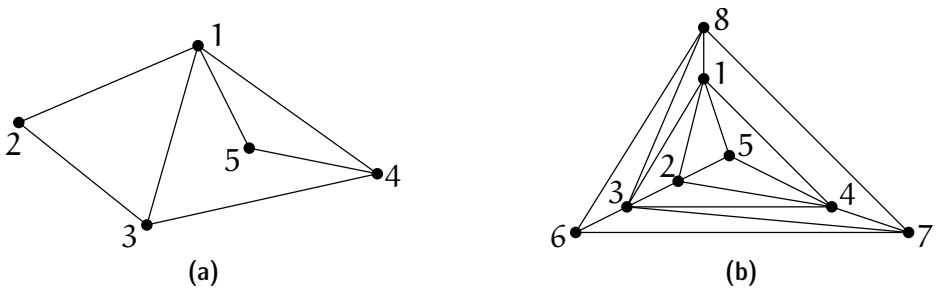


(a) Three-dimensional view. **(b)** Cross-section view.

Figure 2.3: *Stereographic projection.*

Consider the resulting embedding Γ' of G on S: The infinite face of Γ corresponds to the face of Γ' that contains the north pole n of S. Now rotate the embedding Γ' on S such that the desired face contains n. Mapping back to the plane using stereographic projection results in an embedding in which the desired face is the outer face. □

Exercise 2.3. *Consider the plane graphs depicted in Figure 2.4. For both graphs give a plane embedding in which the cycle* $(1, 2, 3)$ *bounds the outer face.*



(a) **(b)**

Figure 2.4: *Make* $(1, 2, 3)$ *bound the outer face.*

Duality. Every plane graph G has a *dual* G^* whose vertices are the faces of G. For every edge in G, we connect its two incident faces by an edge in the dual G^*. Note that in general, G^* is a multigraph (with loops and multiple edges) and may depend on the embedding. So an abstract planar graph G may have several nonisomorphic duals; see Figure 2.5 for an example. If G is a connected plane graph, then $(G^*)^* = G$. We will see later in Section 2.3 that the dual of a 3-connected planar graph is unique (up to isomorphism).

The Euler Formula and its ramifications. One of the most important tools for planar graphs (and more generally, graphs embedded on a surface) is the Euler–Poincaré Formula.

Figure 2.5: *Two plane drawings G_1 and G_2 of the same abstract planar graph and their duals G_1^* and G_2^* with $G_1^* \not\simeq G_2^*$. (To see this, for instance, count the number of vertices of degree greater than three.)*

Theorem 2.4 (Euler's Formula). *For every connected plane graph with n vertices, e edges, and f faces, we have $n - e + f = 2$.*

Proof. Let G be a connected plane graph with n vertices, e edges, and f faces. Note that $e \geqslant n - 1$ as G is connected.

We prove the statement by induction on $e - n$. In the base case $e - n = -1$, the graph G is a (plane) tree and contains exactly one (unbounded) face, and so $n - e + f = 1 + 1 = 2$ as claimed.

In the general case, fix a spanning tree T of G, pick an arbitrary edge e of $G \setminus T$, and consider the graph $G^- = G \setminus e$. By construction it has n vertices and $e - 1$ edges. We claim that it has $f - 1$ faces. To see this observe that $G^- \supset T$ is connected. In particular, the endpoints of e are connected by a path in G^-, which together with e forms a cycle in G. So in G, any two points sufficiently close to but on opposite sides of e are in different faces, whereas they are in the same face of G^-. In other words, the two incident faces of e are distinct in G but merged into one in G^-. All other faces remain untouched. It follows that G^- has $f - 1$ faces, as claimed. Then by the inductive assumption on G^-, we have $n - e + f = n - (e - 1) + (f - 1) = 2$, which concludes the induction. □

In particular, this shows that every plane embedding of a planar graph has the same number of faces. In other words, the number of faces is an invariant of an abstract planar graph. It also follows (as the corollary below) that planar graphs are *sparse*, that is, they have a linear number of edges and faces only. So the asymptotic complexity of a planar graph is already determined by its number of vertices.

Corollary 2.5. *A simple planar graph on $n \geqslant 3$ vertices has at most $3n - 6$ edges and at most $2n - 4$ faces.*

Proof. Without loss of generality we may assume that G is connected. (If not, add edges between components of G until the graph is connected. The number of edges increases and the number of faces remains unchanged.) The statement is easily checked for $n = 3$, where G is either a triangle or a path and therefore has no more than $3 \leqslant 3 \cdot 3 - 6$ edges and no more than $2 \leqslant 2 \cdot 3 - 4$ faces. Next consider a simple connected planar graph G

on $n \geqslant 4$ vertices, and fix any plane embedding of it. Denote by E its set of edges and by F its set of faces. Let

$$X = \{(e, f) \in E \times F : e \text{ bounds } f\}$$

denote the set of incident edge-face pairs. We count X in two different ways.

First note that each edge bounds at most two faces and so $|X| \leqslant 2 \cdot |E|$.

Second note that every face is bounded by at least three edges: If G contains a cycle, then the boundary of every face shall contain a cycle and hence at least three edges. If G is acyclic, then it must be a tree since we assumed it to be connected. Its only face (the outer face) is bounded by all edges; and there are at least three since G contains at least four vertices. In both cases we have $|X| \geqslant 3 \cdot |F|$.

Therefore $3|F| \leqslant 2|E|$. Using Euler's Formula we conclude that

$$\begin{aligned} 4 &= 2(n - |E| + |F|) \leqslant 2n - 3|F| + 2|F| = 2n - |F| \text{ and} \\ 6 &= 3(n - |E| + |F|) \leqslant 3n - 3|E| + 2|E| = 3n - |E| , \end{aligned}$$

which yield the claimed bounds. □

Corollary 2.5 implies that the degree of a "typical" vertex in a planar graph is a small constant.

Corollary 2.6. *The average vertex degree in a simple planar graph is less than six.*

Exercise 2.7. *Prove Corollary 2.6.*

There exist several variations of this statement, a few more of which we will encounter during this course.

Exercise 2.8. *Show that neither K_5 (the complete graph on five vertices) nor $K_{3,3}$ (the complete bipartite graph where both classes have three vertices) is planar.*

Exercise 2.9. *Let P be a set of $n \geqslant 3$ points in the plane such that the distance between every pair of points is at least one. Show that there are at most $3n - 6$ pairs of points in P at distance exactly one.*

Characterizing planarity. The classical theorems of Kuratowski and Wagner provide a characterization of planar graphs in terms of forbidden substructures. A *subdivision* of a graph $G = (V, E)$ is obtained from G by replacing each edge with a path.

Theorem 2.10 (Kuratowski [22, 31]). *A graph is planar if and only if it does not contain a subdivision of $K_{3,3}$ or K_5.*

A *minor* of a graph $G = (V, E)$ is obtained from G using zero or more edge contractions, edge deletions, and/or vertex deletions.

Theorem 2.11 (Wagner [34]). *A graph is planar if and only if it does not contain $K_{3,3}$ or K_5 as a minor.*

In some sense, Wagner's Theorem is a special instance[1] of a much more general theorem.

Theorem 2.12 (Graph Minor Theorem, Robertson/Seymour [28]). *Every minor-closed family of graphs can be described in terms of a finite set of forbidden minors.*

Being *minor-closed* means that any minor of any graph from the family also belongs to the family. For instance, the family of planar graphs is minor-closed because planarity is preserved under removal of edges and vertices and under edge contractions.

Exercise 2.13. *A graph is* 1*-planar if it admits a drawing in the plane in which every edge has at most one crossing. Prove or disprove: The family of* 1*-planar graphs is minor-closed.*

The Graph Minor Theorem is a celebrated result established by Robertson and Seymour in a series of twenty papers, see also the survey by Lovász [23]. They also describe an $O(n^3)$ algorithm (with horrendous constants, though) to decide whether a graph on n vertices contains a fixed (constant-size) minor. As a consequence, every minor-closed property can be tested in polynomial time. Later, Kawarabayashi et al. [20] showed that this problem can be solved in $O(n^2)$ time.

Unfortunately, the Graph Minor Theorem is nonconstructive in the sense that in general we do not know how to obtain the set of forbidden minors for a given family. For instance, for the family of toroidal graphs (graphs that can be embedded without crossings on the torus) more than 16′000 forbidden minors are known, and the theorem tells us that the number is finite, but we still do not know the concrete number. So while we know that there exists a quadratic time algorithm to test membership for minor-closed families, we have no idea what such an algorithm looks like in general.

Graph families other than planar graphs for which the forbidden minors are known include forests (free of K_3 minors) and outerplanar graphs (free of $K_{2,3}$ and K_4 minors). A graph is *outerplanar* if it admits a plane embedding in which all vertices appear on the outer face (Figure 2.6).

Figure 2.6: *An outerplanar graph (left) and a plane embedding of it in which all vertices are incident to the outer face (right).*

Exercise 2.14. *(a) Give an example of a* 6*-connected planar graph or argue that no such graph exists.*

[1] It is more than just a special instance because it also specifies the forbidden minors explicitly.

(b) Give an example of a 5-connected planar graph or argue that no such graph exists.

(c) Give an example of a 3-connected outerplanar graph or argue that no such graph exists.

Planarity testing. To test a given graph for planarity we do not have to contend ourselves with a quadratic-time algorithm. In fact, there exist a number of different linear time algorithms that decide if a given abstract graph is planar; all of them—from a very high-level point of view—can be regarded as an annotated depth-first-search. The first such algorithm was described by Hopcroft and Tarjan [19], while the current state-of-the-art is probably among the "path searching" method by Boyer and Myrwold [6] and the "LR-partition" method by de Fraysseix et al. [14]. Although the overall idea in all these approaches is easy to convey, many technical details make an in-depth discussion rather painful to go through.

2.2 Graph Representations

There are two standard representations for an abstract graph $G = (V, E)$ on $n = |V|$ vertices. For the *adjacency matrix* representation we consider the vertices to be ordered as $V = \{v_1, \ldots, v_n\}$. The adjacency matrix of an undirected graph is a symmetric $n \times n$-matrix $A = (a_{ij})_{1 \leqslant i,j \leqslant n}$ where $a_{ij} = a_{ji} = 1$, if $\{v_i, v_j\} \in E$, and $a_{ij} = a_{ji} = 0$ otherwise. Storing such a matrix explicitly requires $\Omega(n^2)$ space, but it allows testing in constant time whether or not two given vertices are adjacent.

In an *adjacency list* representation, we store for each vertex a list of its neighbors in G. This requires only $O(n+|E|)$ storage, which is better than for the adjacency matrix in case that $|E| = o(n^2)$. On the other hand, the adjacency test for two given vertices is not a constant-time operation, because it requires a search in one of the lists. Depending on the implementation of the lists, the search time ranges from $O(d)$ (for an unsorted list) to $O(\log d)$ (for a sorted dynamic data structure such as a balanced search tree), where d is the minimum degree of the two vertices.

Both representations have their merits. The choice typically depends on what one wants to do with the graph. When dealing with embedded graphs, however, additional information about the embedding is needed beyond the pure incidence structure of the graph. The next section discusses a standard data structure to represent embedded graphs.

2.2.1 The Doubly-Connected Edge List

The *doubly-connected edge list* (DCEL) is a data structure to represent a plane graph in such a way that it is easy to traverse and to manipulate. To avoid complications, let us discuss only connected graphs that contain at least two vertices. It is not hard to extend the data structure to be able to represent all plane graphs. We also assume

that we deal with a straight-line embedding and thus the geometry of edges is defined by the positions of their endpoints already. For more general embeddings, the geometric description of edges has to be stored in addition.

The main building block of a DCEL is a list of *halfedges*. Every actual edge is split into two halfedges going in opposite direction, and these are called *twins*, see Figure 2.7. Along the boundary of each face, halfedges are oriented counterclockwise, that is, the face always stays to the left.

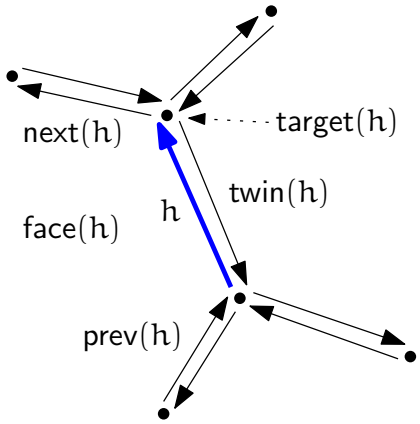


Figure 2.7: *A halfedge in a DCEL.*

A DCEL also stores a list of vertices and a list of faces. These three lists are unordered but interconnected by various pointers. A vertex v stores a pointer $\mathsf{halfedge}(v)$ to an arbitrary halfedge originating from v. Every vertex also records its coordinates $\mathsf{point}(v)$, that is, the point it is mapped to in the embedding. A face f stores a pointer $\mathsf{halfedge}(f)$ to an arbitrary halfedge within the face. A halfedge h stores *five* pointers:

- a pointer $\mathsf{target}(h)$ to its target vertex,
- a pointer $\mathsf{face}(h)$ to its incident face,
- a pointer $\mathsf{twin}(h)$ to its twin halfedge,
- a pointer $\mathsf{next}(h)$ to the halfedge following h along the boundary of $\mathsf{face}(h)$, and
- a pointer $\mathsf{prev}(h)$ to the halfedge preceding h along the boundary of $\mathsf{face}(h)$.

A constant amount of information is stored for every vertex, (half-)edge, and face of the graph. Therefore the whole DCEL needs storage proportional to $|V| + |E| + |F|$, which is $O(n)$ for a plane graph with n vertices by Corollary 2.5.

This information is sufficient for most tasks. For example, traversing all edges around a face f can be done as follows:

$s \leftarrow \mathsf{halfedge}(f)$
$h \leftarrow s$
do

    something with h
    $h \leftarrow \mathsf{next}(h)$
while $h \neq s$

Exercise 2.15. *Give pseudocode to traverse all edges incident to a given vertex v of a DCEL.*

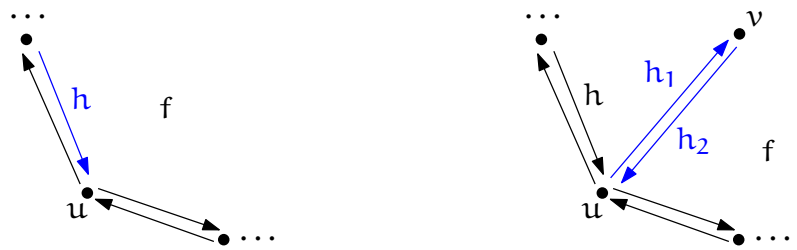
Exercise 2.16. *Why is the previous halfedge* $\mathsf{prev}(\cdot)$ *stored explicitly whereas the source vertex of a halfedge is not?*

2.2.2 Manipulating a DCEL

In many applications, plane graphs do not just appear as static objects but rather evolve over the course of an algorithm. Therefore the data structure must allow for efficient updates. These include, but are not limited to, appending new vertices, edges and faces to the corresponding list within the DCEL and—symmetrically—the ability to delete an existing entity.

First, it should be easy to add a new vertex v to the graph within a given face f and (as we maintain a connected graph) connect v to an existing vertex u. For such a connection to be valid, we require that the open line segment $\overline{uv}$ lies completely in f. Given that we need access to both f and u, it would be convenient to pass the already existing halfedge h that satisfies $\mathsf{face}(h) = f$ and $\mathsf{target}(h) = u$ as an argument. Assuming that $\mathsf{point}(v)$ has already been set to the desired location of the new vertex, our operation then becomes

$\mathsf{add\text{-}vertex\text{-}at}(v, h)$
Precondition: the open line segment $\overline{\mathsf{point}(v)\mathsf{point}(u)}$, where $u := \mathsf{target}(h)$, lies completely in $f := \mathsf{face}(h)$.
Postcondition: the new vertex v has been inserted into f, connected by an edge to u.



(a) before **(b)** after

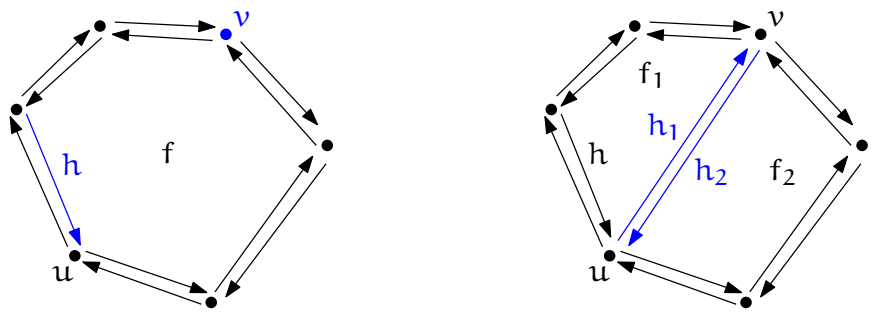
Figure 2.8: *Add a new vertex connected to an existing vertex u.*

See also Figure 2.8. It can be realized by manipulating a constant number of pointers as follows.

add-vertex-at(v, h) {
  $h_1 \leftarrow$ a new halfedge
  $h_2 \leftarrow$ a new halfedge
  halfedge$(v) \leftarrow h_2$
  twin$(h_1) \leftarrow h_2$
  twin$(h_2) \leftarrow h_1$
  target$(h_1) \leftarrow v$
  target$(h_2) \leftarrow u$
  face$(h_1) \leftarrow f$
  face$(h_2) \leftarrow f$
  next$(h_1) \leftarrow h_2$
  next$(h_2) \leftarrow$ next(h)
  prev$(h_1) \leftarrow h$
  prev$(h_2) \leftarrow h_1$
  next$(h) \leftarrow h_1$
  prev(next$(h_2)) \leftarrow h_2$
}

Similarly, it should be possible to add an edge between two existing vertices u and v, provided the open line segment $\overline{uv}$ lies completely within a face f of the graph, see Figure 2.9. Since such an edge insertion splits f into two faces, the operation is called *split-face*. Again we pass as an argument the halfedge h satisfying face$(h) = f$ and target$(h) = u$.

split-face(h, v)
Precondition: v is incident to $f :=$ face(h) but not adjacent to $u :=$ target(h).
  The open line segment $\overline{\text{point}(v)\text{point}(u)}$ lies completely in f.
Postcondition: f has been split by a new edge uv.



(a) before **(b)** after

Figure 2.9: *Split a face by an edge* uv.

The implementation is slightly more complicated compared to add-vertex-at above, because the face f is destroyed and so we have to update the face information of all incident

halfedges. In particular, this is not a constant time operation and has complexity proportional to the size of f.

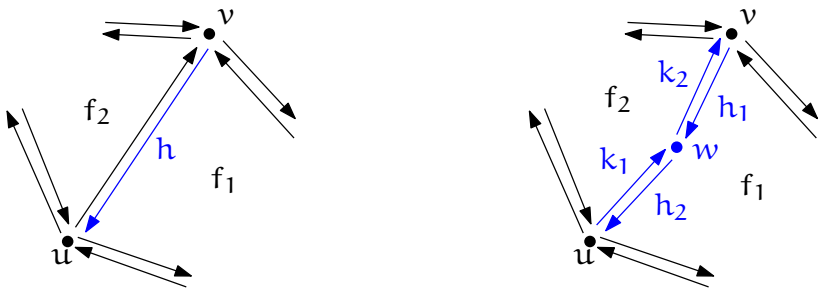
```
split-face(h, v) {
    f1 ← a new face
    f2 ← a new face
    h1 ← a new halfedge
    h2 ← a new halfedge
    halfedge(f1) ← h1
    halfedge(f2) ← h2
    twin(h1) ← h2
    twin(h2) ← h1
    target(h1) ← v
    target(h2) ← u
    next(h2) ← next(h)
    prev(next(h2)) ← h2
    prev(h1) ← h
    next(h) ← h1
    i ← h2
    loop
        face(i) ← f2
        if target(i) = v break the loop
        i ← next(i)
    endloop
    next(h1) ← next(i)
    prev(next(h1)) ← h1
    next(i) ← h2
    prev(h2) ← i
    i ← h1
    do
        face(i) ← f1
        i ← next(i)
    until target(i) = u
    delete the face f
}
```

In a similar fashion one can realize the inverse operation join-face(h) that removes the edge represented by h, thereby joining the faces face(h) and face$(\text{twin}(h))$.

It is easy to see that every connected plane graph on at least two vertices can be constructed using the operations add-vertex-at and split-face, starting from an embedding of K_2 (two vertices connected by an edge).

Exercise 2.17. *Give pseudocode for the operation join-face(h). Specify preconditions if needed.*

Exercise 2.18. *Give pseudocode for the operation split-edge*(h)*, that splits the edge represented by* h *into two by a new vertex* w*, see Figure 2.10.*



(a) before **(b)** after

Figure 2.10: *Split an edge by a new vertex.*

2.2.3 Graphs with Unbounded Edges

In some cases it is convenient to consider plane graphs in which some edges are not mapped to a line segment but to an unbounded curve, such as a ray. This setting is not really much different from the one we studied before, except that one special vertex is placed "at infinity". One way to think of it is in terms of *stereographic projection* (see the proof of Theorem 2.2). The further away a point in $\mathbb{R}^2$ is from the origin, the closer its image on the sphere S gets to the north pole n of S. But there is no way to reach n except in the limit. Therefore, we can imagine drawing the graph on S instead of in $\mathbb{R}^2$ and putting the "infinite vertex" at n.

All this is just for the sake of a proper geometric interpretation. As far as a DCEL of such a graph is concerned, there is no need to consider spheres or anything beyond what we have discussed. The only difference to the case with all finite edges is that there is this special infinite vertex, which does not have any point/coordinates associated to it. Other than that, the infinite vertex is treated in exactly the same way as the finite vertices: it has in- and out-going halfedges along which the unbounded faces can be traversed (Figure 2.11).

Remarks. It is actually not so easy to point exactly to where the DCEL data structure originates from. Often Muller and Preparata [25] are credited, but while they use the term DCEL, the data structure they describe is different from what we discussed above and from what people usually consider a DCEL nowadays. Overall, there are a large number of variants of this data structure, which appear under the names *winged edge* data structure [3], *halfedge* data structure [35], or *quad-edge* data structure [16]. Kettner [21] provides a comparison of all these with some additional references.

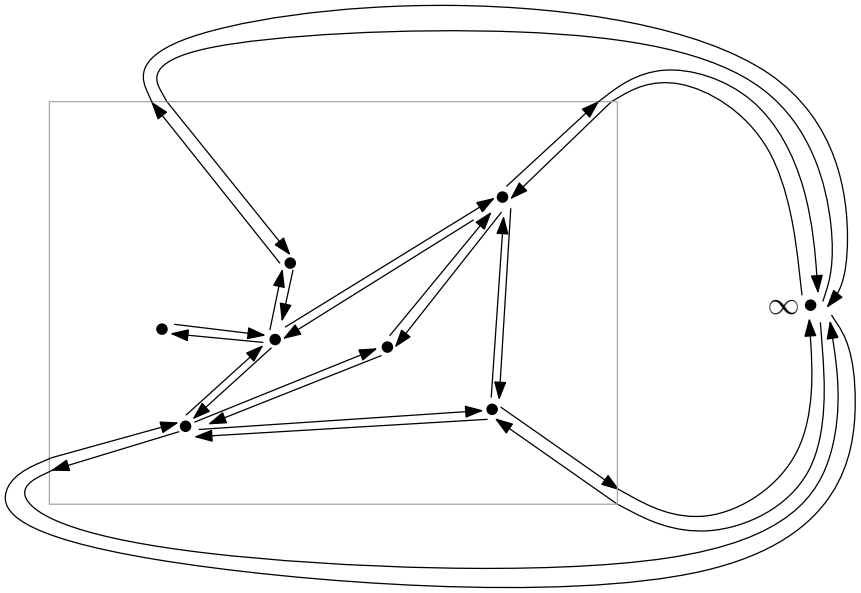


Figure 2.11: *A DCEL with unbounded edges. Usually, we will not show the infinite vertex and draw all edges as straight-line segments. This yields a geometric drawing, like the one within the gray box.*

2.2.4 Combinatorial Embeddings

The basic DCEL omits geometric aspects, such as coordinates of a vertex or the shape of an edge, and only stores incidences and adjacencies between vertices, edges, and faces of an embedding. We call such information the *combinatorial embedding* of the actual plane graph. Conventionally, we write it as a set of face boundaries, where each boundary is encoded as a circular sequence of vertices in counterclockwise order. For instance, the combinatorial embeddings of the plane graphs in Figure 2.12a are

$$\begin{aligned}
&(a) \;:\; \{(1,2,3),(1,3,6,4,5,4),(1,4,6,3,2)\},\\
&(b) \;:\; \{(1,2,3,6,4,5,4),(1,3,2),(1,4,6,3)\}, \text{and}\\
&(c) \;:\; \{(1,4,5,4,6,3),(1,3,2),(1,2,3,6,4)\}.
\end{aligned}$$

Note that a vertex can appear several times along the boundary of a face (if it is a cut-vertex).

This view allows us to compare embeddings easily. Two embeddings (plane graphs) are *combinatorially equivalent* if their combinatorial embeddings are equal up to a global change of orientation (reversing the order of all sequences simultaneously). For example, (b) is not equivalent to (a) nor (c), because it is the only one with a face bounded by seven vertices. However, (a) and (c) turn out to be equivalent: after reverting orientations f_1 takes the role of h_2, f_2 takes the role of h_1, and f_3 takes the role of h_3.

Exercise 2.19. *Let* G *be a planar graph with vertex set* $\{1,\dots,9\}$*. Try to find an embedding corresponding to the following list of circular sequences of faces:*

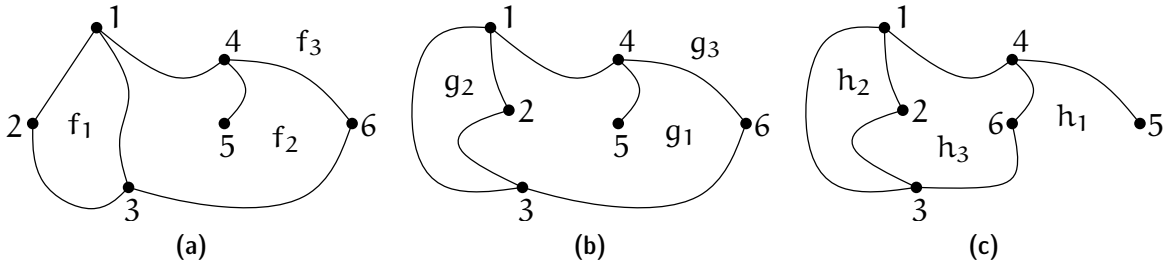


Figure 2.12: *Equivalent embeddings?*

(a) $\{(1,4,5,6,3),(1,3,6,2),(1,2,6,7,8,9,7,6,5),(7,9,8),(1,5,4)\}$
(b) $\{(1,4,5,6,3),(1,3,6,2),(1,2,6,7,8,9,7,6,5),(7,9,8),(1,4,5)\}$

Combinatorial embeddings are not only used to categorize plane graphs. They also play a role in algorithm design. Quite often, algorithms dealing with planar graphs do not need a full-fledged embedding to proceed. It is sufficient to operate on a combinatorial embedding, which is more efficient to handle.

Many people prefer a dual representation which, instead of listing face boundaries, enumerates the neighbors of v in cyclic order for each vertex v. It can avoid the issue of a vertex appearing multiple times in the sequence. However, the following lemma shows that such an issue does not arise when dealing with biconnected graphs.

Lemma 2.20. *In a biconnected plane graph every face is bounded by a cycle.*

We leave the proof as an exercise. Intuitively the statement is clear, but we believe it is instructive to think about a formal argument. An easy consequence is stated below, whose proof is also an exercise.

Corollary 2.21. *For any vertex v in a 3-connected plane graph, there is a cycle that contains all neighbors of v.*

Exercise 2.22. *Prove Lemma 2.20 and Corollary 2.21.*

Given Lemma 2.20, one might wonder the converse question: Which cycles in a planar graph G bound a face (in some plane embedding of G)? Such cycles are said to be *facial*; see Figure 2.13.

Exercise 2.23. *Describe a linear time algorithm that, given an abstract planar graph G and a cycle C in G, tests whether C is a facial cycle. (You may assume that planarity can be tested in linear time.)*

Exercise 2.24. *Let $G = (V, E)$ be a biconnected graph and let $\mathcal{C}$ be a collection of directed cycles of G. Assume that for every directed edge (u, v) whose underlying undirected edge $\{u, v\}$ is in E, there is exactly one cycle $C \in \mathcal{C}$ that contains (u, v). Further assume that for every vertex $v \in V$, the $\deg(v)$ cycles in $\mathcal{C}$ that contain v*

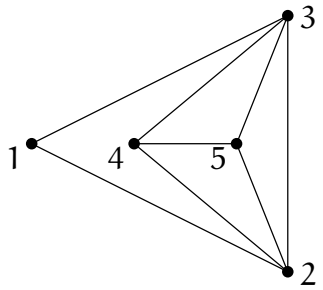


Figure 2.13: *The cycles* $(2,3,5)$ *and* $(1,2,5,3)$*, for example, are both facial. One can show that* $(2,4,3,5)$ *is not.*

can be ordered as $C_0, \dots, C_{\deg(v)-1}$ *such that for every pair* $u, v \in V$ *and* $i \in \mathbb{N}$ *with* $(u,v),(v,w) \in C_i$ *there exists an* $x \in V$ *such that* $(w,v),(v,x) \in C_{i+1}$ *(where indices are taken modulo* $\deg(v)$*). Prove or disprove: The collection* $\mathcal{C}$ *describes a combinatorial plane embedding of* G*.*

2.3 Unique Embeddings

As we have seen, an abstract planar graph may admit many different embeddings, even in the combinatorial sense. Under what condition does it admit a unique combinatorial embedding?

To answer the question, we start by studying cycles that bound a face in *every* plane embedding of G. (Note that this is stronger than being facial.) The lemma below provides a complete characterization of these cycles. Let us agree on some terminology about a cycle C in a graph G. A *chord* of C is an edge in $E(G) \setminus E(C)$ that connects two vertices of C. The cycle C is *induced* if it does not have any chord. It is *separating* if $G \setminus C$ is not connected.

Lemma 2.25. *Let* G *be a planar graph which is neither a cycle, nor a cycle plus a single chord. Then a cycle* C *in* G *bounds a face in every plane embedding of* G *if and only if* C *is induced and not separating.*

Proof. "$\Leftarrow$": Consider any plane embedding Γ of G. By the Jordan Curve Theorem, the cycle C splits the plane into an interior and an exterior region. As $G \setminus C$ is connected, it lies either entirely in the interior or entirely in the exterior. In either case, the other region is bounded by C because C does not have any chord.

"$\Rightarrow$": Using contraposition, suppose that (1) C is not induced or (2) C is separating. We aim to find a plane embedding of G in which C does not bound a face. To this end, let us start from an arbitrary plane embedding Γ of G. If C does not bound a face in Γ then we are done. So next we assume that C bounds a face in Γ.

(1) If C is not induced, then it has a chord c. As $G \neq C \cup c$, the graph G either has some vertex $v \notin C$ or another chord $d \neq c$ of C. We modify Γ by rerouting the

chord c inside the face C and obtain an embedding in which C does not bound a face: one of the two regions split by the Jordan curve C contains the chord c, and the other contains either the vertex v or another chord d.

(2) If C is separating, then $G \setminus C$ is not connected. If $G \setminus C = \emptyset$ then G is either C (which is excluded by assumption) or C plus some chords (which is handled by Case (1)). So from now on we assume $G \setminus C \neq \emptyset$ has two components A and B; see Figure 2.14a. Γ induces plane embeddings Γ_A of $A \cup C$ and Γ_B of $B \cup C$; the cycle C bounds a face in both of them. By the transformation in Theorem 2.2 we can make C bounding the outer face in Γ_A yet an inner face in Γ_B. Then we can glue the two embeddings at C, that is, extend Γ_B by adding Γ_A within the (inner) face bounded by C (Figure 2.14b). The result is a plane embedding of G in which C does not bound a face.

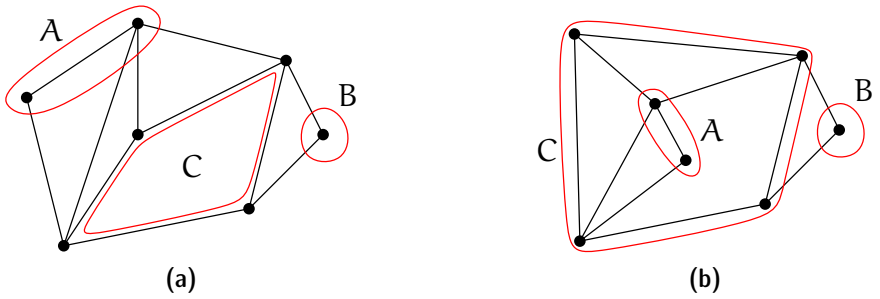


Figure 2.14: *A plane embedding in which C does not bound a face, in Case (2).*

□

For those special graphs G excluded in Lemma 2.25, it is easy to see that all cycles in G bound a face in every plane embedding. This completes the characterization. Since these special graphs are not 3-connected, we have

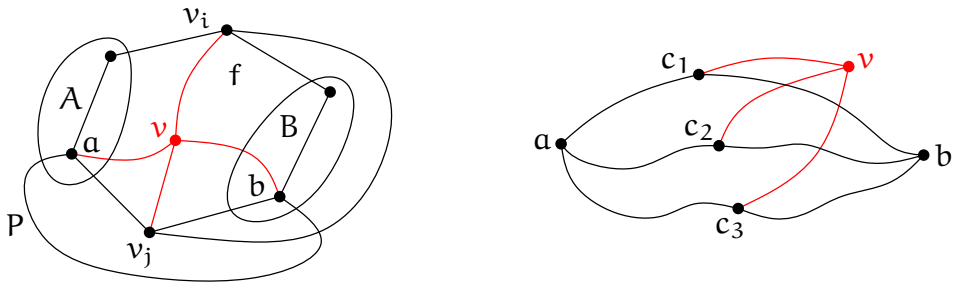
Corollary 2.26. *A cycle C of a 3-connected planar graph G bounds a face in every plane embedding of G if and only if C is induced and not separating.* □

The following theorem tells us that a wide range of graphs have little choice when embedded into the plane, from a combinatorial point of view. Geometrically, though, there is still much freedom.

Theorem 2.27 (Whitney [36]). *A 3-connected planar graph has a unique combinatorial plane embedding (up to equivalence).*

Proof. Let G be a 3-connected planar graph and suppose there exist two embeddings Φ_1 and Φ_2 of G that are not equivalent. So there is a cycle $C = (v_1, \dots, v_k)$ in G that, say, bounds a face f in Φ_1 but does not bound any face in Φ_2. By Corollary 2.26 there are only two options:

Case 1: C has a chord $\{v_i, v_j\}$. Denote $A = \{v_x : i < x < j\}$ and $B = \{v_x : x < i \vee j < x\}$ and observe that both A and B are nonempty because $\{v_i, v_j\}$ is a chord and so v_i and v_j are not adjacent in C. Given that G is 3-connected, there is at least one path P from A to B that avoids both v_i and v_j. Let a denote the last vertex of P that is in A, and let b denote the first vertex of P that is in B. As C bounds f in Φ_1, we can add a new vertex v inside f and connect it to each of v_i, v_j, a and b by four pairwise internally disjoint curves. The result would be a plane graph that contains a K_5 subdivision with branch vertices v, v_i, v_j, a, and b. This contradicts Kuratowski's Theorem (Theorem 2.10).



(a) Case 1. **(b)** Case 2.

Figure 2.15: *Illustration of the two cases in Theorem 2.27.*

Case 2: C is induced and separating. Since C is induced and G is 3-connected, we must have $G \setminus C \neq \emptyset$. So $G \setminus C$ contains two distinct components A and B. Choose vertices $a \in A$ and $b \in B$ arbitrarily. Applying Menger's Theorem (Theorem 1.5) on the 3-connected graph G, there exist three paths $\alpha_1, \alpha_2, \alpha_3$, pairwise internally vertex-disjoint, from a to b. Let c_i be some vertex where α_i intersects C, for $1 \leqslant i \leqslant 3$. Note that c_1, c_2, c_3 exist because C separates A and B, and they are pairwise distinct because $\alpha_1, \alpha_2, \alpha_3$ are pairwise internally (vertex-)disjoint. Therefore, $\{a, b\}$ and $\{c_1, c_2, c_3\}$ form branch vertices of a $K_{2,3}$ subdivision in G. We can add a new vertex v inside f and connect it to each of c_1, c_2 and c_3 by three pairwise internally disjoint curves. The result would be a plane graph that contains a $K_{3,3}$ subdivision. This contradicts Kuratowski's Theorem (Theorem 2.10).

In both cases we arrived at a contradiction and so there does not exist such a cycle C. Thus Φ_1 and Φ_2 are equivalent. □

Whitney's Theorem does not provide a characterization of unique embeddability in general, as there are biconnected graphs with unique combinatorial plane embedding (such as cycles) as well as those with several, non-equivalent combinatorial plane embeddings (such as a triangulated pentagon).

Exercise 2.28. *Describe a family of biconnected planar graphs with exponentially many combinatorial plane embeddings. That is, show that there exists a constant* $c \in \mathbb{R}$

such that for every $n \in \mathbb{N}$ *there exists a biconnected planar graph on* n *vertices that has at least* c^n *different combinatorial plane embeddings.*

2.4 Triangulating a Planar Graph

We like to study worst case scenarios not so much to dwell on "how bad things could get" but rather—phrased positively—because worst case examples provide universal bounds of the form "things are always at least this good". Most questions related to embeddings get harder when the graph contains more edges because every additional edge poses an increasing danger of crossing. So let us study the worst case: planar graphs such that adding any edge shall break its planarity. These graphs are called *maximal planar*. Corollary 2.5 tells us that every (hence also maximal) planar graph on n vertices has at most $3n - 6$ edges. Yet we would like to learn a bit more about how these graphs look like.

Lemma 2.29. *A maximal planar graph on* $n \geqslant 3$ *vertices is biconnected.*

Proof. Consider a maximal planar graph $G = (V, E)$. Note that G is connected because adding an edge between two distinct components of a planar graph maintains planarity. Now if G is not biconnected, then it has a cut-vertex v. Take a plane drawing Γ of G. As $G \setminus v$ is disconnected, removal of v also splits $N_G(v)$ into at least two components. Hence there are two vertices $a, b \in N_G(v)$, consecutive in the circular order around v in Γ, that are in different components of $G \setminus v$. In particular, $ab \notin E$ and we can add this edge to G (routing it very close to the path (a, v, b) in Γ) without violating planarity. This is in contradiction to G being maximal planar, so G must be biconnected. □

Lemma 2.30. *In any embedding of a maximal planar graph on* $n \geqslant 3$ *vertices, all faces are topological triangles, that is, every face is bounded by exactly three edges.*

Proof. Consider a maximal planar graph $G = (V, E)$ and a plane drawing Γ of G. By Lemma 2.29 we know that G is biconnected and so by Lemma 2.20 every face of Γ is bounded by a cycle. Suppose that there is a face f in Γ bounded by a cycle $(v_0, \ldots, v_{k-1}, v_k = v_0)$ of $k \geqslant 4$ vertices. We claim that at least one of the edges v_0v_2 or v_1v_3 is not in E.

Suppose to the contrary that $\{v_0v_2, v_1v_3\} \subseteq E$. Then we can add a new vertex v' in the interior of f and connect it to each of v_0, v_1, v_2, v_3 by a curve inside f without introducing a crossing. In other words, given G is planar, the graph $G' = (V \cup \{v'\}, E \cup \{v'v_i : i \in \{0, 1, 2, 3\}\})$ is also planar. However, v_0, v_1, v_2, v_3, v' are branch vertices of a K_5 subdivision in G': v' is connected to all other vertices within f, each vertex v_i is connected to both $v_{(i-1) \bmod 4}$ and $v_{(i+1) \bmod 4}$ along the boundary of f, and the two missing connections are provided by the edges v_0v_2 and v_1v_3 (Figure 2.16a). This contradicts Kuratowski's Theorem. Therefore, one of the edges v_0v_2 or v_1v_3 must be absent from E, as claimed.

So assume without loss of generality that $v_1v_3 \notin E$. But then we can route a curve from v_1 to v_3 inside f in Γ without introducing a crossing (Figure 2.16b). It follows that

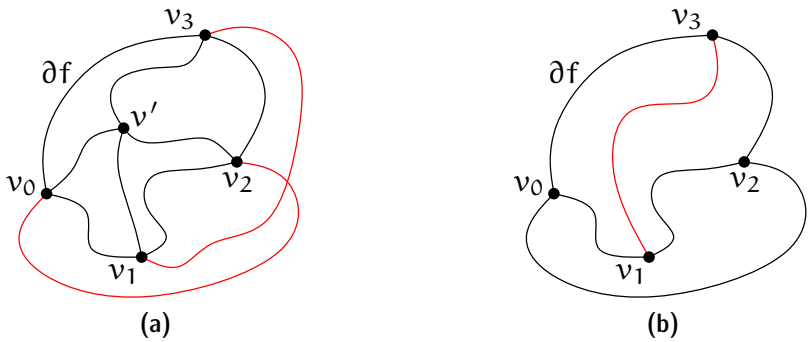


Figure 2.16: *Every face of a maximal planar graph is a topological triangle.*

the edge v_1v_3 can be added to G without sacrificing planarity, which is in contradiction to G being maximal planar. Therefore, there is no such face f bounded by four or more vertices. □

Theorem 2.31. *A maximal planar graph on $n \geqslant 4$ vertices is 3-connected.*

Exercise 2.32. *Prove Theorem 2.31.*

Exercise 2.33. *(a) A* minimal nonplanar *graph is a non-planar graph G which contains an edge e such that $G \setminus e$ is planar. Prove or disprove: Every minimal nonplanar graph contain an edge e such that $G \setminus e$ is maximal planar.*

(b) A maximal-plus-one planar *graph is a graph G that contains an edge e such that $G \setminus e$ is maximal planar. Prove or disprove: Every maximal-plus-one planar graph can be drawn with at most one crossing.*

Many questions about graphs are formulated only for connected graphs because it is easy to add edges to disconnected graphs and make them connected. For similar reason, many questions about planar embeddings are formulated only for maximal planar graphs because it is easy to augment planar graphs and make them maximal planar. Well, this last statement is not entirely obvious. Let us look at it in more detail.

An augmentation of a given planar graph $G = (V, E)$ to a maximal planar graph $G' = (V, E')$ where $E' \supseteq E$ is also called a *topological triangulation*. The proof of Lemma 2.30 already contains the basic algorithmic idea to topologically triangulate a plane graph.

Theorem 2.34. *For a given connected plane graph $G = (V, E)$ on n vertices one can compute in $O(n)$ time and space a maximal plane graph $G' = (V, E')$ with $E \subseteq E'$.*

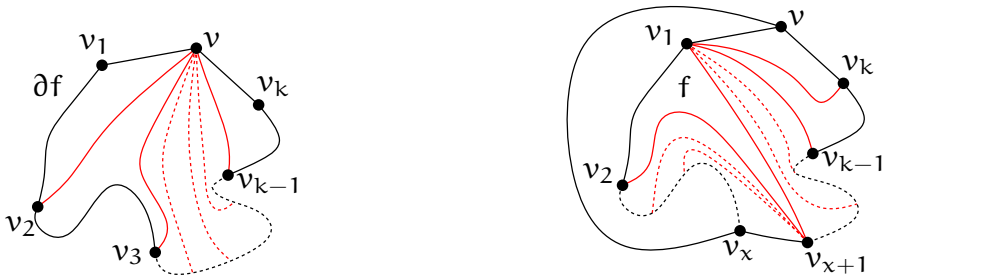
Proof. Suppose, for instance, that G is represented as a DCEL[2], from which one can easily extract the face boundaries. As a clean-up, we walk along the boundary of each

[2]If you wonder how the possibly complicated curves are represented: they do not need to be, since here we need a representation of the combinatorial embedding only.

face. Whenever we see a vertex twice (or more), it must be a cut vertex. We fix this by adding an edge between its current predecessor and successor along the walk, and then continue the walk. Since the total number of traversed edges and vertices of all faces is proportional to $|E|$, which by Corollary 2.5 is linear, the clean-up finishes in $O(n)$ time. Henceforth we may suppose that all faces of G are bounded by cycles.

Every face that is bounded by more than three vertices selects an arbitrary vertex on its boundary. Conversely, every vertex keeps a list of all faces that have selected it. Then we process every vertex $v \in V$ as follows:

1. Mark all neighbors of v.
2. For each face f that selected v, scan its boundary $\partial f = (v, v_1, \ldots, v_k)$ counterclockwise, where $k \geqslant 3$, and find the first marked vertex $v_x \notin \{v_1, v_k\}$.
   - If there is no such vertex, we can safely triangulate f using a star from v, that is, by adding the edges vv_i, for $i \in \{2, \ldots, k-1\}$ (Figure 2.17a). We then mark the new neighbors of v accordingly.
   - Otherwise, the edge vv_x as a curve embedded outside f prevents any vertex in $\{v_1, \ldots, v_{x-1}\}$ from connecting to any vertex in $\{v_{x+1}, \ldots, v_k\}$ by an edge in G. (The reasoning copies the one we made for the edges v_0v_2 and v_1v_3 in the proof of Lemma 2.30 above; see Figure 2.16a.) So we can safely triangulate f using a bi-star from v_1 and v_{x+1}, that is, by adding the edges v_1v_i, for $i \in \{x+1, \ldots, k\}$, and v_jv_{x+1}, for $j \in \{2, \ldots, x-1\}$ (Figure 2.17b).
3. After finishing all faces that seleted v, we conclude the processing of v by clearing all marks on its neighbors.



(a) Case 1: v does not have any neighbor on ∂f other than v_1 and v_k.

(b) Case 2: v has a neighbor v_x on ∂f other than v_1 and v_k.

Figure 2.17: *Topologically triangulating a plane graph.*

Regarding the runtime bound, note that every face is visited only twice: one time when selecting its representative vertex, the other time when scanning its boundary. In this way, each edge is touched a constant number of times in step 2 overall. The marking/unmarking (steps 1 and 3) cost $\sum_{v \in V} \deg(v) = 2|E|$ time by the Handshaking

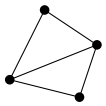
Lemma. Therefore, the total time can be bounded by $O(n + |F| + |E|) = O(n)$ by Corollary 2.5. □

Using any of the standard planarity testing algorithms we can obtain a combinatorial embedding of a planar graph in linear time. Together with Theorem 2.34 this yields:

Corollary 2.35. *For a given planar graph $G = (V, E)$ on n vertices one can compute in $O(n)$ time and space a maximal planar graph $G' = (V, E')$ with $E \subseteq E'$.* □

The results discussed in this section can serve as a tool to fix the combinatorial embedding for a given graph G: augment G using Theorem 2.34 to a maximal planar graph G', whose combinatorial embedding is unique by Theorem 2.27.

Being maximal planar is a property of an abstract graph. In contrast, a geometric graph to which no straight-line edge can be added without crossing is called a *triangulation*. Not every triangulation is maximal planar, as the example depicted to the right shows.



It is also possible to triangulate a geometric graph in linear time. But this problem is much more involved. Triangulating a single face of a geometric graph amounts to what is called "triangulating a simple polygon". This can be done in near-linear[3] time using standard techniques, and in linear time using Chazelle's famous algorithm, whose description spans a fourty pages paper [9].

Exercise 2.36. *We discussed the DCEL structure to represent plane graphs in Section 2.2.1. An alternative way to represent an embedding of a maximal planar graph is the following: For each triangle, store pointers to its three vertices and to its three neighboring triangles. Compare both approaches. Discuss different scenarios where you would prefer one over the other. In particular, analyze the space requirements of both.*

Connectivity serves as an important indicator for properties of planar graphs. Already Wagner showed that a 4-connected graph is planar if and only if it does not contain K_5 as a minor. That is, assuming 4-connectivity the second forbidden minor $K_{3,3}$ becomes "irrelevant". For subdivisions this is a different story. Independently Kelmans and Semour conjectured in the 1970s that 5-connectivity allows to consider K_5 subdivisions only. This conjecture was proven only recently[4] by Dawei He, Yan Wang, and Xingxing Yu.

Theorem 2.37 (He, Wang, and Yu [18]). *Every 5-connected nonplanar graph contains a subdivision of K_5.*

Exercise 2.38. *Give a 4-connected nonplanar graph that does not contain a subdivision of K_5.*

[3] $O(n \log n)$ or—using more elaborate tools—$O(n \log^* n)$ time.
[4] The result was announced in 2015 and published in 2020.

Another example that illustrates the importance of connectivity is the following famous theorem of Tutte that provides a sufficient condition for Hamiltonicity.

Theorem 2.39 (Tutte [32]). *Every 4-connected planar graph is Hamiltonian.*

Moreover, for a given 4-connected planar graph a Hamiltonian cycle can also be computed in linear time [10].

2.5 Compact Straight-Line Drawings

As a next step we consider geometric plane embeddings, where every edge is drawn as a straight-line segment. A classical theorem of Wagner and Fáry states that this is not a restriction to plane embeddability.

Theorem 2.40 (Fáry [13], Wagner [33]). *Every planar graph has a plane straight-line embedding.*

This is quite surprising, considering how much more freedom a simple curve allows, compared to a line segment which is completely determined by its endpoints. To further increase the level of appreciation, let us remark that a similar "straightening" is generally not possible if we fix the point set on which the vertices are to be embedded: On the one hand, Pach and Wenger [27] showed that a given planar graph G on n vertices $v_1, \dots, v_n$ and a given point set $\{p_1, \dots, p_n\} \subset \mathbb{R}^2$, one can always find a plane embedding of G such that $v_i \mapsto p_i$, for all $i \in \{1, \dots, n\}$. On the other hand, this is not possible in general with a plane *straight-line* embedding. For instance, K_4 does not admit a plane straight-line embedding on a set of points that form a convex quadrilateral, such as a rectangle. In fact, it is NP-hard to decide whether a given planar graph admits a plane straight-line embedding on a given point set [7].

Exercise 2.41. *Show the following:*

(a) For every natural number $n \geqslant 4$, there exist a planar graph G on n vertices and a set $P \subset \mathbb{R}^2$ of n points in general position (no three points are collinear) so that G does not admit a plane straight-line embedding on P.

(b) For every natural number $n \geqslant 6$, there exist a planar graph G on n vertices and a set $P \subset \mathbb{R}^2$ of n points in general position (no three points are collinear) so that (1) G does not admit a plane straight-line embedding on P; and (2) there are three points in P forming a triangle that contains all other points from P.

Exercise 2.42. *Show that for every set $P \subset \mathbb{R}^2$ of $n \geqslant 3$ in general position (no three points are collinear) the cycle on n vertices admits a plane straight-line embedding on P.*

Although Fáry-Wagner's theorem has a nice inductive proof, we do not discuss it here. Instead we will soon prove a stronger statement that implies the theorem.

A very desirable property of straight-line embeddings is that they are easy to represent: only the points/coordinates for the vertices are needed. But from an algorithmic and complexity point of view it is also important to learn the space requirement for the coordinates, since it affects the input and output size of algorithms that work on embedded graphs. While the Fáry-Wagner Theorem guarantees the existence of a plane straight-line embedding for every planar graph, it does not bound the size of the coordinates. The following strengthening provides such bounds, via an explicit algorithm that embeds (without crossing) a given planar graph on a linear size integer grid.

Theorem 2.43 (de Fraysseix, Pach, Pollack [15]). *Every planar graph on $n \geqslant 3$ vertices has a plane straight-line drawing on a $(2n-3) \times (n-1)$ integer grid. In fact, it can be computed in $O(n)$ time.*

2.5.1 Canonical Orderings

The key concept behind the algorithm is the notion of a canonical ordering, which is a vertex order that allows building the plane drawing inside out (hence canonical). Reading it backwards one may imagine a shelling or peeling order that destructs the graph from the outside. A canonical ordering also provides a succinct representation for the combinatorial embedding.

Definition 2.44. *A plane graph G is* **internally triangulated** *if it is biconnected and every bounded face is a (topological) triangle. We denote by $C_\circ(G)$ its outer cycle, that is, the cycle bounding its outer face.*

Definition 2.45. *Let G be an internally triangulated plane graph. A permutation $\pi = (v_1, v_2, \ldots, v_n)$ of $V(G)$ is a* **canonical ordering** *for G if for all $k \in \{3, \ldots, n\}$ we have*

(CO1) G_k is internally triangulated;

(CO2) $v_1v_2 \in C_\circ(G_k)$; and

(CO3) v_k is located in the outer face of G_{k-1},

where $G_k := G[\{v_1, \ldots, v_k\}]$ denotes the induced drawing on the first k vertices.

Figure 2.18 shows an example with canonical ordering $(1, 2, \ldots, 8)$. Note that not every permutation is a valid canonical ordering. For instance, if π chooses its first seven vertices from $\{1, 2, 3, 5, 6, 7, 8\}$, then the induced subgraph $G[\{1, 2, 3, 5, 6, 7, 8\}]$ is not biconnected since 1 is a cut vertex. Thus π is not a canonical ordering. Alternatively we may think about it backwards: Suppose we choose the initial three removals from $\{9, 10, 11\}$ as shown in Figure 2.18b, then the next removal cannot be 4 because its removal would create a cut vertex in the resulting graph.

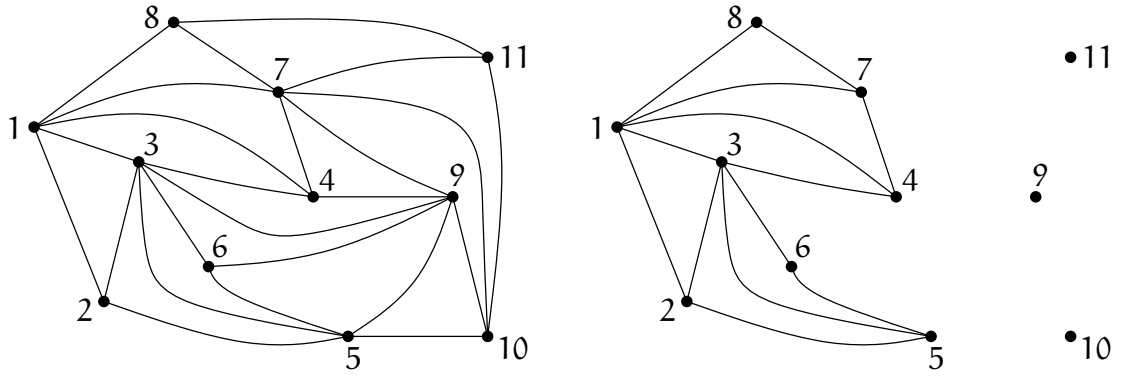


Figure 2.18: *An internally triangulated plane graph with one of its canonical orderings* $(1, 2, \ldots, 8)$.

Theorem 2.46. *For every internally triangulated plane graph* G *and every edge* v_1v_2 *on its outer cycle, there exists a canonical ordering for* G *that starts with* v_1, v_2*. Moreover, such an ordering can be computed in linear time.*

Proof. Induction on n, the number of vertices. For a triangle, any ordering is valid and so the statement holds. Now consider an internally triangulated plane graph $G = (V, E)$ on $n \geqslant 4$ vertices. *Assume* we find a vertex $v_n \in C_\circ(G) \setminus \{v_1, v_2\}$ such that the plane graph $G_{n-1} := G \setminus \{v_n\}$ is internally triangulated. (We will show later that such a vertex always exists.) Then we can apply induction on G_{n-1} to obtain a canonical ordering $(v_1, v_2, \ldots, v_{n-1})$ for G_{n-1}. The extended ordering $(v_1, v_2, \ldots, v_n)$ satisfies (CO1)–(CO3) for $k \in \{3, \ldots, n-1\}$ by the induction hypothesis, but also for $k = n$ by definition of v_n. Hence the induction is complete, assuming the existence of v_n.

It remains to argue that v_n exists. We will show this in two steps:

(1) we can find a $v_n \in C_\circ(G) \setminus \{v_1, v_2\}$ that is not incident to a chord of $C_\circ(G)$; and

(2) such a vertex v_n guarantees that $G_{n-1} := G \setminus \{v_n\}$ is internally triangulated.

First we show (1). If $C_\circ(G)$ does not have any chord, this is obvious because every cycle has at least three vertices, one of which is neither v_1 nor v_2. So suppose that $C_\circ(G)$ has a chord c. The endpoints of c split $C_\circ(G)$ into two paths, one of which does not have v_1 nor v_2 as an internal vertex. We call this path the path *associated* to c. (Such a path has at least two edges because there is always at least one vertex "behind" a chord.) Among all chords of $C_\circ(G)$ we select c such that its associated path has minimal length. Then by this choice of c its associated path together with c forms an induced cycle in G. In particular, none of the (at least one) interior vertices of the path associated to c is incident to a chord of $C_\circ(G)$ because such a chord would either cross c or it would have an associated path that is strictly shorter than the one associated to c. So we can select v_n from these vertices. By definition the path associated to c does not contain v_1 nor v_2, hence this procedure does not select either of these vertices.

Then we look at (2). The way G_{n-1} is obtained from G, every bounded face f of G_{n-1} also appears as a bounded face of G. As G is internally triangulated, f is a triangle. It remains to show that G_{n-1} is biconnected.

Consider the circular sequence of neighbors around v_n in G and break it into a linear sequence $u_1, \dots, u_m$, for some $m \geqslant 2$, that starts and ends with the neighbors of v_n in $C_\circ(G)$. As G is internally triangulated, each of the bounded faces spanned by v_n, u_i, u_{i+1}, for $i \in \{1, \dots, m-1\}$, is a triangle and hence $u_i u_{i+1} \in E$. The boundary of the outer face of G_{n-1} is obtained from $C_\circ(G)$ by replacing v_n with the (possibly empty) sequence $u_2, \dots, u_{m-1}$. As v_n is not incident to a chord of $C_\circ(G)$ (and so none of $u_2, \dots, u_{m-1}$ appeared along $C_\circ(G)$ already), the resulting sequence forms a cycle, indeed. Add a new vertex v in the outer face of G_{n-1} and connect v to every vertex of $C_\circ(G_{n-1})$ to obtain a maximal planar graph $H \supset G_{n-1}$. By Theorem 2.31 the graph H is 3-connected and so G_{n-1} is biconnected, as desired. This also completes the proof of the claim.

Regarding the runtime bound, we maintain for each vertex v whether it is on the current outer cycle and what is the number of incident chords with respect to the current outer cycle. Given a combinatorial embedding of G, it is straighforward to initialize this information in linear time. (Every edge is considered at most twice, once for each endpoint on the outer cycle.) We also maintain an unordered list of the *eligible* vertices, that is, those vertices that are on the outer cycle and not incident to any chord. This list is straightforward to maintain: Whenever a vertex information is updated, check before and after the update whether it is eligible and correspondingly add it to or remove it from the list of eligible vertices. We store with each vertex a pointer to its position in the list (*nil* if it is not eligible currently) so that we can remove it from the list in constant time if needed.

When removing a vertex v_n from G, there are two cases: Either v_n has two neighbors u_1 and u_2 only (Figure 2.19a), in which case the edge u_1u_2 ceases to be a chord. Thus, the chord count for u_1 and u_2 has to be decremented by one. Otherwise, there are $m \geqslant 3$ neighbors $u_1, \dots, n_m$ (Figure 2.19b) and (1) all vertices $u_2, \dots, u_{m-1}$ are new on the outer cycle, and (2) every edge incident to u_i, for $i \in \{2, \dots, m-1\}$, and some other vertex on the outer cycle other than u_{i-1} or u_{i+1} is a new chord. These latter changes have to be reflected in the chord counters at the vertices. So to update these counters, we inspect all edges incident to one of $u_2, \dots, u_{m-1}$. For each such edge, we check whether the other endpoint is on the outer cycle and, if so, increment the counter.

During the course of the algorithm every vertex appears once as a new vertex on the outer cycle. At this point all incident edges (in the current graph G_i) are examined. Similarly, when a vertex v_k is removed from G_K, all edges incident to v_k in G_k are inspected so as to discover if and which new vertices appear on the outer cycle. Finally, each vertex is removed at most once. Therefore, every edge is inspected at most three times: when one of its two endpoints appears first on the outer cycle, and when the first endpoint (and therefore the edge) is removed. Altogether this takes linear time because the number of edges in G is linear by Corollary 2.5. □

Using one of the linear time planarity testing algorithms, we can obtain a combinato-

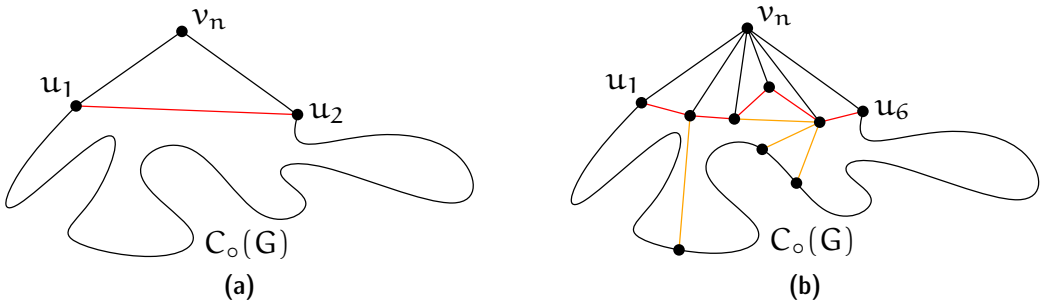
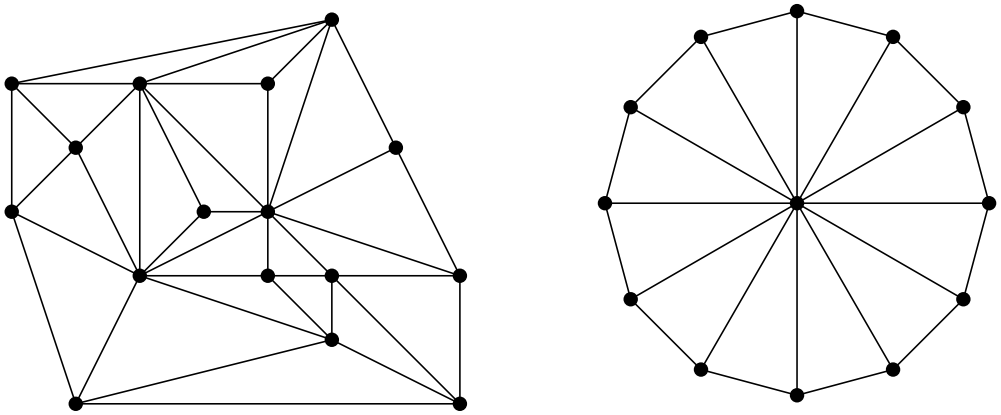


Figure 2.19: *Processing a vertex when computing a canonical ordering.*

rial embedding for a given maximal planar graph G. As every maximal planar graph is 3-connected (Theorem 2.31), this embedding is unique (Theorem 2.27). Then, as every maximal plane graph is also internally triangulated, we can use Theorem 2.46 to provide us with a canonical ordering for (the unique embedding of) G, in overall linear time.

Corollary 2.47. *Every maximal planar graph admits a canonical ordering. Moreover, such an ordering can be computed in linear time.* □

Exercise 2.48. *(a) Compute a canonical ordering for the following internally triangulated plane graphs:*



(b) Design an infinite family of internally triangulated plane graphs on $2k$ *vertices with at least* $k!$ *canonical orderings.*

(c) Design an infinite family of internally triangulated plane graphs, along with specific choices for v_1, v_2*, so that each graph in the family has a unique canonical ordering starting from* v_1, v_2*.*

Exercise 2.49. *(a) Describe a plane graph* G *with* n *vertices that can be embedded (while preserving the outer face) in straight-line on a grid of size* $(2n/3) \times (2n/3)$*, but not on a smaller grid.*

(b) Can you draw G *on a smaller grid if you are allowed to change the outer face?*

As simple as they may appear, canonical orderings are a powerful and versatile tool to work with plane graphs. As an example, consider the following partitioning theorem.

Theorem 2.50 (Schnyder [30]). *For every maximal planar graph* G *on at least three vertices and every fixed face* f *of* G*, the multigraph obtained from* G *by doubling the (three) edges of* f *can be partitioned into three spanning trees.*

Exercise 2.51. *Prove Theorem 2.50.* Hint: *Fix a canonical ordering; for every vertex* v_k *take the edge to its first neighbor on* $C_\circ(G_{k-1})$*; argue that the edges form a spanning tree.*

Of a similar flavor is the following question.

Problem 2.52 (In memoriam Ferran Hurtado (1951–2014)).
Can every complete geometric graph on $n = 2k$ vertices (in general position) be partitioned into k plane spanning trees?

There are several positive results for special point sets [1, 5], and it is also known that there are always $\lfloor n/3 \rfloor$ edge disjoint plane spanning trees [4]. The general statement above has been refuted very recently [26]. However, it remains open if there always exists a partition into $k+1$ plane trees—or more generally, what is the minimum number of plane trees that always suffices.

2.5.2 The Shift-Algorithm

Let $(v_1, \ldots, v_n)$ be a canonical ordering of maximal planar graph G. The plan is to insert vertices in this order and extend the embedding incrementally, starting from the triangle $P(v_1) = (0,0)$, $P(v_3) = (1,1)$, $P(v_2) = (2,0)$; see Figure 2.20.

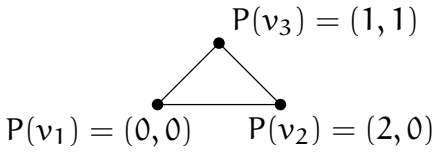


Figure 2.20: *Initialization of the shift algorithm.*

At each step, some vertices are shifted to the right, making room for the insertion of a new vertex. For each vertex v_k inserted, we define a "frozen" set $F(v_k)$ to collect all vertices that move rigidly with v_k in the future. For the first three vertices we define $F(v_i) = \{v_i\}$, for $1 \leqslant i \leqslant 3$. Once defined, the set will not change anymore.

We ensure the following invariants after Step k (that is, after we have inserted v_k):

(i) We obtain a straight-line embedding of $G_k := G[\{v_1, \dots, v_k\}]$ on the integer grid, combinatorially equivalent to the one considered in the canonical ordering. Moreover, $P(v_1) = (0, 0)$ and $P(v_2) = (2k-4, 0)$.

(ii) Denote the outer cycle by $C_\circ(G_k) =: (w_1, \dots, w_t)$ where $w_1 = v_1$ and $w_t = v_2$. The x-coordinates of $w_1, \dots, w_t$ are strictly increasing.[5]

(iii) Each edge $w_i w_{i+1}$, for $1 \leqslant i < t$, of $C_\circ(G_k)$ is drawn as a line segment with slope ± 1. In particular, the Manhattan distance[6] between any two points on $C_\circ(G_k)$ is even.

(iv) The sets $F(w_1), \dots, F(w_t)$ partition $\{v_1, \dots, v_k\}$.

Clearly these invariants hold for G_3, embedded as described above.

Idea for Step $k+1$. We are about to place vertex v_{k+1}. Its neighbors $w_\ell, \dots, w_r$ lie consecutively on $C_\circ(G_k)$ by the property of canonical ordering. Put v_{k+1} at position $\mu(P(w_\ell), P(w_r))$, where

$$\mu((x_\ell, y_\ell), (x_r, y_r)) := \left(\frac{x_\ell - y_\ell + x_r + y_r}{2}, \frac{-x_\ell + y_\ell + x_r + y_r}{2} \right)$$

is the intersection between the line $y = x - x_\ell + y_\ell$ of slope 1 through (x_ℓ, y_ℓ) and the line $y = x_r - x + y_r$ of slope -1 through (x_r, y_r).

Proposition 2.53. *If the Manhattan distance between* $P(w_\ell)$ *and* $P(w_r)$ *is even, then* $\mu(P(w_\ell), P(w_r))$ *is on the integer grid.*

Proof. By (ii) we know that $x_\ell < x_r$. Suppose without loss of generality that $y_\ell \leqslant y_r$. The Manhattan distance of the two points is $d := x_r - x_\ell + y_r - y_\ell$, an even number by assumption. Adding an even number $2x_\ell$ to d yields the even number $x_r + x_\ell + y_r - y_\ell$, half of which is the x-coordinate of $\mu((x_\ell, y_\ell), (x_r, y_r))$. Adding an even number $2y_\ell$ to d yields the even number $x_r - x_\ell + y_r + y_\ell$, half of which is the y-coordinate of $\mu((x_\ell, y_\ell), (x_r, y_r))$. □

However, $\mu(P(w_\ell), P(w_r))$ may be unable to "see" all of $w_\ell, \dots, w_r$, in case that the slope of $w_\ell w_{\ell+1}$ is 1 and/or the slope of $w_{r-1} w_r$ is -1 (Figure 2.21).

In order to resolve these problems, we shift some points to the right so that $w_{\ell+1}$ no longer lies on the line of slope 1 through w_ℓ, and that w_{r-1} no longer lies on the line of slope -1 through w_r. The actual Step $k+1$ then reads:

1. Shift $\bigcup_{i=\ell+1}^{r-1} F(w_i)$ to the right by one unit.
2. Shift $\bigcup_{i=r}^{t} F(w_i)$ to the right by two units.

[5] The notation is a bit sloppy because both t and the w_i depend on k. So in principle we should write w_i^k instead of w_i. But as the k would just make a constant appearance throughout, we omit it to avoid clutter.

[6] The *Manhattan distance* of two points (x_1, y_1) and (x_2, y_2) is $|x_2 - x_1| + |y_2 - y_1|$.

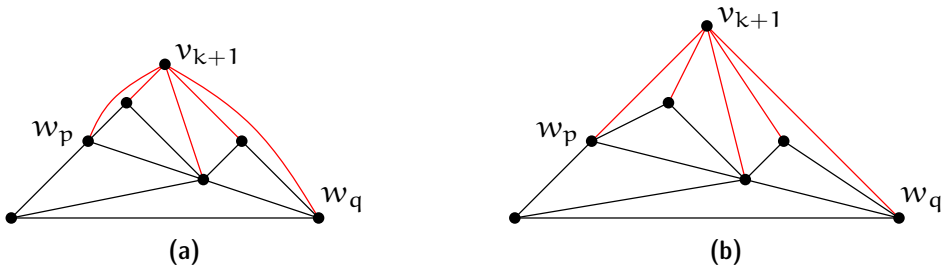


Figure 2.21: *(a) The new vertex v_{k+1} is adjacent to all of $w_\ell, \dots, w_r$. If we place v_{k+1} at $\mu(P(w_\ell), P(w_r))$, then some edges may overlap, in case that $w_{\ell+1}$ lies on the line of slope 1 through w_ℓ or w_{r-1} lies on the line of slope -1 through w_r; (b) shifting $w_{\ell+1}, \dots, w_{r-1}$ by one and $w_r, \dots, w_t$ by two units to the right solves the problem.*

3. $P(v_{k+1}) := \mu(P(w_\ell), P(w_r))$.
4. $F(v_{k+1}) := \{v_{k+1}\} \cup \bigcup_{i=\ell+1}^{r-1} F(w_i)$.

Next we argue that the invariants (i)–(iv) are maintained after Step $k+1$.

For (i), note that the shifting always starts from $w_{\ell+1}$ onward. So $w_1 = v_1$ is never moved and stays at $P(v_1) = (0,0)$. On the other hand, we shift every vertex by two starting from (and including) w_r, hence v_2 moves two units to $P(v_2) = (2(k+1)-4, 0)$.

Also, observe that the Manhattan distance between w_ℓ and w_r remains even because the shift increases their horizontal distance by two and leaves the y-coordinates unchanged. Therefore by Proposition 2.53 the vertex v_{k+1} is embedded on the integer grid indeed.

After shifting, the absolute slopes of the edges $w_\ell w_{\ell+1}$ and $w_{r-1}w_r$ (possibly the same edge) become < 1, and the absolute slopes of all other edges on $C_\circ(G_k)$ remain 1. In contrast, the edges $v_{k+1}w_\ell$ and $v_{k+1}w_r$ both have absolute slope 1, and all edges from v_{k+1} to $w_{\ell+1}, \dots, w_{r-1}$ have absolute slopes > 1. Hence, for all $i \in \{\ell, \dots, r\}$, the edge $v_{k+1}w_i$ intersects $C_\circ(G_k)$ in exactly one point, which is w_i. In other words, these new edges will not cross anything in G_k.

Of course, to conclude that the drawing is plane, we also need to argue that the edges originally in G_k do not clash with each other after shifting. But as this is intuitively clear, we postpone the formal argument for later. Now (i) is complete.

For (ii), clearly both the shifts and the insertion of v_{k+1} maintain the strict order along the outer cycle. For (iii), note that the edges $w_\ell w_{\ell+1}$ and $w_{r-1}w_r$ (possibly equal) are the only edges on the outer cycle $C_\circ(G_k)$ whose slope is changed. But neither edge appears on $C_\circ(G_{k+1})$ any more, as they are covered by the two new edges $v_{k+1}w_\ell$ and $v_{k+1}w_r$; these new edges have slope 1 and -1, respectively. Regarding (iv), the set $F(v_{k+1})$ by definition includes the new vertex v_{k+1} and encompasses the sets of all outer cycle vertices that it covers (that is, they are on $C_\circ(G_k)$ but not on $C_\circ(G_{k+1})$). So the sets on $C_\circ(G_{k+1})$ partition $\{v_1, \dots, v_{k+1}\}$.

So (i)–(iv) are invariants of the algorithm, indeed. Let us look at the consequences. During the entire procedure, invariants (i)(ii) and the definition of μ ensures that each point is placed on a $(2n-3) \times (n-2)$ integer grid. In fact, the final vertex v_n is always placed at $\mu(P(v_1), P(v_2)) = \mu((0,0),(2n-4,0)) = (n-2, n-2)$ since both v_1 and v_2 are its neighbors.

Finally, we return to provide a formal argument that the "interior part" of the drawing remains plane under shifts.

Lemma 2.54. *Let G_k, $k \geqslant 3$, be straight-line embedded on grid as described by the algorithm. Assume $C_\circ(G_k) = (w_1, \ldots, w_t)$, and let $\delta_1 \leqslant \ldots \leqslant \delta_t$ be nonnegative integers. If for each i we shift $F(w_i)$ by δ_i to the right, then the resulting straight-line drawing is plane.*

Proof. Induction on k. For the base case G_3 this is obvious. Now for G_k, assume $v_k = w_\ell$, where $2 \leqslant \ell < t$. Denote its $m \geqslant 2$ neighbors as $u_1, \ldots, u_m$ where $u_1 = w_{\ell-1}$ and $u_m = w_{\ell+1}$. Then we have

$$C_\circ(G_{k-1}) = (w_1, \ldots, w_{\ell-1}, \underbrace{u_2, \ldots, u_{m-1}}_{\text{could be empty}}, w_{\ell+1}, \ldots, w_t).$$

Recall that the algorithm defines $F(v_k) = \{v_k\} \cup \bigcup_{i=1}^m F(u_i)$. Hence, to shift each $F(w_i)$ by δ_i is equivalent to applying the sequence

$$\Delta := (\delta_1, \ldots, \delta_{\ell-1}, \underbrace{\delta_\ell, \ldots, \delta_\ell}_{m-2 \text{ times}}, \delta_{\ell+1}, \ldots, \delta_t)$$

to G_{k-1} and then shifting v_k by δ_ℓ.

Clearly Δ is monotonically increasing, so by the inductive assumption the shifted drawing of G_{k-1} is plane. After shifting v_k by δ_ℓ, the drawing of G_k is plane: Vertex v_k moves rigidly with its neighbors $u_2, \ldots, u_{m-1}$, and the two extreme neighbors u_1 and u_m move relatively to the left and right, respectively. The corresponding edges cannot cross anything during this movement. □

Linear time. The challenge in implementing the shift algorithm efficiently lies in the eponymous shift operations, which modify the x-coordinates of potentially many vertices. In fact, it is not hard to see that a naive implementation—which keeps track of all coordinates explicitly—may use quadratic time. De Fraysseix et al. described an implementation of the shift algorithm that uses $O(n \log n)$ time. Then Chrobak and Payne [11] observed how to improve the runtime to linear, using the following ideas.

Recall that $P(v_{k+1})$ is placed at the coordinates

$$x = \frac{x_\ell - y_\ell + x_r + y_r}{2} \quad \text{and} \quad y = \frac{(x_r - x_\ell) + y_\ell + y_r}{2}, \tag{2.55}$$

and thus

$$x - x_\ell = \frac{(x_r - x_\ell) + y_r - y_\ell}{2}. \tag{2.56}$$

In other words, to determine the y-coordinate and the x-offset relative to the leftmost neighbor w_ℓ, we only need to know the y-coordinates of $P(w_\ell)$ and $P(w_r)$ together with x-offset of $P(w_r)$ relative to $P(w_\ell)$.

To exploit these relations, we represent the geometric embedding of G_k as follows. For each vertex v_i, for $1 \leqslant i \leqslant k$, we store three integers:

1. the y-coordinate $y(v_i)$, which is determined when v_i is inserted and remains the same throughout the algorithm,
2. an index (or pointer) to a parent vertex $p(v_i) \in V(G_k)$, and
3. the x-offset $\Delta(v_i)$ of $P(v_i)$ relative to $P(p(v_i))$.

The *parent vertex* $p(v_i)$ of a vertex v_i on $C_\circ(G_k)$ is its predecessor on $C_\circ(G_k)$, that is, if $C_\circ(G_k) = (w_1, \ldots, w_t)$, then $p(w_j) = w_{j-1}$, for $j > 1$, and $p(w_1) = \textit{nil}$. For a vertex v_i in $G_k \setminus C_\circ(G_k)$ its reference vertex $p(v_i)$ is the vertex v_x, where $i < x \leqslant k$, that covered v_i, that is, such that v_i is on $C_\circ(G_{x-1})$ but not on $C_\circ(G_x)$. Note that in the latter case $p(v_i)$ is can equivalently be described as the neighbor of v_i in G that appears last in the canonical ordering used.

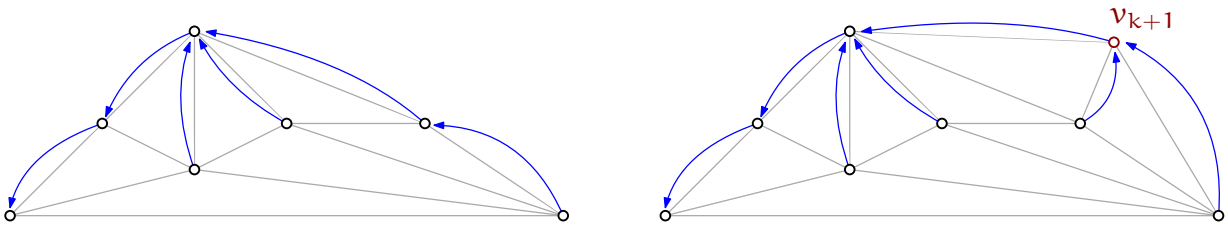


Figure 2.22: *Maintaining the reference tree structure under insertion of a vertex* v_{k+1}.

The parent structure represents a tree that is rooted at v_1, see Figure 2.22 for an example. Knowing the x-coordinate of $P(p(v))$ and the offset $\Delta(v)$, we can easily compute the x-coordinate of $P(v)$. So fixing the x-coordinate of $P(v_1)$, which is zero by Invariant (i), fixes the x-coordinates of all vertices in the drawing. Thus, using this information for $G = G_n$, we can compute all x-coordinates in the final embedding using a simple loop in $O(n)$ time:

```
x[1] = 0
for i = n downto 2:
  x[i] = x[p[i]] + delta[i]
```

Algorithmically, we compute both $p(v_i)$ and $\Delta(v_i)$ in the step when v_i is covered by v_{k+1}, and then these values remain the same for the rest of the algorithm, as the edge $v_i v_{k+1}$ is frozen. Let $w_{\ell+1}, \ldots, w_{r-1}$ denote the vertices covered by inserting v_{k+1}. If $r = \ell + 1$, no vertex is covered and there is nothing to do. So suppose that $r - \ell \geqslant 2$.

We first update the parent information and compute the x-offset of w_r relative to w_ℓ, by iterating over $w_{\ell+1}, \ldots, w_r$, setting their parent to v_{k+1} and summing up the individual offsets. This allows us to compute $\Delta(v_{k+1})$ using (2.56). Then we iterate a second time over $w_{\ell+1}, \ldots, w_r$ to compute the x-offset $\lambda(w_j)$ of $P(w_j)$ relative to $P(w_\ell)$, for all $j \in \{\ell+1, \ldots, r\}$ and use these values to rebase the offset to be relative to $P(v_{k+1})$ instead, as shown in the code fragment below.

```
lambda = 0
for j = l+1 to r:
  lambda = lambda + delta[j]
  delta[j] = lambda - delta[k+1]
```

This concludes the description of the algorithm and its runtime analysis.

2.5.3 Remarks and Open Problems

From a geometric complexity point of view, Theorem 2.43 provides very good news for planar graphs in a similar way that the Euler Formula does from a combinatorial complexity point of view. Euler's Formula tells us that we can obtain a combinatorial representation (for instance, as a DCEL) of any plane graph using $O(n)$ space, where n is the number of vertices. Now the shift algorithm tells us that for any planar graph we can even find a geometric plane (straight-line) representation using $O(n)$ space. In addition to the combinatorial information, we only have to store $2n$ numbers from the range $\{0, 1, \ldots, 2n-4\}$.

When we make such claims regarding space complexity we implicitly assume the so-called *word RAM model*. In this model each memory cell stores a *word* of b bits, which may represent any integer in $\{0, \ldots, 2^b - 1\}$. One also assumes that b is sufficiently large, in our case $b \geqslant \log n$.

There are also different models such as the *bit complexity model*, where one is charged for every bit used to store information. In our case that would already incur an additional factor of $\log n$ for the combinatorial representation: for instance, for each halfedge we store its endpoint, which is an index from $\{1, \ldots, n\}$.

Edge lengths. Theorem 2.43 shows that planar graphs admit a plane straight-line drawing where all vertices have integer coordinates. It is an open problem whether a similar statement can be made for edge lengths.

Problem 2.57 (Harborth's Conjecture [17]). Every planar graph admits a plane straight-line drawing where all Euclidean edge lengths are integral.

Without the planarity restriction such a drawing is possible because for every $n \in \mathbb{N}$ one can find a set of n points in the plane, not all collinear, such that their distances are all integral. In fact, such a set of points can be constructed to lie on a circle of integral radius [2]. When mapping the vertices of K_n onto such a point set, all edge lengths are

integral. In the same paper it is also shown that there exists no infinite set of points in the plane so that all distances are integral, unless all of these points are collinear. Unfortunately, collinear point sets are not very useful for drawing graphs. The existence of a dense subset of the plane where all distances are rational would resolve Harborth's Conjecture. However, it is not known whether such a set exists, and in fact the suspected answer is "no".

Problem 2.58 (Erdős–Ulam Conjecture [12]). There is no dense set of points in the plane whose Euclidean distances are all rational.

Generalizing the Fáry-Wagner Theorem. As discussed earlier, not every planar graph on n vertices admits a plane straight-line embedding on every set of n points. But Theorem 2.40 states that for every planar graph G on n vertices there *exists* a set P of n points in the plane so that G admits a plane straight-line embedding on P. It is an open problem whether this statement can be generalized to hold for several graphs, in the following sense.

Problem 2.59. What is the largest number $k \in \mathbb{N}$ for which the following statement holds? For every collection of k planar graphs $G_1, \ldots, G_k$ on n vertices each, there exists a set P of n points so that G_i admits a plane straight-line embedding on P, for every $i \in \{1, \ldots, k\}$.

By Theorem 2.40 we know that the statement holds for $k = 1$. Already for $k = 2$ it is not known whether the statement holds. However, it is known that k is finite [8]. Specifically, there exists a collection of 49 planar graphs on 11 vertices each so that for every set P of 11 points in the plane at least one of these graphs does not admit a plane straight-line embedding on P [29]. Therefore we have $k \leqslant 49$.

Questions

1. *What is an embedding? What is a planar/plane graph?* Give the definitions and explain the difference between planar and plane.
2. *How many edges can a planar graph have? What is the average vertex degree in a planar graph?* Explain Euler's formula and derive your answers from it.
3. *How can plane graphs be represented on a computer?* Explain the DCEL data structure and how to work with it.
4. *How can a given plane graph be (topologically) triangulated efficiently?* Explain what it is, including the difference between topological and geometric triangulation. Give a linear time algorithm, for instance, as in Theorem 2.34.
5. *What is a combinatorial embedding? When are two combinatorial embeddings equivalent? Which graphs have a unique combinatorial plane embedding?* Give the definitions, explain and prove Whitney's Theorem.

6. *What is a canonical ordering and which graphs admit such an ordering? For a given graph, how can one find a canonical ordering efficiently?* Give the definition. State and prove Theorem 2.46.

7. *Which graphs admit a plane embedding using straight line edges? Can one bound the size of the coordinates in such a representation?* State and prove Theorem 2.43.

References

[1] Oswin Aichholzer, Thomas Hackl, Matias Korman, Marc van Kreveld, Maarten Löffler, Alexander Pilz, Bettina Speckmann, and Emo Welzl, Packing plane spanning trees and paths in complete geometric graphs. *Inform. Process. Lett.*, 124, (2017), 35–41.

[2] Norman H. Anning and Paul Erdős, Integral distances. *Bull. Amer. Math. Soc.*, 51/8, (1945), 598–600.

[3] Bruce G. Baumgart, A polyhedron representation for computer vision. In *Proc. AFIPS Natl. Comput. Conf.*, vol. 44, pp. 589–596, AFIPS Press, Alrington, Va., 1975.

[4] Ahmad Biniaz and Alfredo García, Packing plane spanning trees into a point set. *Comput. Geom. Theory Appl.*, 90, (2020), 101653.

[5] Prosenjit Bose, Ferran Hurtado, Eduardo Rivera-Campo, and David R. Wood, Partitions of complete geometric graphs into plane trees. *Comput. Geom. Theory Appl.*, 34/2, (2006), 116–125.

[6] John M. Boyer and Wendy J. Myrvold, On the cutting edge: simplified O(n) planarity by edge addition. *J. Graph Algorithms Appl.*, 8/3, (2004), 241–273.

[7] Sergio Cabello, Planar embeddability of the vertices of a graph using a fixed point set is NP-hard. *J. Graph Algorithms Appl.*, 10/2, (2006), 353–363.

[8] Jean Cardinal, Michael Hoffmann, and Vincent Kusters, On universal point sets for planar graphs. *J. Graph Algorithms Appl.*, 19/1, (2015), 529–547.

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

[10] Norishige Chiba and Takao Nishizeki, The Hamiltonian cycle problem is linear-time solvable for 4-connected planar graphs. *J. Algorithms*, 10/2, (1989), 187–211.

[11] Marek Chrobak and Thomas H. Payne, A linear-time algorithm for drawing a planar graph on a grid. *Inform. Process. Lett.*, 54, (1995), 241–246.

[12] Paul Erdős, Ulam, the man and the mathematician. *J. Graph Theory*, 9/4, (1985), 445–449.

[13] István Fáry, On straight lines representation of planar graphs. *Acta Sci. Math. Szeged*, 11/4, (1948), 229–233.

[14] Hubert de Fraysseix, Patrice Ossona de Mendez, and Pierre Rosenstiehl, Trémaux trees and planarity. *Internat. J. Found. Comput. Sci.*, 17/5, (2006), 1017–1030.

[15] Hubert de Fraysseix, János Pach, and Richard Pollack, How to draw a planar graph on a grid. *Combinatorica*, 10/1, (1990), 41–51.

[16] Leonidas J. Guibas and Jorge Stolfi, Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Trans. Graph.*, 4/2, (1985), 74–123.

[17] Heiko Harborth and Arnfried Kemnitz, Plane integral drawings of planar graphs. *Discrete Math.*, 236/1–3, (2001), 191–195.

[18] Dawei He, Yan Wang, and Xingxing Yu, The Kelmans-Seymour conjecture IV: A proof. *J. Combin. Theory Ser. B*, 144, (2020), 309–358.

[19] John Hopcroft and Robert E. Tarjan, Efficient planarity testing. *J. ACM*, 21/4, (1974), 549–568.

[20] Ken-ichi Kawarabayashi, Yusuke Kobayashi, and Bruce Reed, The disjoint paths problem in quadratic time. *J. Combin. Theory Ser. B*, 102/2, (2012), 424–435.

[21] Lutz Kettner, *Software design in computational geometry and contour-edge based polyhedron visualization*. Ph.D. thesis, ETH Zürich, Zürich, Switzerland, 1999.

[22] Kazimierz Kuratowski, Sur le problème des courbes gauches en topologie. *Fund. Math.*, 15/1, (1930), 271–283.

[23] László Lovász, Graph minor theory. *Bull. Amer. Math. Soc.*, 43/1, (2006), 75–86.

[24] Bojan Mohar and Carsten Thomassen, *Graphs on surfaces*, Johns Hopkins University Press, Baltimore, 2001.

[25] David E. Muller and Franco P. Preparata, Finding the intersection of two convex polyhedra. *Theoret. Comput. Sci.*, 7, (1978), 217–236.

[26] Johannes Obenaus and Joachim Orthaber, Edge Partitions of Complete Geometric Graphs (Part 1). *CoRR*, abs/2108.05159, (2021), 1–32.

[27] János Pach and Rephael Wenger, Embedding planar graphs at fixed vertex locations. *Graphs Combin.*, 17, (2001), 717–728.

[28] Neil Robertson and Paul Seymour, Graph Minors. XX. Wagner's Conjecture. *J. Combin. Theory Ser. B*, 92/2, (2004), 325–357.

[29] Manfred Scheucher, Hendrik Schrezenmaier, and Raphael Steiner, A Note on Universal Point Sets for Planar Graphs. *J. Graph Algorithms Appl.*, 24/3, (2020), 247–267.

[30] Walter Schnyder, Planar graphs and poset dimension. *Order*, 5, (1989), 323–343.

[31] Carsten Thomassen, Kuratowski's Theorem. *J. Graph Theory*, 5/3, (1981), 225–241.

[32] William T. Tutte, A theorem on planar graphs. *Trans. Amer. Math. Soc.*, 82/1, (1956), 99–116.

[33] Klaus Wagner, Bemerkungen zum Vierfarbenproblem. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 46, (1936), 26–32.

[34] Klaus Wagner, Über eine Eigenschaft der ebenen Komplexe. *Math. Ann.*, 114/1, (1937), 570–590.

[35] Kevin Weiler, Edge-based data structures for solid modeling in a curved surface environment. *IEEE Comput. Graph. Appl.*, 5/1, (1985), 21–40.

[36] Hassler Whitney, Congruent graphs and the connectivity of graphs. *Amer. J. Math.*, 54/1, (1932), 150–168.

Chapter 3

Crossings

So far we have mostly studied planar graphs which allow us to avoid crossings altogether. However, there are many interesting graphs that are not planar, and still we would like to draw them in a reasonable fashion. An obvious quantitative approach is to minimize the number of crossings, even if they are inevitable.

3.1 Crossing Numbers

For an abstract graph $G = (V, E)$, the *crossing number* $\mathrm{cr}(G)$ is defined as the minimum number of edge crossings over all drawings of G. Analogously, the *rectilinear crossing number* $\overline{\mathrm{cr}}(G)$ is defined as the minimum number of edge crossings over all straight-line drawings of G. A drawing of G that achieves $\mathrm{cr}(G)$ or $\overline{\mathrm{cr}}(G)$ crossings is called a *minimum-crossing drawing* or *minimum-crossing straight-line drawing*, respectively.

These notions are well-defined since $\mathrm{cr}(G) \leqslant \overline{\mathrm{cr}}(G) \leqslant \binom{|E|}{2}$ are finite. To see the upper bound, we construct a straight-line drawing of G as follows. Bijectively map the vertices of V onto a set of $n = |V|$ points in general position (that is, such that no three points are collinear), then draw every edge as a straight-line segment. This is a valid drawing in which every pair of distinct edges share at most one point.

Actually, this last property also holds for all minimum-crossing drawings, as the following lemma demonstrates.

Lemma 3.1. *In every minimum-crossing drawing of a graph, every pair of distinct edges share at most one point.*

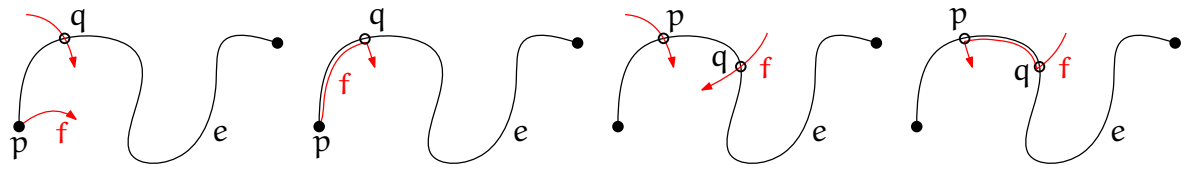
Proof. Consider a graph G and a minimum-crossing drawing Γ of G. Suppose for contradiction that two edges $e \neq f$ share distinct points $p \neq q$ in Γ. Let e_p^q be the part of e from p to q in Γ, and similarly define f_p^q. Without loss of generality, suppose that e_p^q has no more crossings than f_p^q. Then we redraw f_p^q to closely follow e_p^q, as illustrated in Figure 3.1.

- If p (or q) is a common vertex of the edges e and f, then we can choose the side so that the crossing at q (or p) is eliminated.

- Otherwise, both p and q are crossing points. Depending on how f approaches p and q, we are able to eliminate either one (if approached from the same side of e) or two (if approached from opposite sides of e) of these crossings.

The number of crossings other than p and q does not increase, due to our assumption that e_p^q has at most as many crossings as f_p^q. Hence, the total number of crossings strictly decreases.

Finally, if f should cross itself due to this modification, we can eliminate each self-crossing by omitting the curve between the two occurrences along f. The result is a drawing of G with strictly fewer crossings than Γ, a contradiction to Γ being a minimum-crossing drawing of G. □



(a) before **(b)** after **(c)** before **(d)** after

Figure 3.1: *Redraw f_p^q by the side of e_p^q to reduce the overall number of crossings. (a) and (b) depict the situation where both edges e and f are incident to a vertex p, in which case the crossing at q can be eliminated. (c) and (d) depict the situation where both p and q are crossings; in this example we can remove a crossing at p or q.*

A drawing in which every pair of edges has at most one point in common is called *simple*, and a graph drawn together with a simple drawing of it is called a *simple topological graph*. Using this terminology we can rephrase Lemma 3.1 as follows: "Every minimum-crossing drawing is simple." A simple drawing implies that no two adjacent edges cross. Drawings that satisfy this latter (and weaker) property are called *star-simple* because the incident edges to any vertex form a plane star.[1]

It is quite easy to certify an upper bound on the crossing number of a graph—just present a drawing that has a small number of crossings. But it is conceptually harder to certify a lower bound because it needs to account for *all* possible drawings of this graph. The following lower bound, though, can be obtained by simple counting.

Lemma 3.2. *For a graph G with $n \geqslant 3$ vertices and e edges, we have $\mathrm{cr}(G) \geqslant e-(3n-6)$.*

Proof. Consider a drawing of $G = (V, E)$ with $\mathrm{cr}(G)$ crossings. For each crossing, we pick one of the two involved edges arbitrarily. Obtain a new graph $G' = (V, E')$ from G by removing all picked edges. By construction G' is plane and, therefore, $|E'| \leqslant 3n - 6$ by Corollary 2.5. As at most $\mathrm{cr}(G)$ edges were picked ("at most" because some edge might

[1] In the literature also the terms *semi-simple* or *semisimple* are used.

be picked by several crossings), we have $|E'| \geqslant |E| - \mathrm{cr}(G)$. Combining both bounds completes the proof. □

Exercise 3.3. *Consider two edges e and f in a topological plane drawing so that e and f cross at least twice. Prove or disprove: There always exist two distinct crossings p and q of e and f so that the portion of e between p and q is not crossed by f, and the portion of f between p and q is not crossed by e.*

Exercise 3.4. *Let G be a graph with $n \geqslant 3$ vertices, e edges, and $\mathrm{cr}(G) = e - (3n-6)$. Show that in every drawing of G with $\mathrm{cr}(G)$ crossings, every edge is crossed at most once.*

Exercise 3.5. *Consider the abstract graph G that is obtained as follows: Start from a plane embedding of the 3-dimensional cube, and add in every face a pair of (crossing) diagonals. Show that $\mathrm{cr}(G) = 6 < \overline{\mathrm{cr}}(G)$.*

Exercise 3.6. *A graph is* 1-planar *if it can be drawn in the plane so that every edge is crossed at most once. Show that a 1-planar graph G on $n \geqslant 3$ vertices has at most $4n-8$ edges and $\mathrm{cr}(G) \leqslant n-2$.*

3.2 The Crossing Lemma

The bound in Lemma 3.2 is quite good if the number of edges is close to $3n$ but not so good for dense graphs. For instance, for the complete graph K_n the lemma guarantees a quadratic number of crossings, whereas the Guy-Harary-Hill Conjecture [8] claims

$$\mathrm{cr}(K_n) = \frac{1}{4}\left\lfloor\frac{n}{2}\right\rfloor\left\lfloor\frac{n-1}{2}\right\rfloor\left\lfloor\frac{n-2}{2}\right\rfloor\left\lfloor\frac{n-3}{2}\right\rfloor \in \Theta(n^4).$$

The conjecture has been verified, in part with extensive computer help, for the complete graph on $n \leqslant 14$ vertices [2, 9, 11]; it remains open for $n \geqslant 15$.

So for dense graphs we would like to have a better lower bound for the crossing number. Given that the bound in Lemma 3.2 is reasonably good for sparse graphs, why not apply it to some sparse subgraph of G and then try scaling back to G? This simple idea turns out to work astonishingly well, as the following theorem demonstrates.

Theorem 3.7 (Crossing Lemma [4]). *For a graph G with n vertices and $e \geqslant 4n$ edges, we have*

$$\mathrm{cr}(G) \geqslant \frac{e^3}{64n^2}.$$

Proof. Consider a minimum-crossing drawing Γ of G, with $\mathrm{cr}(G)$ crossings. We select each vertex independently with probability p (a suitable value for p will be determined later). By this process we obtain a random subset $U \subseteq V$, the corresponding induced subgraph $G[U]$, along with its induced drawing $\Gamma[U]$. Consider the following three random variables:

- $N = |U|$, the number of selected vertices, with $\mathbb{E}[N] = pn$;
- M, the number of edges in $G[U]$, with $\mathbb{E}[M] = p^2 e$; and
- C, the number of crossings in $\Gamma[U]$, with $\mathbb{E}[C] = p^4 \mathrm{cr}(G)$. (Here we use Lemma 3.1, which says that adjacent edges do not cross in the minimum-crossing drawing Γ.)

According to Lemma 3.2, these quantities satisfy $C \geqslant \mathrm{cr}(G[U]) \geqslant M - 3N$ under all outcomes of the random experiment. Taking expectations on both sides and using linearity of expectation yields $\mathbb{E}[C] \geqslant \mathbb{E}[M] - 3\mathbb{E}[N]$ and so $p^4 \mathrm{cr}(G) \geqslant p^2 e - 3pn$. Setting $p = 4n/e$ (which is $\leqslant 1$ due to the assumption $e \geqslant 4n$) gives

$$\mathrm{cr}(G) \geqslant \frac{e}{p^2} - 3\frac{n}{p^3} = \frac{e^3}{16n^2} - 3\frac{e^3}{64n^2} = \frac{e^3}{64n^2}. \qquad \square$$

The beautiful proof described above is attributed to Chazelle, Sharir, and Welzl and listed in "Proofs from THE BOOK" [3, Chapter 40], a collection inspired by Paul Erdős' belief in "a place where God keeps aesthetically perfect proofs". The original proof of the Crossing Lemma was more complicated and had a worse constant.

Asymptotically the bound in Theorem 3.7 is tight: Pach and Tóth [10] describe graphs with $n \ll e \ll n^2$ that have crossing number at most

$$\frac{16}{27\pi^2}\frac{e^3}{n^2} < \frac{1}{16.65}\frac{e^3}{n^2}.$$

Hence it is not possible to replace $1/64$ by $1/16.65$ in the statement of the theorem. However, the constant $1/64$ is not the best possible: Ackerman [1] showed that $1/64$ can be replaced by $1/29$, at the cost of requiring $e \geqslant 6.95n$. Very recently, Büngener and Kaufmann [5] further improved the constant to $1/27.48$, at the cost of requiring $e \geqslant 6.77n$.

Exercise 3.8. *Show that the bound from the Crossing Lemma is asymptotically tight: There exists a constant c so that for every $n, e \in \mathbb{N}$ with $e \leqslant \binom{n}{2}$ there is a graph with n vertices and e edges that admits a plane drawing with at most ce^3/n^2 crossings.*

Exercise 3.9. *We call a graph* quasiplanar *if it admits a simple drawing in the plane such that no three edges pairwise cross. Denote by $\mathrm{qp}(n)$ the maximum number of edges in a quasiplanar graph on n vertices. Show that $\mathrm{qp}(n) \in O(n^{3/2})$.*

3.3 Applications of the Crossing Lemma

In the remainder of this chapter, we will discuss several nontrivial bounds on the size of combinatorial structures that can be obtained by judicious application of the Crossing Lemma. These beautiful connections were observed by Székely [13]; their original proofs were different and more involved.

We say that a point and a geometric object (such as a line or a circle) are *incident* if the former lies on the latter.

Theorem 3.10 (Szemerédi-Trotter [14]). *The maximum number of incidences between n points and m lines in $\mathbb{R}^2$ is at most $2^{5/3} \cdot n^{2/3}m^{2/3} + 4n + m$.*

Proof. Let P denote the given set of n points, and let L denote the given set of m lines. We may suppose that every line from L contains at least one point from P. (Discard all lines that do not, as they contribute no incidence.) Denote by I the number of incidences between P and L. Consider the graph $G = (P, E)$ whose vertices are the points P, and where two points p, q are joined by an edge if they appear consecutively along some line $\ell \in L$ (that is, $p, q \in \ell$ and no other point from P lies on the line segment $\overline{pq}$). The arrangement of P and L naturally induces a straight-line drawing of G. It has at most $\binom{m}{2}$ crossings because every crossing must be an intersection of two lines, and any two lines can intersect at most once.

Each line $\ell \in L$ is incident to some $I_\ell \geqslant 1$ point(s) from P and contributes $I_\ell - 1$ edge(s) to E. Hence $|E| = \sum_{\ell \in L}(I_\ell - 1) = I - m$. If $|E| \leqslant 4n$, then $I \leqslant 4n + m$ and the theorem holds. Otherwise, we can apply the Crossing Lemma to obtain

$$\binom{m}{2} \geqslant \mathrm{cr}(G) \geqslant \frac{|E|^3}{64n^2} = \frac{(I-m)^3}{64n^2}$$

and so $I \leqslant 2^{5/3}\, n^{2/3}m^{2/3} + m$. □

The bound in Theorem 3.10 is asymptotically tight, in the following sense [10, Remark 4.2]. There exist sets of n points and m lines in $\mathbb{R}^2$ that have $c \cdot n^{2/3}m^{2/3}$ incidences, for some constant $c > 0.42$ that is independent of n and m.

Theorem 3.11. *The maximum number of unit distances between n points in $\mathbb{R}^2$ is at most $5n^{4/3}$.*

Proof. Let P be the given set of n points, and consider the set C of n unit circles centered at the points in P. Then the number I of incidences between P and C is exactly twice the number of unit distances between points from P. So it suffices to upper bound I.

Define a graph $G = (P, E)$ on P as follows. For each circle $c \in C$, we list the points from $P \cap c$ in circular order, and add a new edge between every pair of consecutive points. By construction, if c contains I_c points from P, then it contributes exactly I_c edges to E, hence $I = |E|$. Note however that G is not necessarily simple, as it may contain loops (if some $I_c = 1$) and parallel edges (if some $I_c = 2$, or if some $p, q \in P$ are consecutive along different circles).

Obtain a new graph $G' = (P, E')$ from G by removing all edges along circles $c \in C$ of $I_c \leqslant 2$. Since at most $|C| = n$ circles are removed and each removed circle contributed at most two edges to E, we have $|E'| \geqslant |E| - 2n$. In G' there are neither loops, nor parallel edges contributed by the same circle. Therefore, between any two points p and q there are at most two parallel edges in G' because at most two different unit circles can pass through any two points $p \neq q$ in $\mathbb{R}^2$.

Obtain a new graph $G'' = (P, E'')$ from G' by removing one copy of every double edge. Clearly G'' is a simple graph with $|E''| \geqslant |E'|/2 \geqslant |E|/2 - n$. Rearranging, we have $I = |E| \leqslant 2(|E''| + n)$.

If $|E''| \leqslant 4n$, then $I \leqslant 10n < 10n^{4/3}$ and the theorem holds. Otherwise, by the Crossing Lemma we have

$$n^2 > 2\binom{n}{2} \geqslant cr(G'') \geqslant \frac{|E''|^3}{64n^2}.$$

Here the upper bound on $cr(G'')$ is due to that every pair of circles can intersect at most twice. Rearranging, it follows that $|E''| < 4n^{4/3}$ and so $I < 8n^{4/3} + 2n < 10n^{4/3}$. □

Exercise 3.12. *Show that the maximum number of unit distances determined by n points in $\mathbb{R}^2$ is $\Omega(n \log n)$. Hint: Consider the hypercube.*

The final application comes from arithmetic combinatorics. Given a set $A \subset \mathbb{R}$, we denote the *sum set* by $A + A := \{a + a' : a, a' \in A\}$ and similarly the *product set* by $A \cdot A := \{a \cdot a' : a, a' \in A\}$. It is easy to construct ground sets that have a small, that is, linear size sum set: Just take an arithmetic progression, such as $2, 4, 6, 8, 10, \ldots$. Similarly, geometric progressions exhibit a small product set. However, it is much more challenging to find a ground set A for which both the sum set and the product set are small. In fact, Erdős conjectured [7] that for every set A of n numbers, we have $\max\{|A+A|, |A \cdot A|\} \in \Omega(n^{2-\epsilon})$, for every $\epsilon > 0$. The general conjecture is still open. But the statement is known to hold for *some* reasonably small values of ϵ. At a first glance, it is not so clear why there should be a connection between this problem and questions about crossings in drawings of graphs. But there is such a connection, as discovered by Elekes [6]. He used the Crossing Lemma to give an elegant proof of the following bound.

Theorem 3.13 (Elekes [6]). *For $A \subset \mathbb{R}$ with $|A| = n \geqslant 3$ we have*

$$\max\{|A + A|, |A \cdot A|\} \geqslant \frac{1}{4} n^{\frac{5}{4}}.$$

Proof. Let $A = \{a_1, \ldots, a_n\}$. Set $X = A + A$ and $Y = A \cdot A$. We will show that $|X||Y| \geqslant \frac{1}{16} n^{5/2}$, which proves the theorem. Let $P = X \times Y \subset \mathbb{R}^2$ be the set of points whose x-coordinate is in X and whose y-coordinate is in Y. So we have $|P| = |X||Y|$. Next define a set L of lines by $\ell_{ij} = \{(x, y) \in \mathbb{R}^2 : y = a_i(x - a_j)\}$, for $i, j \in \{1, \ldots, n\}$. Clearly, we have $|L| = n^2$.

On the one hand, every line ℓ_{ij} contains at least n points from P because for each $k \in \{1, \ldots, n\}$, the point $(x_k, y_k) := (a_j + a_k, a_i a_k) \in X \times Y$ satisfies the equation $y_k = a_i(x_k - a_j)$ and thus is on ℓ_{ij}. Therefore the number I of incidences between P and L is at least $|L| \cdot n = n^3$.

On the other hand, by the Szemerédi-Trotter Theorem we have

$$I \leqslant 2^{5/3}|P|^{2/3}n^{4/3} + 4|P| + n^2.$$

Combining both bounds we obtain

$$2^{5/3}|P|^{2/3}n^{4/3} + 4|P| + n^2 \geqslant n^3.$$

Hence, at least one of the two summands $2^{5/3}|P|^{2/3}n^{4/3}$ and $4|P| + n^2$ is at least half of the sum, that is, at least $n^3/2$. If it is the latter, then we have

$$|P| \geqslant \frac{n^2}{4}\left(\frac{n}{2} - 1\right).$$

Using that $n \geqslant 3$ and therefore $\sqrt{n} \geqslant 3/2$, we continue to bound

$$\frac{n^2}{4}\left(\frac{n}{2} - 1\right) = \frac{n^2}{4}\left(\frac{\sqrt{n}\sqrt{n}}{6} + \frac{n}{3} - 1\right) \geqslant \frac{n^2}{4}\frac{\sqrt{n}}{4} = \frac{n^{5/2}}{16}.$$

To conclude the proof it remains to consider the former case, in which

$$|P|^{2/3} \geqslant \frac{n^3}{2 \cdot 2^{5/3} n^{4/3}} = \left(\frac{n^5}{256}\right)^{1/3} \Longrightarrow |P| \geqslant \frac{n^{5/2}}{16}.$$

□

The lower bound has been gradually improved in a series of papers. The current state of the art is

$$\max\{|A + A|, |A \cdot A|\} \geqslant n^{\frac{4}{3} + \frac{2}{1167}} > n^{1.335}$$

by Rudnev and Stevens [12].

Questions

8. *What is the crossing number of a graph? What is the rectilinear crossing number?* Give the definitions and examples. Explain the difference.
9. *For a nonplanar graph, the more edges it has, the more crossings we would expect. Can you quantify such a correspondence more precisely?* State and prove Lemma 3.2 and Theorem 3.7 (The Crossing Lemma).
10. *Why is it called "Crossing Lemma" rather than "Crossing Theorem"?* Explain at least two applications of the Crossing Lemma, for instance, your pick out of the Theorems 3.10, 3.11, and 3.13.

References

[1] Eyal Ackerman, On topological graphs with at most four crossings per edge. *Comput. Geom. Theory Appl.*, 85, (2019), 101574.

[2] Oswin Aichholzer, Another Small but Long Step for Crossing Numbers: cr(13) = 225 and cr(14) = 315. In *Proc. 33rd Canad. Conf. Comput. Geom.*, pp. 72–77, 2021.

[3] Martin Aigner and Günter M. Ziegler, *Proofs from THE BOOK*, Springer, Berlin, 4th edn., 2010.

[4] Miklós Ajtai, Václav Chvátal, Monroe M. Newborn, and Endre Szemerédi, Crossing-free subgraphs. *Ann. Discrete Math.*, 12, (1982), 9–12.

[5] Aaron Büngener and Michael Kaufmann, Improving the Crossing Lemma by Characterizing Dense 2-Planar and 3-Planar Graphs. *CoRR*, abs/2409.01733, (2024), 1–25.

[6] György Elekes, On the number of sums and products. *Acta Arithmetica*, 81, (1997), 365–367.

[7] Paul Erdős, Some recent problems and results in graph theory, combinatorics and number theory. In *Proc. 7th Southeastern Conf. on Combinatorics, Graph Theory and Computing*, vol. 17 of *Congr. Numer.*, pp. 3–14, 1976.

[8] Frank Harary and Anthony Hill, On the number of crossings in a complete graph. *Proc. Edinburgh Math. Soc.*, 13/4, (1963), 333–338.

[9] Dan McQuillan and R. Bruce Richter, On the crossing number of K_n without computer assistance. *J. Graph Theory*, 82/4, (2016), 387–432.

[10] János Pach and Géza Tóth, Graphs drawn with few crossings per edge. *Combinatorica*, 17/3, (1997), 427–439.

[11] Shengjun Pan and R. Bruce Richter, The crossing number of K_{11} is 100. *J. Graph Theory*, 56/2, (2007), 128–134.

[12] Misha Rudnev and Sophie Stevens, An update on the sum-product problem. *Math. Proc. Camb. Phil. Soc.*, 173/2, (2022), 411–430.

[13] László A. Székely, Crossing numbers and hard Erdős problems in discrete geometry. *Combinatorics, Probability and Computing*, 6/3, (1997), 353–358.

[14] Endre Szemerédi and William T. Trotter, Jr., Extremal problems in discrete geometry. *Combinatorica*, 3/3–4, (1983), 381–392.

Chapter 4

Polygons

Although a line $\ell \subset \mathbb{R}^2$ can be treated as an infinite point set, it has a finite description. For instance, we may encode it by three coefficients $a, b, c \in \mathbb{R}$ with $(a, b) \neq (0, 0)$ and interpret it as all the points satisfying the equation $ax + by = c$. Actually all of the fundamental geometric objects that we mentioned in Chapter 1 can be described by a constant number of parameters; hence they have constant *description complexity* or, informally, just *size*.[1]

In this course we typically deal with objects that have unbounded size. Sometimes they are formed by merely aggregating constant-size objects. For instance, aggregation of points forms a finite point set. At other times we also demand additional structure beyond aggregation. Probably the most fundamental example is what we call a *polygon*. You probably learned this term in school, but what *is* a polygon precisely? Consider the examples shown in Figure 4.1. Are these all polygons? If not, where would you draw the line?

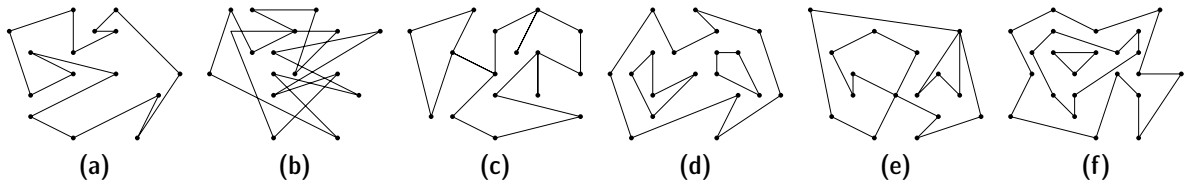


Figure 4.1: *What is a polygon?*

4.1 Classes of Polygons

Ironically, there is no *the* right answer to the question, and there are different types of polygons. Usually, the sloppy term "polygon" refers to what we call a *simple polygon* defined below.

[1] Unless specified otherwise, we always assume that the dimension is a small constant. If we work in high-dimensional space $\mathbb{R}^d$ where d varies, then their description complexity become $\Theta(d)$.

Definition 4.1. *A* **simple polygon** *is a compact region* $P \subset \mathbb{R}^2$ *whose boundary is a simple closed curve* $\partial P : [0,1] \to \mathbb{R}^2$ *consisting of finitely many consecutive line segments.*

Out of the examples shown above only Polygon 4.1a is simple. For each of the remaining polygons the bounding segments do not make a simple closed curve.

When describing a simple polygon P it is sufficient to describe only its boundary ∂P.[2] As ∂P by definition consists of finitely many consecutive line segments, it can be encoded by a sequence $p_1, \ldots, p_n$ of points, so that ∂P is formed by the line segments $\overline{p_1p_2}, \overline{p_2p_3}, \ldots, \overline{p_{n-1}p_n}, \overline{p_np_1}$. These points and segments are referred to as the *vertices* and the *edges* of the polygon, respectively. The set of vertices of a polygon P is denoted by $V(P)$, and the set of edges of P is denoted by $E(P)$.

Recall from Theorem 2.1 (Jordan curve theorem) that any simple closed curve separates the plane $\mathbb{R}^2$ into a (bounded) interior and an (unbounded) exterior. To prove this theorem in its full generality is surprisingly difficult. But for simple polygons the situation is easier, and in fact we can readily tell apart the interior from the exterior algorithmically, which we leave as an exercise.

Exercise 4.2. *Describe an algorithm to decide whether a point lies inside or outside of a simple polygon. More precisely, given a simple polygon* $P \subset \mathbb{R}^2$ *as a list of its vertices* $(v_1, v_2, \ldots, v_n)$ *in counterclockwise order and a query point* $q \in \mathbb{R}^2$*, decide whether* q *is inside* P*, on the boundary of* P*, or outside. The runtime of your algorithm should be* $O(n)$*.*

There are good reasons to ask for the boundary of a polygon to form a simple curve: For instance, in the example depicted in Figure 4.1b there are several regions for which it is completely unclear whether they should belong to the interior or to the exterior of the polygon. A similar problem arises for the interior regions in Figure 4.1f. But there are more general classes of polygons that some of the remaining examples fall into. We will discuss only one such class here. It comprises polygons like the one from Figure 4.1d.

Definition 4.3. *A region* $P \subset \mathbb{R}^2$ *is a* **simple polygon with holes** *if it can be described as* $P = F \setminus \bigcup_{H \in \mathcal{H}} H^\circ$*, where* $\mathcal{H}$ *is a finite collection of pairwise disjoint simple polygons (called holes) and* F *is a simple polygon for which* $F^\circ \supset \bigcup_{H \in \mathcal{H}} H$*.*

The way we define them through the notion of simple polygons makes a trichotomy immediate, just as for simple polygons: Every point in the plane can be either inside, on the boundary, or outside of P.

4.2 Polygon Triangulation

Topologically speaking, a simple polygon is nothing but a disk and thus a very elementary object. But geometrically a simple polygon can be—as if mocking the label we attached

[2] In fact this is not obvious. To see the subtlety, consider a polygon P and the point set $Q := P^\circ \cup (\mathbb{Q}^2 \cap \partial P)$. Note that $\partial P = \partial Q$, so it is possible to recover different sets out from the same boundary. The catch here is that Q is not compact. One can show that we can recover only one *compact* set out from a given boundary, and that is why we need compactness in the definition of a simple polygon.

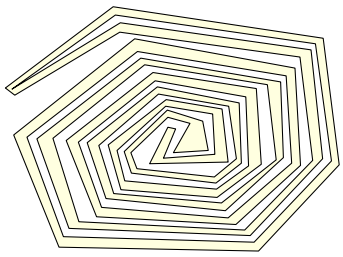


Figure 4.2: *A simple (?) polygon.*

to it—pretty complicated in shape, see Figure 4.2 for an example. While it has a succinct one-dimensional representation as the sequence of the boundary vertices, we often want to work with a more structured representation that retains the two-dimensional shape. For instance, computing the area of a general simple polygon is not so straighforward out of a one-dimensional representation. But if we manage to represent the polygon as a disjoint union of simpler geometric shapes such as triangles, rectangles or trapezoids, then its area simply sums up all the area of the individual shapes, which are easy to compute. This motivates the definition of a triangulation.

Definition 4.4. *A* **triangulation** *of a simple polygon* P *with vertex set* $V(P)$ *is a collection* $\mathcal{T}$ *of triangles, such that*

(1) $P = \bigcup_{T \in \mathcal{T}} T$*;*

(2) $V(P) = \bigcup_{T \in \mathcal{T}} V(T)$*; and*

(3) for every distinct pair $T_1, T_2 \in \mathcal{T}$*, the intersection* $T_1 \cap T_2$ *is either a common vertex, a common edge, or empty.*

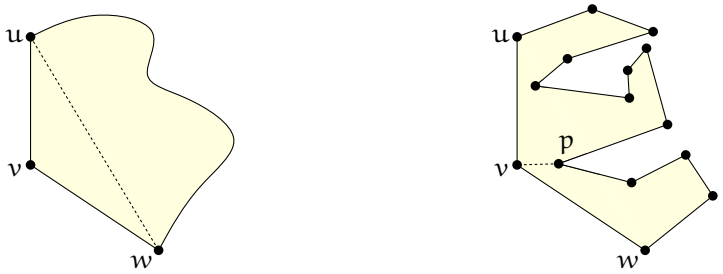
Exercise 4.5. *Show that each condition in Definition 4.4 is necessary in the following sense: Give an example of a non-triangulation that would form a triangulation if the condition was omitted. Is the definition equivalent if (3) is replaced by* $T_1^\circ \cap T_2^\circ = \emptyset$*, for every distinct pair* $T_1, T_2 \in \mathcal{T}$*?*

Beyond area computation, triangulations are incredibly useful in planar geometry. The significance roots in the fact that every simple polygon admits a triangulation.

Theorem 4.6. *Every simple polygon has a triangulation.*

Proof. Let P be a simple polygon on n vertices. We prove the statement by induction on n. For $n = 3$, the polygon P *is* a triangle, which forms a triangulation by itself. For $n > 3$, consider the lexicographically smallest vertex v of P. That is, among all vertices with the smallest x-coordinate we pick the one with the smallest y-coordinate. Let vertices u and w be the predecessor and successor of v along a counterclockwise traversal of ∂P, respectively, so that P is locally to the left of $u \to v \to w$. By choice of v, the walk $u \to v \to w$ forms a strict left turn. Hence $T := uvw$ forms a triangle that lies in P. We distinguish two cases.

Case 1: relint($\overline{uw}$) $\subset P^\circ$ (Figure 4.3a). That is, the segment $\overline{uw}$ except for its two endpoints is fully contained in P°. Hence P can be split into two interior-disjoint simple polygons: the triangle T and a polygon $P' := (P \setminus T) \cup \overline{uw}$ on $n - 1$ vertices. By the inductive hypothesis, the polygon P' admits a triangulation. This triangulation together with T yields a triangulation of P.



(a) Case 1. **(b)** Case 2.

Figure 4.3: *Cases in the proof of Theorem 4.6.*

Case 2: relint($\overline{uw}$) $\not\subset P^\circ$ (Figure 4.3b). Then some point from ∂P must be in T° or on $\overline{uw}$. As ∂P is composed of line segments, some vertex of P must be in T° or on $\overline{uw}$. Among all such vertices select p to maximize the distance to the line uw. Imagine a line through p that is parallel to uw. This line together with uv, vw bounds a triangle that does not contain any part of ∂P in its interior. It follows that relint($\overline{vp}$) $\subset P^\circ$. Therefore, the segment vp splits P into two interior-disjoint polygons P_1 and P_2 on fewer than n vertices each (the vertex u does not appear in one of them, whereas w does not appear in the other). By the inductive hypothesis, both P_1 and P_2 have triangulations, and their union yields a triangulation of P. □

Exercise 4.7. *In the proof of Theorem 4.6, would the argument in Case 2 also work if the point p was chosen to be a vertex of P in T° that minimizes the Euclidean distance to v?*

The configuration from Case 1 above is called an *ear*: three consecutive vertices u, v, w of a simple polygon P such that the relative interior of $\overline{uw}$ lies in P°. In fact, we could have skipped the analysis for Case 2 by referring to the following theorem.

Theorem 4.8 (Meisters [13, 14]). *Every simple polygon with $n \geqslant 4$ vertices has two ears A and B such that $A^\circ \cap B^\circ = \emptyset$.*

But conversely, knowing Theorem 4.6 and the theorem below, we can recover Theorem 4.8 as a direct consequence.

Theorem 4.9. *Every triangulation of a simple polygon P with $n \geqslant 4$ vertices contains at least two triangles that serve as ears in P.*

Exercise 4.10. *Prove Theorem 4.9.*

Exercise 4.11. *Let* P *be a simple polygon with vertices* $v_1, v_2, \ldots, v_n$ *in counterclockwise order, where* v_i *has coordinates* (x_i, y_i)*. Show that the area of* P *is*

$$\frac{1}{2}\sum_{i=1}^{n} x_i y_{i+1} - x_{i+1} y_i,$$

where we agree that $(x_{n+1}, y_{n+1}) = (x_1, y_1)$*.*

A triangulation naturally induces a plane straight-line graph, whose vertices are the polygon vertices and whose edges come from the sides of triangles. The number of edges and triangles (faces) are completely determined by the number of vertices, as the following simple lemma shows.

Lemma 4.12. *Every triangulation of a simple polygon on* $n \geqslant 3$ *vertices consists of* $n-2$ *triangles and* $2n-3$ *edges.*

Proof. We prove the statement by induction on n. For $n = 3$, a triangulation has $n-2 = 1$ triangle and $2n-3 = 3$ edges. Thus, the stamement holds.

In the general case, take any triangulation T of a simple polygon P on $n \geqslant 4$ vertices. Let X be a triangle of T that serves as an ear of P, which exists by Theorem 4.9. Then $T-X$ is a triangulation of a simple polygon on $n-1$ vertices. By the inductive hypothesis, the triangulation $T-X$ consists of $(n-1)-2 = n-3$ triangles and $2(n-1)-3 = 2n-5$ edges. Putting X back adds one more triangle and two more edges. Therefore, the triangulation T consists of $n-3+1 = n-2$ triangles and $2n-5+2 = 2n-3$ edges. □

Tetrahedralizations in $\mathbb{R}^3$. The universal presence of triangulations is something particular about the plane: The natural generalization of Theorem 4.6 to dimension three and higher does not hold. Here we discuss the issue in $\mathbb{R}^3$.

A simple polygon is a topological disk in $\mathbb{R}^2$ that is locally bounded by patches of lines. The corresponding term in $\mathbb{R}^3$ is a *polyhedron*, which can be informally defined via a literal translation of the previous sentence: it is a topologically ball that is locally bounded by patches of planes. A triangle in $\mathbb{R}^2$ corresponds to a tetrahedron in $\mathbb{R}^3$ and a *tetrahedralization* is a nice partition into tetrahedra. Being "nice" means that the union of the tetrahedra covers the object, the vertices of the tetrahedra are vertices of the polyhedron, and any two distinct tetrahedra intersect in either a common triangular face, a common edge, or a common vertex, or not at all.[3]

Unfortunately, there are polyhedra in $\mathbb{R}^3$ that do not admit a tetrahedralization. The following construction is due to Schönhardt [18]. It is based on a triangular prism, that is, two congruent triangles placed in parallel planes, where the corresponding sides of both triangles are connected by a rectangle (Figure 4.4a). We slightly rotate one

[3]These "nice" subdivisions can be defined in an abstract combinatorial setting, where they are called *simplicial complexes*.

triangle within its plane, thus twisting the prism. As a consequence, the rectangles are dented inward along their diagonal in direction of the rotation, and are no longer plane. (Figure 4.4b). The other diagonals of the (former) rectangles—labeled ab', bc', and

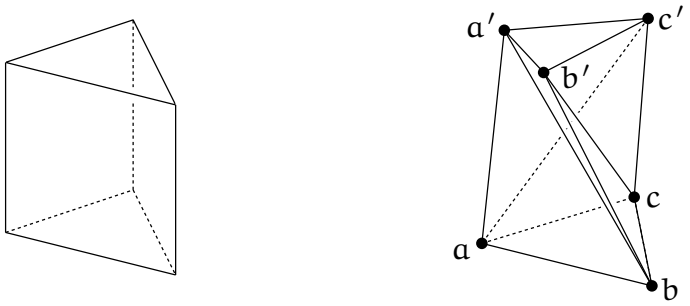


Figure 4.4: *The Schönhardt polyhedron cannot be subdivided into tetrahedra without adding new vertices.*

ca' —are now epigonals, that is, they lie in the exterior of the polyhedron. Since these epigonals are the only missing edges between the vertices, there is no way to add edges to form a tetrahedron for a subdivision. Clearly the polyhedron is not a tetrahedron by itself, and so we conclude that it does not admit a tetrahedralization. Actually, it is NP-complete to decide whether a non-convex polyhedron has a tetrahedralization [15]. However, if adding new vertices—which are called Steiner vertices—is allowed, then a tetrahedralization is possible, both in this example and in general.

Even if a tetrahedralization of a polyhedron exists, there is another significant difference to polygons in $\mathbb{R}^2$. While the number of triangles in a triangulation of a polygon depends only on the number of vertices, the number of tetrahedra in two different tetrahedralization of the same polyhedron may be different. See Figure 4.5 for a simple example of a polyhedron that has tetrahedralizations with two or three tetrahedra. Deciding whether a convex polyhedron has a tetrahedralization with at most a given number of tetrahedra is NP-complete [6].

Exercise 4.13. *Characterize all possible tetrahedralizations of the three-dimensional cube.*

Triangulation algorithms. Knowing that a triangulation exists is nice, but it is even nicer to know that it can also be constructed efficiently.

Exercise 4.14. *Convert Theorem 4.6 into an $O(n^2)$ time algorithm to construct a triangulation for a given simple polygon with n vertices.*

The runtime of this straightforward application of Theorem 4.6 is not optimal. For those who are interested in more efficient algorithms, please refer to Appendices A and C.

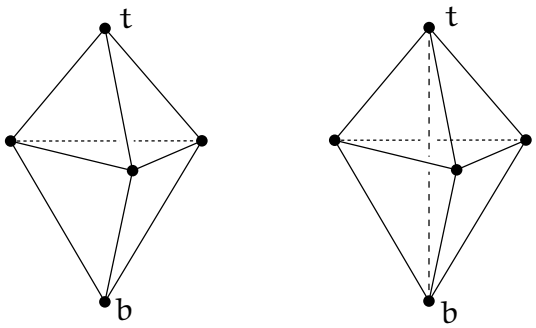


Figure 4.5: *Two tetrahedralizations of the same polyhedron, a triangular bipyramid. The left partition uses two polyhedra; both the top vertex* t *and the bottom vertex* b *belong to only one tetrahedron. The right partition uses three polyhedra that all share the dashed diagonal* bt*.*

The best (in terms of worst-case runtime) algorithm known due to Chazelle [7] computes a triangulation in linear time, but it is very complicated. There is also a somewhat simpler randomized algorithm in expected linear time [4]. We will not cover either of them in the notes. It remains open whether there exists a simple (which is not really well-defined, except that Chazelle's Algorithm does not qualify) deterministic linear time algorithm to triangulate a simple polygon [10].

Polygons with holes. It is interesting to note that the complexity of the triangulation problem changes to $\Theta(n \log n)$, if the polygon may contain holes [5]. This means that (1) there is an algorithm to construct a triangulation for a given simple polygon with holes on n vertices (counting both the vertices on the outer boundary and on the holes' boundaries) in $O(n \log n)$ time; and (2) there is a lower bound of $\Omega(n \log n)$ operations in any model of computation that is subject to the same lower bound for comparison-based sorting. This difference in complexity is a very common pattern: There are many problems that are (sometimes much) harder for simple polygons with holes than for simple polygons. So maybe the term "simple" has some justification, after all...

General triangle covers. What if we drop the "niceness" conditions for triangulations and just want to describe a given simple polygon as a union of triangles? It turns out this is a rather drastic change. For instance, it is unlikely that we can efficiently find an optimal/minimal description of this type: Christ has shown [8] that it is NP-hard to decide whether for a simple polygon P with n vertices and a positive integer k, there exists a set of at most k triangles whose union is P. In fact, the problem is not even known to be in NP, because it is not clear whether the coordinates of solutions can always be encoded succinctly.

4.3 The Art Gallery Problem

In 1973 Victor Klee posed the following question: "How many guards are necessary, and how many are sufficient to patrol the paintings and works of art in an art gallery with n walls?" Thinking in geometric terms, we may model "an art gallery with n walls" as a simple polygon P bounded by n edges, hence also n vertices. And a guard is modeled as a point $g \in P$ where he stands. The edges are opaque and prevent one from seeing what is behind, thus g watches over all the points p for which the line segment $\overline{gp}$ lies completely in P; see Figure 4.6. The task is then to place as least points as possible so that the entire P is being watched.

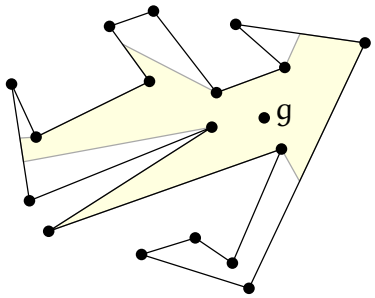


Figure 4.6: *The region that a guard g can observe.*

It is not hard to see that $\lfloor n/3 \rfloor$ guards are necessary in some cases.

Exercise 4.15. *Describe a family $(P_n)_{n\geqslant 3}$ of simple polygons such that P_n has n vertices and requires at least $\lfloor n/3 \rfloor$ guards.*

What is more surprising: $\lfloor n/3 \rfloor$ guards are always sufficient for all P with n vertices. Chvátal [9] was the first to show it, but then Fisk [11] gave a much simpler proof using—as you may have guessed—triangulations. Fisk's proof was considered so beautiful that was selected in "Proofs from THE BOOK" [3]. It is based on the following lemma.

Lemma 4.16. *Every triangulation of a simple polygon is 3-colorable. That is, each vertex can be assigned one of three colors so that adjacent vertices receive different colors.*

Proof. Induction on n. For $n = 3$ the statement is obvious. For $n > 3$, by Theorem 4.9 the triangulation contains an ear uvw. Cutting off the ear creates a triangulation of a polygon on $n-1$ vertices, which by the inductive hypothesis admits a proper 3-coloring. Now whichever two colors the vertices u and w receive in this coloring, there remains a spare color for v. □

Theorem 4.17 (Fisk [11]). *Every simple polygon with n vertices can be guarded using at most $\lfloor n/3 \rfloor$ guards.*

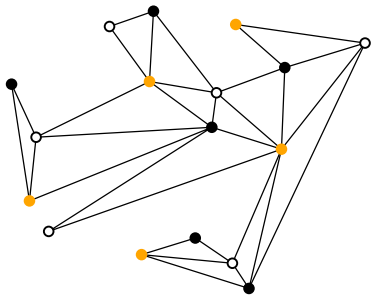


Figure 4.7: *A triangulation of a simple polygon on* 17 *vertices and a 3-coloring of it. The orange vertices form the smallest color class and guard the polygon using* $5 \leqslant \lfloor 17/3 \rfloor$ *guards.*

Proof. Fix a triangulation of the polygon and then a 3-coloring of the vertices, as ensured by Lemma 4.16. Place a guard at each vertex of the smallest color class, which clearly amounts to at most $\lfloor n/3 \rfloor$ vertices. As any point p in the polygon lands in some triangle T and exactly one of T's vertices has the selected color, the point p is watched by that vertex. Hence the whole polygon is guarded. □

4.4 Optimal Guarding

While Exercise 4.15 shows that the bound in Theorem 4.17 is tight in general, it is easy to see that Fisk's method does not necessarily minimize the number of guards. Also, we do not have to restrict ourselves to place the guards at vertices only, but can rather place them anywhere on the boundary or even in the interior of the polygon. In all these cases, we can ask for the minimum number of guards required to guard a given polygon P. These problems have been shown to be NP-hard by Lee and Lin [12] already in the 1980s. Actually, if the guards are not constrained to lie on vertices, it is not even clear whether the corresponding decision problem is in NP. In fact, recent results by Abrahamsen et al. suggest the opposite. In the remainder of this section we will briefly discuss some of these results.

Recall that a decision problem is in NP if for any "yes" instance, one can present a *certificate* that allows polynomial-time verification of the "yes" status.[4] In our context, if we restrict the guards to be on vertices, a natural certificate is the set of vertices where the guards stand. It allows us to verify that the guards indeed watch the entire polygon and that the number of guards is within the specified limit.

However, if we drop the restriction, a natural certificate would be the coordinates of the guards. Since no more than $\lfloor n/3 \rfloor$ guards are required, this seems a reasonable certificate. But what if the number of bits needed to explicitly represent these coordinates are exponential in input size? One might be tempted to think that any reasonable guard can be placed at an intersection point of some lines that are defined by polygon vertices.

[4] And of course, for any "no" instance there should not be any fake certificate that tricks the verification...

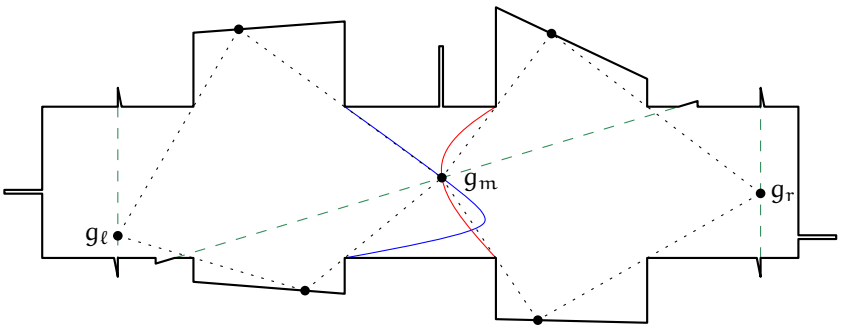


Figure 4.8: *To guard this polygon with three guards, there must be one guard on each of the green dashed segments. The middle guard g_m must be to the left of the blue curve, to the right of the red curve, and on the dashed green line. The intersection point of these three curves has irrational coordinates.*

Alas, in general this is not correct: some guards with irrational coordinates may be required, even if all vertices of P have integral coordinates! This surprising result has been presented in 2017 and we will sketch its main ideas, referring to the paper by Abrahamsen, Adamaszek, and Miltzow [1] for more details and the exact construction.

Consider the polygon shown in Figure 4.8, which consists of a main rectangular region with triangular, rectangular, and trapezoidal regions attached. On the one hand, it can be watched by three guards. On the other hand, we will argue that, if this polygon is guarded with less than four guards, at least one of the guards has an irrational coordinate. The polygon contains three pairs of triangular regions with the following structure. Each pair is connected by a green dashed segment in the figure. This segment contains one edge of each of the two triangles and separates their interiors. Hence, a single guard that sees both of these triangles has to be placed on this separating segment. Further, there is no other point that can guard two of these six triangles. Therefore, if we have only three guards, each of them must be placed on one of these three disjoint segments. The small rectangular regions to the left, top, and bottom outside the main rectangular region further constrain the positions of the guards along these segments.

Let the guards be g_ℓ, g_m, and g_r, as in the figure. The guard g_ℓ cannot see all the points inside the left two trapezoidal regions, and thus g_m has to be placed appropriately. For each position of g_ℓ on its segment, we get a unique rightmost position on which a second guard can be placed to guard the two trapezoids. The union of these points defines an arc that is a segment of a quadratic curve (the roots of a quadratic polynomial). We get an analogous curve for g_r and the two trapezoids attached to the right. By a careful choice of the vertex coordinates, these two curves cross at a point p that also lies on the segment for the guard g_m and has irrational coordinates. It then follows from a detailed argument (see [1]) that p is the only feasible placement of g_m. Let us point out that the choice of the vertex coordinates to achieve this is far from trivial. For example, there can only be a single line defined by two points with rational coordinates that passes through p, and this is the line on which the guard g_m is constrained to lie on.

Exercise 4.18. *Let* P *be a polygon with vertices on the integer grid, and let* g *be a point inside that polygon with at least one irrational coordinate. Show that there can be at most one diagonal of* P *passing through* g*.*

Nevertheless, the sketched construction leads to the following result.

Theorem 4.19 (Abrahamsen et al. [1])**.** *For any* k*, there is a simple polygon* P *with integer vertex coordinates such that* P *can be guarded by* $3k$ *guards, while a guard set having only rational coordinates requires* $4k$ *guards.*

Abrahamsen, Adamaszek, and Miltzow [2] showed recently that the art gallery problem is actually complete in a complexity class called $\exists\mathbb{R}$. The *existential theory of the reals* (see [16, 17] for details) is the set of true sentences of the form $\exists x_1, \ldots, x_n \in \mathbb{R} : \phi(x_1, \ldots, x_n)$ for a quantifier-free Boolean formula ϕ without negation[5] that can use the constants 0 and 1 as well as the operators $+$, $*$, and $<$. For example, $\exists x, y : (x < y) \wedge (x * y < 1 + 1)$ is such a formula. A problem is in the complexity class $\exists\mathbb{R}$ if it allows for a polynomial-time reduction to the problem of deciding such formulas, and it is complete if in addition every problem in $\exists\mathbb{R}$ can be reduced to it in polynomial time.

Exercise 4.20. *Show that* $\mathrm{NP} \subseteq \exists\mathbb{R}$*.*

For the art gallery problem, the result by Abrahamsen et al. [2] implies that the coordinates of an optimal guard set may be doubly-exponential in the input size. This statement does not exclude the possibility of a more concise, implicit way to express the existence of an optimal solution. However, if we found such a way, then this would imply that the art gallery problem is in NP, which, in turn, would imply $\mathrm{NP} = \exists\mathbb{R}$.

Questions

11. *What is a simple polygon/a simple polygon with holes?* Explain the definitions and provide some examples of members and non-members of the respective classes. For a given polygon you should be able to tell which of these classes it belongs to or does not belong to and provide justifications.
12. *What is a closed/open/bounded set in* $\mathbb{R}^d$*? What is the interior/closure of a point set?* Explain the definitions and provide some illustrative examples. For a given set you should be able to argue which of the aforementioned properties it possesses.
13. *What is a triangulation of a simple polygon? Does it always exist?* Explain the definition and provide some illustrative examples. Present the proof of Theorem 4.6 in detail.

[5] If we also allowed negation (and hence also the relations $\leqslant$ and $=$), then it would be possible to express statements like $\exists x : x * x = 1 + 1$ that have only irrational solutions, which is impossible to achieve by using only strict inequalities $<$. Interestingly, however, the complexity class $\exists\mathbb{R}$, when defined in terms of these more expressive formulas, would remain the same, see [17].

14. *How many points are needed to guard a simple polygon?* Present the proofs of Theorem 4.9, Lemma 4.16, and Theorem 4.17 in detail.

15. *How about higher dimensional generalizations? Can every polyhedron in* $\mathbb{R}^3$ *be nicely subdivided into tetrahedra?* Explain Schönhardt's construction.

16. **(This topic was not covered in HS25 and therefore the question will not be asked in the exam.)** *Is there a succinct representation for optimal guard placements?* State Theorem 4.19 and sketch the construction.

References

[1] Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow, Irrational guards are sometimes needed. In *Proc. 33rd Internat. Sympos. Comput. Geom.*, pp. 3:1–3:15, 2017.

[2] Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow, The art gallery problem is $\exists\mathbb{R}$-complete. In *Proc. 50th Annu. ACM Sympos. Theory Comput.*, pp. 65–73, 2018.

[3] Martin Aigner and Günter M. Ziegler, *Proofs from THE BOOK*, Springer, Berlin, 3rd edn., 2003.

[4] Nancy M. Amato, Michael T. Goodrich, and Edgar A. Ramos, A randomized algorithm for triangulating a simple polygon in linear time. *Discrete Comput. Geom.*, 26/2, (2001), 245–265.

[5] Takao Asano, Tetsuo Asano, and Ron Y. Pinter, Polygon triangulation: efficiency and minimality. *J. Algorithms*, 7/2, (1986), 221–231.

[6] Alexander Below, Jesús A. De Loera, and Jürgen Richter-Gebert, The complexity of finding small triangulations of convex 3-polytopes. *J. Algorithms*, 50/2, (2004), 134–167.

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

[8] Tobias Christ, Beyond triangulation: covering polygons with triangles. In *Proc. 12th Algorithms and Data Struct. Sympos.*, vol. 6844 of *Lecture Notes Comput. Sci.*, pp. 231–242, Springer, 2011.

[9] Václav Chvátal, A combinatorial theorem in plane geometry. *J. Combin. Theory Ser. B*, 18/1, (1975), 39–41.

[10] Erik D. Demaine, Joseph S. B. Mitchell, and Joseph O'Rourke, The Open Problems Project, Problem #10. https://topp.openproblem.net/p10.

[11] Steve Fisk, A short proof of Chvátal's watchman theorem. *J. Combin. Theory Ser. B*, 24/3, (1978), 374.

[12] Der-Tsai Lee and Arthur K. Lin, Computational complexity of art gallery problems. *IEEE Trans. Inform. Theory*, 32/2, (1986), 276–282.

[13] Gary H. Meisters, Polygons have ears. *Amer. Math. Monthly*, 82/6, (1975), 648–651.

[14] Gary H. Meisters, Principal vertices, exposed points, and ears. *Amer. Math. Monthly*, 87/4, (1980), 284–285.

[15] Jim Ruppert and Raimund Seidel, On the difficulty of triangulating three-dimensional nonconvex polyhedra. *Discrete Comput. Geom.*, 7, (1992), 227–253.

[16] Marcus Schaefer, Complexity of some geometric and topological problems. In *Proc. 17th Int. Sympos. Graph Drawing (GD 2009)*, pp. 334–344, 2009.

[17] Marcus Schaefer and Daniel Štefankovič, Fixed Points, Nash Equilibria, and the Existential Theory of the Reals. *Theory Comput. Syst.*, 60, (2017), 172–193.

[18] Erich Schönhardt, Über die Zerlegung von Dreieckspolyedern in Tetraeder. *Math. Ann.*, 98, (1928), 309–312.

Chapter 5

Convexity and Convex Hulls

There is an incredible variety of point sets and polygons, but some of them are "nicer" than others in some respect. Look at the two polygons below, for instance:

(a) A convex polygon. **(b)** A non-convex polygon.

Figure 5.1: *Examples of polygons: Which one do you prefer?*

The polygon shown on the left is visually and geometrically much simpler than the one on the right. But let us take a more algorithmic stance, as aesthetics is hard to argue about. When designing algorithms, the left polygon turns out to be much easier to deal with. A particular exploitable property is that one can walk straight between any two points in it without ever leaving it. A polygon, or more generally a point set, with this property is called *convex*.

Definition 5.1. *A point set* $P \subseteq \mathbb{R}^d$ *is* **convex** *if* $\overline{pq} \subseteq P$ *for every pair* $p, q \in P$. *Equivalently, the intersection of* P *with any line is a connected segment.*

The polygon in Figure 5.1b is not convex because the line segment between some pair of points does not completely lie within the polygon. An immediate consequence of the definition is the following:

Observation 5.2. *For any family* $(P_i)_{i \in I}$ *of convex sets, the intersection* $\bigcap_{i \in I} P_i$ *is convex.*

Many problems are comparatively easy to solve for convex sets but very hard in general, and we will encounter some instances of this phenomenon in the course. However, many polygons are not convex, and a discrete point set is never convex (unless it contains one or no point). In such cases it is useful to approximate or encompass a given set P by a convex set $H \supseteq P$. Ideally, H should differ from P as little as possible, so we want it to be the smallest convex set encompassing P:

Definition 5.3. *The* **convex hull** $\operatorname{conv}(P)$ *of a set* $P \subseteq \mathbb{R}^d$ *is the intersection of all convex supersets of* P*.*

At first glance this definition is a bit scary: There can be infinitely many convex supersets, whose intersection might not yield something sensible to work with. But at least, (i) the intersection is well-defined, as the whole space $\mathbb{R}^d$ is certainly a convex superset which takes part in the intersection; (ii) the resulting intersection is convex due to Observation 5.2; and so (iii) the convex hull is the inclusion-wise smallest convex set containing P.

To see what it really *looks like*, we appeal to an algebraic characterization to be introduced in the next section.

5.1 Algebraic Characterizations

In this section we develop algebraic characterizations of convexity. They are indispensable tools in studying convex sets in general dimension d.

Consider $P \subseteq \mathbb{R}^d$. In linear algebra course you have learnt the notion of *linear hull* $\operatorname{lin}(P)$, which is the smallest linear subspace of $\mathbb{R}^d$ that contains P. For instance, the linear hull of $\{(1,2)\} \subset \mathbb{R}^2$ is the line through $(0,0)$ and $(1,2)$; the linear hull of $\{(1,2),(3,4)\}$ is the whole space $\mathbb{R}^2$. One can show that $\operatorname{lin}(P)$ is exactly the set of all *linear combinations* of P:

$$\operatorname{lin}(P) = \left\{ \sum_{i=1}^{n} \lambda_i p_i \;\middle|\; n \in \mathbb{N};\quad p_i \in P,\ \lambda_i \in \mathbb{R} \text{ for } 1 \leqslant i \leqslant n \right\}.$$

A finite set $P = \{p_1, \ldots, p_N\}$ is *linearly independent* if no point in P is a linear combination of the others. Equivalently, the equation $\sum_{i=1}^{N} \lambda_i p_i = 0$ has only the trivial solution $\lambda_1 = \cdots = \lambda_N = 0$. Indeed, if some $\lambda_j \neq 0$ then p_j is a linear combination of the other points with coefficients $\{-\lambda_i/\lambda_j\}_{i \neq j}$. Vice versa, if p_j is a linear combination of the others, this gives us a non-trivial solution to the equation with $\lambda_j = -1$.

In analogue, the *affine hull* of P is the smallest affine subspace[1] of $\mathbb{R}^d$ that contains P. For instance, the affine hull of $\{(1,2),(3,4)\} \subset \mathbb{R}^2$ is the line through $(1,2)$ and $(3,4)$.

[1] An affine space is simply a linear space "shifted" by an offset. That is, adding a constant vector to all vectors in a linear space yields an affine space; conversely, subtracting a fixed vector in the affine space from all vectors sends us back to a linear space. In view of this correspondence, all concepts related to linear space can be translated directly to affine space.

One can show that $\operatorname{aff}(P)$ is exactly the set of all *affine combinations* of P.

$$\operatorname{aff}(P) = \left\{ \sum_{i=1}^{n} \lambda_i p_i \;\middle|\; n \in \mathbb{N};\quad p_i \in P,\ \lambda_i \in \mathbb{R} \text{ for } 1 \leqslant i \leqslant n;\quad \sum_{i=1}^{n} \lambda_i = 1 \right\}.$$

A finite set $P = \{p_1, \ldots, p_N\}$ is *affinely independent* if no point of P is an affine combination of the others. Equivalently, the equation system $\sum_{i=1}^{N} \lambda_i p_i = 0$, $\sum_{i=1}^{N} \lambda_i = 0$ has only the trivial solution $\lambda_1 = \cdots = \lambda_N = 0$. This equivalence can be argued as we did for linear independence. The following proposition is then immediate.

Proposition 5.4. *Let $P \subseteq \mathbb{R}^d$ be a finite point set, and obtain a point set $P' \subseteq \mathbb{R}^{d+1}$ by appending a new coordinate 1 to each point in P. For example, from $P = \{(2,3),(0,4)\} \subseteq \mathbb{R}^2$ we obtain $P' = \{(2,3,1),(0,4,1)\} \subseteq \mathbb{R}^3$. Then P is affinely independent if and only if P' is linearly independent.*

Corollary 5.5. *Any set of $d+2$ points in $\mathbb{R}^d$ is affinely dependent, as any set of $d+2$ points in $\mathbb{R}^{d+1}$ is linearly dependent.*

It turns out that convex hulls can be described algebraically in a very similar way.

Proposition 5.6. *For any $P \subseteq \mathbb{R}^d$ we have*

$$\operatorname{conv}(P) = \left\{ \sum_{i=1}^{n} \lambda_i p_i \;\middle|\; n \in \mathbb{N};\quad p_i \in P,\ \lambda_i \geqslant 0 \text{ for } 1 \leqslant i \leqslant n;\quad \sum_{i=1}^{n} \lambda_i = 1 \right\}.$$

the set of all **convex combinations** *of P.*

To prove it, we need a powerful characterization of convexity.

Proposition 5.7. *A set $P \subseteq \mathbb{R}^d$ is convex if and only if it is closed under convex combination (i.e. any convex combination of P lands in P).*

Proof. "$\Leftarrow$": Convexity only requires closure under convex combination of $n = 2$ points, a special case of $n \in \mathbb{N}$.

"$\Rightarrow$": By induction on n, the number of points taking part in the convex combination. For $n = 1$ the statement is trivial. For $n \geqslant 2$, consider an arbitrary convex combination $p := \sum_{i=1}^{n} \lambda_i p_i$ where $p_i \in P$ and $\lambda_i > 0$ for $1 \leqslant i \leqslant n$, and $\sum_{i=1}^{n} \lambda_i = 1$. Here we assumed $\lambda_i > 0$ because otherwise we can just omit those points whose coefficients are zero. We need to show that $p \in P$.

Let us write

$$p = \left(\sum_{i=1}^{n-1} \lambda_i p_i \right) + \lambda_n p_n = \lambda \left(\sum_{i=1}^{n-1} \frac{\lambda_i}{\lambda} p_i \right) + (1-\lambda) p_n$$

where $\lambda := \sum_{i=1}^{n-1} \lambda_i = 1 - \lambda_n \in [0,1]$. Note that $q := \sum_{i=1}^{n-1} \frac{\lambda_i}{\lambda} p_i$ is a convex combination of $n-1$ points of P, so $q \in P$ by the inductive hypothesis. Consequently $p = \lambda q + (1-\lambda) p_n \in P$ by convexity of P. □

Proof of Proposition 5.6. Denote the set on the right hand side by R.

- $\mathrm{conv}(P) \supseteq R$: Consider an arbitrary convex superset $C \supseteq P$. By Proposition 5.7 ("$\Rightarrow$" direction), any convex combination of C (and in particular of P) is contained in C. Hence $C \supseteq R$, and it follows that $\mathrm{conv}(P) \supseteq R$.
- $\mathrm{conv}(P) \subseteq R$: Clearly R is a superset of P. We will show that R is convex, so it participates in the intersection that defines $\mathrm{conv}(P)$.

  To this end, take any two points $p, q \in R$. We may express $p =: \sum_{i=1}^{n} \lambda_i p_i$ and $q =: \sum_{i=1}^{n} \mu_i p_i$ as convex combinations of a common collection of points $p_1, \ldots, p_n \in P$. This is always possible because we may take the union of their individual collections and set irrelevant coefficients to zero.

  Now for any $\lambda \in [0,1]$ we have $\lambda p + (1-\lambda)q = \sum_{i=1}^{n} (\lambda\lambda_i + (1-\lambda)\mu_i)p_i \in R$, as $\underbrace{\lambda\lambda_i}_{\geqslant 0} + \underbrace{(1-\lambda)}_{\geqslant 0}\underbrace{\mu_i}_{\geqslant 0} \geqslant 0$ for all $1 \leqslant i \leqslant n$, and $\sum_{i=1}^{n} (\lambda\lambda_i + (1-\lambda)\mu_i) = \lambda + (1-\lambda) = 1$.

  Therefore $\overline{pq} \in R$, meaning that R is convex, indeed. □

In a linear space, the notion of a basis plays a fundamental role. It is a minimal description of the linear space of interest. Similarly, we want to describe convex sets using as few entities as possible, which leads to the notion of extreme points.

Definition 5.8. *The convex hull* $\mathrm{conv}(P)$ *of a finite point set* $P \subset \mathbb{R}^d$ *is called a* **convex polytope** *(or a* **convex polygon** *when* $d = 2$*). Every* $p \in P$ *such that* $p \notin \mathrm{conv}(P \setminus \{p\})$ *is called an* **extreme point** *of* P*.*

Exercise 5.9. *Show that a "convex polygon" defined above is really a "simple polygon that is convex".*

Proposition 5.10. *Any convex polytope* $\mathrm{conv}(P)$ *is the convex hull of the extreme points of* P*.*

Proof. Let $P = \{p_1, \ldots, p_n\}$. Assume without loss of generality that its extreme points are $p_1, \ldots, p_k$. We will prove by induction on $i = n, \ldots, k$ that $\mathrm{conv}(P) = \mathrm{conv}\{p_1, \ldots, p_i\}$.

For $i = n$ the statement is trivial. For $k \leqslant i < n$, we have $\mathrm{conv}(P) = \mathrm{conv}\{p_1, \ldots, p_{i+1}\}$ by induction hypothesis. Since the point p_{i+1} is not extreme, it can be expressed as a convex combination $p_{i+1} = \sum_{j=1}^{i} \lambda_j p_j$. Thus any $x \in \mathrm{conv}(P)$ can be expressed as

$$x = \sum_{j=1}^{i+1} \mu_j p_j = \sum_{j=1}^{i} \mu_j p_j + \mu_{i+1} p_{i+1} = \sum_{j=1}^{i} (\mu_j + \mu_{i+1}\lambda_j) p_j.$$

Note that the coefficients are non-negative and sum up to 1, thus $x \in \mathrm{conv}\{p_1, \ldots, p_i\}$. So we conclude $\mathrm{conv}(P) \subseteq \mathrm{conv}\{p_1, \ldots, p_i\}$; the reverse inclusion is trivial. □

5.2 Classic Theorems for Convex Sets

Next we will discuss a few fundamental theorems about convex sets in $\mathbb{R}^d$. The proofs typically employ the algebraic characterization of convexity and some techniques from linear algebra.

Theorem 5.11 (Radon [8]). *Any set $P \subset \mathbb{R}^d$ of $d+2$ points can be partitioned into two disjoint subsets P^+ and P^- such that $\operatorname{conv}(P^+) \cap \operatorname{conv}(P^-) \neq \emptyset$.*

Proof. Let $P = \{p_1, \ldots, p_{d+2}\}$, which by Corollary 5.5 is affinely dependent. Hence $\sum_{i=1}^{d+2} \lambda_i p_i = 0$ and $\sum_{i=1}^{d+2} \lambda_i = 0$ for some $\lambda_1, \ldots, \lambda_{d+2} \in \mathbb{R}$ that are not all zero. In particular, there exist strictly positive and strictly negative coefficients.

Let P^+ be the set of all points p_i for which $\lambda_i > 0$, and denote $P^- := P \setminus P^+$. Then $P^+, P^- \neq \emptyset$ and $\sum_{p_i \in P^+} \lambda_i p_i = \sum_{p_i \in P^-} (-\lambda_i) p_i$. Observe that $\sum_{p_i \in P^+} \lambda_i = \sum_{p_i \in P^-} -\lambda_i =: s > 0$. So with renormalized coefficients

$$\mu_i := \begin{cases} \lambda_i / s & p_i \in P^+ \\ -\lambda_i / s & p_i \in P^- \end{cases} \geqslant 0$$

we have $\sum_{p_i \in P^+} \mu_i p_i = \sum_{p_i \in P^-} \mu_i p_i$, which describes a common point of $\operatorname{conv}(P^+)$ and $\operatorname{conv}(P^-)$. □

Theorem 5.12 (Carathéodory [3]). *For any $P \subset \mathbb{R}^d$ and $q \in \operatorname{conv}(P)$ there exist $k \leqslant d+1$ points $p_1, \ldots, p_k \in P$ such that $q \in \operatorname{conv}(p_1, \ldots, p_k)$.*

Exercise 5.13. *Prove Theorem 5.12.*

Exercise 5.14. *Given a finite point set $P \subset \mathbb{R}^d$, a point $q \in \operatorname{conv}(P)$, and a point $x \in \mathbb{R}^d$. Show that there exists a subset $P' \subseteq P$ of at most d points such that $q \in \operatorname{conv}(P' \cup \{x\})$.*

Theorem 5.15 (Helly). *Consider a collection $\mathcal{C} = \{C_1, \ldots, C_n\}$ of $n \geqslant d+1$ convex subsets of $\mathbb{R}^d$, such that any $d+1$ sets from $\mathcal{C}$ have non-empty intersection. Then $\bigcap_{i=1}^n C_i \neq \emptyset$, i.e. all sets from $\mathcal{C}$ have non-empty intersection.*

Proof. Induction on n. The base case $n = d+1$ holds by assumption. Hence suppose that $n \geqslant d+2$. Define sets $D_i = \bigcap_{j \neq i} C_j$, for $i \in \{1, \ldots, n\}$. As D_i is an intersection of $n-1$ sets from $\mathcal{C}$, by the inductive hypothesis we know that $D_i \neq \emptyset$. Hence we may take an arbitrary point $p_i \in D_i$, for each $i \in \{1, \ldots, n\}$. By Theorem 5.11 the set $\{p_1, \ldots, p_n\}$ can be partitioned into two disjoint subsets P^+ and P^- such that there exists a point $p \in \operatorname{conv}(P^+) \cap \operatorname{conv}(P^-)$. We claim that $p \in \bigcap_{i=1}^n C_i$, which completes the proof.

Fix any $i \in \{1, \ldots, n\}$ and consider C_i. By construction $p_{i'} \in D_{i'} \subseteq C_i$ for all $i' \neq i$. Suppose, say, $p_i \in P^-$, then $P^+ \subseteq \{p_{i'}\}_{i' \neq i} \subseteq C_i$. By convexity of C_i we see $\operatorname{conv}(P^+) \subseteq C_i$ and thus $p \in C_i$. The other case that $p_i \in P^+$ is symmetric. □

There is a nice application of Helly's theorem showing the existence of so-called centerpoints of finite point sets. Basically, a centerpoint is one way to generalize the notion of a median to higher dimensions.

Definition 5.16. *Let $P \subset \mathbb{R}^d$ be a set of n points. A point $p \in \mathbb{R}^d$, not necessarily in P, is a* **centerpoint** *of P if every open halfspace containing more than $\frac{dn}{d+1}$ points of P also contains p.*

Stated differently, every closed halfspace containing a centerpoint also contains at least $\frac{n}{d+1}$ points of P (which is equivalent to containing at least $\left\lceil \frac{n}{d+1} \right\rceil$ points). We have the following result.

Theorem 5.17. *Every set $P \subset \mathbb{R}^d$ of n points has a centerpoint.*

Proof. We may assume that P contains at least $d+1$ points; otherwise, we may embed P in a lower-than-d-dimensional affine subspace and reduce d.

Define a family of subsets of P by

$$\mathcal{A} := \left\{ P \cap H \;\middle|\; H \text{ an open halfspace}, |P \cap H| > \frac{dn}{d+1} \right\}.$$

Since $|P| = n$, the number of subsets in $\mathcal{A} =: \{A_1, \ldots, A_m\}$ is also finite. For each $1 \leqslant i \leqslant m$, we denote $C_i := \operatorname{conv}(A_i)$ which, due to convexity, is contained in the open halfspaces that define A_i.

Suppose there is a point $c \in \bigcap_{i=1}^{m} C_i$, then c is also contained in every open halfspace $H : |P \cap H| > \frac{dn}{d+1}$ and thus is a centerpoint. So it suffices to show the existence of c. To this end, we will prove that any $d+1$ sets in $\mathcal{A}$ have a common point; so do any $d+1$ sets among $C_1, \ldots, C_m$. The claim then follows via Theorem 5.15.

For any $d+1$ sets in $\mathcal{A}$, each set by definition contains more than $\frac{dn}{d+1}$ points of P, so the total number of point occurrences is more than $(d+1)\frac{dn}{d+1} = dn$. Therefore, there exists a point $p \in P$ that occurs more than d times, that is, in all $d+1$ sets. This completes the proof. □

Exercise 5.18. *Show that the number of points in Definition 5.16 is best possible, that is, for every n there is a set of n points in $\mathbb{R}^d$ such that for any $p \in \mathbb{R}^d$ there is an open halfspace containing $\left\lfloor \frac{dn}{d+1} \right\rfloor$ points but not p.*

Theorem 5.19 (Separation Theorem). *Any two compact convex sets $C, D \subset \mathbb{R}^d$ with $C \cap D = \emptyset$ can be separated strictly by a hyperplane, that is, there exists a hyperplane h such that C and D lie in the opposite open halfspaces bounded by h.*

Proof. Consider the distance function $\delta : C \times D \to \mathbb{R}$ with $(c, d) \mapsto \|c - d\|$. Since $C \times D$ is compact and δ is continuous, the function δ attains its minimum at some point $(c_0, d_0) \in C \times D$. Note that $\delta(c_0, d_0) > 0$ because $C \cap D = \emptyset$. Let h be the hyperplane perpendicular to the line segment $\overline{c_0 d_0}$ and passing through its midpoint; see Figure 5.2. We claim that h strictly separates C and D.

To see this, suppose first that that there was a point $c' \in C \cap h$, say. Then by convexity of C we have $\overline{c_0 c'} \subseteq C$. But some point along this segment is closer to d_0 than is c_0, in contradiction to the choice of c_0. Suppose, then, that C has points on both

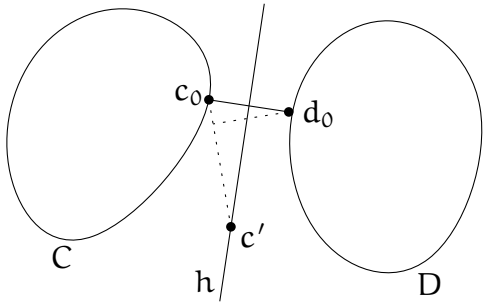


Figure 5.2: *The hyperplane* h *strictly separates the compact convex sets* C *and* D.

sides of h. Then by convexity of C it has also a point on h, but we just saw that it is impossible. The argument for D is symmetric. Therefore, C and D must lie in opposite open halfspaces bounded by h. □

The statement above is wrong for arbitrary (not necessarily compact) convex sets. Only if we allow non-strict separation (i.e. the hyperplane may intersect both sets), can we guarantee such a separation. However, the proof is a bit more involved (cf. Matoušek's book [7], but also check the errata on his webpage).

Exercise 5.20. *Show that the Separation Theorem does not hold in general if not both of the sets are convex.*

Exercise 5.21. *Prove or disprove:*

a) The convex hull of a compact subset of $\mathbb{R}^d$ *is compact.*

b) The convex hull of a closed subset of $\mathbb{R}^d$ *is closed.*

Altogether we obtain various equivalent definitions for the convex hull, summarized in the following theorem.

Theorem 5.22. *For a compact set* $P \subset \mathbb{R}^d$ *we can characterize* $\operatorname{conv}(P)$ *equivalently as one of*

1. *the smallest (w. r. t. set inclusion) convex subset of* $\mathbb{R}^d$ *that contains* P*;*
2. *the set of all convex combinations of points from* P*;*
3. *the set of all convex combinations formed by* $d+1$ *or fewer points from* P*;*
4. *the intersection of all convex supersets of* P*;*
5. *the intersection of all closed halfspaces containing* P*.*

Exercise 5.23. *Prove Theorem 5.22.*

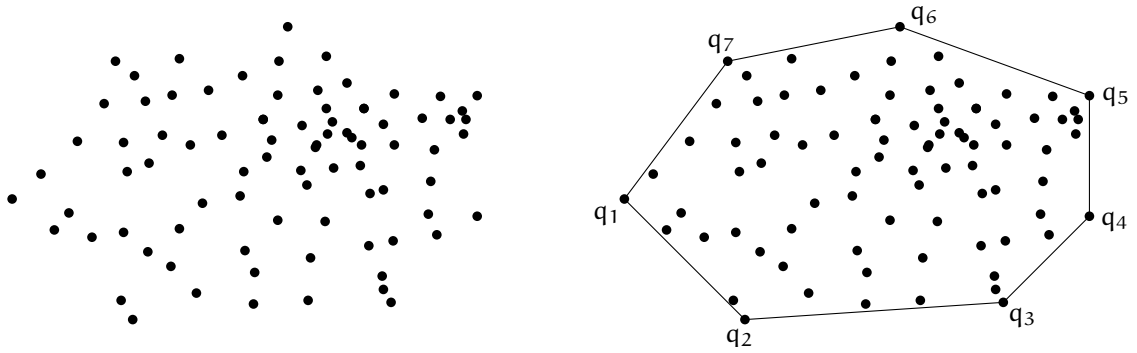
5.3 Planar Convex Hull

Although we know by now what is the convex hull of a point set, it is not yet clear how to construct it algorithmically. As a first step, we have to find a suitable representation for convex hulls. In this section we focus on the problem in $\mathbb{R}^2$, where the convex hull of a finite point set forms a convex polygon. A convex polygon is easy to represent, for instance, as a sequence of its vertices in counterclockwise orientation. In higher dimensions finding a suitable representation for convex polytopes is a much more delicate task.

Problem 5.24 (Convex hull).

Input: $P = \{p_0, \ldots, p_{n-1}\} \subset \mathbb{R}^2$, for some $n \in \mathbb{N}$.

Output: A sequence $(q_0, \ldots, q_{h-1})$ of the vertices of $\mathrm{conv}(P)$, ordered counterclockwise.



(a) Input. **(b)** Output.

Figure 5.3: *Convex Hull of a set of points in* $\mathbb{R}^2$.

Another possible algorithmic formulation of the problem is to ignore the structure of the convex hull and just consider it as a point set.

Problem 5.25 (Extreme points).

Input: $P = \{p_0, \ldots, p_{n-1}\} \subset \mathbb{R}^2$, for some $n \in \mathbb{N}$.

Output: The set of vertices of $\mathrm{conv}(P)$.

Degeneracies. A couple of further clarifications regarding the above problem definitions are in order.

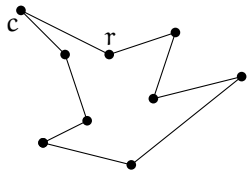
First of all, for efficiency reasons an input is usually specified as a sequence of points. Do we insist that this sequence forms a set or are duplications of points allowed?

What if three points are collinear? Are all of them considered extreme? They are not, according to our definition from above; and that is what we will stick to. But there may be situations where one wants to include these points nevertheless.

By the Separation Theorem, every extreme point p can be separated from the convex hull of the remaining points by a line. If we translate the line so that it passes through p, then every point in P other than p shall strictly lie in one side of it. In $\mathbb{R}^2$ it turns out convenient to work with the following "directed" reformulation.

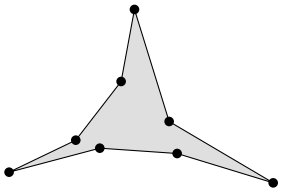
Proposition 5.26. *Let $P \subset \mathbb{R}^2$ be a finite point set. A point $p \in P$ is extreme for P $\iff$ there is a directed line ℓ through p such that $P \setminus \{p\}$ is (strictly) to the left of ℓ.*

The *interior angle* at a vertex v of a polygon P is the angle between the two edges of P incident to v whose corresponding angular domain lies in P°. If this angle is smaller than π, the vertex is called *convex*; if the angle is larger than π, the vertex is called *reflex*. In the polygon depicted on the right, the vertex c is convex whereas the vertex r is reflex.



Exercise 5.27.

A set $S \subset \mathbb{R}^2$ is star-shaped *if there exists a point $c \in S$, such that for every point $p \in S$ the line segment $\overline{cp}$ is contained in S. A simple polygon with exactly three convex vertices is called a pseudotriangle (see the example shown on the right).*



In the following we consider subsets of $\mathbb{R}^2$. Prove or disprove:

a) Every convex vertex of a simple polygon lies on its convex hull.

b) Every star-shaped set is convex.

c) Every convex set is star-shaped.

d) The intersection of two convex sets is convex.

e) The union of two convex sets is convex.

f) The intersection of two star-shaped sets is star-shaped.

g) The intersection of a convex set with a star-shaped set is star-shaped.

h) Every triangle is a pseudotriangle.

i) Every pseudotriangle is star-shaped.

5.4 Trivial algorithms

One can compute the extreme points using Carathéodory's Theorem as follows: Test for every point $p \in P$ whether there are $q, r, s \in P \setminus \{p\}$ such that p is inside the triangle qrs. Runtime $O(n^4)$.

Another option, inspired by the Separation Theorem: Test for every pair $(p, q) \in P^2$ whether all points from $P \setminus \{p, q\}$ are to the left of the directed line $\overrightarrow{pq}$ (or on the line segment $\overline{pq}$). Runtime $O(n^3)$.

Exercise 5.28. *Let $P = (p_0, \ldots, p_{n-1})$ be a sequence of n points in $\mathbb{R}^2$. Someone claims that you can check by means of the following algorithm whether or not P describes the boundary of a convex polygon in counterclockwise order:*

```
bool IsConvex(p_0, ..., p_{n-1}) {
    for i = 0, ..., n − 1:
        if (p_i, p_{(i+1) mod n}, p_{(i+2) mod n}) form a rightturn:
            return false;
    return true;
}
```

Disprove the claim and describe a correct algorithm to solve the problem.

Exercise 5.29. *Let $P \subset \mathbb{R}^2$ be a convex polygon, given as an array $p[0 \ldots n-1]$ of its n vertices in counterclockwise order.*

a) Describe an $O(\log n)$ time algorithm to determine whether a point q lies inside, outside or on the boundary of P.

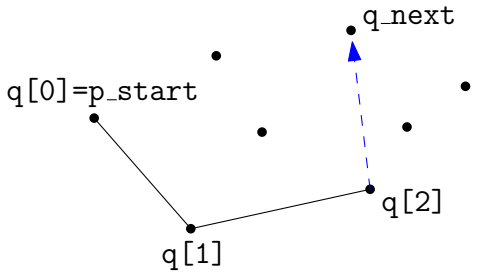
b) Describe an $O(\log n)$ time algorithm to find a (right) tangent to P from a query point q located outside P. That is, find a vertex $p[i]$, such that P is contained in the closed halfplane to the left of the oriented line $qp[i]$.

5.5 Jarvis' Wrap

We are now ready to describe a first simple algorithm to construct the convex hull. It is inspired by Proposition 5.26 and works as follows:

> Find a vertex q_0 of $\text{conv}(P)$ (e.g., the point in P with smallest x-coordinate).
> "Wrap" P starting from q_0, i.e., always find the next vertex q_i of $\text{conv}(P)$ as the rightmost point with respect to the directed line $\overrightarrow{q_{i-2}q_{i-1}}$.

Besides comparing x-coordinates, the only geometric primitive needed is an *orientation* test: For three points $p, q, r \in \mathbb{R}^2$, the predicate $\texttt{rightturn}(p, q, r)$ is true if and only if r is (strictly) to the right of the directed line pq.



Code for Jarvis' Wrap.

`p[0..n-1]` contains a sequence of n points.
`p_start` is the point with smallest x-coordinate.
`q_next` is some *other* point in `p[0..n-1]`.

```
int h = 0;
Point q_now = p_start;
do {
  q[h] = q_now;
  h = h + 1;

  for (int i = 0; i < n; i = i + 1)
    if (rightturn(q_now, q_next, p[i]))
      q_next = p[i];

  q_now = q_next;
  q_next = p_start;
} while (q_now != p_start);
```

`q[0..h-1]` describes a convex polygon bounding the convex hull of `p[0..n-1]`.

Analysis. For every output point the above algorithm spends n rightturn tests, which is $O(nh)$ in total.

Theorem 5.30. [6] *Jarvis' Wrap computes the convex hull of n points in $\mathbb{R}^2$ using $O(nh)$ rightturn tests, where h is the number of hull vertices.*

In the worst case we have $h = n$, that is, $O(n^2)$ rightturn tests. Jarvis' Wrap has a remarkable property called *output sensitivity*: the runtime depends not only on the size of the input but also on the size of the output. For a huge point set whose convex hull consists of a constant number of vertices only, the algorithm constructs the convex hull in optimal linear time. But the worst case performance of Jarvis' Wrap is suboptimal, as we will see soon.

Degeneracies. The algorithm may have to cope with some degeneracies.

- Several points have smallest x-coordinate $\Rightarrow$ sort by lexicographical order:

$$(x_p, y_p) < (x_q, y_q) \iff (x_p < x_q) \vee (x_p = x_q \wedge y_p < y_q) .$$

- Three or more points collinear, so potentially multiple points are rightmost $\Rightarrow$ choose the farthest one.

Predicates. As mentioned above, the Jarvis' Wrap (and most other 2D convex hull algorithms) need the `rightturn` predicate, or more generally, orientation tests. The `rightturn` computation amounts to evaluating a polynomial of degree two, see the exercise below. We therefore say that it has *algebraic degree* two. In contrast, the lexicographic comparison has degree one only. Higher algebraic degree not only means more time-consuming multiplications, but also creates large intermediate results which may lead to overflows as well as challenges for storage and exact computation.

Exercise 5.31. *Prove that for three points* $(x_p, y_p), (x_q, y_q), (x_r, y_r) \in \mathbb{R}^2$*, the sign of the determinant*

$$\begin{vmatrix} 1 & x_p & y_p \\ 1 & x_q & y_q \\ 1 & x_r & y_r \end{vmatrix}$$

determines if r *lies to the right, to the left or on the directed line* $\overrightarrow{pq}$*.*

Exercise 5.32. *The* `InCircle` *predicate: Given four points* $p, q, r, s \in \mathbb{R}^2$*, is* s *located inside the circle defined by* p, q, r*? The goal of this exercise is to derive an algebraic formulation of this predicate as a determinant, similar to the* `rightturn` *predicate in Exercise 5.31. To this end we employ the so-called parabolic lifting map, which will also play a prominent role in later chapters.*

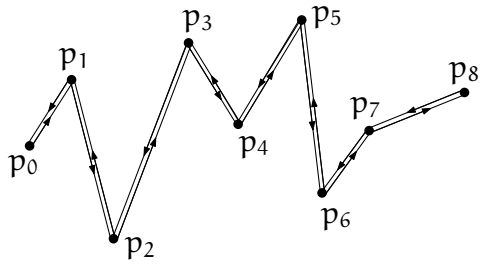
The parabolic lifting map $\ell : \mathbb{R}^2 \to \mathbb{R}^3$ *lifts a point* $p = (x, y) \in \mathbb{R}^2$ *to* $\ell(p) = (x, y, x^2 + y^2) \in \mathbb{R}^3$*. For a circle* $C \subseteq \mathbb{R}^2$ *of positive radius, show that the "lifted circle"* $\ell(C) := \{\ell(p) \mid p \in C\}$ *is contained in a unique plane* $h_C \subseteq \mathbb{R}^3$*. Moreover, show that a point* $p \in \mathbb{R}^2$ *is strictly inside (outside, respectively) of* C *if and only if the lifted point* $\ell(p)$ *is strictly below (above, respectively)* h_C*.*

Use these insights to formulate the `InCircle` *predicate for given points* (x_p, y_p)*,* (x_q, y_q)*,* (x_r, y_r)*,* $(x_s, y_s) \in \mathbb{R}^2$ *as a determinant.*

5.6 Graham Scan (Successive Local Repair)

There exist many algorithms that exhibit a better worst-case runtime than Jarvis' Wrap. Here we discuss only one of them: a particularly elegant and easy-to-implement variant of the so-called *Graham Scan* [5]. This algorithm is referred to as *Successive Local Repair* because it starts with some (possibly non-convex) polygon enclosing all the points and then step-by-step repairs the deficiencies by removing reflex vertices. It goes as follows:

Sort the points lexicographically to obtain a sequence $p_0, \ldots, p_{n-1}$ and build a corresponding circular sequence $p_0, \ldots, p_{n-1}, \ldots, p_0$ that walks around the point set in counterclockwise direction.



$p_0, p_1, \ldots, p_7, p_8, p_7, \ldots, p_1, p_0$

As long as there is a consecutive triple (p, q, r) such that r is to the right of or on the directed line $\overrightarrow{pq}$, remove q from the sequence.

Code for Graham Scan.

`p[0..n-1]` is a lexicographically sorted sequence of $n \geqslant 2$ distinct points.

```
q[0] = p[0];
int h = 0;
// Lower convex hull (left to right):
for (int i = 1; i < n; i = i + 1) {
  while (h>0 && !leftturn(q[h-1], q[h], p[i]))
    h = h - 1;
  h = h + 1;
  q[h] = p[i];
}

// Upper convex hull (right to left):
for (int i = n-2; i >= 0; i = i - 1) {
  while (!leftturn(q[h-1], q[h], p[i]))
    h = h - 1;
  h = h + 1;
  q[h] = p[i];
}
```

`q[0..h-1]` describes a convex polygon bounding the convex hull of `p[0..n-1]`.

Correctness. We argue for the lower convex hull only. The argument for the upper hull is symmetric. A point p is on the lower convex hull of P if there is a rightward directed line g through p such that $P \setminus \{p\}$ is strictly to the left of g. A directed line is *rightward* if it forms an absolute angle of at most π with the positive x-axis. (Compare this statement with the one in Proposition 5.26.)

First, we claim that every point that the algorithm discards does not appear on the lower convex hull. A point q_h is discarded only if there exist points q_{h-1} and p_i with

$q_{h-1} < q_h < p_i$ (lexicographically) so that $q_{h-1}q_hp_i$ does not form a leftturn. Thus, for every rightward directed line g through q_h at least one of q_{h-1} or p_i lies on or to the right of g. It follows that q_h is not on the lower convex hull, as claimed.

Upon finishing the construction of lower hull, in the sequence $q_0, \ldots, q_{h-1}$ every consecutive triple $q_iq_{i+1}q_{i+2}$, for $0 \leqslant i \leqslant h-3$, forms a leftturn with $q_i < q_{i+1} < q_{i+2}$. Thus, for every such triple there exists a rightward directed line g through q_{i+1} such that $P \setminus \{p\}$ is (strictly) to the left of g (for instance, take g to be perpendicular to the angular bisector of $\angle q_{i+2}q_{i+1}q_i$). It follows that every inner point of the sequence $q_0, \ldots, q_{h-1}$ is on the lower convex hull. The extreme points q_0 and q_{h-1} are the lexicographically smallest and largest point of P, respectively, both of which are easily seen to be on the lower convex hull as well. Therefore, $q_0, \ldots, q_{h-1}$ form the lower convex hull of P, which proves the correctness of the algorithm.

Analysis.

Theorem 5.33. *The convex hull of a set $P \subset \mathbb{R}^2$ of n points can be computed using $O(n \log n)$ geometric operations.*

Proof. 1. Sorting and removal of duplicate points: $O(n \log n)$.

2. At the beginning we have a sequence of $2n-1$ points; at the end the sequence consists of h points. Observe that for every "false" `leftturn`, one point is discarded from the sequence for ever. Therefore, we have exactly $2n-h-1$ such tests. In addition there are at most $2n-2$ "true" `leftturn`, as bounded by the number of iterations of the outer `for` loop. Altogether we have at most $4n-h-3$ tests.

In total the algorithm uses $O(n \log n)$ geometric operations. Note that the number of `leftturn` tests is linear only, whereas we need worst-case $\Theta(n \log n)$ lexicographic comparisons which dominates the runtime. □

5.7 Lower Bound

It is not hard to see that the runtime of Graham Scan is asymptotically optimal in the worst-case.

Theorem 5.34. *$\Omega(n \log n)$ geometric operations are needed to construct the convex hull of n points in $\mathbb{R}^2$ (in the algebraic computation tree model).*

Proof. Reduction from the sorting problem, for which $\Omega(n \log n)$ comparisons are needed in the algebraic computation tree model. Given n real numbers $x_1, \ldots, x_n$, we construct a point set $P = \{(x_i, x_i^2) \mid 1 \leqslant i \leqslant n\} \subseteq \mathbb{R}^2$. This construction can be regarded as embedding the numbers into $\mathbb{R}^2$ along the x-axis and then lifting them vertically onto the unit parabola. The counterclockwise order in which the points appear along the lower convex hull of P corresponds to the sorted order of the x_i's. Therefore, if we could construct the convex hull in $o(n \log n)$ time, then we could also sort in $o(n \log n)$ time, a contradiction. □

Clearly this reduction does not work for the Extreme Points problem. But using a reduction from Element Uniqueness (see Section 1.1) instead, one can show that $\Omega(n \log n)$ operations is also needed for computing merely the set of extreme points. This was first shown by Avis [1] for linear computation trees, then by Yao [9] for quadratic computation trees, and finally by Ben-Or [2] for general algebraic computation trees.

5.8 Chan's Algorithm

Given matching upper and lower bounds we may be tempted to consider the algorithmic complexity of the planar convex hull problem settled. However, there are fine-grained structures to be discovered: Recall that the lower bound is a worst case bound. For instance, the Jarvis' Wrap runs in $O(nh)$ time and thus beats the $\Omega(n \log n)$ bound whenever $h = o(\log n)$. The question remains whether one can achieve both output sensitivity and optimal worst case performance at the same time. Indeed, Chan [4] presented an algorithm to achieve this by cleverly combining the best of Jarvis' Wrap and Graham Scan. Let us look at this algorithm in detail. It first guesses an upper bound H for the number of vertices h. Then it proceeds in two phases that are executed one after another.

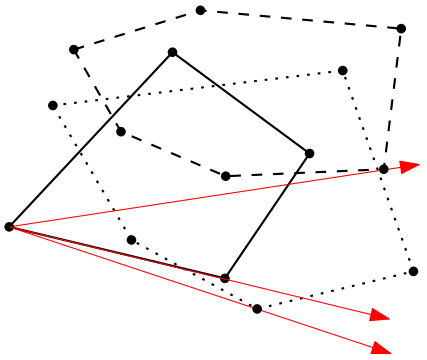
Divide. *Input:* a set $P \subset \mathbb{R}^2$ of n points and a number $H \in \{1, \dots, n\}$.

1. Divide P into $k = \lceil n/H \rceil$ sets $P_1, \dots, P_k$ with $|P_i| \leqslant H$.
2. Construct $\operatorname{conv}(P_i)$ using Graham Scan for $i \in \{1, \dots, k\}$.

Analysis. Step 1 takes $O(n)$ time. Step 2 can be handled in $O(H \log H)$ time for each P_i, hence $O(kH \log H) = O(n \log H)$ time in total.

Conquer. *Output:* the first H vertices of $\operatorname{conv}(P)$ in counterclockwise order.

1. Find the lexicographically smallest point $p_<$ in P.
2. Starting from $p_<$, find the first H vertices of $\operatorname{conv}(P)$ in counterclockwise order by Jarvis' Wrap on the convex polygons $\operatorname{conv}(P_1), \dots, \operatorname{conv}(P_k)$. Specifically, in each wrap step, determine for every i the right tangent t_i to $\operatorname{conv}(P_i)$ from the current vertex (see Exercise 5.29 for definition, and the figure below for illustration). Select our next vertex among the k candidates $t_1, \dots, t_k$ such that it is rightmost with respect to the direction of the last two vertices.



Analysis. Step 1 takes $O(n)$ time. Step 2 consists of at most H wrap steps. Each wrap step needs $O(k \log H)$ time for finding the right tangents (see Exercise 5.29) and $O(k)$ time for selecting the rightmost candidate. That amounts to $O(Hk \log H) = O(n \log H)$ time in total.

Remark. Using a more clever strategy instead of many tangency searches one can handle the conquer phase in $O(n)$ time, see Exercise 5.35 below. However, this is irrelevant as far as the asymptotic runtime is concerned, since already the divide phase takes $O(n \log H)$ time.

Exercise 5.35. *Consider k convex polygons $P_1, \dots P_k$, for some constant $k \in \mathbb{N}$, where each polygon is given as a list of its vertices in counterclockwise order. Show how to construct the convex hull of $P_1 \cup \dots \cup P_k$ in $O(n)$ time, where $n = \sum_{i=1}^{k} n_i$ and n_i is the number of vertices of P_i, for $1 \leqslant i \leqslant k$.*

Searching for h. While the runtime bound for $H \approx h$ is exactly what we were heading for, we still need a means to estimate h closely, whose exact value is unknown in general. Fortunately we can address this problem rather easily, by applying what is called a *doubly exponential search*. It works as follows.

> Try the algorithm from above iteratively with parameter $H = \min\{2^{2^t}, n\}$, for $t = 0, 1 \dots$ until the conquer phase finds all vertices of $\mathrm{conv}(P)$ (i.e., the wrap returns to its starting point).

Analysis: Let 2^{2^s} be the last parameter for which the algorithm is called. Since the previous trial with $H = 2^{2^{s-1}}$ did not find all vertices, we know that $2^{2^{s-1}} < h$, namely $2^{s-1} < \log h$, where h is the actual number of vertices of $\mathrm{conv}(P)$. The total runtime is therefore at most

$$\sum_{t=0}^{s} cn \log 2^{2^t} = cn \sum_{t=0}^{s} 2^t = cn(2^{s+1} - 1) < 4cn \log h = O(n \log h),$$

for some constant $c \in \mathbb{R}$. In summary, we obtain the following theorem.

Theorem 5.36. *The convex hull of a set $P \subset \mathbb{R}^2$ of n points can be computed using $O(n \log h)$ geometric operations, where h is the number of convex hull vertices.*

Questions

17. *How is convexity defined? What is the convex hull of a set in* $\mathbb{R}^d$*?* Give at least three possible definitions and show that they are equivalent.
18. *What is a centerpoint of a finite point set in* $\mathbb{R}^d$*?* State and prove the centerpoint theorem (Theorem 5.17) and the two classic theorems used in its proof (Theorems 5.11 and 5.15).
19. *What does it mean to compute the convex hull of a set of points in* $\mathbb{R}^2$*?* Discuss input and expected output and possible degeneracies.
20. *How can the convex hull of a set of* n *points in* $\mathbb{R}^2$ *be computed efficiently?* Describe and analyze (including proofs) Jarvis' Wrap, Graham Scan, and Chan's Algorithm.
21. *Is there a linear time algorithm to compute the convex hull of* n *points in* $\mathbb{R}^2$*?* Prove the lower bound and define/explain the model in which it holds.
22. *Which geometric predicates are used to compute the convex hull of* n *points in* $\mathbb{R}^2$*?* Explain the two predicates and how to compute them.

References

[1] David Avis, Comments on a lower bound for convex hull determination. *Inform. Process. Lett.*, 11/3, (1980), 126.

[2] Michael Ben-Or, Lower bounds for algebraic computation trees. In *Proc. 15th Annu. ACM Sympos. Theory Comput.*, pp. 80–86, 1983.

[3] Constantin Carathéodory, Über den Variabilitätsbereich der Fourierschen Konstanten von positiven harmonischen Funktionen. *Rendiconto del Circolo Matematico di Palermo*, 32, (1911), 193–217.

[4] Timothy M. Chan, Optimal output-sensitive convex hull algorithms in two and three dimensions. *Discrete Comput. Geom.*, 16/4, (1996), 361–368.

[5] Ronald L. Graham, An efficient algorithm for determining the convex hull of a finite planar set. *Inform. Process. Lett.*, 1/4, (1972), 132–133.

[6] Ray A. Jarvis, On the identification of the convex hull of a finite set of points in the plane. *Inform. Process. Lett.*, 2/1, (1973), 18–21.

[7] Jiří Matoušek, *Lectures on discrete geometry*, Springer, New York, NY, 2002.

[8] Johann Radon, Mengen konvexer Körper, die einen gemeinsamen Punkt enthalten. *Math. Annalen*, 83/1–2, (1921), 113–115.

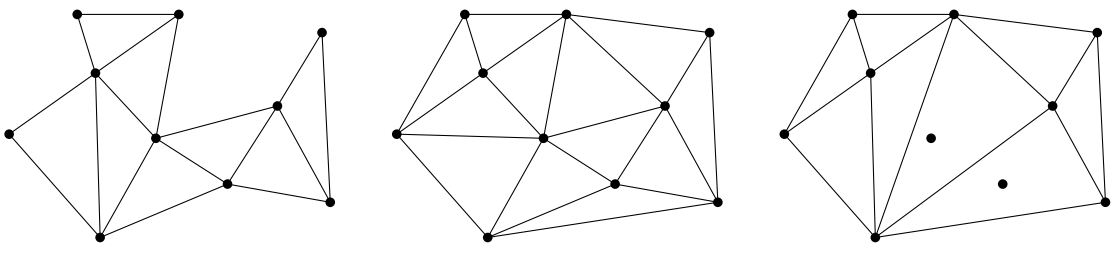
[9] Andrew C. Yao, A lower bound to finding convex hulls. *J. ACM*, 28/4, (1981), 780–787.

Chapter 6

Delaunay Triangulations

In Chapter 4 we have discussed triangulations of simple polygons. A triangulation partitions a polygon into triangles, which allows to easily compute the total area, or to derive a small guarding set, for instance. Another typical application is interpolation: Suppose a function f is defined on the vertices of the polygon P, and we want to extend it "reasonably" and continuously to the entire P. To this end we take a triangulation $\mathcal{T}$. Given any point $p \in P$ we find a triangle $v_1v_2v_3 \in \mathcal{T}$ that contains p, and so $p = \sum_{i=1}^{3} \lambda_i v_i$ can be written as a (unique) convex combination of the three vertices. We may use the same coefficients to define an interpolated value $f(p) := \sum_{i=1}^{3} \lambda_i f(v_i)$.

If triangulations are a useful tool when working with polygons, they might also turn out useful for other geometric objects, such as point sets. But what could be a triangulation of a point set? Polygons have a clearly defined interior, which naturally lends itself to be covered by triangles. A point set does not have an interior, unless... we make one. Here the notion of convex hull comes handy. One way to think of a point set is as a convex polygon (its convex hull) potentially with some little holes (those points in the interior of the hull). A triangulation should then partition the convex hull while *respecting* the points in the interior. Figure 6.1b shows an example. In contrast, Figure 6.1c gives a counterexample: although the triangles do partition the convex hull, some points in the interior are not respected as they are swallowed by large triangles.



(a) Simple polygon triangulation. **(b)** Point set triangulation. **(c)** Not a triangulation.

Figure 6.1: *Examples of (non-)triangulations.*

This interpretation directly leads to the following adaption of Definition 4.4.

Definition 6.1. *A* **triangulation** *of a finite point set* $P \subset \mathbb{R}^2$ *is a collection* $\mathcal{T}$ *of triangles, such that*

(1) $\operatorname{conv}(P) = \bigcup_{T \in \mathcal{T}} T$*;*

(2) $P = \bigcup_{T \in \mathcal{T}} V(T)$*; and*

(3) for every distinct pair $T, T' \in \mathcal{T}$*, the intersection* $T \cap T'$ *is either a common vertex, or a common edge, or empty.*

Just as for polygons, triangulations are universally available for point sets, meaning that (almost) every point set admits at least one.

Proposition 6.2. *Every set* $P \subseteq \mathbb{R}^2$ *of* $n \geqslant 3$ *points has a triangulation, unless all points in* P *are collinear.*

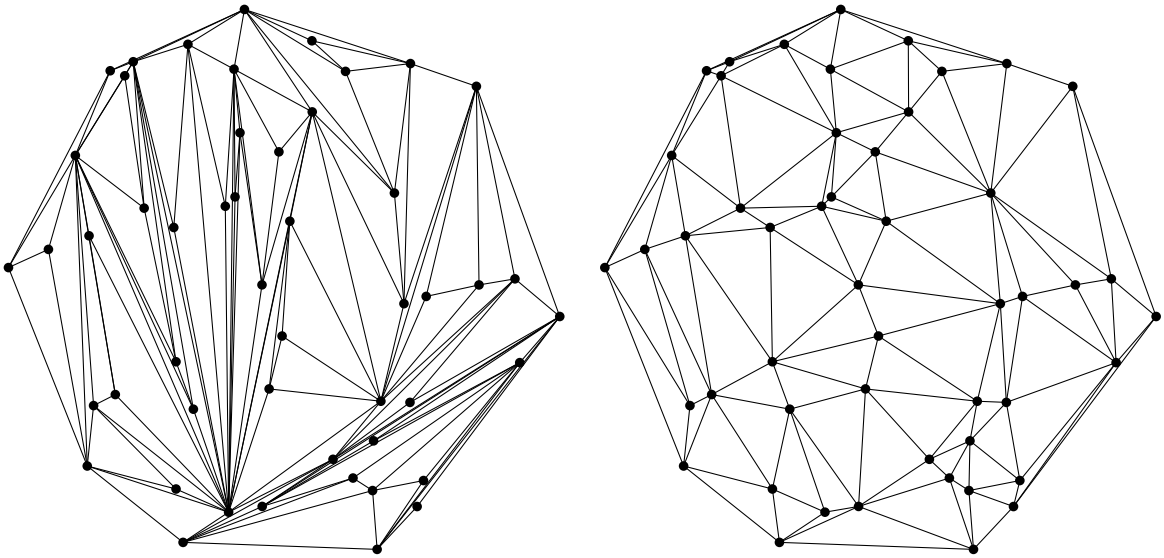
Proof. In order to construct a triangulation for P, consider the lexicographically sorted sequence $p_1, \ldots, p_n$ of points in P. Let m be minimal such that $p_1, \ldots, p_m$ are not collinear. We triangulate $p_1, \ldots, p_m$ by connecting p_m to all of $p_1, \ldots, p_{m-1}$ (which *are* on a common line), see Figure 6.2a.

(a) Getting started. **(b)** Adding a point.

Figure 6.2: *Constructing the scan triangulation of* P*.*

Then we add $p_{m+1}, \ldots, p_n$ one by one. Let us inductively assume that we had built a triangulation of $P_{i-1} := \{p_1, \ldots, p_{i-1}\}$, and we are about to add p_i. Note that p_i is not contained in $C_{i-1} := \operatorname{conv}(P_{i-1})$ because of the lexicographic order. We connect it with all "visible" vertices of C_{i-1}; that is, every vertex v of C_{i-1} for which $\overline{p_i v} \cap C_{i-1} = \{v\}$. Among these vertices are two tangent points from p_i to C_{i-1}, and the vertices in between are exactly the visible ones. After adding these connections, we have covered $C_i \setminus C_{i-1}$ by several new disjoint triangles, so overall we obtain a triangulation of P_i. □

The triangulation constructed in Proposition 6.2 is called a *scan triangulation*. Figure 6.3a shows a larger example. It is usually "ugly", though, as the lexicographic order tends to produce many long and skinny triangles. This is not only an aesthetic deficit, but also a practical concern in the context of interpolation, for example, since long and skinny triangles imply a less local interpolation. In contrast, the *Delaunay triangulation* of the same point set (Figure 6.3b) looks much nicer, and we will discuss in the next section how to get this triangulation.



(a) Scan triangulation. **(b)** Delaunay triangulation.

Figure 6.3: *Two triangulations of the same set of* 50 *points.*

Exercise 6.3. *Describe how to implement the scan triangulation in* $O(n \log n)$ *time for a set of* n *points in* $\mathbb{R}^2$*.*

On another note, if you look closely into the Graham Scan algorithm for planar convex hulls in Chapter 5, then you will realize that we also could have used it to prove Proposition 6.2. Whenever a point q is discarded during Graham Scan due to a right turn $p \to q \to r$, we add the triangle pqr to fill the space. Eventually this leads to a triangulation of the point set.

Every triangulation of P induces a plane straight-line graph $G = (P, E)$, where the edges are the sides of the triangles. As shown by the lemma below (cf. Corollary 2.5), the counts of edges and triangles are determined by P.

Lemma 6.4. *Any triangulation of a set* $P \subset \mathbb{R}^2$ *of* n *points has exactly* $3n-h-3$ *edges and* $2n-h-1$ *faces in its induced graph, where* $h := |P \cap \partial\mathrm{conv}(P)|$ *is the number of points on the outer cycle.*

Proof. Consider the graph induced by an arbitrary triangulation of P. Denote by E the set of edges and by F the set of faces. We count the number of edge-face incidences in two ways. Denote $X = \{(e, f) \in E \times F : e \text{ bounds } f\}$.

On the one hand, every edge is incident to exactly two faces and therefore $|X| = 2|E|$. On the other hand, every inner face is a triangle and the outer face is bounded by h edges, therefore $|X| = 3(|F| - 1) + h$. Together we obtain $3|F| = 2|E| - h + 3$. Combining with Euler's formula $n - |E| + |F| = 2$ we can solve for $|E| = 3n - h - 3$ and $|F| = 2n - h - 1$. □

In graph theory, the term "triangulation" is sometimes used as a synonym for "maximal planar graph". But geometric triangulations are somewhat weaker: They are not

maximal in the sense that no abstract edge can be added; rather, only in the sense that no *straight-line* edge can be added without sacrificing planarity.

Corollary 6.5. *A triangulation of a set* $P \subset \mathbb{R}^2$ *of* n *points is maximal planar, if and only if* $\mathrm{conv}(P)$ *is a triangle.*

Proof. Combine Corollary 2.5 and Lemma 6.4. □

Exercise 6.6. *Find for every* $n \geqslant 3$ *a simple polygon* P *with* n *vertices such that* P *has exactly one triangulation.* P *should be in general position, meaning that no three vertices are collinear.*

Exercise 6.7. *Show that every set of* $n \geqslant 5$ *points in general position (no three points are collinear) has at least two different triangulations.*
Hint: Show first that every set of five points in general position contains a convex 4*-hole, that is, a subset of four points that span a convex quadrilateral that does not contain the fifth point.*

6.1 The Empty Circle Property

We will now move on to study the ominous and supposedly nice Delaunay triangulations mentioned above. They are defined in terms of an "empty circumcircle" property. The *circumcircle* of a triangle is the unique circle passing through the three vertices of the triangle, see Figure 6.4. Observe that long and skinny triangles usually have unproportionally large circumcircles, which tend to (though not always) enclose other points inside. A Delaunay triangulation forbids such enclosure, in hope of avoiding skinny triangles as much as possible.

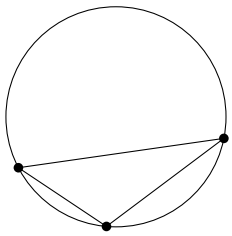


Figure 6.4: *Circumcircle of a triangle.*

Definition 6.8. *A triangulation* $\mathcal{T}$ *of a finite point set* $P \subset \mathbb{R}^2$ *is a* **Delaunay triangulation**, *if the circumcircle of every triangle* $T \in \mathcal{T}$ *is* **empty**, *that is, the circle does not enclose any point from* P *strictly inside.*

Consider the example depicted in Figure 6.5. It shows a Delaunay triangulation of a set of six points: The circumcircles of all five triangles are empty (we also say that the

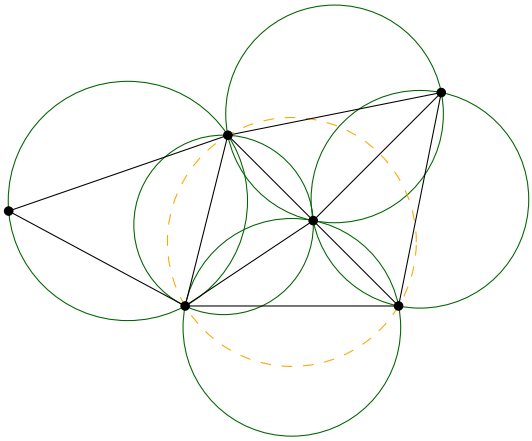
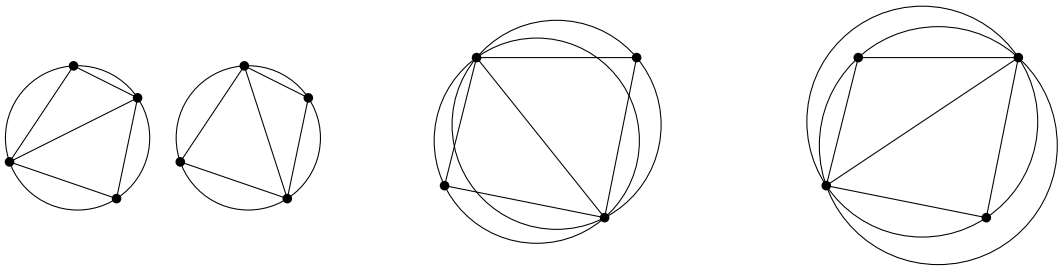


Figure 6.5: *All triangles satisfy the empty circle property.*

triangles satisfy the empty circle property). The dashed circle is not empty, but that is fine since it is not a circumcircle of any triangle.

It is instructive to look at the toy example where four points are arranged in convex position. Obviously, there are two possible triangulations. If the four points happen to lie on the same cycle C, the circumcircle of any three points is exactly C, which is empty, so both triangulations shall be Delaunay (see Figure 6.6a). But in general position, i.e. when the four points are not cocircular, only one triangulation is Delaunay (see Figures 6.6b and 6.6c). This case is formalized in the proposition below, whose proof technique will show up frequently in this chapter.



(a) Two Delaunay triangulations. **(b)** Delaunay triangulation. **(c)** Non-Delaunay triangulation.

Figure 6.6: *Triangulations of four points in convex position.*

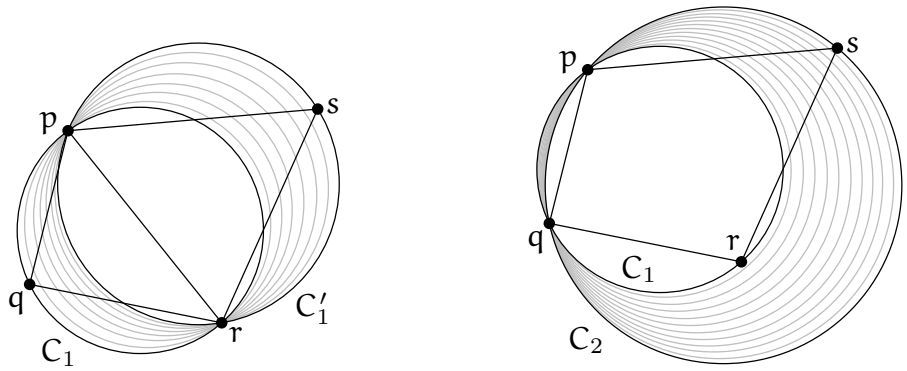
Proposition 6.9. *Given a set $P \subset \mathbb{R}^2$ of four points that are in convex position but not cocircular. Then P has exactly one Delaunay triangulation.*

Proof. Consider a set of four points $P = \{p, q, r, s\}$ arranged counterclockwise in convex position. There are only two possible triangulations: $\mathcal{T}_1 := \{prq, prs\}$ and $\mathcal{T}_2 := \{qsp, qsr\}$.

Let C_1 be the circumcircle of triangle $prq \in \mathcal{T}_1$, and C_1' be the circumcircle of the other triangle $prs \in \mathcal{T}_1$. Since the four points are not cocircular, we have only two cases:

s **is strictly outside** C_1. First we argue that q must be strictly outside C_1'. Imagine the process of continuously moving from C_1 to C_1' while keeping p, r on the cycle (Figure 6.7a). More precisely, we move the center towards s along the perpendicular bisector of pr. At some point the cycle hits s and becomes C_1' and the point q must be "left behind". Thus q is strictly outside C_1', indeed.

As both C_1 and C_1' are empty, $\mathcal{T}_1$ is a Delaunay triangulation. Next we argue that $\mathcal{T}_2$ is not Delaunay. Consider the continuous motion from C_1 to C_2, the circumcircle of $qsp \in \mathcal{T}_2$, while keeping p, q intact (Figure 6.7b). The point r is on C_1 and remains within the circle all the way up to C_2. This means C_2 is not empty, thus $\mathcal{T}_2$ is not Delaunay.



(a) Going from C_1 to C_1' in $\mathcal{C}_1$. **(b)** Going from C_1 to C_2 in $\mathcal{C}_2$.

Figure 6.7: *Circumcircles and containment for triangulations of four points.*

s **is strictly inside** C_1. The case is symmetric: just shift the roles of $pqrs$ to $qrsp$.

□

Exercise 6.10. *Prove or disprove that every minimum weight triangulation (that is, a triangulation for which the sum of edge lengths is minimum) is a Delaunay triangulation.*

6.2 The Lawson Flip algorithm

It is not clear yet that every point set P of n points actually has a Delaunay triangulation (given that not all points are collinear). In this and the next two sections, we will prove that this is the case, via the *Lawson flip algorithm*:

1. Compute some triangulation of P (for example, the scan triangulation).
2. While there exist two adjacent triangles Δ, Δ' such that the circumcircle of Δ encloses a vertex of Δ' (see Figure 6.6c; observe that the four vertices must be in

convex position), replace them by the other pair of adjacent triangles (Figure 6.6b). In other words, we flip the diagonal of the convex quadrilateral.

We call the replacement operation in the second step a *(Lawson) flip*.

Theorem 6.11. *Let $P \subseteq \mathbb{R}^2$ be a set of n points, equipped with some triangulation $\mathcal{T}$. The Lawson flip algorithm terminates after at most $\binom{n}{2} = O(n^2)$ flips, and the resulting triangulation $\mathcal{D}$ is a Delaunay triangulation of P.*

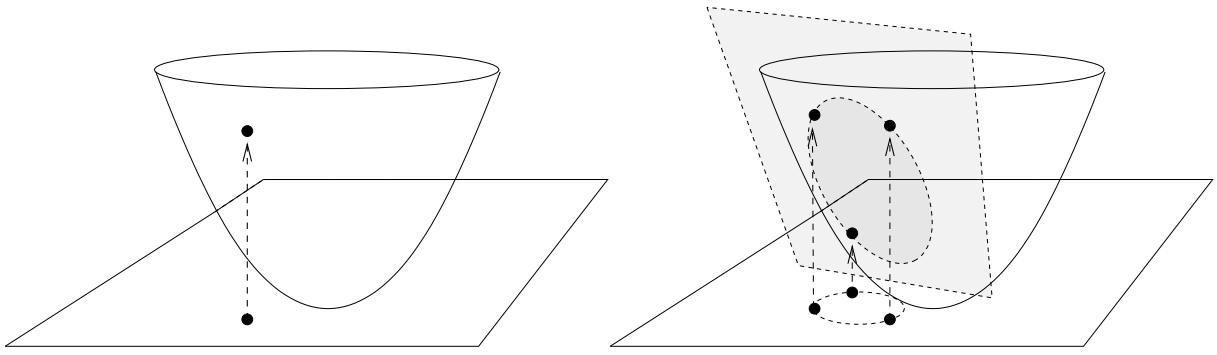
We will prove Theorem 6.11 in two steps: In Section 6.3 we show that the program described above always terminates and, therefore, is an algorithm indeed. (If you think about it a little, it is not obvious whether the algorithm would loop indefinitely.) Then in Section 6.4 we show that the algorithm does produce a Delaunay triangulation upon termination.

6.3 Termination of the Lawson Flip Algorithm

For the termination proof, we make use of the (parabolic) *lifting map* ℓ:

$$p = (x, y) \in \mathbb{R}^2 \quad \mapsto \quad \ell(p) = (x, y, x^2 + y^2) \in \mathbb{R}^3.$$

Geometrically, ℓ "lifts" the point vertically up until hitting the *unit paraboloid* $\{(x, y, z) \mid z = x^2 + y^2\} \subseteq \mathbb{R}^3$, see Figure 6.8a.



(a) The lifting map.

(b) Points on/inside/outside a circle are lifted to points on/below/above a plane.

Figure 6.8: *The lifting map: circles map to planes.*

Recall the following important property of the lifting map that we proved in Exercise 5.32. It is illustrated in Figure 6.8b.

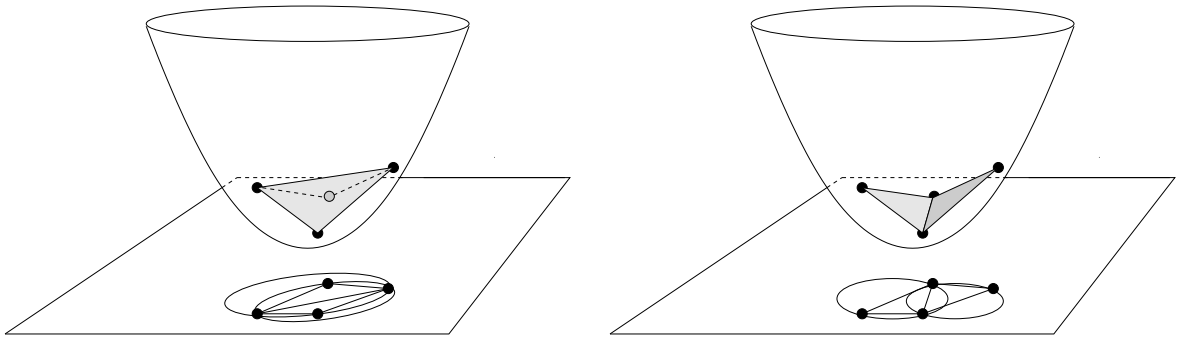
Lemma 6.12. *Let $C \subseteq \mathbb{R}^2$ be a circle of positive radius. The "lifted circle" $\ell(C) := \{\ell(p) \mid p \in C\}$ is contained in a unique plane $h_C \subseteq \mathbb{R}^3$. Moreover, a point $p \in \mathbb{R}^2$ is strictly inside (respectively outside) C if and only if the lifted point $\ell(p)$ is strictly below (respectively above) h_C.*

Using the lifting map, we can interpret triangulations in the 3D space. For each triangle $\Delta = pqr$, we define its "lifted version" as $\ell(\Delta) := \text{conv}\{\ell(p), \ell(q), \ell(r)\}$, which is a triangle hanging in the space with Δ being its "shadow". This way, the triangulation is lifted to a piecewise linear surface in the space consisting of patches of triangles.

Consider a Lawson flip on adjacent triangles $\Delta = pqr$ and $\Delta' = pqs$, where p, q, r, s are in convex position. Let C and C' be their respective circumcircles. By the condition of a flip, C encloses s, and similarly C' encloses r. In the lifted picture, Lemma 6.12 states that $\ell(s)$ is strictly below the plane that contains $\ell(\Delta)$, and similarly $\ell(r)$ is strictly below the plane that contains $\ell(\Delta')$. In other words, the triangles $\ell(\Delta)$ and $\ell(\Delta)$ form a mountain that protrudes upward; see Figure 6.9a.

After the flip, the two triangles are replaced by prs and qrs. In the lifted picture, triangles form a valley that protrudes downward by a similar reasoning; see Figure 6.9b.

More pictorially, imagine an opaque tetrahedron $\text{conv}\{\ell(p), \ell(q), \ell(r), \ell(s)\}$ in the space. When you look at it from the top, you see two faces corresponding to the two triangles before the flip; and when you look from the bottom, you see the other two faces corresponding to the two triangles after the flip. (You cannot see three faces from either direction, since p, q, r, s are in convex position.) Hence a Lawson flip can be interpreted as replacing the two top faces by the two bottom faces of the tetrahedron.



(a) Before the flip: the top two triangles of the tetrahedron and the corresponding non-Delaunay triangulation in the plane.

(b) After the flip: the bottom two triangles of the tetrahedron and the corresponding Delaunay triangulation in the plane.

Figure 6.9: *Lawson flip: the height of the surface of lifted triangles decreases.*

It follows that the 3D surface can only grow strictly downward pointwise. In particular, once an edge pq has been flipped, it becomes strictly above the surface thereafter and thus can never show up again. Since n points can span at most $\binom{n}{2}$ edges, the bound on the number of flips follows.

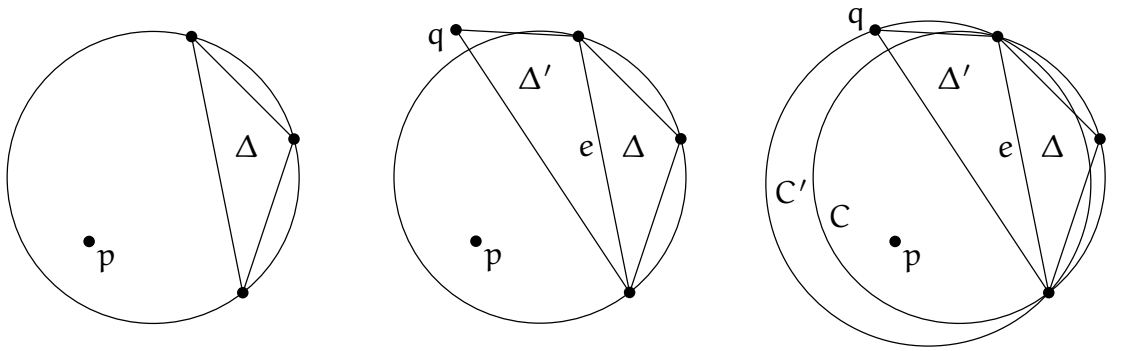
6.4 Correctness of the Lawson Flip Algorithm

The triangulation of P that we get upon termination of the Lawson flip algorithm is "locally Delaunay": it checks the empty circle property for adjacent triangles only. But

in fact it is "globally Delaunay", too.

Proposition 6.13. *The triangulation $\mathcal{D}$ that results from the Lawson flip algorithm is a Delaunay triangulation.*

Proof. Suppose for contradiction that there is some triangle $\Delta \in \mathcal{D}$ and some point $p \in P$ strictly inside the circumcircle C of Δ. Among all such pairs (Δ, p), we choose one that minimizes the distance from p to Δ. Note that this distance is positive by definition of a triangulation. We assume for now that the point on Δ closest to p lies on the relative interior of some edge e of Δ; we will deal with the other case later. The situation is as depicted in Figure 6.10a.



(a) A point p inside the circumcircle C of a triangle Δ.

(b) The edge e of Δ closest to p and the second triangle Δ' incident to e.

(c) The circumcircle C′ of Δ' also contains p, and p is closer to Δ' than to Δ.

Figure 6.10: *Correctness of the Lawson flip algorithm.*

There must be another triangle Δ' in $\mathcal{D}$ that is incident to the edge e. By the local Delaunay property of $\mathcal{D}$, the third vertex q of Δ' is on or outside of C, see Figure 6.10b. But then the circumcircle C′ of Δ' contains the whole portion of C on p's side of e, hence it also contains p; moreover, p is closer to Δ' than to Δ (Figure 6.10c). But this is a contradiction to our choice of Δ and p. Hence there was no (Δ, p), and $\mathcal{D}$ is a Delaunay triangulation.

Consider now the special case where the point on Δ closest to p happens to be a vertex v of the triangle Δ. In this case, we need some additional care when choosing Δ. Among all triangles that use v as a vertex and that have p in their circumcircle, we choose our actual Δ as the one for which the edge e (as before, this is the edge that faces p in the circumcircle) and the segment $\overline{pv}$ form an angle closest 90 degrees.

From here, the proof proceeds as in the first case. We construct a new triangle Δ' that also uses the edge e and that also contains the point p in its circumcircle. The difference is that we do not necessarily get the same type of contradiction because the point on Δ' closest to p might still be v. If that is the case, however, the angle between the edge e' (this is the edge that faces p in the circumcircle of Δ') and the segment $\overline{pv}$

has will be closer to $90°$ compared to e. This now stands in contradiction to our more careful choice of the triangle Δ, which finishes the proof. □

Exercise 6.14. *The Euclidean minimum spanning tree (EMST) of a finite point set $P \subset \mathbb{R}^2$ is a spanning tree for which the sum of the edge lengths is minimum (among all spanning trees of P). Show:*

(a) Every EMST of P is a plane graph.

(b) Every EMST of P contains a closest pair, that is, an edge between two points $p, q \in P$ that have minimum distance to each other among all point pairs in $\binom{P}{2}$.

(c) Every Delaunay Triangulation of P contains an EMST of P.

Exercise 6.15. *(a) Show that for any two triangulations T_1 and T_2 on a point set P, it is possible to transform T_1 into T_2 using $O(n^2)$ edge flips.*

(b) Let $D = P \cup Q$ where $P = \{p_1, p_2, \ldots, p_n\}$ and $Q = \{q_1, q_2, \ldots, q_n\}$ each forms a slightly bent arc, facing against each other. For any line q_iq_j the set P is on its left; and symmetrically, for any line p_ip_j the set Q is on its right. Show that there are two triangulations T_1 and T_2 on D such that at least $\Omega(n^2)$ edge flips are needed to transform T_1 into T_2.

(c) Show that D can be constructed in such a way that one of the triangulations from (b), say, T_1 is a Delaunay triangulation.

6.5 The Delaunay Graph

Despite the fact that a point set may have more than one Delaunay triangulation, there are certain edges that are present in every Delaunay triangulation, for instance, the edges of the convex hull.

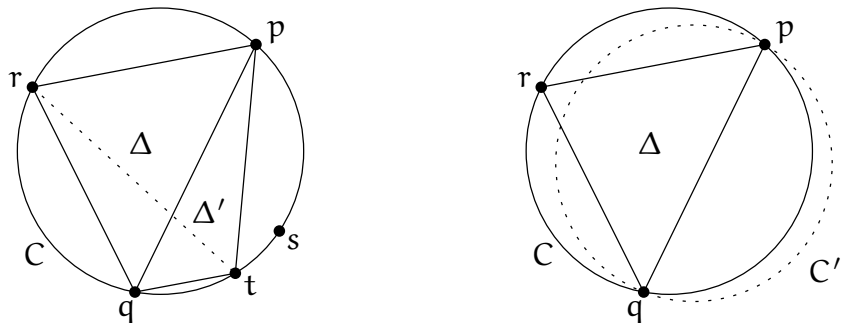
Definition 6.16. *The* **Delaunay graph** *of $P \subseteq \mathbb{R}^2$ consists of all line segments $\overline{pq}$, for $p, q \in P$, that are contained in every Delaunay triangulation of P.*

The following characterizes the edges of the Delaunay graph.

Lemma 6.17. *The segment $\overline{pq}$, for $p, q \in P$, is in the Delaunay graph of P if and only if there exists a circle through p and q for which all other points of P are strictly outside.*

Proof. "⇒": Let pq be an edge in the Delaunay graph of P, and let $\mathcal{D}$ be a Delaunay triangulation of P. Then there exists a triangle $\Delta = pqr$ in $\mathcal{D}$, whose circumcircle C does not enclose any point from P strictly inside.

If there is a point s on C such that $\overline{rs}$ intersects $\overline{pq}$, then let $\Delta' = pqt \neq \Delta$ denote the other triangle in $\mathcal{D}$ that is incident to pq (Figure 6.11a). Note that t must be on C, for



(a) Another point $s \in \partial C$. **(b)** Moving C away from s.

Figure 6.11: *Characterization of edges in the Delaunay graph (I).*

otherwise the circumcircle of Δ' would enclose s. Now flipping the edge pq to rt yields another Delaunay triangulation that does not contain the edge pq, in contradiction to pq being an edge in the Delaunay graph. Therefore, there is no such point s.

Otherwise we can slightly change the circle C by moving away from r while keeping p and q on the circle. As P is a finite point set, we can do such a modification without catching another point from P with the circle. In this way we obtain a circle C' through p and q such that all other points from P are strictly outside C' (Figure 6.11b).

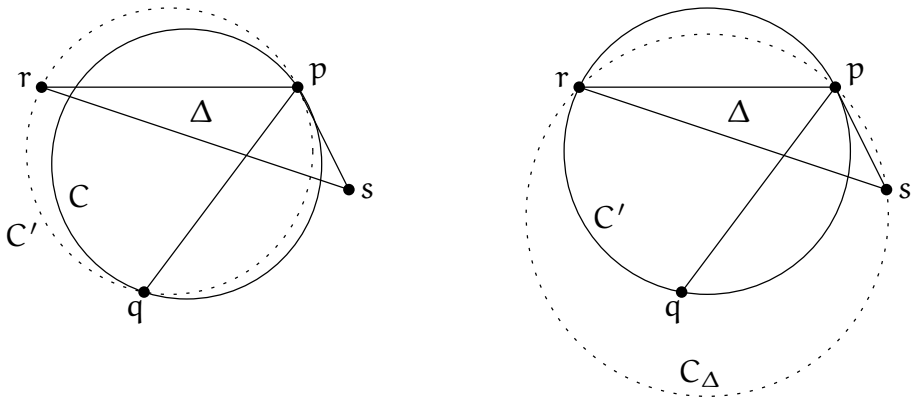
"$\Leftarrow$": Let $\mathcal{D}$ be a Delaunay triangulation of P. If $\overline{pq}$ is not an edge of $\mathcal{D}$, there must be another edge of $\mathcal{D}$ that crosses $\overline{pq}$ (otherwise, we could add $\overline{pq}$ to $\mathcal{D}$ and still have a plane graph, a contradiction to $\mathcal{D}$ being a triangulation of P). Let rs denote the first edge of $\mathcal{D}$ that the directed line segment $\overrightarrow{pq}$ intersects.

Consider the triangle Δ of $\mathcal{D}$ that is incident to rs on the side that faces p (given that $\overline{rs}$ intersects $\overline{pq}$ this is a well defined direction). By the choice of rs neither of the other two edges of Δ intersects $\overline{pq}$, and $p \notin \Delta^\circ$ because Δ is part of a triangulation of P. The only remaining option is that p is a vertex of $\Delta = prs$. As Δ is part of a Delaunay triangulation, its circumcircle C_Δ needs to be empty.

Consider now a circle C through p and q for which all other points are strictly outside. Fixing p and q, we expand C towards r to eventually obtain the circle C' through p, q, r (Figure 6.12a). Recall that r and s are on different sides of the line through p and q. Therefore, s lies strictly outside C'. Next fix p and r and expand C' towards s to eventually obtain the circle C_Δ through p, r, s (Figure 6.12b). Recall that s and q are on the same side of the line through p and r. Therefore, $q \in C_\Delta$, which is in contradiction to C_Δ being empty. It follows that there is no Delaunay triangulation of P that does not contain the edge pq. □

The Delaunay graph is useful to prove uniqueness of the Delaunay triangulation in case of general position.

Corollary 6.18. *Let $P \subset \mathbb{R}^2$ be a finite set of points in general position (no four points of P are cocircular). Then P has a unique Delaunay triangulation.*



(a) Expanding C towards r. **(b)** Expanding C' towards s.

Figure 6.12: *Characterization of edges in the Delaunay graph (II).*

Exercise 6.19. *Prove Corollary 6.18.*

6.6 Every Delaunay Triangulation Maximizes the Smallest Angle

Why are we interested in Delaunay triangulations? It turns out that Delaunay triangulations satisfy a number of interesting properties. Here we give a scientific explanation for their nice looks.

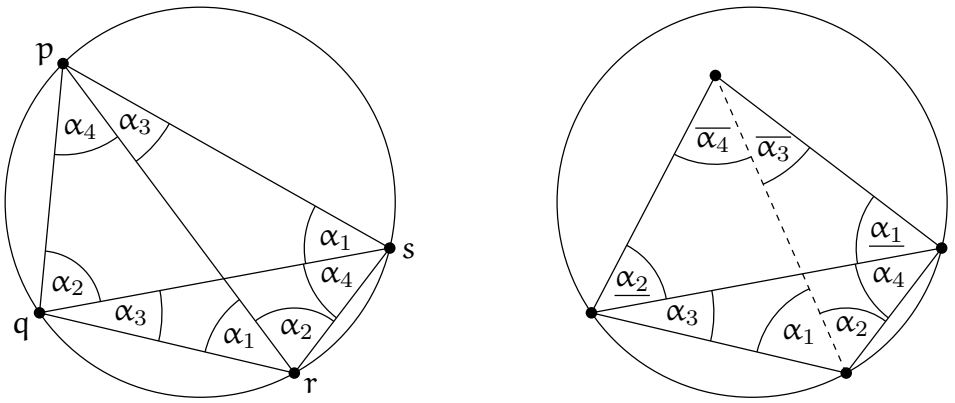
Recall that when we compared a scan triangulation with a Delaunay triangulation of the same point set in Figure 6.3, we claimed that the scan triangulation is "ugly" because it contains many long and skinny triangles. The triangles of the Delaunay triangulation, at least in this example, look much nicer, that is, much closer to an equilateral triangle. One way to quantify this "niceness" is to look at the angles that appear in a triangulation: If all angles are large, then all triangles are reasonably close to an equilateral triangle. Indeed, we will show that Delaunay triangulations maximize the smallest angle among all triangulations of a given point set. This is not saying that there are no long and skinny triangles in a Delaunay triangulation. But if there is one, then the small angle is inherent: there would exist at least as skinny triangle in *every* triangulation of the point set.

Every triangulation $\mathcal{T}$ of P induces an *angle sequence* $A(\mathcal{T}) = (\theta_1, \theta_2, \ldots, \theta_{3m})$ which lists the measures of interior angles of all $T \in \mathcal{T}$, sorted increasingly so that $0 < \theta_1 \leqslant \theta_2 \leqslant \cdots \leqslant \theta_{3m} < \pi$. Here m is the number of triangles, which is a constant determined by P; see Lemma 6.4. Let $\mathcal{T}, \mathcal{T}'$ be two triangulations of P. We say that $A(\mathcal{T}) < A(\mathcal{T}')$ if there is some i for which $\theta_i < \theta'_i$ and $\theta_j = \theta'_j$, for all $j < i$. (This is nothing but the lexicographic order on angle sequences.) We write $A(\mathcal{T}) \leqslant A(\mathcal{T}')$ if $A(\mathcal{T}) < A(\mathcal{T}')$ or $A(\mathcal{T}) = A(\mathcal{T}')$.

Theorem 6.20. *Let $P \subseteq \mathbb{R}^2$ be a finite set of points in general position (not all collinear and no four cocircular). Let $\mathcal{D}^*$ be the unique Delaunay triangulation of P, and let*

$\mathcal{T}$ *be any triangulation of* P. *Then* $A(\mathcal{T}) \leqslant A(\mathcal{D}^*)$.

In particular, $\mathcal{D}^*$ maximizes the smallest angle among all triangulations of P.



(a) Four cocircular points and the induced eight angles.

(b) The situation before a flip.

Figure 6.13: *Angle-optimality of Delaunay triangulations.*

Proof. We know that $\mathcal{T}$ can be transformed into $\mathcal{D}^*$ through the Lawson flip algorithm, and we are done if we can show that each flip lexicographically increases the angle sequence. Recall that a flip involves two triangles and thus effectively expels six angles from the sequence and injects another six. We claim that the minimum of the six new angles is *strictly* larger than the minimum of the six old angles. This claim, once proven, would imply that the sequence increases lexicographically: Before flipping, let $0 < \theta < \pi$ be the minimum of the six old angles and $i \in \{1, \ldots, 3m\}$ be the last position that the value occurs in the sequence; after flipping, all values at positions $j < i$ shall persist whereas the value at position i shall strictly increase.

Next we proceed to show the claim. Let us first look at the situation of four cocircular points; see Figure 6.13a. The *inscribed angle theorem* (a generalization of Thales' Theorem, stated below as Theorem 6.21) tells us that the eight depicted angles come in four equal pairs. For instance, the angles labeled α_1 at s and r are angles on the same side of the chord pq of the circle.

In our situation, however, no four points are cocircular. When we perform a Lawson flip, the picture is as in Figure 6.13b where we are about to replace the solid with the dashed diagonal. Here we use under- and over-lines to suggest the relation between angles; angle $\underline{\alpha}$ (repectively $\overline{\alpha}$) is strictly smaller (respectively larger) than α. At the flip, the six old angles are

$$\alpha_1 + \alpha_2, \quad \alpha_3, \quad \alpha_4, \quad \underline{\alpha_1}, \quad \underline{\alpha_2}, \quad \overline{\alpha_3} + \overline{\alpha_4},$$

and the six new angles are

$$\alpha_1, \quad \alpha_2, \quad \overline{\alpha_3}, \quad \overline{\alpha_4}, \quad \underline{\alpha_1} + \alpha_4, \quad \underline{\alpha_2} + \alpha_3.$$

Now, *every* new angle is larger than *some* old angle:

$$\begin{aligned} \alpha_1 &> \underline{\alpha_1}, \\ \alpha_2 &> \underline{\alpha_2}, \\ \overline{\alpha_3} &> \alpha_3, \\ \overline{\alpha_4} &> \alpha_4, \\ \underline{\alpha_1} + \alpha_4 &> \alpha_4, \\ \underline{\alpha_2} + \alpha_3 &> \alpha_3. \end{aligned}$$

So the minimum of the new angles is strictly larger than the minimum of the old angles, as claimed. □

Theorem 6.21 (Inscribed Angle Theorem). *Let* pq *be a chord on a circle* C*. Then* $\angle prq$ *stays constant when the point* r *moves along any of the two arcs between* p *and* q*.*

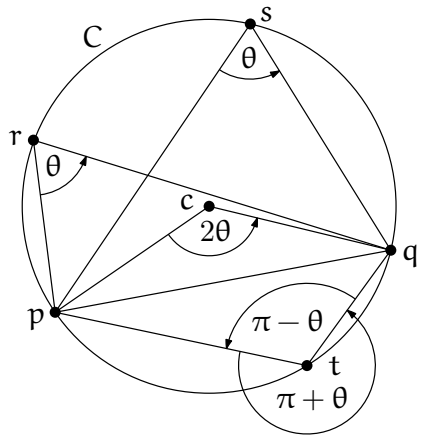
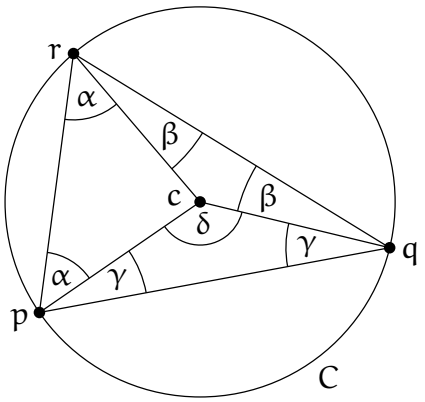


Figure 6.14: *The Inscribed Angle Theorem with* $\theta := \angle prq$*.*

Proof. Without loss of generality we may assume that c is located to the left of or on the oriented line pq.

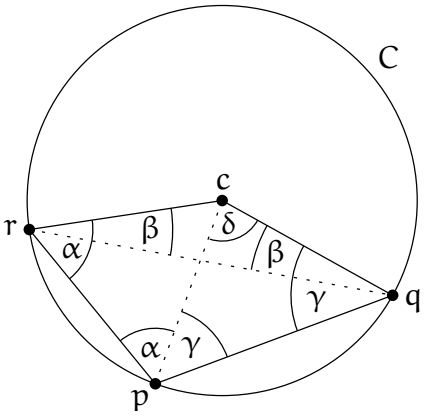
Consider first the case that the triangle $\Delta = pqr$ contains c. Then Δ can be partitioned into three triangles: pcr, qcr, and cpq. All three triangles are isosceles, because two sides of each form the radius of C. Denote $\alpha = \angle prc$, $\beta = \angle crq$, $\gamma = \angle cpq$, and $\delta = \angle pcq$ (see the figure shown to the right). The angles we are interested in are $\theta = \angle prq = \alpha + \beta$ and δ, and we will show that $\delta = 2\theta$.



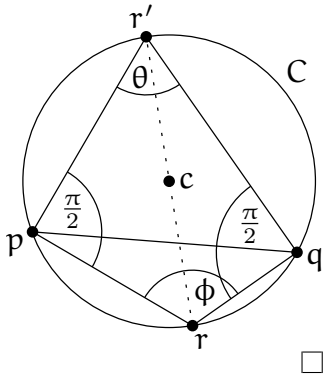
Indeed, the angle sum in Δ is $\pi = 2(\alpha + \beta + \gamma)$ and the angle sum in the triangle cpq is $\pi = \delta + 2\gamma$. Combining both yields $\delta = 2(\alpha + \beta) = 2\theta$.

Next suppose that $pqcr$ are in convex position and r is to the left of or on the oriented line pq. Without loss of generality let r be to the left of or on the oriented line qc. (The case that r lies to the right of or on the oriented line pc is symmetric.) Define α, β, γ, δ as above and observe that $\theta = \alpha - \beta$. Again we show that $\delta = 2\theta$.

The angle sum in the triangle cpq is $\pi = \delta + 2\gamma$ and the angle sum in the triangle rpq is $\pi = (\alpha - \beta) + \alpha + \gamma + (\gamma - \beta) = 2(\alpha + \gamma - \beta)$. Combining both yields $\delta = \pi - 2\gamma = 2(\alpha - \beta) = 2\theta$.



It remains to consider the case that r is to the right of the oriented line pq. Consider the point r' that is antipodal to r on C, and the quadrilateral $Q = prqr'$. We are interested in the angle ϕ of Q at r. By Thales' Theorem the inner angles of Q at p and q are both $\pi/2$. Hence the angle sum of Q is $2\pi = \theta + \phi + 2\pi/2$ and so $\phi = \pi - \theta$. As shown in the first two cases, θ is a constant and thus ϕ is also a constant.



□

What happens in the case where the Delaunay triangulation is not unique? The following still holds.

Theorem 6.22. *Let $P \subseteq \mathbb{R}^2$ be a finite set of points, not all on a line. Every Delaunay triangulation $\mathcal{D}$ of P maximizes the smallest angle among all triangulations $\mathcal{T}$ of P.*

Proof. Let $\mathcal{D}$ be some Delaunay triangulation of P. We infinitesimally perturb the points in P such that no four are on a common circle anymore. Then the Delaunay triangulation becomes unique (Corollary 6.18). Starting from $\mathcal{D}$, we keep applying Lawson flips until we reach the unique Delaunay triangulation $\mathcal{D}^*$ of the perturbed point set. Now we examine this sequence of flips on the original *unperturbed* point set. All these flips must involve four cocircular points (only in the cocircular case, an infinitesimal perturbation can change "good" edges into "bad" edges that still need to be flipped). But as Figure 6.13 (a) easily implies, such a "degenerate" flip does not change the smallest of the six involved angles. It follows that $\mathcal{D}$ and $\mathcal{D}^*$ have the same smallest angle, and since $\mathcal{D}^*$ maximizes the smallest angle among all triangulations $\mathcal{T}$ (Theorem 6.20), so does $\mathcal{D}$. □

6.7 Constrained Triangulations (not covered in 2025)

Sometimes one would like to have a Delaunay triangulation, but certain edges are already prescribed. Of course, one cannot expect to be able to get a proper Delaunay triangulation where all triangles satisfy the empty circle property. But it is possible to

obtain some triangulation that comes as close as possible to a proper Delaunay triangulation, given that we are forced to include the edges in E. Such a triangulation is called a *constrained Delaunay triangulation*, a formal definition of which follows.

Let $P \subseteq \mathbb{R}^2$ be a finite point set and $G = (P, E)$ a geometric graph with vertex set P and straight-line edges E. A triangulation $\mathcal{T}$ of P is said to be a *constrained Delaunay triangulation* with respect to G if it contains all edges in E and, for every triangle $\Delta \in \mathcal{T}$,

> The circumcircle of Δ does not enclose any point $q \in P$ visible from Δ°. A point $q \in P$ is *visible* from Δ° if there exists a point $p \in \Delta^\circ$ such that the line segment $\overline{pq}$ does not cross any $e \in E$. We can thus imagine the line segments of E as "blocking the view".

For illustration, consider the simple polygon and its constrained Delaunay triangulation shown in Figure 6.15, where the thick edges are prescribed. The circumcircle of the shaded triangle Δ contains a lot of points in its interior, but that does not matter since the points are blocked by the edge e and are thus invisible from Δ°.

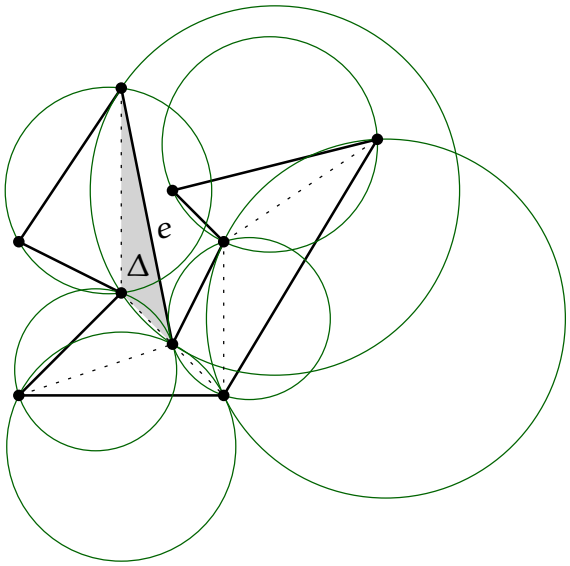


Figure 6.15: *Constrained Delaunay triangulation of a simple polygon.*

Theorem 6.23. *For every finite point set P and every plane graph $G = (P, E)$, there exists a constrained Delaunay triangulation of P with respect to G.*

Exercise 6.24. *Prove Theorem 6.23. Also describe a polynomial algorithm to construct such a triangulation.*

Questions

23. *What is a triangulation?* Provide the definition and prove a basic property: every triangulation with the same set of vertices and the same outer face has the same

number of triangles.

24. *What is a triangulation of a point set?* Give a precise definition.
25. *Does every point set (not all points on a common line) have a triangulation?* You may, for example, argue with the scan triangulation.
26. *What is a Delaunay triangulation of a set of points*? Give a precise definition.
27. *What is the Delaunay graph of a point set?* Give a precise definition and a characterization.
28. *How can you prove that every set of points (not all on a common line) has a Delaunay triangulation?* You can for example sketch the Lawson flip algorithm and the Lifting Map, and use these to show the existence.
29. *When is the Delaunay triangulation of a point set unique?* Show that general position is a sufficient condition. Is it also necessary?
30. *What can you say about the "quality" of a Delaunay triangulation?* Prove that every Delaunay triangulation maximizes the smallest interior angle in the triangulation, among the set of all triangulations of the same point set.

Chapter 7

Incremental Construction of Delaunay Triangulation

We have learned about the Lawson flip algorithm which computes a Delaunay triangulation of a given n-point set $P \subseteq \mathbb{R}^2$ by performing $O(n^2)$ flips. With some care, the algorithm can be implemented to run in $O(n^2)$ time. On the other hand, we have also seen in an exercise that certain point sets require $\Omega(n^2)$ flips, meaning that the worst-case running time is $\Theta(n^2)$.

Here we will present a different, randomized algorithm which runs in $O(n \log n)$ time in expectation. (The probability comes from the random choices made by the algorithm, not from the input P.) Throughout we assume general position (no three points collinear and no four points cocircular), so that the Delaunay triangulation is unique by Corollary 6.18. There are techniques to deal with non-general position, but we will leave that out.

7.1 Incremental construction

To avoid special cases, we augment the set P with three "far-out" points a, b and c. For now suffice it to say that the huge triangle abc contains P with abundant space cushion.

The idea is to start from the triangle abc and insert other points one after another according to a uniformly random order $(p_1, p_2, \ldots, p_n)$ of P. For $1 \leqslant s \leqslant n$, we denote $P_s = \{p_1, \ldots, p_s\}$ and $P_s^+ = \{a, b, c\} \cup P_s$. Suppose that in the first $s-1$ rounds we had built the Delaunay triangulation $\mathcal{D}_{s-1}$ of P_{s-1}^+. At round s we shall insert point p_s and repair the structure to get the Delaunay triangulation $\mathcal{D}_s$ of P_s^+. In the end, we obtain the Delaunay triangulation $\mathcal{D}_n$ of P_n^+.

From $\mathcal{D}_n$ we want to "read off" the Delaunay triangulation of P by simply ignoring the three artificial points. For this to work, the convex hull boundary $\partial\mathrm{conv}(P)$ should be respected by $\mathcal{D}_n$. It can be ensured by placing a, b, c far enough so that they are not enclosed by the empty circumcircles going through adjacent convex hull vertices. But practically speaking, a simpler approach is to choose $a = (-\infty, -\infty)$, $b = (\infty, -\infty)$ and

$c = (0, \infty)$ and extend the algebra to handle symbols $-\infty, \infty$.

Below is the outline of round s, which will be fleshed out in subsequent sections. In our figures, we suppress the artificial points since they are merely a technicality.

(a) Find the triangle $\Delta \in \mathcal{D}_{s-1}$ that contains p_s, and split it into three triangles by connecting p_s with the three vertices of Δ. We now have a triangulation $\mathcal{T}$ of P_s^+. (Figure 7.1a)

(b) Perform Lawson flips on $\mathcal{T}$ until we obtain the Delaunay triangulation $\mathcal{D}_s$. (Figure 7.1b)

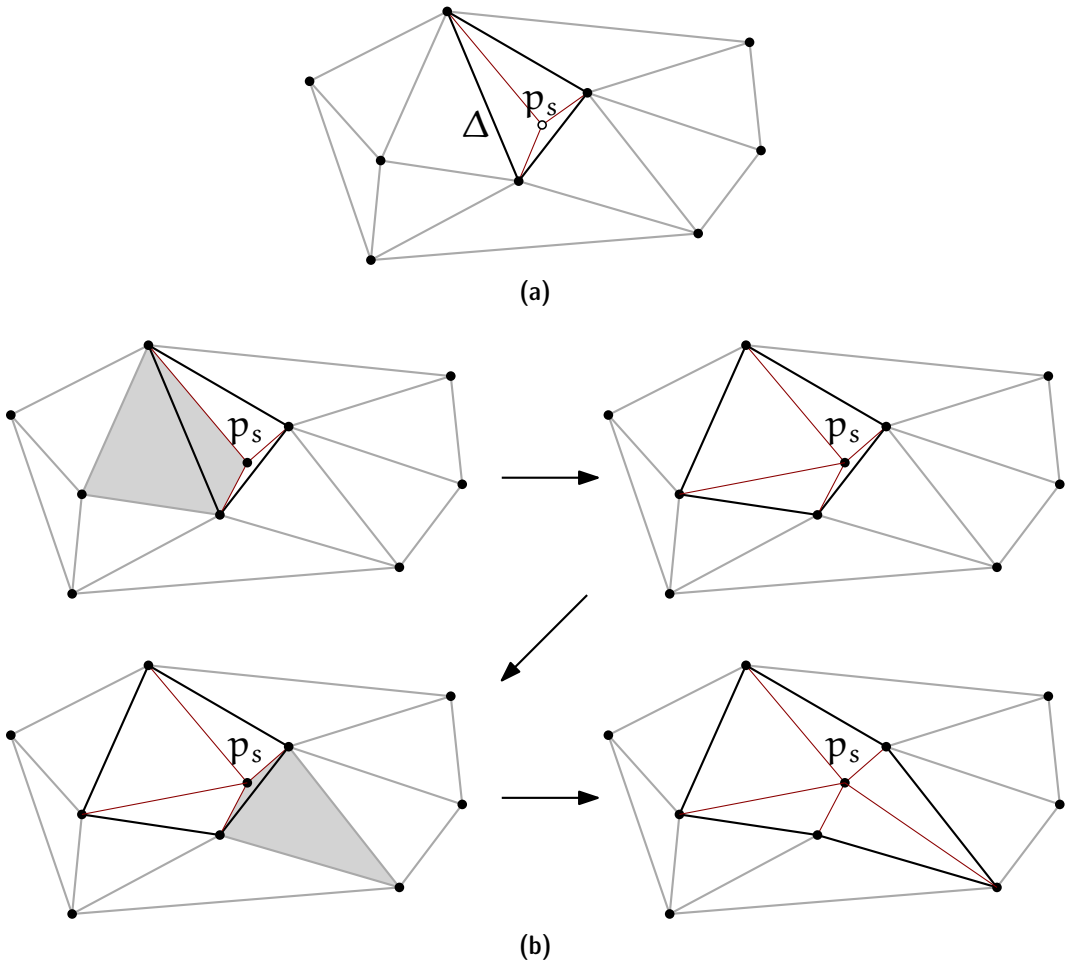


Figure 7.1: *Insert p_s to $\Delta \in \mathcal{D}_{s-1}$ and perform Lawson flips.*

7.2 Organizing the Lawson flips

First off, let us implement (b) in the outline. It turns out that the Lawson flips proceed quite systematically.

Lemma 7.1. *The following invariants hold at any particular moment in round s:*

(i) Every edge incident to p_s must belong to $\mathcal{D}_s$; in particular, it cannot be flipped away in the rest of round s.

(ii) Every applicable Lawson flip at this moment involves some triangle $p_s uv$ and some triangle $uvw \in \mathcal{D}_{s-1}$. It replaces them with triangles $p_s uw$ and $p_s vw$, both incident to p_s, thus the degree of p_s increases by one.

Proof. We argue by strong induction over time. As the base case, we consider the moment before any flip is performed.

(i) Let us take any incident edge $p_s w$, where w must be a vertex of Δ. Since $\Delta \in \mathcal{D}_{s-1}$, its circumcircle C encloses nothing but the new point p_s. We can thus shrink C to an empty circle C' passing through p_s and w only, see Figure 7.2a. So the edge $p_s w$ must be in $\mathcal{D}_s$ by Lemma 6.17.

(ii) Only the three edges of Δ are potentially flippable, since they are the only edges whose incident triangles have changed and form a convex quadrilateral. So any next flip must adhere to the claimed format.

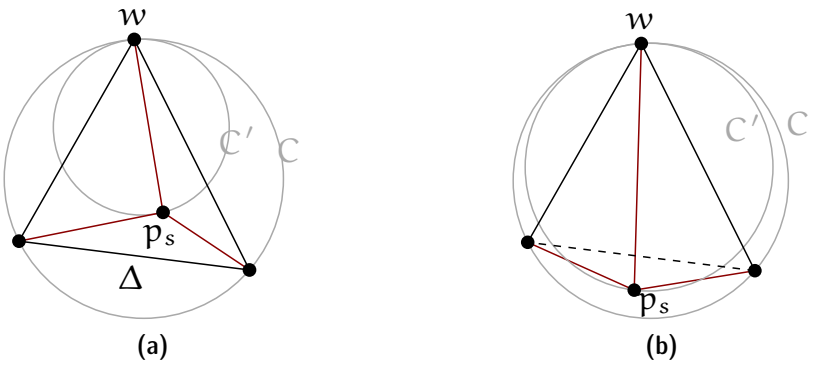


Figure 7.2: *Newly created edges incident to p_s are in the Delaunay graph*

Next we consider the moment after some flip(s) have been performed. Denote by $R \subseteq \mathbb{R}^2$ the union of all triangles incident to p_s right now. Note that R is a star-shaped polygon. One can see by inductively applying (ii) that the affected region of the previous flips is restricted in R. In other words, all triangles outside R are not yet touched, meaning they must belong to $\mathcal{D}_{s-1}$.

(i) Let us take the incident edge $p_s w$ generated by the last flip. By induction hypothesis (ii), this flip destroys exactly one triangle in $\mathcal{D}_{s-1}$. Its circumcircle C contains p_s only, and shrinking it yields an empty circle C' through p_s and w, see Figure 7.2b. Thus $p_s w$ must be in $\mathcal{D}_s$ by Lemma 6.17.

(ii) As established in (i), all edges incident to p_s are not flippable. So any flippable edges has to be a boundary edge of polygon R, say uv. On one side it is incident to some triangle $p_s uv$ (by definition of R); on the other side it is incident to a triangle $uvw \in \mathcal{D}_{s-1}$ (as we argued above).

This completes the induction. □

The lemma suggests that we can maintain a queue of potentially flippable edges that we process in turn. Initially the queue contains only the three edges of Δ. In each step, we remove an edge uv from the queue. If its two incident triangles $p_s uv$ and uvw are not locally Delaunay, then we perform the flip and push uw and vw to the queue. Otherwise we simply discard it because it cannot become flippable in the future. (Suppose to contradiction that it becomes flippable, then by Lemma 7.1 the flip must involve $p_s uv$ and some $uvw \in \mathcal{D}_{s-1}$. But the two triangles are in place right now, so we should have performed the flip right away.)

Corollary 7.2. *Let $d_s := \deg_{\mathcal{D}_s}(p_s)$ be the degree of vertex p_s in the (graph of) triangulation $\mathcal{D}_s$. Then in round s we perform exactly $d_s - 3$ Lawson flips. Moreover, these flips can be implemented to consume time only linear in d_s. The total number of triangles created in round s is $2d_s - 3$ (although some of them can be flipped away within the same round).*

Proof. By Lemma 7.1, every Lawson flip increases the number of edges incident to p_s by exactly one. So the number of flips is equal to the final degree d_s minus the initial degree 3. Each flip creates two new triangles, along with the initial three triangles we get $2d_s - 3$ in total. Using the queue implementation discussed above, every flip needs only a constant number of operations, so the total running time is linear in d_s. □

7.3 The History Graph

Let us get back to part (a) in the outline and specify how we find the triangle $\Delta \in \mathcal{D}_{s-1}$ that contains the point p_s. Doing this in the naïve way (checking all triangles) is not a good idea, as it would then amount to $\Theta(n^2)$ work throughout the whole algorithm. Here is a smarter method, based on a data structure called *history graph*.

Definition 7.3. *For $1 \leqslant s \leqslant n$, the* **history graph** *$\mathcal{H}_s$ is a directed acyclic graph whose nodes are all triangles ever been created in the first s rounds. Whenever the algorithm splits a triangle Δ, we add a directed edge from Δ to the three new triangles (Figure 7.3a). Whenever the algorithm flips triangles Δ_1, Δ_2 to Δ_1', Δ_2', we add directed edges $\Delta_i \to \Delta_j'$ for $i, j \in \{1, 2\}$. (Figure 7.3b)*

The history graph $\mathcal{H}_s$ contains triangles of outdegrees 3, 2 and 0, where the ones with zero outdegree are exactly the triangles of $\mathcal{D}_s$. It can be built during the incremental construction at asymptotically no extra cost; but it may need extra space to keeps all triangles ever created.

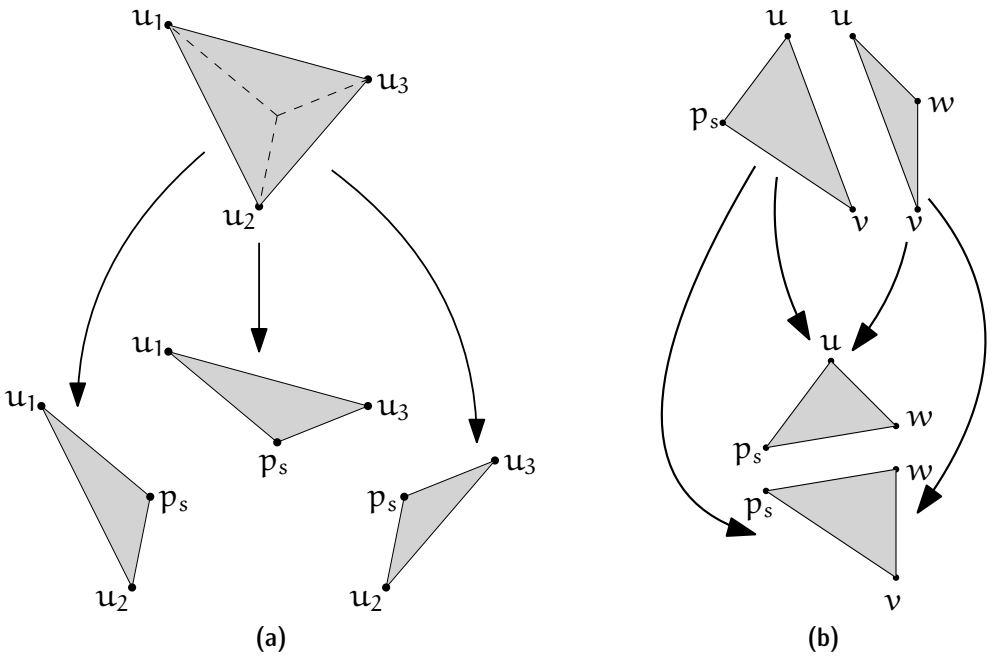


Figure 7.3: *The history graph*

Given $\mathcal{H}_{s-1}$, we can search for the triangle $\Delta \in \mathcal{D}_{s-1}$ that contains p_s by starting from the big triangle abc—it certainly contains p_s—and tracking down a directed path in $\mathcal{H}_{s-1}$. If the current triangle still has outneighbors, we move on to the unique outneighbor containing p_s (recall that we assume general position) and search iteratively. If the current triangle has no outneighbors, it must be in $\mathcal{D}_{s-1}$ and contains p_s, so we are done.

7.4 Analysis of the algorithm

The runtime analysis heavily exploits conditional expectations. Here is a quick refresher. Let X, Y be two random variables in a finite probability space. When we "condition on" variable X, what we mean is to "freeze" or "reveal" the outcome of X as a concrete value. Consequently some randomness dissipates, and the distribution of Y is thus biased. In general this distribution shall depend on the concrete X-value. For example, suppose we sample a uniform permutation π of $\{1, 2, 3\}$, and define $X = \pi(1)$ and $Y = \pi(2)$. So Y by itself is uniformly distributed over $\{1, 2, 3\}$. Conditioning on X shall make Y uniformly distributed on $\{1, 2, 3\} \setminus X$ instead.

The conditional expectation $\mathbb{E}(Y \mid X)$ is defined as the expectation of Y taken with respect to this now-biased distribution. Hence $\mathbb{E}(Y \mid X)$ is a function of X in general. In

our illustrative example,

$$\mathbb{E}(Y \mid X) = \begin{Bmatrix} 2.5, & X = 1 \\ 2, & X = 2 \\ 1.5, & X = 3 \end{Bmatrix} = 3 - \frac{X}{2}.$$

It is easy to prove the so-called total expectation formula $\mathbb{E}(Y) = \mathbb{E}[\mathbb{E}(Y \mid X)]$, but it might be more important to remember its interpretation. To compute $\mathbb{E}(Y)$, we first partition the universe depending on the outcome of X. Then for each part, we compute the expectation $\mathbb{E}(Y \mid X)$ individually, which are our "partial results". Finally, we put these pieces together by a weighted average. One can view this as a natural generalization of elementary counting principle: To count the number of certain objects, we could partition them into several types, count each type individually, and then sum them up.

Cost of Lawson flips. Recall from Corollary 7.2 that $d_s := \deg_{\mathcal{D}_s}(p_s)$ captures the running time of Lawson flips as well as the growth of history graph in round s. This leads us to study the expected value of d_s.

Lemma 7.4. $\mathbb{E}[d_s] \leqslant 6$ *for all* s.

Proof. Let us condition on the set P_s, i.e. we freeze the set of the first s points. Note that the exact ordering of these points is not revealed, and remains uniformly random. In particular, p_s is uniformly distributed in P_s. On the other hand, the Delaunay triangulation $\mathcal{D}_s$ is no longer random because it is uniquely determined by P_s.

Hence $\mathbb{E}[d_s \mid P_s]$ means "the expected degree of p_s in the fixed graph $\mathcal{D}_s = \mathcal{D}_s(P_s)$, where the point p_s is sampled from the fixed set P_s uniformly at random".

Since $\mathcal{D}_s$ is a triangulation on $s+3$ points with triangular convex hull, it follows from Lemma 6.4 that it has $3(s+3)-6$ edges. Excluding the three edges of the convex hull, the total degree of all points in P_s is at most $2(3(s+3)-9) = 6s$. This implies that $\mathbb{E}[d_s \mid P_s] \leqslant 6$. The lemma follows by removing the condition via total expectation. □

By combining the above lemmas, we can also prove the following bound on the expected number of triangles created by the algorithm. Note that this is at the same time a bound on the expected size of the history graph.

Corollary 7.5. *The expected number of triangles ever created in* n *rounds is at most* $9n + 1 = O(n)$. *All the same, the expected running time of all Lawson flips in* n *rounds is* $O(n)$.

Proof. Before inserting any points from the set P, we only have the artificial triangle abc. During round s of the algorithm, we know from Corollary 7.2 that the number of new triangles created is $2d_s - 3$. Combined with Lemma 7.4, the expected number of created triangles in all n iterations is

$$1 + \mathbb{E}\left[\sum_{s=1}^{n} 2d_s - 3\right] = 1 + \sum_{s=1}^{n} (2\mathbb{E}[d_s] - 3) \leqslant 1 + (2 \cdot 6 - 3)n = 9n + 1. \qquad \square$$

Note that we cannot say that every round creates at most 9 triangles; as there could be very costly insertions with some probability. But the claim holds in expectation which is enough to provide a linear expected runtime.

Cost of locating points. We proceed now to the most difficult part of the analysis: to bound the time for finding the triangle in $\mathcal{H}_{s-1}$ that contains p_s. This is proportional to the number of triangles in $\mathcal{H}_{s-1}$ that contains p_s. Hence let us take a closer look at all the triangles in the history graph $\mathcal{H}_{s-1}$.

Suppose a triangle Δ was added to the graph in round r. If $\Delta \in \mathcal{D}_r$ then we call it *valid*, as it survived the round that it was born. Otherwise we call it *ephemeral*, as it got flipped away in the very same round it was born. To make the analysis possible, we want to express the running time in terms of valid triangles only.

Observation 7.6. *The number of triangles in $\mathcal{H}_{s-1}$ that contains p_s is proportional to the number of* valid *triangles in $\mathcal{H}_{s-1}$ whose* circumcircle *contains p_s.*

Indeed, recall from Lemma 7.1 that at every Lawson flip in some round r, one of the replaced triangles is in $\mathcal{D}_{r-1}$ (hence valid) and the other one was created in the current round r (hence ephemeral). That is, a flip always destroys valid and ephemeral triangles in pair. Therefore, for any ephemeral triangle $\Delta \in \mathcal{H}_{s-1}$ that contains p_s, we may charge it to its partner Δ', the valid triangle that was destroyed together with Δ. It is clear that the triangle Δ' is charged at most once. We also know from the condition of Lawson flip that Δ, hence also p_s, is contained in the circumcircle of Δ'. So the observation is established.

Back to time analysis, let us introduce some handy random variables. For every $1 \leqslant r < s \leqslant n$,

- $\tau_r = \mathcal{D}_r \setminus \mathcal{D}_{r-1}$ consists of all triangles in $\mathcal{D}_r$ newly created in round r;
- $\varphi_{r,s}$ is the number of triangles in τ_r whose circumcircle contains the point p_s.

Then the observation implies that the searching time in round s is proportional to $\sum_{r=1}^{s-1} \varphi_{r,s}$. This works since any valid triangle in $\mathcal{H}_{s-1}$ that contains p_s must be in τ_r for some $r < s$.

Instead of bounding this cost for a particular round s, we try to bound the combined cost over all rounds, i.e.

$$T := \sum_{s=1}^{n} \sum_{r=1}^{s-1} \varphi_{r,s} = \sum_{r=1}^{n} \sum_{s=r+1}^{n} \varphi_{r,s}$$

where we exchanged the summations in the second equality.

Lemma 7.7 (The proof of this lemma is not exam-relevant in 2025.). *It holds that* $\mathbb{E}[T] = O(n \log n)$.

Proof. Using linearity of expectation, we have

$$\mathbb{E}[T] = \sum_{r=1}^{n} \sum_{s=r+1}^{n} \mathbb{E}[\varphi_{r,s}].$$

Observe that the variables $\varphi_{r,r+1}, \varphi_{r,r+2}, \ldots, \varphi_{r,n}$ are identically distributed due to symmetry. To see this more clearly, recall that $\varphi_{r,s}$ is defined in terms of τ_r and p_s. Let us condition on (i.e. freeze) the ordering $(p_1, \ldots, p_r)$. Then τ_r is fixed, whereas each of $p_{r+1}, p_{r+2}, \ldots, p_n$ is uniformly distributed over the fixed set $P \setminus \{p_1, \ldots, p_r\}$. It follows that the variables of interest are identically distributed under the condition; but we may remove the condition nonetheless via total probability.

Hence we may simplify the expectation as

$$\mathbb{E}[T] = \sum_{r=1}^{n} (n - r) \cdot \mathbb{E}[\varphi_{r,r+1}] \tag{7.8}$$

It remains to analyze the expected value $\mathbb{E}[\varphi_{r,r+1}]$ for every particular $1 \leqslant r \leqslant n$. Let Γ consist of all triangles in $\mathcal{D}_r$ whose circumcircle contains p_{r+1}. This is nothing but "all triangles in $\mathcal{D}_r$ that are destroyed in round $r+1$". From Lemma 7.1 we immediately have $|\Gamma| = d_{r+1} - 2$ (we also count the triangle that is split into three).

On the other hand, by definition we may rewrite $\varphi_{r,r+1} = \sum_{\Delta \in \Gamma} X_\Delta$. Here X_Δ is the indicator variable for the event $\Delta \notin \mathcal{D}_{r-1}$, which takes value 1 if the event happens and value 0 otherwise. In order to apply linearity of expectation, the summation must be "derandomized"; that is, it should not run over a random set Γ. Hence we condition on (i.e. freeze) the *set* P_r as well as the point p_{r+1}. We stress that the concrete ordering of P_r is not revealed. Nevertheless, the Delaunay triangulation $\mathcal{D}_r$ is uniquely determined, so is Γ. Therefore,

$$\begin{aligned}
\mathbb{E}[\varphi_{r,r+1} \mid P_r, p_{r+1}] &= \sum_{\Delta \in \Gamma} \mathbb{E}[X_\Delta \mid P_r, p_{r+1}] \\
&= \sum_{\Delta \in \Gamma} \Pr[\Delta \notin \mathcal{D}_{r-1} \mid P_r, p_{r+1}] \\
&\leqslant \sum_{\Delta \in \Gamma} \frac{3}{r} \\
&= \frac{3}{r} \cdot |\Gamma| \;=\; \frac{3}{r} \cdot (d_{r+1} - 2)
\end{aligned}$$

To see the inequality, observe that if a triangle $\Delta \in \Gamma$ is not contained in $\mathcal{D}_{r-1}$, then it must be created in round r. In particular, p_r needs to be its vertex by Lemma 7.1. As p_r is uniformly distributed over the set P_r (under conditions P_r, p_{r+1}), the event happens with probability at most $3/r$. ("At most" because some vertex of Δ might be the artificial points a, b or c; in that case p_r cannot hit it).

Now we remove the condition via total expectation and obtain

$$\mathbb{E}[\varphi_{r,r+1}] \leqslant \frac{3}{r} \cdot (\mathbb{E}[d_{r+1}] - 2) \leqslant \frac{12}{r}, \tag{7.9}$$

where we used Lemma 7.4 in the last step. We are finally able to plug (7.9) back into (7.8) to conclude the proof:

$$\mathbb{E}[T] \leqslant \sum_{r=1}^{n} \frac{12(n-r)}{r} \leqslant 12n \sum_{r=1}^{n} \frac{1}{r} = O(n \log n). \qquad \square$$

The main theorem. Having the previous lemmas at hand, assembling our main result is now straightforward.

Theorem 7.10. *The Delaunay triangulation of a set P of n points in the plane can be computed in $O(n \log n)$ expected time, using $O(n)$ expected space.*

Proof. The correctness of the algorithm follows from the correctness of the Lawson flip algorithm, and from the fact that we perform all possible Lawson flips in each round. For the space consumption, only the history graph might use more than linear space, but Lemma 7.5 bounds its expected size by $O(n)$, so the claim follows.

For the running time, Lemma 7.7 bounds the expected time spent on point location (over all n rounds) by $O(n \log n)$, and Lemma 7.5 bounds the expected time spent on Lawson flips (over all n rounds) by $O(n)$. So the algorithm runs in $O(n \log n)$ time in expectation. $\square$

Exercise 7.11. *For a sequence of n pairwise distinct numbers $y_1, \ldots, y_n$ consider the sequence of pairs $(\min\{y_1, \ldots, y_i\}, \max\{y_1, \ldots, y_i\})_{i=0,1,\ldots,n}$ ($\min \emptyset := +\infty, \max \emptyset := -\infty$). How often do these pairs change in expectation if the sequence is permuted randomly, each permutation appearing with the same probability? Determine the expected value.*

Exercise 7.12. *Given a set P of n points in convex position represented by the clockwise sequence of the vertices of its convex hull, provide an algorithm to compute its Delaunay triangulation in $O(n)$ time.*

Questions

31. *What conditions should the three "far-out" points a, b, c satisfy? Explain the reason.*
32. *Describe the algorithm for the incremental construction of $\mathcal{DT}(P)$*: how do we find the triangle containing the point p_s to be inserted into $\mathcal{D}_{s-1}$? How do we transform $\mathcal{D}_{s-1}$ into $\mathcal{D}_s$? How many steps does the latter transformation take?
33. *What are the two types of triangles that the history graph contains?*

Chapter 8

Voronoi Diagrams

8.1 The Post Office Problem

Suppose there are n post offices in a city, and a citizen would like to know which one is closest to him.[1] Modeling the city in the plane, we think of the post offices as a point set $P = \{p_1, \dots p_n\} \subset \mathbb{R}^2$, and the query location as a point $q \in \mathbb{R}^2$. The task is to find $p_i \in P$ that minimizes $\|p_i - q\|$.

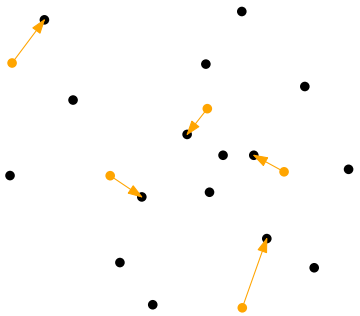


Figure 8.1: *Closest post offices for various query points.*

While the post offices P are considered stable, the query point q is not known in advance and can be changing frequently. Therefore, our long term goal is to come up with a (static) data structure on top of P that allows to answer *any* possible query efficiently.

As there can be only n possible answers, the idea is to apply the so-called *locus approach*: we subdivide the query space (in our case $\mathbb{R}^2$) into n regions according to the answer; the i-th region contains all points for which p_i is the closest. The resulting structure is called a *Voronoi diagram*; see Figure 8.2 for an example.

[1] Another—possibly historically more accurate—way to think of the problem: You want to send a letter to a person living in the city. For this you should know his zip code, which is the code of the post office closest to him.

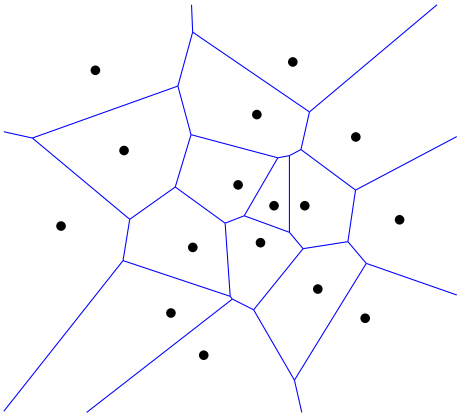


Figure 8.2: *The Voronoi diagram of a point set.*

Let us remark right away that such approach works for a variety of distance functions and spaces [2, 7]. So the Voronoi diagram can be viewed as a broadly applicable paradigm. Without further qualification, the underlying distance function is Euclidean.

What exactly does a Voronoi diagram look like? As a warmup, suppose there are only two post offices: $P = \{p, p'\}$. Then the plane subdivides into two regions delimited by the *bisector* of p and p', i.e. the points that are equidistant to p and p'. The following proposition characterizes the shape of the bisector.

Proposition 8.1. *The bisector of two distinct points $p, p' \in \mathbb{R}^d$ is a hyperplane (a line when $d = 2$). It is orthogonal to the line pp' and goes through the midpoint of $\overline{pp'}$.*

Proof. Let us understand points as column vectors, so for any points $a = (a_1, \ldots, a_d)$ and $b = (b_1, \ldots, b_d)$ in $\mathbb{R}^d$ we have the identity $\|a-b\|^2 = \sum_{i=1}^d (a_i - b_i)^2 = \sum_{i=1}^d a_i^2 - 2\sum_{i=1}^d a_i b_i + \sum_{i=1}^d b_i^2 = \|a\|^2 - 2a^\top b + \|b\|^2$.

The bisector of p, p', by definition, consists of all points $x \in \mathbb{R}^d$ such that

$$\begin{aligned} \|p - x\| = \|p' - x\| &\iff \|p - x\|^2 = \|p' - x\|^2 \\ &\iff \|p\|^2 - 2p^\top x + \|x\|^2 = \|p'\|^2 - 2p'^\top x + \|x\|^2 \\ &\iff 2(p' - p)^\top x = \|p'\|^2 - \|p\|^2. \end{aligned}$$

As $p \neq p'$, this is the equation of a hyperplane orthogonal to the vector $p' - p$ (hence the line pp'). One can easily verify that the midpoint $x = (p + p')/2$ fits in the equation. □

Let us then denote by $H(p, p')$ the closed halfspace bounded by the bisector of p, p' that contains p. In this chapter we only study $\mathbb{R}^2$, so $H(p, p')$ is a halfplane (Figure 8.3). As we noted earlier, when there are only two post offices p and p', the plane is subdivided by $H(p, p')$ and $H(p', p)$.

Exercise 8.2.

(a) What is the bisector of a line ℓ and a point $p \in \mathbb{R}^2 \setminus \ell$, that is, the set of all points $x \in \mathbb{R}^2$ with $\|x - p\| = \min_{r \in \ell} \|x - r\|$?

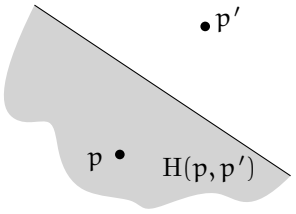


Figure 8.3: *The bisector of two points in* $\mathbb{R}^2$.

(b) For two distinct points $p, p' \in \mathbb{R}^2$*, what is the region that contains all points whose distance to* p *is exactly twice their distance to* p'*?*

8.2 Voronoi Diagram

Understanding the situation for two points essentially teaches us the law for the general case. In the following we formally define and study the Voronoi diagram for a given point set $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$.

Definition 8.3. *For* $i \in \{1, \ldots, n\}$*, the* **Voronoi cell** *of point* p_i *is defined as*

$$V_P(i) := \left\{ q \in \mathbb{R}^2 : \|q - p_i\| \leqslant \|q - p_j\|,\ \forall j \in \{1, \ldots, n\} \right\}.$$

Observe that (1) each Voronoi cell is non-empty since $p_i \in V_P(i)$; (2) the interiors of the cells are disjoint; and (3) the cells cover the entire plane. So these cells form a *subdivision* of the plane. It turns out that every cell looks quite regular:

Proposition 8.4. *For every* $i \in \{1, \ldots, n\}$*,*

$$V_P(i) = \bigcap_{j \neq i} H(p_i, p_j)\,.$$

In particular, it is a convex set whose boundary is piecewise linear (i.e. consisting of segments, rays or lines).

Proof. For every $j \neq i$, we have $\|q - p_i\| \leqslant \|q - p_j\|$ if and only if $q \in H(p_i, p_j)$. Hence $V_P(i)$ is exactly the intersection of these halfplanes (which are all convex); this is a convex set with piecewise linear boundary. □

Definition 8.5. *The* **Voronoi Diagram** $VD(P)$ *of a set* $P = \{p_1, \ldots, p_n\}$ *of points in* $\mathbb{R}^2$ *is the subdivision of the plane induced by the Voronoi cells* $V_P(i)$*, for* $i = 1, \ldots, n$*. We denote by* $VV(P)$ *the set of vertices, by* $VE(P)$ *the set of edges, and by* $VR(P)$ *the set of regions/cells.*

Lemma 8.6. *Every vertex* $v \in VV(P)$ *satisfies the following statements:*

(a) v *is incident to at least three cells from* $VR(P)$*;*

(b) v *is the common endpoint of at least three edges from* $\mathrm{VE}(P)$*;*

(c) v *is the center of an empty circle* $C(v)$ *through at least three points from* P*; "empty" means that no point from* P *is strictly enclosed by* $C(v)$*.*

Proof. Consider a vertex $v \in \mathrm{VV}(P)$. As all Voronoi cells are convex, $k \geqslant 3$ of them must be incident to v. This proves a) and b).

Without loss of generality let these incident cells be $V_P(1), \ldots, V_P(k)$ in circular order; see Figure 8.4. Since $v \in V_P(i)$ for all $i \leqslant k$, the points $p_1, \ldots, p_k$ are simultaneously closest to v. (Any p_j where $j > k$ is strictly farther away; for otherwise v should have been incident to $V_P(j)$, too.) With r denoting this smallest distance, we have $\|v - p_i\| = r < \|v - p_j\|$ for $1 \leqslant i \leqslant k < j \leqslant n$. In other words, $p_1, \ldots, p_k$ are on an empty circle of radius r centered at v. This proves (c). □

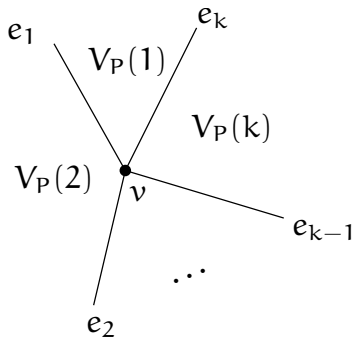


Figure 8.4: *Voronoi cells around* v*.*

Corollary 8.7. *If* P *is in general position (no four points cocircular), then every vertex* $v \in \mathrm{VV}(P)$ *satisfies the following statements:*

(a) v *is incident to exactly three cells from* $\mathrm{VR}(P)$*;*

(b) v *is the common endpoint of exactly three edges from* $\mathrm{VE}(P)$*;*

(c) v *is the center of an empty circle* $C(v)$ *through exactly three points from* P*.* □

Lemma 8.8. *There is an unbounded Voronoi edge shared by* $V_P(i)$ *and* $V_P(j)$*, if and only if* $\overline{p_i p_j} \cap P = \{p_i, p_j\}$ *and* $\overline{p_i p_j} \subseteq \partial\mathrm{conv}(P)$*.*

Proof. Denote by $b_{i,j}$ the bisector of p_i and p_j, and let $\mathcal{D}$ denote the family of circles with center on $b_{i,j}$ and passing through p_i, p_j. It is not hard to see that the following statements are equivalent:

- There is an unbounded Voronoi edge shared by $V_P(i)$ and $V_P(j)$.
- There is a ray $\rho \subset b_{i,j}$ such that for all $r \in \rho$ and $k \notin \{i, j\}$, we have $\|r - p_k\| > \|r - p_i\| = \|r - p_j\|$.

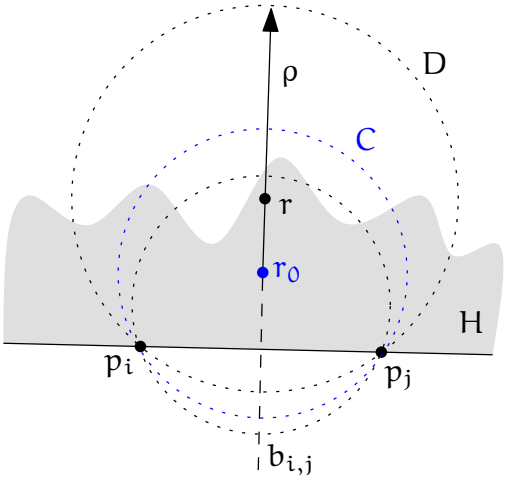


Figure 8.5: *The correspondence between* $\overline{p_ip_j}$ *appearing on* $\partial\text{conv}(P)$ *and the existence of a divergent family of empty disks.*

- There is a ray $\rho \subset b_{i,j}$ such that every circle $D \in \mathcal{D}$ with center on ρ is empty. (Figure 8.5)

Assuming the last statement, we have two observations. First, no point from P is on the segment $\overline{p_ip_j}$ except p_i, p_j. Second, the open halfplane H bounded by line p_ip_j and containing the infinite part of ρ contains no point from P. Therefore $\overline{p_ip_j}$ appears on $\partial\text{conv}(P)$.

Conversely, assume $\overline{p_ip_j} \cap P = \{p_i, p_j\}$ and $\overline{p_ip_j} \subseteq \partial\text{conv}(P)$. Then one of the open halfplanes H bounded by line p_ip_j contains no point from P. Since all points from $P \setminus \{p_i, p_j\}$ are strictly away from the segment $\overline{p_ip_j}$, there exists an empty circle $C \in \mathcal{D}$ provided its center r_0 is sufficiently far away from the line. Let $\rho \subseteq b_{i,j} \cap H$ be a ray emanating from r_0. Any circle $D \in \mathcal{D}$ centered on ρ encloses only a subset of $H \cup C$. As neither H nor C contains any point from P, the circle D is empty. This establishes the last statement, and the proof is complete. □

8.3 Duality With Delaunay Triangulations

A *straight-line dual* of a plane graph G is a geometric graph G' defined as follows: For each face of G, designate a point in $\mathbb{R}^2$ as its representative. Connect two representatives if their corresponding faces are adjacent in G.

Note that the notion depends heavily on the plane embedding G (the word "face" does not make sense for an abstract G), as well as the choice for representatives. In general, G' may have crossings.

Every Voronoi diagram can be treated as a plane graph. It is particularly natural to pick p_i as the representative for face $V_P(i)$. With this choice, the dual has no crossing and satisfies interesting properties. We thus call it *the* straight-line dual of a Voronoi diagram.

Theorem 8.9 (Delaunay [3]). *Let $P \subset \mathbb{R}^2$ be a set of $n \geqslant 3$ points in general position (no three points collinear and no four points cocircular). The straight-line dual of $\mathrm{VD}(P)$ is exactly the (unique) Delaunay triangulation of P.*

Proof. We write $G := \mathrm{VD}(P)$ (understood as a plane graph). Denote by G' its straight-line dual, and by D the Delaunay triangulation of P. Note that $V(G') = P = V(D)$. We aim to show $E(G') = E(D)$.

First we argue $E(G') \subseteq E(D)$. By construction of the dual, every edge $p_ip_j \in E(G')$ originates from adjacent cells $V_P(i), V_P(j)$ in G.

- If the cells share an unbounded Voronoi edge, then by Lemma 8.8, p_ip_j is on $\partial\mathrm{conv}(P)$ which is also contained in $E(D)$.
- Otherwise, the cells share a bounded Voronoi edge uv. By Corollary 8.7(b)(c), the Voronoi vertex v is incident to exactly three Voronoi cells $V_P(i)$, $V_P(j)$ and some $V_P(k)$, and the circle through p_i, p_j, p_k is empty. In particular p_ip_j is in $E(D)$ by Lemma 6.17.

Conversely we argue $E(G') \supseteq E(D)$. Any edge in $E(D)$ appears in some Delaunay triangle $p_ip_jp_k$ with empty circumcircle. The center v of the circle thus has p_i, p_j, p_k as its closest points. So v must be incident to the cells $V_P(i), V_P(j), V_P(k)$. Therefore, by construction of the dual we know that p_ip_j, p_jp_k, p_kp_i are edges in $E(G')$. □

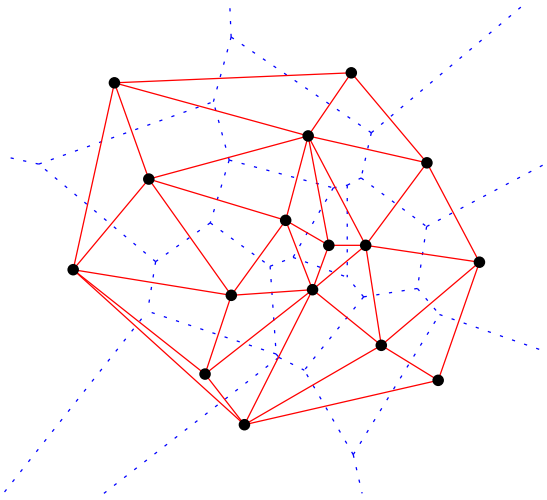


Figure 8.6: *The Voronoi diagram of a point set and its dual Delaunay triangulation.*

As a remark, the proof in fact establishes a correspondence between Voronoi vertices and Delaunay triangles: Given a Voronoi vertex, the three points from its incident cells form a Delaunay triangle; vice versa, given a Delaunay triangle, the center of its circumcircle is a Voronoi vertex.

It is not hard to remove the general position assumption in Theorem 8.9. In this case, a Voronoi vertex of degree $k > 3$ corresponds in the dual to a convex k-gon with

cocircular vertices. If we triangulate all these polygons in the dual arbitrarily, then we obtain a Delaunay triangulation of P. In fact, the dual of the Voronoi diagram for points in non-general position turns out to be equal to the Delaunay graph.

Corollary 8.10. $|VE(P)| \leqslant 3n-3-h$ *and* $|VV(P)| \leqslant 2n-2-h$*, where* $h := |P \cap \partial\mathrm{conv}(P)|$ *is the number of points on the convex hull boundary.*

Proof. We assume general position (otherwise the proof can be adapted easily). Every Voronoi edge corresponds to an edge in the Delaunay triangulation. Every Voronoi vertex corresponds to a triangle in the Delaunay triangulation. So the counts follow from Lemma 6.4. □

Corollary 8.11. *For a set* $P \subset \mathbb{R}^2$ *of* n *points in general position, the Voronoi diagram of* P *can be constructed in expected* $O(n \log n)$ *time and* $O(n)$ *space.*

Proof. We have seen that a Delaunay triangulation for P can be obtained using randomized incremental construction within the asserted time and space bounds. Using the correspondence between Voronoi vertices/edges and Delaunay triangles/edges, we may generate the Voronoi diagram in $O(n)$ additional time and space. □

Exercise 8.12. *Consider the Delaunay triangulation* $\mathcal{T}$ *for a set* $P \subset \mathbb{R}^2$ *of* $n \geqslant 3$ *points in general position. Prove or disprove:*

(a) Every edge of $\mathcal{T}$ *intersects its dual Voronoi edge.*

(b) Every vertex of $VD(P)$ *is contained in its dual Delaunay triangle.*

Exercise 8.13. *Given a Voronoi diagram of some unknown point set* P*, can you compute* P *along with a Delaunay triangulation in linear time?*

8.4 A Lifting Map View (not exam-relevant in HS2025)

Recall that the lifting map $\ell\colon (x, y) \mapsto (x, y, x^2+y^2)$ raises a point in the plane vertically to the unit paraboloid $\mathcal{U}$ in $\mathbb{R}^2$. We used it in Section 6.3 to prove that the Lawson Flip Algorithm terminates. Interestingly, it also plays a role here with Voronoi diagrams.

For $p \in \mathbb{R}^2$ let $H_p \subseteq \mathbb{R}^3$ be the plane tangent to $\mathcal{U}$ at $\ell(p)$. We denote by $h_p(q)$ the vertical projection of a point $q \in \mathbb{R}^2$ onto the plane H_p (see Figure 8.7).

Lemma 8.14. $\|\ell(q) - h_p(q)\| = \|p - q\|^2$*, for any points* $p, q \in \mathbb{R}^2$*.*

Exercise 8.15. *Prove Lemma 8.14.* Hint: *First determine the equation of the plane* H_p *tangent to* $\mathcal{U}$ *at* $\ell(p)$*.*

Theorem 8.16. *Let* H_p^+ *be the closed halfspace above plane* H_p*. Define* $\mathcal{H} := \bigcap_{p \in P} H_p^+$*. Then the vertical projection of* $\partial\mathcal{H}$ *onto the* xy*-plane forms the Voronoi diagram of* P*. That is, the faces/edges/vertices of* $\partial\mathcal{H}$ *project to Voronoi cells/edges/vertices.*

Proof. Consider a point q' on the face defined by the plane H_p. Let $q \in \mathbb{R}^2$ be its vertical projection onto the xy-plane, so $q' = h_p(q)$. Note that $\ell(q)$ is above q', while all planes other than H_p are below q', thus $\|\ell(q) - h_p(q)\| \leqslant \|\ell(q) - h_r(q)\|$ for all $r \in P$. By Lemma 8.14, p is the closest point from P to q. □

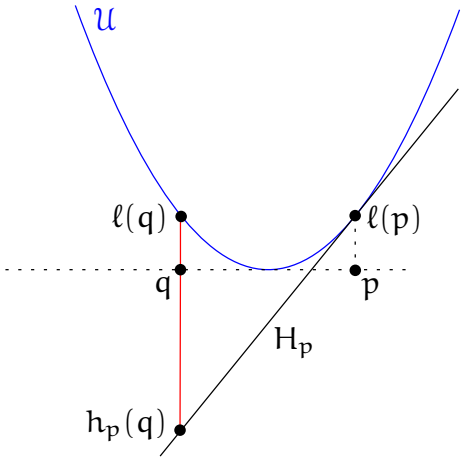


Figure 8.7: *A cross section of the lifting map interpretation for Voronoi diagram.*

8.5 Planar Point Location

One last bit is still missing in order to solve the post office problem optimally.

Theorem 8.17. *Given a triangulation $\mathcal{T}$ for a set $P \subset \mathbb{R}^2$ of n points, one can build in $O(n)$ time and space a data structure which allows for finding in $O(\log n)$ time a triangle $\Delta \in \mathcal{T}$ that contains any given query point $q \in \text{conv}(P)$.*

The data structure is known as *Kirkpatrick's hierarchy*. Before discussing it in detail, let us put things together to solve the post office problem.

Corollary 8.18 (Nearest Neighbor Search). *Given a set $P \subset \mathbb{R}^2$ of n points, one can build in expected $O(n \log n)$ time an $O(n)$ size data structure which allows for reporting in $O(\log n)$ time a nearest point $p \in P$ to any given query point $q \in \text{conv}(P)$.*

Proof. First we construct the Voronoi diagram $\text{VD}(P)$ in expected $O(n \log n)$ time; see Corollary 8.11. It has exactly n convex cells. Truncate every unbounded cell by $\partial\text{conv}(P)$ into a bounded one, since we are concerned with query points within $\text{conv}(P)$ only.[2] Now that all the cells are convex polygons, we may triangulate all of them in overall $O(n)$ time (the procedure only traverses each edge twice, and the number of edges is $O(n)$ by Corollary 8.10). We put a label "p_i" on all triangles in cell $V_p(i)$. Now we have a triangulation of the point set P. Apply Theorem 8.17 to build Kirkpatrick's hierarchy, which takes $O(n)$ time and space.

When we receive a query point $q \in \text{conv}(P)$, we use the data structure to find in $O(\log n)$ time a triangle containing q. Output the label of the triangle, which is exactly the nearest point from P to q. □

[2] We even know how to decide in $O(\log n)$ time whether or not a given point lies within $\text{conv}(P)$, see Exercise 5.29.

8.6 Kirkpatrick's Hierarchy

We will now develop a data structure for point location in a triangulation, as described in Theorem 8.17. For simplicity we assume that the triangulation $\mathcal{T}$ we work with is maximal planar, that is, the outer face is a triangle as well. This can easily be achieved by wrapping a huge triangle Δ around $\mathcal{T}$ and triangulating the vacuum in between Δ and $\mathcal{T}$ in linear time (how?).

The main idea for the data structure is to construct a sequence

$$\mathcal{T} = \mathcal{T}_0,\ \mathcal{T}_1,\ \ldots,\ \mathcal{T}_{h-1},\ \mathcal{T}_h = \{\Delta\}$$

of triangulations such that the vertices of $\mathcal{T}_i$ are a proper subset of the vertices of $\mathcal{T}_{i-1}$, for $i = 1, \ldots, h$. Hence the triangulations get coarser as we move forward.

Given a query point q, we hunt for a triangle in $\mathcal{T}_0$ that contains q by tracing the sequence backwards:

- Start from the big triangle $\Delta \in \mathcal{T}_h$ which certainly contains q;
- Then find a triangle in the finer triangulation $\mathcal{T}_{h-1}$ that contains q;
- ...
- Finally, find a triangle in the target triangulation $\mathcal{T}_0$ that contains q.

Locating the query point.

1. Let $T_h := \Delta$.
2. For each $i = h, \ldots, 1$, examine all triangles in $\mathcal{T}_{i-1}$ that intersects T_i, until we find a triangle T_{i-1} that contains q.
3. Output T_0.

Proposition 8.19. *The search procedure can be implemented to use at most* $3ch$ *orientation tests, provided every triangle in* $\mathcal{T}_i$ *intersects at most* c *triangles in* $\mathcal{T}_{i-1}$.

Proof. In the data structure we link each triangle in $\mathcal{T}_i$ to at most c intersecting triangles in $\mathcal{T}_{i-1}$. With this implementation, step 2 examines at most ch triangles in total. For each triangle, three orientation tests suffice to determine if it contains q. □

We will show next how to construct the sequence so that both c and h are small. Concretely, we will make c a constant and $h = O(\log n)$.

Thinning. Suppose we have $\mathcal{T}_{i-1}$ at hand and want to construct $\mathcal{T}_i$ by removing several vertices and re-triangulating. Note that removing a vertex p (and its incident edges) from $\mathcal{T}_{i-1}$ creates a hole, which is a star-shaped polygon with p being its star-point.

Lemma 8.20. *A star-shaped polygon, given as a sequence of* $n \geqslant 3$ *vertices and a star-point, can be triangulated in* $O(n)$ *time.*

Exercise 8.21. *Prove Lemma 8.20.*

As a side remark, the *kernel* of a simple polygon, that is, the (possibly empty) set of all star-points, can be constructed in linear time as well [8].

Since we want h to be small, we had better remove a decent number of vertices. These vertices should have low degrees, since the degree is a natural upper bound for the number of triangles in $\mathcal{T}_{i-1}$ intersecting the triangles after re-triangulation.

Our working plan is thus to remove a constant proportion of *independent* (i.e. pairwise non-adjacent) low-degree vertices. The following lemma asserts the existence of such a set of vertices in every triangulation.

Lemma 8.22. *In every triangulation of $n \geqslant 3$ points, there is a set of at least $\lceil n/18 \rceil$ independent vertices whose degrees are at most 8. Moreover, such a set can be found in $O(n)$ time.*

Proof. Let $G = (V, E)$ denote the graph of the triangulation, which we treat as an abstract planar graph. We may assume without loss of generality that G is maximal planar. (Otherwise use Theorem 2.34 to combinatorially triangulate G arbitrarily in linear time. Any independent set in the resulting graph is independent in the old graph, and the degree of a vertex can only increase.)

For $n = 3$ the statement is trivially true. Next assume $n \geqslant 4$. The total degree of G is $\sum_{v \in V} \deg_G(v) = 2|E| < 6n$ by Corollary 2.5. On the other hand, G is 3-connected by Theorem 2.31, so every vertex has degree at least 3. Let $W \subseteq V$ denote the set of vertices of degree at most 8. Then we have

$$\begin{aligned} 6n > \sum_{v \in V} \deg_G(v) &= \sum_{v \in W} \deg_G(v) + \sum_{v \in V \setminus W} \deg_G(v) \\ &\geqslant 3|W| + 9(n - |W|) = 9n - 6|W|, \end{aligned}$$

hence $|W| > n/2$.

Let us pick an independent set greedily: In each iteration, pick a remaining vertex in W, then eliminate itself and its neighbors. Repeat until all vertices in W have been eliminated.

By construction, the picked vertices are independent and have degrees at most 8. Each iteration eliminates at most nine vertices (the picked vertex and its at most eight neighbors) from W, so upon termination we have picked at least $|W|/9 \geqslant \lceil n/18 \rceil$ vertices.

Regarding the running time, if G is represented by adjacency lists, for example, we can obtain the neighborhood of any vertex $v \in W$ in $\deg_G(v) = O(1)$ time. As there are at most $|W|$ iterations, the greedy procedure runs in overall $O(n)$ time. □

Proof of Theorem 8.17. We construct the hierarchy $\mathcal{T}_0, \ldots \mathcal{T}_h$ iteratively. Let $\mathcal{T}_0 = \mathcal{T}$, and derive $\mathcal{T}_i$ from $\mathcal{T}_{i-1}$ as follows. We remove an independent set of $\mathcal{T}_{i-1}$ provided by Lemma 8.22. This creates several holes, each being a star-shaped polygon. We re-triangulate the holes by Lemma 8.20, and the result is $\mathcal{T}_i$. Each new triangle in $\mathcal{T}_i$ keeps

pointers to the intersecting triangles in $\mathcal{T}_{i-1}$; the number of needed pointers is at most $c = 8$.

The above steps cost time linear n_i, the number of vertices in $\mathcal{T}_i$. Since at least $\lceil n_{i-1}/18 \rceil$ vertices are removed, we have

$$n_i \leqslant \frac{17}{18} n_{i-1} \leqslant \cdots \leqslant \left(\frac{17}{18}\right)^i n$$

Therefore, the total cost for building the hierarchy is proportional to

$$\sum_{i=0}^{h} n_i \leqslant n \cdot \sum_{i=0}^{h} \left(\frac{17}{18}\right)^i < n \cdot \sum_{i=0}^{\infty} \left(\frac{17}{18}\right)^i = 18n \in O(n).$$

Similarly the space consumption is linear.

The number of levels amounts to $h = \log_{18/17} n$. Thus by Proposition 8.19 each query needs at most $3ch = 3 \cdot 8 \cdot \log_{18/17} n < 292 \log n$ orientation tests. □

Improvements. As the name suggests, the hierarchical approach discussed above is due to David Kirkpatrick [6]. The constant 292 that appears in the query time is somewhat formidable. There has been a whole line of research trying to improve it using different techniques.

- Sarnak and Tarjan [9]: $4 \log n$.
- Edelsbrunner, Guibas, and Stolfi [4]: $3 \log n$.
- Goodrich, Orletsky, and Ramaiyer [5]: $2 \log n$.
- Adamy and Seidel [1]: $1 \log n + 2\sqrt{\log n} + O(\sqrt[4]{\log n})$.

Comparison with history graph. Similar to Kirkpatrick's hierarchy, the history graph for the incremental construction (Chapter 7) is also used to locate query points in a triangulation. But the two data structures have fundamental differences. First, the history graph is built during the construction of a Delaunay triangulation, whereas Kirkpatrick's hierarchy is built on top of any given triangulation (in our case a triangulation of the Voronoi diagram). Second, the history graph does not guarantee a logarithmic time for an arbitrary query point—not even in the probabilistic sense. In fact, the analysis there only bounds the expected total running time over all rounds, and the random choice of insertion (=query) points turns out crucial.

Exercise 8.23. *Let $\{p_1, p_2, \ldots, p_n\}$ be a set of points in the plane, which we call* obstacles*. Imagine there is a disk of radius r centered at the origin which can be moved around the obstacles but is not allowed to intersect them (touching the boundary is okay). Is it possible to move the disk out of these obstacles? See Figure 8.8.*

More formally, the question is whether there is a continuous curve $\gamma : [0, 1] \to \mathbb{R}^2$ with $\gamma(0) = (0, 0)$ and $\|\gamma(1)\| \geqslant \max\{\|p_1\|, \ldots, \|p_n\|\}$, such that at any time $t \in [0, 1]$

and $\|\gamma(t)-p_i\| \geqslant r$*, for any* $1 \leqslant i \leqslant n$*. Describe an algorithm to decide this question and to construct such a path—if one exists—given arbitrary points* $\{p_1, p_2, \ldots, p_n\}$ *and a radius* $r > 0$*. Argue why your algorithm is correct and analyze its running time.*

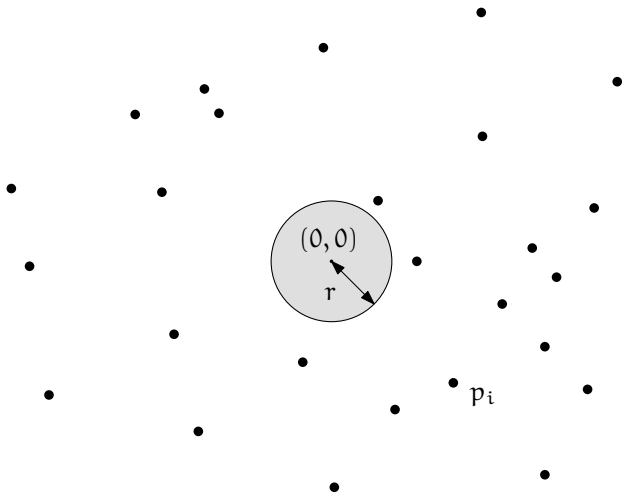


Figure 8.8: *Motion planning: Illustration for Exercise 8.23.*

Exercise 8.24. *This exercise is about an application from* Computational Biology*: You are given a set of disks* $P = \{a_1, .., a_n\}$ *in* $\mathbb{R}^2$*, all with the same radius* $r_a > 0$*. Each of these disks represents an atom of a protein. A water molecule is represented by a disc with radius* $r_w > r_a$*. A water molecule cannot intersect the interior of any protein atom, but it can be tangent to one. We say that an atom* $a_i \in P$ *is* accessible *if there exists a placement of a water molecule such that it is tangent to* a_i *and does not intersect the interior of any other atom in* P*. Given* P*, find an* $O(n \log n)$ *time algorithm which determines all atoms of* P *that are inaccessible.*

Exercise 8.25. *Let* $P \subset \mathbb{R}^2$ *be a set of* n *points. Describe a data structure to find in* $O(\log n)$ *time a point in* P *that is furthest from a given query point* q *among all points in* P*.*

Exercise 8.26. *Show that the bounds given in Theorem 8.17 are optimal in the algebraic computation tree model.*

Questions

34. *What is the Voronoi diagram of a set of points in* $\mathbb{R}^2$*?* Give a precise definition and explain/prove the basic properties: convexity of cells, why is it a subdivision of the plane?, Lemma 8.6, Lemma 8.8.

35. *What is the correspondence between the Voronoi diagram and the Delaunay triangulation for a set of points in* $\mathbb{R}^2$*?* Prove duality (Theorem 8.9) and explain where general position is needed.

36. *How to construct the Voronoi diagram of a set of points in* $\mathbb{R}^2$*?* Describe an $O(n \log n)$ time algorithm, for instance, via Delaunay triangulation.

37. *What is the Post-Office Problem and how can it be solved optimally?* Describe the problem and a solution using linear space, $O(n \log n)$ preprocessing, and $O(\log n)$ query time.

38. *How does Kirkpatrick's hierarchical data structure for planar point location work exactly?* Describe how to build it and how the search works, and prove the runtime bounds. In particular, you should be able to state and prove Lemma 8.22 and Theorem 8.17.

39. **(This topic was not covered in HS25 and therefore the question will not be asked in the exam.)***How can the Voronoi diagram be interpreted in context of the lifting map?* Describe the transformation and prove its properties to obtain a formulation of the Voronoi diagram as an intersection of halfspaces one dimension higher.

References

[1] Udo Adamy and Raimund Seidel, On the exaxt worst case query complexity of planar point location. *J. Algorithms*, 37, (2000), 189–217.

[2] Franz Aurenhammer, Voronoi diagrams: A survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23/3, (1991), 345–405.

[3] Boris Delaunay, Sur la sphère vide. A la memoire de Georges Voronoi. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskih i Estestvennyh Nauk*, 6, (1934), 793–800.

[4] Herbert Edelsbrunner, Leonidas J. Guibas, and Jorge Stolfi, Optimal point location in a monotone subdivision. *SIAM J. Comput.*, 15/2, (1986), 317–340.

[5] Michael T. Goodrich, Mark W. Orletsky, and Kumar Ramaiyer, Methods for achieving fast query times in point location data structures. In *Proc. 8th ACM-SIAM Sympos. Discrete Algorithms*, pp. 757–766, 1997.

[6] David G. Kirkpatrick, Optimal search in planar subdivisions. *SIAM J. Comput.*, 12/1, (1983), 28–35.

[7] Rolf Klein, *Concrete and abstract Voronoi diagrams*, vol. 400 of *Lecture Notes Comput. Sci.*, Springer, 1989.

[8] Der-Tsai Lee and Franco P. Preparata, An optimal algorithm for finding the kernel of a polygon. *J. ACM*, 26/3, (1979), 415–421.

[9] Neil Sarnak and Robert E. Tarjan, Planar point location using persistent search trees. *Commun. ACM*, 29/7, (1986), 669–679.

Chapter 9

Arrangements

During this course we encountered several situations where it was convenient to assume a point set "in general position". In the plane it usually means no three points are collinear and/or no four points are cocircular. This raises an algorithmic question: How do we test for n given points whether or not three of them are collinear? The exact computational complexity of this innocent-looking problem is a major open problem in theoretical computer science. Obviously we could test all triples in $O(n^3)$ time, but can we do better? As it turns out: Yes, we can! We will see how to employ the so-called *projective duality transform* to solve the problem in $O(n^2)$ time. Despite some interest, nobody knows if and how general position testing can be done in subquadratic time; the only known lower bound is $\Omega(n \log n)$.

The aforementioned duality transformation is interesting because it offers an equivalent *dual* perspective to a problem. Sometimes the dual form is easier to work with, and its solution can be efficiently translated back into a solution to the original or *primal* form. So what is this transformation about? Recall that a hyperplane in $\mathbb{R}^d$ is the set of solutions $x \in \mathbb{R}^d$ to a linear equation $\sum_{i=1}^{d} h_i x_i = h_{d+1}$, where at least one of $h_1, \ldots, h_d$ is nonzero. If $h_d = 1$, we call the hyperplane *non-vertical*. Now observe that points and non-vertical hyperplanes in $\mathbb{R}^d$ can both be described by d real numbers. It is thus tempting to map them to each other. In $\mathbb{R}^2$, hyperplanes are lines and the standard **projective duality transform** maps a point $p = (a, b)$ to the non-vertical line $p^* : y = ax - b$, and a non-vertical line $\ell : y = ax + b$ to the point $\ell^* := (a, -b)$.

Proposition 9.1. *The standard projective duality transform is*

- *incidence preserving:* $p \in \ell \iff \ell^* \in p^*$ *and*
- *order preserving:* p *is above* $\ell \iff \ell^*$ *is above* p^*.

Exercise 9.2. *Prove Proposition 9.1.*

Exercise 9.3. *For each of the following point sets, what image do we get after applying the duality transform pointwise?*

(a) $k \geqslant 3$ *collinear points;*

(b) a line segment;

(c) a halfplane;

(d) the boundary points of the upper convex hull of a finite point set.

One can also visualize duality in terms of the parabola $\mathcal{P} : y = \frac{1}{2}x^2$. Let $p = (a, b)$ be any point. If it is on $\mathcal{P}$, then its dual line p^* is the tangent to $\mathcal{P}$ at p. Otherwise we consider its vertical projection $p' := (a, \frac{1}{2}a^2)$ onto $\mathcal{P}$. As we argued, $(p')^*$ is the tangent to $\mathcal{P}$ at p'. Note that the slopes of p^* and $(p')^*$ are the same, and p^* is just $(p')^*$ shifted vertically by $\frac{1}{2}a^2 - b$.

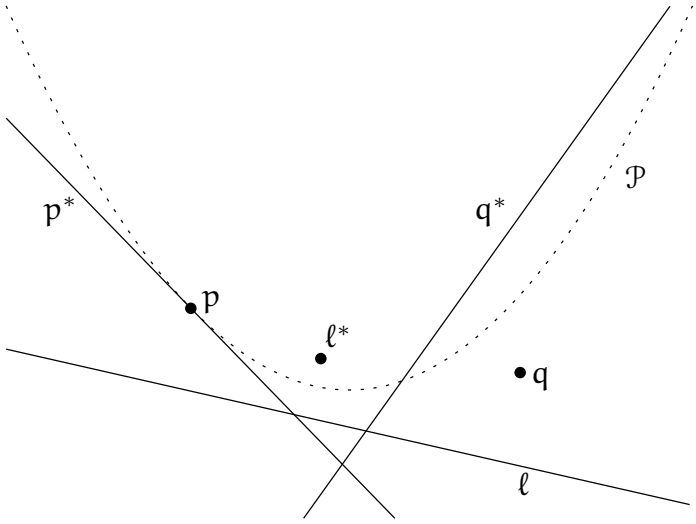


Figure 9.1: *Point ↔ line duality through the lens of the parabola* $\mathcal{P} : y = \frac{1}{2}x^2$.

The problem of whether or not three points in the primal plane are collinear transforms to whether or not three lines in the dual plane meet in a common point. We will solve the latter problem with the help of *line arrangements*, as defined below.

9.1 Line Arrangements

The subdivision of the plane induced by a finite set L of lines is called the **line arrangement** $\mathcal{A}(L)$. We may imagine its creation as follows. First, from the plane $\mathbb{R}^2$ we subtract all lines in L (considered as point sets), thus breaking $\mathbb{R}^2$ into one or more open connected regions. Note that every region is an intersection of open halfplanes and hence convex. We call these regions the (2-dimensional) **cells** of the arrangement. Next, from each line $\ell \in L$ we subtract all the other lines, thus splitting ℓ into one or more open connected components. These collectively form the 1-dimensional cells or **edges**. What remains are the intersection points of lines from L. They are called the 0-dimensional cells or **vertices**.

The notion naturally generalizes to curve arrangements in $\mathbb{R}^2$ and hyperplane arrangements in $\mathbb{R}^d$. Without further specification, the word "arrangement" refers to line

arrangements in this chapter. The **complexity** of an arrangement is just the total number of vertices, edges and cells (in general, the total number of cells of any dimension).

A line arrangement is called **simple** if no two lines are parallel and no three lines meet at a common point.

Theorem 9.4. *Every simple arrangement of n lines has $\binom{n}{2}$ vertices, n^2 edges, and $\binom{n}{2} + n + 1$ cells.*

Proof. Since every pair of lines intersect and all the intersection points are distinct, there are $\binom{n}{2}$ vertices.

We count the number of edges by induction on n. For $n = 1$ we have $1^2 = 1$ edge. By adding a new line to an arrangement of $n-1$ lines, we split $n-1$ existing edges into two and also introduce n edges along the new line. So inductively there are $(n-1)^2 + (n-1) + n = n^2$ edges in total.

The number f of cells can be obtained from Euler's formula. For this we need to treat the arrangement as a planar graph $G = (V, E)$ by adding a vertex at "infinity" which absorbs all unbounded edges. Then the faces of G one-to-one correspond to the cells of the arrangement, with $|V| = \binom{n}{2} + 1$ and $|E| = n^2$. By Euler's formula we have $|V| - |E| + f = 2$. It follows that

$$f = 2 - |V| + |E| = 1 - \binom{n}{2} + n^2 = 1 + \binom{n}{2} + n\,. \qquad \square$$

So the complexity of a simple arrangement is $\Theta(n^2)$. Tweaking the above proof, it is easy to see that the complexity of *any* line arrangement is $O(n^2)$.

Exercise 9.5. *Consider a set of lines in $\mathbb{R}^2$ with no three meeting at a common point. Form a plane graph G whose vertices are the intersection points of the lines. Two vertices are adjacent if and only if they appear consecutively along some line. Prove that G is 3-colorable. That is, we can paint the vertices using at most three colors so that adjacent vertices receive different colors.*

9.2 Constructing Line Arrangements

How do we store a line arrangement in computers? Although some cells and edges are unbounded, we can effectively bound the arrangement by a sufficiently large box that cages all vertices. Such a box can be constructed in $O(n \log n)$ time for n lines.

Exercise 9.6. *How?*

Moreover, as we have seen in the previous proof, we can view the arrangement as a planar graph by adding a symbolic vertex that absorbs all (unbounded) edges leaving the box. For algorithmic purposes, we usually represent the graph by a doubly connected edge list (DCEL), cf. Section 2.2.1.

How do we construct an arrangement algorithmically? We would be satisified with any $O(n^2)$ algorithm, as the worst case complexity of line arrangements is quadratic already. A natural approach is incremental construction: just insert the lines one by one in some arbitrary order $\ell_1, \ldots, \ell_n$.

At Step i, we want to insert the line ℓ_i into the arrangement $\mathcal{A}(\{\ell_1, \ldots, \ell_{i-1}\})$ of the first $i-1$ lines. Suppose that the edges that leave the left side of the bounding box are ordered by slope. Hence we can locate in $O(i)$ time the leftmost cell f in $\mathcal{A}(\{\ell_1, \ldots, \ell_{i-1}\})$ that ℓ_i intersects. Then we traverse the boundary of f counterclockwise, until we intersect ℓ_i again when walking along some halfedge h (see Figure 9.2 for illustration). Insert a new vertex at this intersection point, split f and h accordingly, and continue in the same way with the cell on the twin side of h.

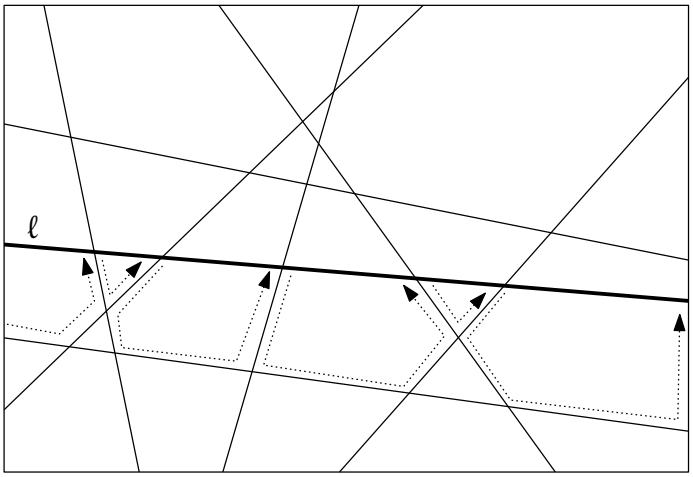


Figure 9.2: *Incremental construction: Insertion of a line ℓ. (Only part of the arrangement is shown in order to increase readability.)*

Each insertion and split is a constant time operation. But what is the time needed for the traversal? The worst case complexity of $\mathcal{A}(\{\ell_1, \ldots, \ell_{i-1}\})$ is $\Omega(i^2)$, but maybe not all cells and edges are relevant to our traversal. In fact, the relevant zone has linear complexity only, as we will show next.

9.3 Zone Theorem

For an arrangement $\mathcal{A}(L)$ and an arbitrary line ℓ (not necessarily from L), the **zone** $Z_{\mathcal{A}(L)}(\ell)$ is the set of cells from $\mathcal{A}(L)$ whose closure intersects ℓ.

Theorem 9.7. *Given an arrangement $\mathcal{A}(L)$ of n lines and a line ℓ, there are at most $10n$ edges in all cells of $Z_{\mathcal{A}(L)}(\ell)$.*

Proof. Fix the line ℓ and assume with loss of generality that it is horizontal (otherwise rotate the plane accordingly). For each cell, orient its horizontal edges from left to right,

and the other edges from bottom to top. An oriented edge is *left-bounding* if the cell is to its right; otherwise it is *right-bounding*. Figure 9.3 gives an example.

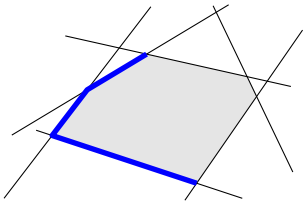


Figure 9.3: *Left-bounding edges (blue and bold) of a cell.*

We will show that there are at most $5n$ left-bounding edges in all cells of $Z_{\mathcal{A}(L)}(\ell)$ by induction on $n = |L|$. By symmetry, the same also holds for the number of right-bounding edges, thus the theorem follows.

For $n = 1$, there is at most $1 < 5n$ left-bounding edge in $Z_{\mathcal{A}(L)}(\ell)$. (The number could be zero because the only line in L might be horizontal and above ℓ.)

Now assume the statement is true for $n-1$. If ℓ does not intersect with L, then all lines are horizontal and there is at most $1 < 5n$ left-bounding edge in $Z_{\mathcal{A}(L)}(\ell)$. Else let p be the rightmost point where ℓ intersects with L. We distinguish two cases:

Case 1: there is a unique line $r \in L$ through p. Consider the zone $Z_{\mathcal{A}(L\setminus\{r\})}(\ell)$, namely the cells of the arrangement $\mathcal{A}(L \setminus \{r\})$ that touch ℓ. They have at most $5n-5$ left-bounding edges by the induction hypothesis. Now we add r back to the arrangement and see what happens. Let R be the rightmost cell in $Z_{\mathcal{A}(L\setminus\{r\})}(\ell)$. The line r intersects ∂R in at most two points and thus splits at most two edges of R, both of which may be left-bounding. Denote the underlying lines of these edges by ℓ_0 and ℓ_1. Further, the new edge $r \cap R$ is left-bounding for the rightmost cell R' of $Z_{\mathcal{A}(L)}(\ell)$. So locally the number of left-bounding edges increases by at most three.

Note that r does not contribute left-bounding edges to any cell of $Z_{\mathcal{A}(L)}(\ell)$ other than R': To any cell of $Z_{\mathcal{A}(L)}(\ell)$ that lies to the left of r, the line r can contribute right-bounding edges only; and any cell other than R' that lies to the right of r is shielded away from ℓ by one of ℓ_0 or ℓ_1, that is, any such cell it not part of $Z_{\mathcal{A}(L)}(\ell)$. Therefore, the total number of left-bounding edges in $Z_{\mathcal{A}(L)}(\ell)$ is at most $(5n-5)+3 < 5n$.

Case 2: there are multiple lines through p. We add these lines to the arrangement of the remaining lines one by one. Adding the first such line r creates at most three left-bounding edges by the analysis from Case 1. We claim that adding any further line r' that passes through p creates at most five left-bounding edges. To see this, we argue as in Case 1: The line r' splits at most two left-bounding edges of R, and it also splits the left-bounding edge of R' induced by r. Finally, the line r' itself provides two left-bounding edges (joined at the point p). Thus, the number of left-bounding edges increases by at most five for each additional edge. This proves the claim and concludes the proof of the theorem. □

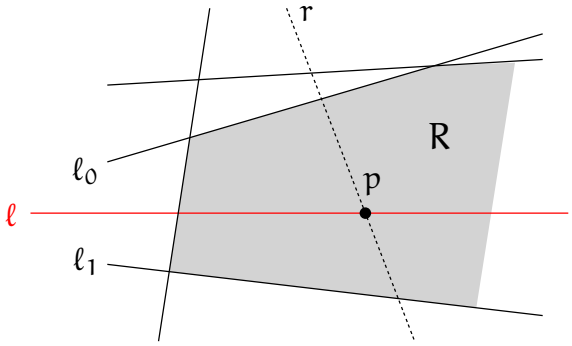


Figure 9.4: *At most three new left-bounding edges are created by adding* r *to* $\mathcal{A}(L\setminus\{r\})$.

Corollary 9.8. *The arrangement of* n *lines in* $\mathbb{R}^2$ *can be constructed in optimal* $\Theta(n^2)$ *time and space.*

Proof. Use the incremental construction described earlier. At Step $i = 1, \ldots, n$, we do a linear search among $i-1$ leftmost edges to find the starting cell and then traverse (part of) the zone of the line ℓ_i in the arrangement $\mathcal{A}(\{\ell_1, \ldots, \ell_{i-1}\})$. By Theorem 9.7 the complexity of this zone and hence the time complexity of Step i altogether is $O(i)$. Overall we obtain $\sum_{i=1}^{n} ci = O(n^2)$ time (and space), for some constant $c > 0$, which is optimal by Theorem 9.4. □

Generally in $\mathbb{R}^d$, a simple hyperplane arrangement has complexity $\Theta(n^d)$, and a zone of a hyperplane has complexity $O(n^{d-1})$.

Exercise 9.9. *For an arrangement* $\mathcal{A}$ *of a set of* n *lines in* $\mathbb{R}^2$*, let*

$$\mathcal{F} := \bigcup_{C \text{ is a bounded cell of } \mathcal{A}} \overline{C}$$

be the union of the closure of all bounded cells. Show that the complexity (number of vertices and edges of the arrangement lying on the boundary) of $\mathcal{F}$ *is* $O(n)$.

9.4 General Position and Minimum Triangle

The real beauty and power of line arrangements manifest through the projective duality. It is often convenient to assume that no two points in the primal plane have the same x-coordinate, so that no line through two points in the primal is vertical (and hence becomes an infinite point in the dual). This degeneracy can be discovered by sorting the points according to x-coordinate, and resolved by rotating the whole plane by a sufficiently small angle ε. To select ε, we can iterate over all non-vertical lines through two points, compute its (absolute) angle to verticality, and then choose ε to be strictly less than all these angles. The procedure clearly runs in $O(n^2)$ time.

Therefore, the following two problems can be solved in $O(n^2)$ time and space by constructing the dual arrangement.

General position test. Given n points in $\mathbb{R}^2$, are there three collinear points? (Dual: do three of the n dual lines meet at a common point?)

Minimum area triangle. Given a set $P \subset \mathbb{R}^2$ of n points, what is the minimum area triangle spanned by three distinct points from P? This can be viewed as a quantified version of general position test: the minimum area is zero if and only if there are three collinear points.

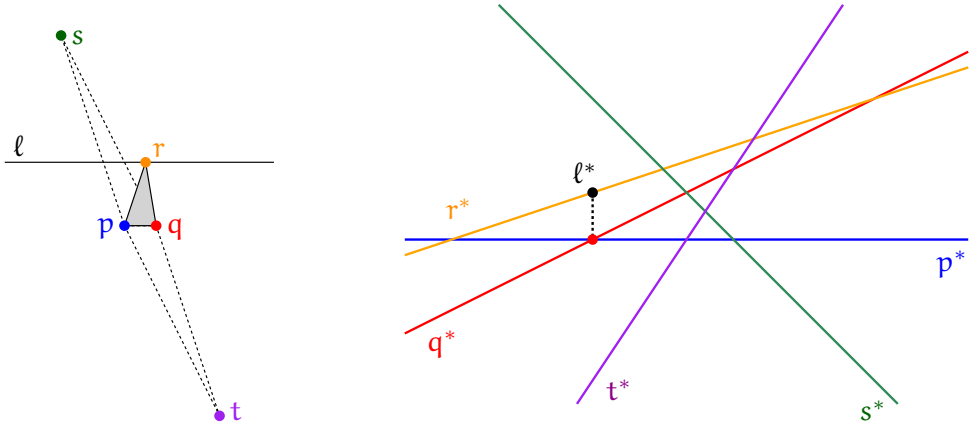
Let us make the problem easier by fixing two distinct points $p, q \in P$ and ask for a minimum area triangle pqr, where $r \in P \setminus \{p, q\}$. With pq fixed, the area of pqr is proportional to the distance between r and the line pq. Thus we want to find

> a closest line ℓ parallel to pq and passing through some point $r \in P \setminus \{p, q\}$. $(\star)$

Consider the set $P^* := \{p^* : p \in P\}$ of dual lines and their arrangement $\mathcal{A}$. In $\mathcal{A}$ the statement $(\star)$ translates to

> a closest point ℓ^* with the same x-coordinate as the vertex $p^* \cap q^*$ and lying on some line $r^* \in P^*$.

See Figure 9.5 for illustration. In other words, for the vertex $p^* \cap q^*$ of $\mathcal{A}$ we want to find a line $r^* \in P^*$ closest to it vertically—above or below. Of course, in the end we want this information not only for one particular vertex (which provides the minimum area triangle for fixed p, q) but for *all* vertices of $\mathcal{A}$, that is, for all possible pairs of fixed points $\{p, q\} \in \binom{P}{2}$.



(a) primal **(b)** dual

Figure 9.5: *Minimum area triangle spanned by two fixed points* p, q.

Luckily, such information can be maintained over the incremental construction of $\mathcal{A}$. When inserting a new line ℓ, it may become the vertically closest line from vertices of the already computed partial arrangement. However, only vertices in the zone of ℓ may be affected. The zone is traversed anyway, so in the same pass we can update the

information for vertices immediately and vertically above or below ℓ, at no extra cost asymptotically.

By the end of the incremental construction, each vertex $p^* \cap q^*$ has recorded a vertically closest line, for $\{p, q\} \in \binom{P}{2}$. These correspond to minimum area triangles for every fixed p, q. The smallest such candidate can be found by simply comparing their areas, which takes $O(n^2)$ time.

Exercise 9.10. *A set* P *of* n *points in the plane is said to be in* ε-general position *for* $\varepsilon > 0$ *if no three points of the form*

$$p + (x_1, y_1),\ q + (x_2, y_2),\ r + (x_3, y_3)$$

are collinear, where $p, q, r \in P$ *and* $|x_i|, |y_i| < \varepsilon$*, for* $i \in \{1, 2, 3\}$*. In words: The set* P *remains in general position under changing point coordinates by less than* ε *each.*

Give an algorithm with runtime $O(n^2)$ *for checking whether a given point set* P *is in* ε*-general position.*

Exercise 9.11. *Let* F *be a family of vertical line segments such that for each three of them, there exists a line that intersects all three. Show that there exists a line which intersects all line segments in* F*.*

9.5 Constructing Rotation Systems

Here is an application of line arrangements with a different flavor. Recall the notion of a combinatorial embedding from Chapter 2. It is specified by the circular order of boundary edges of each face. Equivalently, we may represent it by the circular order of edges around each vertex. This latter view is called a rotation system.

In a similar way we can also condense the geometry of a finite point set $P \subset \mathbb{R}^2$ combinatorially. For a point $q \in P$ let $c_P(q)$ denote the circular sequence of points from $P \setminus \{q\}$ around q (that is, in the order as they would be encountered by a ray sweeping around q). The **rotation system** of P is nothing but $\{c_P(q) : q \in P\}$.[1]

Given a set P of n points, it is trivial to construct its rotation system in $O(n^2 \log n)$ time, by sorting each of the n lists independently. But in fact we can do it optimally in $O(n^2)$ time by duality transform.

Consider a directed line sweeping counterclockwise around a point $q \in P$ in the primal plane, initially vertically downward. The slope increases from $-\infty$ to ∞ until the line becomes vertically upward. This finishes the right half of the circular sweep. The left half is similar: the slope changes from $-\infty$ to ∞, until the line completes a full circle.

In the dual plane, this corresponds to traversing the line q^* from $-\infty$ to ∞ twice. (The sweeping line ℓ always goes through the point q in the primal, so ℓ^* always sits on q^* in the dual, and its x-coordinate reflects the slope of ℓ.) Hence we may retrieve the

[1] You may also think of it as the "combinatorial embedding" of the complete geometric graph on P. But as these graphs are not planar for $|P| \geqslant 5$, they do not have the notion of faces.

circular sequence $c_P(q)$ by two passes in the dual arrangement: In the first (respectively the second) traversal of q^* we record the sequence of intersections with p^* for all $p \in P$ to the right (respectively left) of q. The circular sequence is exactly the concatenation of the two. Clearly, the traversals along all lines in the dual arrangement can be done in $O(n^2)$ time.

Exercise 9.12 (Eppstein [6]). *Describe an $O(n^2)$ time algorithm that given a set P of n points in the plane finds a subset of five points that form a strictly convex empty pentagon (or reports that there is none if that is the case). Empty means that the convex pentagon may not contain any other points of P.*

Hint: *For each $p \in P$ discard all points to the left of p and consider the polygon $S(p)$ formed by p and the remaining points taken in circular order around p. Explain why it suffices to check for all p whether $S(p)$ has four vertices other than p that form an empty convex quadrilateral. How do you check this in $O(n^2)$ time?*

Remark: *It was shown by Harborth [11] that every set of ten or more points in general position contains a subset of five points that form a strictly convex empty pentagon.*

9.6 Segment Endpoint Visibility Graphs

In this section we present a more complex application of duality and line arrangements, in the context of motion planning. Here a fundamental problem is to find a short(est) path between two given positions in some domain, subject to certain constraints. As an example, suppose we are given two points $p, q \in \mathbb{R}^2$ and a set $S \subset \mathbb{R}^2$ that models obstacles. What is the shortest path between p and q that avoids S?

Observation 9.13. *Let S be a finite set of polygonal obstacles. The shortest path between two points that avoids S, if it exists, is a polygonal path whose vertices (except the two ends) are obstacle vertices.*

A simplest type of obstacles conceivable is a line segment. In general they might separate the two query points, say when they form a closed curve that surrounds one of the points. However, if we require the obstacles to be pairwise disjoint line segments then there is always a free path between any query points. Hence by the above observation we may restrict our attention to straight line edges connecting obstacle vertices—in our case, segment endpoints.

Definition 9.14. *Consider a set S of n disjoint line segments in $\mathbb{R}^2$. The* **segment endpoint visibility graph** *$\mathcal{V}(S)$ is a geometric graph defined on the segment endpoints. Two segment endpoints p and q are adjacent if they see each other; that is, if*

- *the line segment $\overline{pq}$ is in S, or*
- *$\overline{pq} \cap s \subseteq \{p, q\}$ for all $s \in S$.*

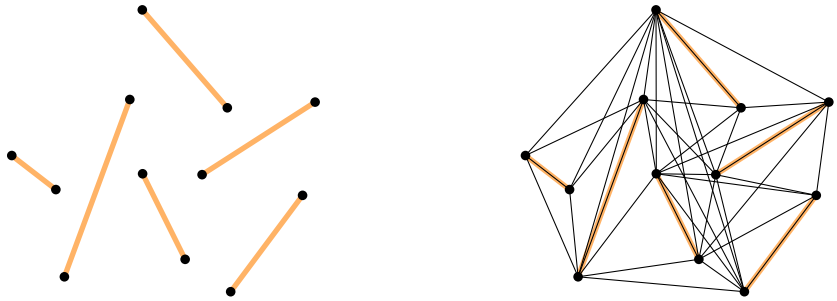


Figure 9.6: *A set of disjoint line segments and their endpoint visibility graph.*

If all segments are on the convex hull boundary, then the visibility graph is complete. If the segments form parallel chords of a convex polygon, then the visibility graph consists of copies of K_4 glued together side by side, and the number of edges is linear only.

On another note, these graphs also arise in the context of the following question: Given a set of disjoint line segments, can we link them at their endpoints via additional segments to form a closed Jordan curve? This is not always possible: Just consider three parallel chords of a convex polygon (Figure 9.7a). However, if we do not insist that all segments appear in the curve, but allow them to be diagonals or epigonals of the curve, then it is always possible [13, 14]. To rephrase: the segment endpoint visibility graph of disjoint line segments is Hamiltonian (unless all segments are collinear). It is actually essential to allow epigonals and not only diagonals [10, 22] (Figure 9.7b).

Figure 9.7: *Sets of disjoint line segments that do not admit certain polygons.*

Back to motion planning, our problem essentially reduces to finding a shortest path in the visibility graph $\mathcal{V}(S)$. But first of all, we need to construct the graph. Doing it in brute force takes $O(n^3)$ time where n denotes the number of segments in S. (Take all pairs of endpoints and check all other segments for obstruction.)

Theorem 9.15 (Welzl [23]). *The segment endpoint visibility graph of n disjoint line segments can be constructed in worst case optimal $O(n^2)$ time.*

Proof. Let P be the set of endpoints of S. As before we assume general position, so that no three points in P are collinear and no two have the same x-coordinate. (One can handle such degeneracy explicitly.)

Conceptually we perform a *rotational sweep*. That is, we rotate a direction vector v, initially pointing vertically downwards, in a counterclockwise fashion until it points

vertically upwards. While rotating, we maintain for each point $p \in P$ the segment $s(p)$ that it "sees" in direction ν (if any). Figure 9.8 shows an example. If ν is parallel to some segment $\overline{pq}$, say pointing in the direction of $\overrightarrow{pq}$, then we assign $s(p) := \overline{pq}$.

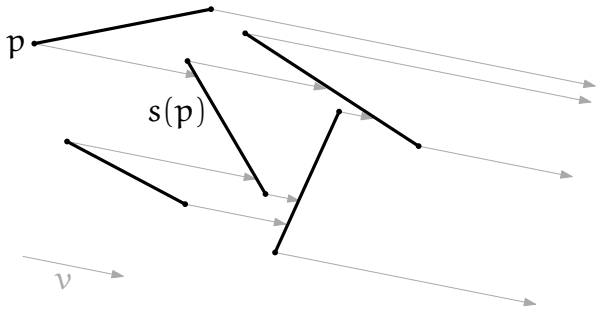


Figure 9.8: *Visible segments along a rotating direction ν. The arrow from an endpoint p indicates the segment $s(p)$ it sees; if the arrow extends to infinity, then no segment is visible from p.*

During the sweep, we can output the edges of the visibility graph on the fly: If currently q is an endpoint of the segment $s(p)$, and ν is in the direction of $\overrightarrow{pq}$, then we output the edge $\{p, q\}$.

Why does it work? Every output edge $\{p, q\}$ must be in $\mathcal{V}(S)$ because p sees $s(p) \ni q$ by definition. Conversely, let $\{p, q\}$ be an edge in $\mathcal{V}(S)$ where p is to the left of q, say. Assume that q is an endpoint of segment s. When ν sweeps over the direction of $\overrightarrow{pq}$, we must have $s(p) = s$ and the output condition is met.

To perform the actual sweep, we first observe that changes of visible segments can only occur discretely. Indeed, for any point $p \in P$, the segment $s(p)$ can only change when p sees some point $q \in P$ exactly in direction ν. Hence, we only need to consider directions $\overrightarrow{pq}$ with $p, q \in P$.

Hence let us call a pair $(p, q) \in \binom{P}{2}$, q to the right of p, an *event*. Its *slope* is the slope of the line through p and q. Then the sweep can be implemented as follows: process the events in order of increasing slope (by general position, all slopes are distinct); whenever we get to process an event (p, q), there are four cases (Figure 9.9):

(1) p and q belong to the same input segment $\implies$ output the edge $\{p, q\}$.

(2) q is obscured from p by $s(p)$ $\implies$ no change.

(3) q is an endpoint of $s(p)$ $\implies$ output $\{p, q\}$ and update $s(p)$ to $s(q)$.

(4) q is an endpoint of a segment s that now obscures $s(p)$ $\implies$ output $\{p, q\}$ and update $s(p)$ to s.

What is the runtime of this rotational sweep? We have $O(n^2)$ events, and sorting them in increasing slope takes $O(n^2 \log n)$ time. After this, the actual sweep takes $O(n^2)$ time, as each event is processed in constant time.

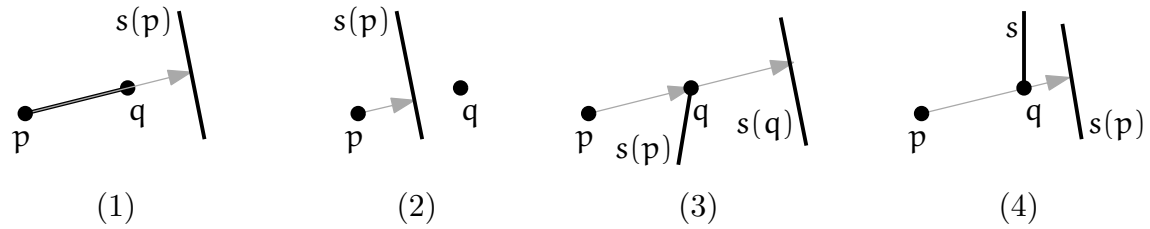


Figure 9.9: *Processing an event during the rotational sweep. Arrows indicate the segment* $s(p)$ *just before the event.*

To get rid of the $O(\log n)$ factor—we promised an $O(n^2)$ algorithm—we replace the sweep by a *topological* sweep, based on the observation that we do not strictly need to proceed by increasing slope. We just need property (a) below in order to correctly handle cases (1)(2)(4), while property (b) takes care of case (3).

(a) For each fixed $p \in P$, the events (p, q) are processed in order of increasing slope.

(b) When processing event (p, q), we have already processed the events (q, r) of smaller slope but not any event (q, r) of larger slope.

Any order of events that satisfies properties (a) and (b) will work. We can easily come by such an order when we interpret these properties in the arrangement $\mathcal{A}(P^*)$, where $P^* := \{p^* : p \in P\}$ is the projective dual of P. Recall that the slope of a line through two points $p, q \in P$ corresponds to the x-coordinate of the intersection of the dual lines p^*, q^*. Moreover, q is to the right of p if and only if q^* has larger slope than p^*. Hence, events in the dual are pairs of lines (p^*, q^*) where q^* has larger slope than p^*. The x-coordinate of an event is defined as the x-coordinate of the arrangement vertex $p^* \cap q^*$. Then, properties (a) and (b) translate to

(a*) For each fixed $p^* \in P^*$, the events (p^*, q^*) are processed in increasing x-coodinate.

(b*) When processing event (p^*, q^*), we have already processed the events (q^*, r^*) of smaller x-coordinate but not any event (q^*, r^*) of larger x-coodinate.

As the dual events correspond to arrangement vertices, properties (a*) and (b*) essentially say that if vertex u is to the left of vertex v *on the same line*, then u should appear before v. This notion coincides with the *topological order* on the *directed* arrangement graph with all edges directed from left to right. Clearly this directed graph is acyclic, so a topological order exists and can be computed, for instance, via (reversed) post order DFS in time linear in the size of the graph, which in our case is $O(n^2)$. □

Although the topological sweep is easy, the reader may ask whether the real sweep is also possible in $O(n^2)$ time. In other words: given a set of n points, is it possible to sort the $\binom{n}{2}$ lines defined by the points according to slope in $O(n^2)$ time? Standard lower bounds for sorting do not apply, since the $\binom{n}{2}$ slopes are highly interdependent.

The answer is unknown, and according to Exercise 9.16, the problem is at least as hard as another, rather prominent, open problem.

Exercise 9.16. *The $X+Y$ sorting problem is the following: given two sets X and Y of n distinct numbers each, sort the set $X+Y=\{x+y : x\in X,\ y\in Y\}$. It is an open problem whether this can be done in $o(n^2\log n)$ time.*

Prove that $X+Y$ sorting reduces (in $O(n)$ time) to the problem of sorting the $\binom{2n}{2}$ lines pq by slope, for all pairs $\{p,q\}$ from a set of $2n$ points.

9.7 3-Sum

We have seen a quadratic time algorithm to test whether a point set is in general position (no three points on a common line). But as we mentioned, the exact computational complexity of this problem is unsettled. In fact, it is closely related to an abstract problem called **3-Sum**, in the sense that resolving the complexity of one of the problems also resolves the complexity of the other. However, doing so is one of the major open problems in theoretical computer science.

The 3-Sum problem is the following: Given a set S of n integers, is there a triple[2] of elements from S that sum up to zero? Obviously, by testing all triples this can be solved in $O(n^3)$ time. If we pick the triples to be tested more cleverly, we obtain an $O(n^2)$ algorithm. To this end, we sort the elements from S in increasing order and obtain the sequence $s_1<\ldots<s_n$. This takes $O(n\log n)$ time. Then we test the triples as follows.

```
For i = 1,...,n {
    j = i, k = n.
    While k ⩾ j {
        If s_i + s_j + s_k = 0 then output the triple s_i, s_j, s_k and stop.
        If s_i + s_j + s_k > 0 then k = k − 1 else j = j + 1.
    }
}
```

The runtime is clearly quadratic. Regarding the correctness, observe the following invariant at the start of every iteration of the inner loop: $s_i+s_x+s_k<0$ for all $x\in\{i,\ldots,j-1\}$, and $s_i+s_j+s_x>0$ for all $x\in\{k+1,\ldots,n\}$.

Interestingly, until recently this was the fastest algorithm known for 3-Sum. But at FOCS 2014, Grønlund and Pettie [9] presented a deterministic algorithm that solves 3-Sum in $O(n^2(\log\log n/\log n)^{2/3})$ time.

They also gave an upper bound of $O(n^{3/2}\sqrt{\log n})$ on the decision tree complexity of 3-Sum, which since then has been further improved in a series of papers. The latest improvement is due to Kane, Lovett, and Moran [15] who showed that $O(n\log^2 n)$ linear

[2] That is, an element of S may be chosen twice or even three times, although the latter makes sense for the number 0 only.

queries suffice (each query amounts to asking for the sign of the weighted sum of six numbers in S, with coefficients in $\{-1, 0, 1\}$). In this decision tree model, only queries that involve the input numbers are counted, all other computation, for instance, using these query results to analyze the parameter space are for free. So the results demonstrate that the (supposed) hardness of 3-Sum does not originate from the complexity of the decision tree.

The big open question remains whether an $O(n^{2-\varepsilon})$ algorithm can be achieved. Only in some very restricted models of computation—such as the 3-linear decision tree model where a decision can only be based on the sign of a linear expression in 3 input variables—it is known that 3-Sum requires quadratic time [7].

There is a whole class of problems that are equivalent to 3-Sum up to sub-quadratic time reductions [8]; such problems are referred to as **3-Sum-hard**.

Definition 9.17. *A problem X is* **3-Sum-hard** *if every 3-Sum instance of size n reduces to solving a constant number of X instances of size $O(n)$, within $O(n^{2-\varepsilon})$ reduction time for some constant $\varepsilon > 0$.*

Exercise 9.18. *Show that the following variation of 3-Sum—call it 3-Sum$^\circ$—is 3-Sum-hard: Given a set S of n integers, are there three (distinct) elements of S that sum up to zero?*

As another example, consider the problem **GeomBase**: Given n points with integer coordinates on the horizontal lines $y = 0$, $y = 1$, and $y = 2$, is there a non-horizontal line that passes through at least three points?

Given any 3-Sum instance $S = \{s_1, \ldots, s_n\}$, we can reduce it to a GeomBase instance P of size $3n$: For each s_i we create three points $(2s_i, 0)$, $(-s_i, 1)$ and $(2s_i, 2)$ in P. Now, if there is a non-horizontal line through three points in P, then each level contributes one, and so the three collinear points have form $p = (2s_i, 0)$, $q = (-s_j, 1)$ and $r = (2s_k, 2)$. The inverse slopes of lines pq and qr must be equal, hence $\frac{-s_j - 2s_i}{1-0} = \frac{2s_k + s_j}{2-1}$, or $s_i + s_j + s_k = 0$. Conversely, if there is a triple $s_i + s_j + s_k = 0$, then by a reverse argument we see that the points $(2s_i, 0), (-s_j, 1), (2s_k, 2) \in P$ are collinear.

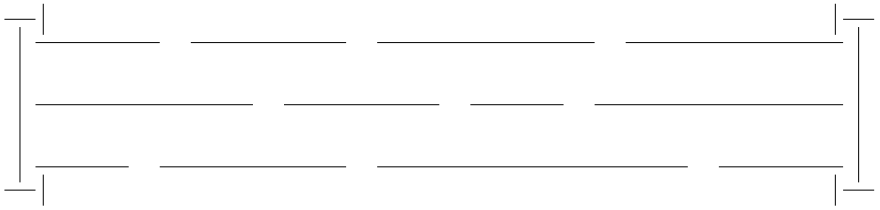
A very similar problem is **General Position** that we studied earlier. For a 3-Sum$^\circ$ instance S, we create a General Position instance P by lifting the numbers onto the curve $y = x^3$, that is $P := \{(a, a^3) \mid a \in S\}$. Three distinct points $(a, a^3), (b, b^3), (c, c^3) \in P$ are collinear if and only if the slopes of the lines through each pair are equal; that is

$$\begin{aligned}
& (b^3 - a^3)/(b - a) = (c^3 - b^3)/(c - b) \\
\iff \quad & b^2 + a^2 + ab = c^2 + b^2 + bc \\
\iff \quad & b = (c^2 - a^2)/(a - c) \\
\iff \quad & b = -(a + c) \\
\iff \quad & a + b + c = 0.
\end{aligned}$$

Hence P has a solution if and only if S has a solution. Note that the "if" part needs a, b, c to be distinct, and this is why we reduce from 3-Sum$^\circ$.

Minimum Area Triangle is a strict generalization of General Position and, therefore, also 3-Sum-hard.

In **Segment Separation**, we are given a set of n line segments and have to decide whether there exists a line that does not intersect the segments but separates them into two non-empty subsets. To show that this problem is 3-Sum-hard, we can reduce from GeomBase, where we emulate the points along the three levels $y = 0$, $y = 1$, and $y = 2$ by punching sufficiently small "holes". The resulting segments form the Segment Separation instance. Horizontal splits can be prevented by constant size gadgets next to the left and right ends:



Constructing such an instance requires sorting the points along each of the three levels, which can be done in $O(n \log n) = O(n^{2-\varepsilon})$ time. It remains to specify how "sufficiently small" are those holes. As all input numbers are integers, it is not hard to show that punching a hole of range $\pm 1/4$ around each point is small enough.

In **Segment Visibility**, we are given a set S of n horizontal line segments and two specific $s_1, s_2 \in S$. The question is: Are there two points, $p_1 \in s_1$ and $p_2 \in s_2$ that can see each other, that is, $\text{relint}(\overline{p_1 p_2})$ does not intersect any segment from S? The reduction from GeomBase is basically the same as for Segment Separation, just put s_1 above and s_2 below the three levels.

In **Motion Planning**, we are given a robot (modeled as a line segment), some obstacles (modeled as a set of disjoint line segments), and a source and a target position. The question is: Can the robot move (by translation and rotation) from the source to the target position, without ever intersecting the obstacles? To show that Motion Planning is 3-Sum-hard, employ the reduction from GeomBase as above. The holes on the three punched lines form the doorways from "the top room" to "the bottom room". Each room is surrounded by walls on the outer sides, so a robot in the top room can only reach the bottom room through the doorways. By specifying a long enough robot, the only way that it can pass is via three collinear holes.

Exercise 9.19. *The 3-Sum' problem asks: Given three sets S_1, S_2, S_3 of n integers each, are there $a_1 \in S_1$, $a_2 \in S_2$, $a_3 \in S_3$ such that $a_1 + a_2 + a_3 = 0$? Prove that 3-Sum' and 3-Sum are equivalent; more precisely, that they are reducible to each other in subquadratic time.*

9.8 Ham Sandwich Theorem

Suppose two thieves have stolen a necklace that contains rubies and diamonds. Now it is time to distribute the prey. Both, of course, should get the same number of rubies and the same number of diamonds. On the other hand, it would be a pity to completely disintegrate the beautiful necklace. Hence they want to use as few cuts as possible to achieve a fair gem distribution.

To phrase the problem in a geometric (and somewhat more general) setting: Given two finite sets R and D of points in $\mathbb{R}^2$, how do we find a line that bisects both sets? This means that each side of the line contains half of the points from R and half of the points from D. To solve this problem, we will make use of the concept of *levels* in arrangements.

Definition 9.20. *Consider an arrangement $\mathcal{A}(L)$ of a set L of n non-vertical lines in the plane. We say that a point p is on the k-***level** *in $\mathcal{A}(L)$ if there are at most $k-1$ lines below and at most $n-k$ lines above p. The 1-level and the n-level are also referred to as* **lower** *and* **upper envelope***, respectively.*

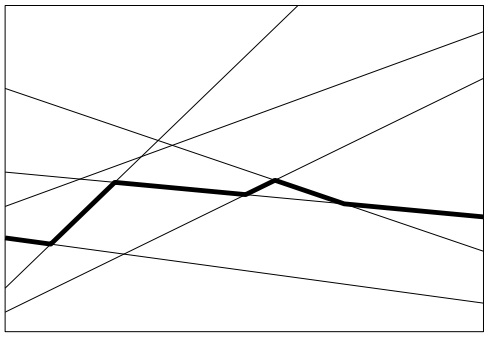


Figure 9.10: *The 3-level of an arrangement.*

Another way to look at the k-level is to consider the lines as real functions; then the lower/upper envelope is the pointwise minimum/maximum of those functions, and the k-level is defined by taking pointwise the k^{th}-smallest function value.

Theorem 9.21. *Let $R, D \subset \mathbb{R}^2$ be finite sets of points. Then there exists a line that bisects both R and D. That is, in either open halfplane bounded by ℓ, there are no more than $|R|/2$ points from R and no more than $|D|/2$ points from D.*

Proof. Without loss of generality suppose that both $|R|$ and $|D|$ are odd. (If, say, $|R|$ is even, simply remove an arbitrary point from R. Any bisector for the resulting set is also a bisector for R.) We may also suppose that no two points from $R \cup D$ have the same x-coordinate; otherwise we rotate the plane infinitesimally.

Let R^* and D^* denote the set of lines dual to the points from R and D, respectively. Consider the arrangement $\mathcal{A}(R^*)$. Every point on the median level of $\mathcal{A}(R^*)$ corresponds

to a primal line that bisects R. As $|R^*| = |R|$ is odd, both the leftmost and the rightmost segments of the median level are contributed by the same line $\ell_r \in R^*$: the one with median slope. Similarly there is a line $\ell_d \in D^*$ that contributes to the leftmost and rightmost segments of the median level in $\mathcal{A}(D^*)$.

Since no two points from $R \cup D$ have the same x-coordinate, no two lines from $R^* \cup D^*$ have the same slope, and thus ℓ_r and ℓ_d intersect. Consequently, being piecewise linear continuous functions, the median level of $\mathcal{A}(R^*)$ and the median level of $\mathcal{A}(D^*)$ also intersect (see Figure 9.11 for an example). Any point that lies on this intersection corresponds to a primal line that bisects both point sets simultaneously. □

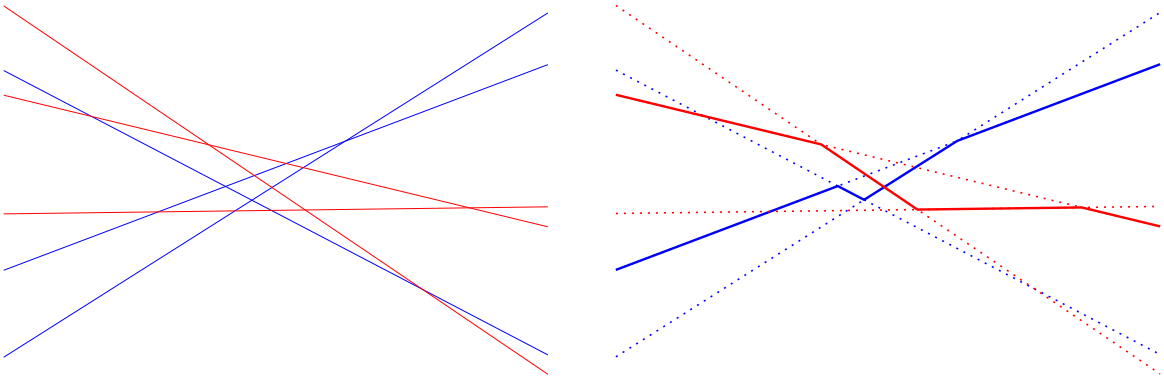


Figure 9.11: *An arrangement of 3 red lines and an arrangement of 3 blue lines, drawn in one picture. Their median levels are marked bold on the right.*

How can the thieves use Theorem 9.21? If they are smart, they drape the necklace along some convex curve, say a circle. Then by Theorem 9.21 there exists a line that simultaneously bisects the set of diamonds and the set of rubies. As any line intersects the circle at most twice, they need to cut the necklace at only two points.

You can also think of the two point sets as a discrete distribution of a ham sandwich that is to be cut fairly, that is, in such a way that both parts have the same amount of ham and the same amount of bread. That is where the name "ham sandwich cut" comes from. The theorem generalizes both to higher dimension and to more general types of measures (here we study the discrete setting only where we simply count points). These generalizations can be proven using the *Borsuk-Ulam Theorem*, which states that any continuous map from S^d to $\mathbb{R}^d$ must map some pair of antipodal points to the same point. For a proof of both theorems and many applications see Matoušek's book [19].

Theorem 9.22. *Let $P_1, \dots, P_d \subset \mathbb{R}^d$ be finite sets of points. Then there exists a hyperplane h that simultaneously bisects all of $P_1, \dots, P_d$. That is, in either open halfspace defined by h there are no more than $|P_i|/2$ points from P_i, for every $i \in \{1, \dots, d\}$.*

This implies that the thieves can fairly distribute a necklace consisting of d types of gems using at most d cuts.

Exercise 9.23. *Prove or disprove the following statement: Given three finite sets* A, B, C *of points in the plane, there is always a circle or a line that bisects* A, B *and* C *simultaneously (that is, no more than half of the points of each set are inside or outside the circle or on either side of the line, respectively).*

Knowing about the existence of a ham sandwich cut certainly is not good enough. It is not hard to turn the proof given above into an $O(n^2)$ algorithm, but we can do better...

9.9 Constructing Ham Sandwich Cuts in the Plane

The algorithm outlined below is interesting not only in itself, but also because it illustrates a fundamental paradigm for designing optimization algorithms: *prune & search*. The basic idea behind prune & search is to exclude part of the search space from further consideration ("prune") at each step, and then search in the remaining space. A well-known example is binary search: every step takes constant time and discards about half of the possible solutions, resulting in a logarithmic runtime overall. As another example, if at each step some $1/d$ fraction of all potential solutions is discarded, and the step costs linear time cn in the number n of (remaining) solutions, then the runtime T satisfies

$$T(n) \leqslant cn + T\left(\left(1 - \frac{1}{d}\right) n\right) < cn \sum_{i=0}^{\infty} \left(1 - \frac{1}{d}\right)^i = cdn,$$

which gives a linear time performance.

Theorem 9.24 (Edelsbrunner and Waupotitsch [5]). *Let* $R, D \subset \mathbb{R}^2$ *be finite sets of points with* $n = |R| + |D|$. *Then in* $O(n \log n)$ *time one can find a line* ℓ *that simultaneously bisects* R *and* D.

Proof. We design a recursive algorithm $\text{Find}(L_1, k_1;\ L_2, k_2;\ (x_1, x_2))$ that, for sets of lines L_1, L_2, non-negative integers k_1, k_2 and an open interval (x_1, x_2), finds an intersection between the k_1-level of $\mathcal{A}(L_1)$ and the k_2-level of $\mathcal{A}(L_2)$. It operates under two premises:

Distinct slopes: the lines in $L_1 \cup L_2$ have pairwise distinct slopes.

Odd-intersection: the two levels of interest intersect an odd number of times in (x_1, x_2) and do not intersect at x_1 or x_2. Equivalently, the level above the other at x_1 runs below the other at x_2.

In the end, we are interested in $\text{Find}\left(R^*, \frac{|R|+1}{2};\ D^*, \frac{|D|+1}{2};\ (-\infty, \infty)\right)$. Rotating the plane if necessary, we may assume without loss of generality that points in $R \cup D$ have distinct x-coordinates and thus lines in $R^* \cup D^*$ have distinct slopes. Also, as shown in the proof of Theorem 9.21, the odd-intersection property holds for these initial parameters.

Now we describe the algorithm. Let $L := L_1 \cup L_2$ and find the line $\mu \in L$ of median slope. Partition $L =: L_< \cup \{\mu\} \cup L_>$ depending on whether a line has slope less than or

greater than μ. Pair the lines of $L_<$ with those of $L_>$ arbitrarily (one remains unpaired if $|L|$ is even). Denote by I the $\lfloor(|L|-1)/2\rfloor$ intersection points generated by these pairs, and let j be the median x-coordinate of the points in I.

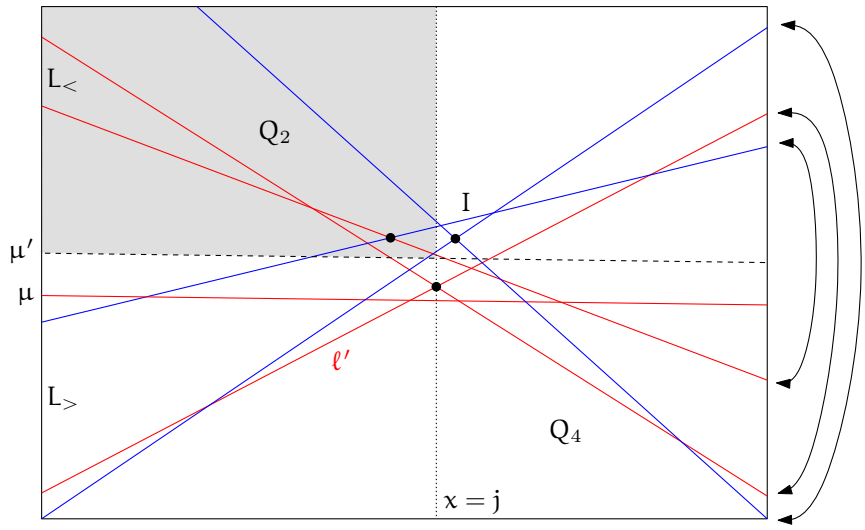


Figure 9.12: *An example with a set L_1 of four red lines and a set L_2 of three blue lines. Suppose that $k_1 = 3$ and $k_2 = 2$. Then the interesting quadrant is the top-left quadrant Q_2 (shaded), and the red line ℓ' (the line with the smallest slope in L_1) is discarded because it does not intersect Q_2.*

Find the point on the k_1-level of $\mathcal{A}(L_1)$ at j, and the point on the k_2-level of $\mathcal{A}(L_2)$ at j. If these points coincide then we simply return it. Otherwise, if $j \in (x_1, x_2)$ then exactly one of the open intervals (x_1, j) or (j, x_2) satisfies the odd-intersection property, and we restrict our attention to this interval. If $j \notin (x_1, x_2)$, then we keep the original interval (x_1, x_2). In either case, we may assume by symmetry that the interval of interest is to the left of j.

In the following it is our goal to discard a constant fraction of the lines in L from future consideration. To this end, let $I_>$ denote the set of points from I with x-coordinate greater than j. Let μ' be a line parallel to μ that bisects $I_>$. The two lines $x = j$ and μ' subdivide the plane into four quadrants. By definition, each quadrant contains about a quarter of the points from I, that is, about $|L|/8$ points.

By assumption the interval of interest is (x_1, t) for some $t \leqslant j$. In particular, exactly one of the left two quadrants is *interesting* in the sense that $((x_1, t) \times \mathbb{R}) \cap Q$ contains an odd number of intersections between the two levels. Denote this interesting quadrant by Q. We will later argue how to algorithmically determine Q. For now, suppose that the upper left quadrant Q_2 is interesting (Figure 9.12). Consider its opposite (that is, the lower right) quadrant Q_4. Any line in $L_>$ that passes through Q_4 is completely below the interesting quadrant Q_2, so all such lines can be safely discarded from further consideration. How many are they? Recall that every point in I is generated by paired lines, one from $L_>$ and the other from $L_<$. So there are at least $|I \cap Q_4| \geqslant \lfloor |L|/8 \rfloor$ such

lines.

After we discard these lines, we want to resume our search recursively. For every discarded line from L_1, we decrease the parameter k_1 by one. Analogously, for every discarded line from L_2, we decrease the parameter k_2 by one. In the other case where the quadrant Q_3 is interesting and we discard lines passing through Q_1, the parameters k_1 and k_2 stay the same. Denote the remaining sets of lines by L_1' and L_2', and the adjusted parameters by k_1' and k_2'.

We want to apply the algorithm recursively to compute an intersection between the k_1'-level of $\mathcal{A}(L_1')$ and the k_2'-level of $\mathcal{A}(L_2')$. However, discarding lines changes the arrangement and its levels. As a result, it is not clear that the odd-intersection property holds for the k_1'-level of $\mathcal{A}(L_1')$ and the k_2'-level of $\mathcal{A}(L_2')$ on the interval (x_1, t). Note that we do know that these levels intersect in the interesting quadrant, and this intersection persists because none of the involved lines is removed. However, it is conceivable that the removal of lines changes the parity of intersections in the non-interesting quadrant of the interval (x_1, t). Luckily, this issue can be resolved as a part of the algorithm to determine the interesting quadrant, which we will discuss next. More specifically, we will show how to determine an interval $(x_1', x_2') \subseteq (x_1, x_2)$ on which the odd-intersection property holds for the k_1'-level of $\mathcal{A}(L_1')$ and the k_2'-level of $\mathcal{A}(L_2')$.

So let us argue how to determine the interesting quadrant, that is, how to test whether the k_1-level of $\mathcal{A}(L_1)$ and the k_2-level of $\mathcal{A}(L_2)$ intersect an odd number of times in $((x_1, t) \times \mathbb{R}) \cap H^+$, where H^+ is the open halfplane above μ'. For this it is enough to sweep the line μ' from left to right in the arrangement $\mathcal{A}(L)$ (conceptually) while keeping track of the relative ordering of the two levels and μ'. Initially at $x = x_1$, we know which level is above the other. Then we inspect all intersections between μ and L from left to right. If the current intersection point is on one of the two levels, then the relative ordering might change. So we check whether it is still consistent with that initial ordering. For instance, if the initial ordering is "the k_1-level above the k_2-level above μ'" and the k_1-level intersects μ' before the k_2-level does, then we can deduce that there must be an intersection of the two levels above μ'. As the relative position of the two levels is reversed at $x = x_2$, at some point an inconsistency, that is, the presence of an intersection will be detected and we will be able to tell whether it is above or below μ'. (There could be many more intersections between the two levels, but finding just one intersection is good enough.) Along with this above/below information we also obtain a suitable interval (x_1', x_2') for which the odd-intersection property holds because the levels of interest do not change in that interval.

The sweep amounts to computing all intersections of μ' with L, sorting them by x-coordinate, and keeping track of the number of lines from L_1 (resp. L_2) below μ'. At every point of intersection, these counters can be adjusted and the inconsistency can be tested in constant time. Overall, this step takes $O(|L| \log |L|)$ time. It dominates the whole algorithm, noting that all other operations are based on rank-i element selection, which can be done in linear time [4]. Altogether, we obtain as a recursion for the runtime

$$T(n) \leqslant cn \log n + T(7n/8),$$

which solves to $T(n) \in O(n \log n)$. □

In the plane, a ham sandwich cut can actually be found in linear time using a sophisticated prune and search algorithm by Lo, Matoušek and Steiger [18]. But in higher dimension, the algorithmic problem gets harder. In fact, already for $\mathbb{R}^3$ the complexity of finding a ham sandwich cut is wide open: The best algorithm known, from the same paper by Lo et al. [18], has runtime $O(n^{3/2} \log^2 n / \log^* n)$ and no non-trivial lower bound is known. If the dimension d is not fixed, it is both NP-hard and W[1]-hard[3] in d to decide the following question [17]: Given $d \in \mathbb{N}$, finite point sets $P_1, \dots, P_d \subset \mathbb{R}^d$, and a point $p \in \bigcup_{i=1}^{d} P_i$, is there a ham sandwich cut through p?

Exercise 9.25. *The goal of this exercise is to develop a data structure for halfspace range counting.*

(a) Given a set $P \subset \mathbb{R}^2$ of n points in general position, show that it is possible to partition this set by two lines such that each region contains at most $\lceil \frac{n}{4} \rceil$ points.

(b) Design a data structure of size $O(n)$ that can be constructed in time $O(n \log n)$ and allows you, for any halfspace h, to output the number of points $|P \cap h|$ of P contained in this halfspace h in time $O(n^\alpha)$, for some $0 < \alpha < 1$.

9.10 Davenport-Schinzel Sequences

The complexity of a simple arrangement of n lines in $\mathbb{R}^2$ is The complexity of a simple arrangement of n lines in $\mathbb{R}^2$ is $\Theta(n^2)$, so every algorithm that uses the whole arrangement explicitly needs $\Omega(n^2)$ time. However, there are many scenarios where we only need a local part of the arrangement. For instance, if we only care about the cells touching a specific line, then the zone theorem ensures a linear complexity overall. As another example, to construct a ham sandwich cut for two sets in $\mathbb{R}^2$ we only need the median levels of their dual line arrangements. As mentioned in the previous section, the relevant information can actually be obtained in linear time.

Hence it can be rewarding to study the "local" complexity of arrangements. Here we pursue one such direction: to analyze the complexity—the number of vertices and edges—of a single cell in an arrangement of n *simple curves* in $\mathbb{R}^2$.

If the curves are lines then this is of little interest: Every line can appear at most once on the boundary of each cell; and it is possible that all lines appear on the boundary of one cell simultaneously.

But if the curves are line segments rather than lines, the situation changes in a surprising way. Certainly a single segment can appear several times along the boundary of a cell, see the example in Figure 9.13. Make a guess: What is the maximal complexity of a cell in an arrangement of n line segments in $\mathbb{R}^2$? You will find out the correct answer

[3] Essentially this means that it is unlikely to be solvable in time $O(f(d)p(n))$, for an arbitrary function f and a polynomial p.

soon, although we only prove part of it. But my guess would be that it is rather unlikely that your guess is correct, unless, of course, you knew the answer already. :-)

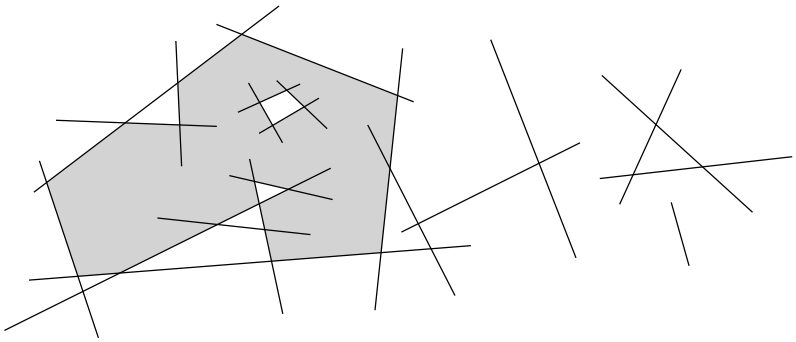


Figure 9.13: *A single cell in an arrangement of line segments.*

To start our program, we will focus on one particular cell in the curve arrangement: its lower envelope, or intuitively everything that can be seen vertically from below. To analyze the lower envelopes we will describe them combinatorially using sequences with forbidden substructures. These sequences, named after Davenport and Schinzel, are of independent interest as they appear in a number of combinatorial problems [2] and in the analyses of data structures [21]. The techniques apply not only to lower envelopes but also to arbitrary cells of curve arrangements.

Definition 9.26. *Let* $n, s \in \mathbb{N}$. *An* (n, s)-**Davenport-Schinzel sequence** *is a sequence over an alphabet of* n *symbols, such that*

- *no two consecutive elements are the same, and*
- *no subsequence of length* $s + 2$ *alternates between two different symbols.*[4]

Let $\lambda_s(n)$ *be the length of a longest* (n, s)*-Davenport-Schinzel sequence.*

So, for instance, an $(n, 2)$-Davenport-Schinzel sequence forbids alternations of the form $\ldots a \ldots b \ldots a \ldots b \ldots$. As another example, consider the sequence abcbacb. It is a $(3, 4)$-Davenport-Schinzel sequence but not a $(3, 3)$-Davenport-Schinzel sequence, because it contains a subsequence bcbcb of length 5 that alternates between b and c.

Note that the Davenport-Schinzel property is invariant of the alphabet in the following sense. If a sequence $\sigma_1, \ldots, \sigma_\ell$ over alphabet A is (n, s)-Davenport-Schinzel and $\varphi : A \to B$ is a bijection, then the sequence $\varphi(\sigma_1), \ldots, \varphi(\sigma_\ell)$ over alphabet B is again (n, s)-Davenport-Schinzel. In other words, we can change the alphabet whenever it is convenient.

Exercise 9.27. *Show that* $\lambda_1(n) = n$ *and* $\lambda_2(n) = 2n - 1$.

Exercise 9.28. *Prove that* $\lambda_s(n)$ *is finite for all* $n, s \in \mathbb{N}$. *Can you give explicit upper and lower bounds?*

[4] Note that a subsequence need not be contiguous. For example, geo is a subsequence of congestion.

Proposition 9.29. $\lambda_s(m) + \lambda_s(n) \leqslant \lambda_s(m+n)$.

Proof. Consider any (m, s)-Davenport-Schinzel sequence of length ℓ and (n, s)-Davenport-Schinzel sequence of length ℓ'. Assume without loss of generality that their alphabets are disjoint, so concatenating them yields a sequence of length $\ell + \ell'$ over an alphabet of $m+n$ symbols. It is straightforward to check that the sequence is $(m+n, s)$-Davenport-Schinzel. □

Let us now see how Davenport-Schinzel sequences are connected to lower envelopes. Consider a set $\mathcal{F} = \{f_1, \ldots, f_n\}$ of real-valued continuous functions that are defined on a common closed interval $I \subset \mathbb{R}$. The *lower envelope* $\mathcal{L}_\mathcal{F}$ of $\mathcal{F}$ is defined as the pointwise minimum of the functions. Formally, $\mathcal{L}_\mathcal{F}(x) := \min\{f_j(x) : 1 \leqslant j \leqslant n\}$ for $x \in I$.

Suppose that the graphs of any two functions in $\mathcal{F}$ intersect at finitely many points. Then each function can appear on the lower envelope $\mathcal{L}_\mathcal{F}$ only finitely many times. Imagine scanning the lower envelope from left to right, then the indices of functions that appear on it form the *lower envelope sequence* $\phi(\mathcal{F})$; see Figure 9.14 for an illustration.

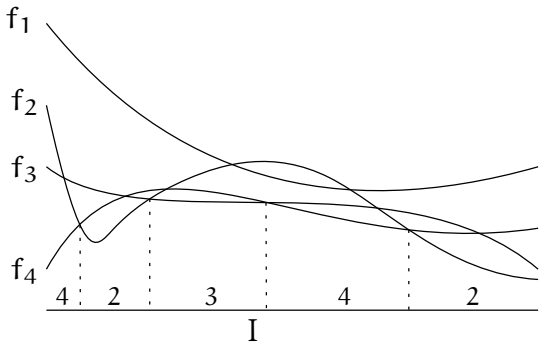


Figure 9.14: *The lower envelope sequence* $(4, 2, 3, 4, 2)$ *of a set of functions.*

Each intersection between the graphs of f_j and f_k can lead to at most one alternation between j and k in $\phi(\mathcal{F})$. Hence we have the following:

Observation 9.30. *If the graphs of any two functions in* $\mathcal{F}$ *intersect at most* s *times, then* $\phi(\mathcal{F})$ *is an* (n, s)*-Davenport-Schinzel sequence.*

But in order to deal with line segments (understood as linear functions over interval domains), the above machinery needs slight extension because the segments may have different domains. So let us allow each function in $\mathcal{F}$ having an individual closed interval as its domain. The lower envelope $\mathcal{L}_\mathcal{F}$, now defined over the real line, is given by $\mathcal{L}_\mathcal{F}(x) := \inf\{f_j(x) : 1 \leqslant j \leqslant n \text{ and } x \text{ is in the domain of } f_j\}$. In the case where no f_j is defined at x, we have $\mathcal{L}_\mathcal{F}(x) = \infty$.

Again assuming that the graphs of any two functions intersect at finitely many points, the lower envelope sequence $\phi(\mathcal{F})$ records the indices of the functions that appear on $\mathcal{L}_\mathcal{F}$ as we scan from left to right. By convention we use index 0 for intervals where $\mathcal{L}_\mathcal{F}(x) = \infty$; see Figure 9.15 for an illustration.

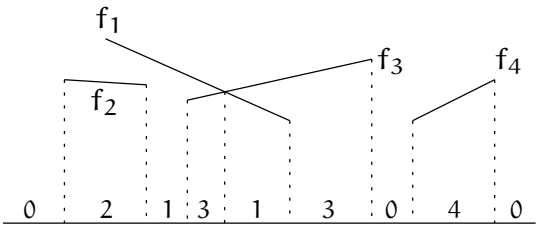


Figure 9.15: *The lower envelope sequence of a set of segments.*

In Figure 9.15 we already see that two segments f_j, f_k, despite crossing only once, may lead to a lower envelope subsequence jkjk of length 4 (instead of 2 as it would be if they were defined over a common interval). Generally we have:

Proposition 9.31. *Let $\mathcal{F}$ be a collection of n real-valued continuous functions, each defined on some closed interval, and the graphs of any two functions intersect at most s times. Then $\phi(\mathcal{F})$ is an $(n+1, s+2)$-Davenport-Schinzel sequence.*

Proof. We first show that there is no subsequence of length $s+4$ that alternates between two symbols $j, k \in \{1, \ldots, n\}$. For this, we consider the at most s points at which f_j, f_k are equal, plus the at most 4 domain endpoints of f_j and f_k. These together subdivide the real line into at most $s+5$ intervals. Within each of the at most $s+3$ inner intervals, f_j is either always above or always below f_k by continuity; only the lower one may appear on $\mathcal{L}_{\mathcal{F}}$ (possibly zero or multiple times). Hence in $\phi(\mathcal{F})$, the symbols j, k cannot alternate within this interval. It follows that any j-k alternating subsequence in $\phi(\mathcal{F})$ has length most $s+3$.

Finally, we observe that $\phi(\mathcal{F})$ cannot contain subsequence j0j for $j \in \{1, \ldots, n\}$, since the domain of f_j is an interval. Hence in particular there is no subsequence 0j0j or j0j0; that is, no subsequence of length 4 that alternates between symbols 0 and $j \in \{1, \ldots, n\}$. The statement then follows. □

So let us now focus on obtaining upper bounds for $\lambda_s(n)$. You have seen linear bounds for $s = 1, 2$ in Exercise 9.27. But the situation for $s = 3$ is more complicated.

Lemma 9.32. $\lambda_3(n) \leqslant 2n(1 + \log n)$.

Proof. For $n = 1$ we have $\lambda_3(1) = 1 \leqslant 2$. For $n > 1$ consider any $(n, 3)$-Davenport-Schinzel sequence σ of length $\lambda_3(n)$. Let a be a symbol that appears least frequently in σ; so it appears at most $\frac{1}{n}\lambda_3(n)$ times. Delete all appearances of a from σ to obtain a sequence σ' of length at least $(1-\frac{1}{n})\lambda_3(n)$ over $n-1$ symbols. But σ' is not necessarily a Davenport-Schinzel sequence because it might contain consecutive bb for some symbol b; this happens when $\sigma = \ldots bab \ldots$.

We claim that there are at most two places where such a situation may arise. In fact, the only two possible places are around the first and last appearances of a. If any intermediate appearance of a had resulted in a consecutive pair bb after deletion,

then $\sigma = \ldots a \ldots bab \ldots a \ldots$. So σ would contain the subsequence $ababa$, in contradiction to it being an $(n, 3)$-Davenport-Schinzel sequence.

Given the claim, one can repair σ' by deleting at most two characters from it. This produces an $(n-1, 3)$-Davenport-Schinzel sequence of length at least $(1 - \frac{1}{n})\lambda_3(n) - 2$, which by definition cannot exceed $\lambda_3(n-1)$. Rearranging terms yields the recursion

$$\frac{\lambda_3(n)}{n} \leqslant \frac{\lambda_3(n-1)}{n-1} + \frac{2}{n-1}.$$

Denoting $\overline{\lambda}_3(n) := \frac{\lambda_3(n)}{n}$, this implies

$$\overline{\lambda}_3(n) \leqslant \overline{\lambda}_3(n-1) + \frac{2}{n-1} \leqslant \cdots \leqslant \overline{\lambda}_3(1) + \sum_{i=2}^{n} \frac{2}{i-1} = 1 + 2H_{n-1},$$

where H_{n-1} is the $(n-1)$-st harmonic number. Together with $2H_{n-1} < 1 + 2\log n$ we obtain $\lambda_3(n) \leqslant 2n(1 + \log n)$. □

Bounds for higher-order Davenport-Schinzel sequences. As we have seen, $\lambda_1(n)$ (no aba) and $\lambda_2(n)$ (no $abab$) are both linear in n. It turns out that for $s \geqslant 3$, $\lambda_s(n)$ is slightly superlinear in n (with s fixed). The bounds are known almost exactly, and they involve the inverse Ackermann function $\alpha(n)$, which grows extremely slowly.

To define the inverse Ackermann function, we first define a hierarchy of functions $\alpha_1(n), \alpha_2(n), \alpha_3(n), \ldots$ where $\alpha_k(n)$ grows much more slowly than $\alpha_{k-1}(n)$ for every fixed k.

We let $\alpha_1(n) = \lceil n/2 \rceil$. Then, for each $k \geqslant 2$, we define $\alpha_k(n)$ as the number of repeating applications of α_{k-1} on n until the value becomes no larger than 1. In other words, $\alpha_k(n)$ is defined recursively by:

$$\alpha_k(n) = \begin{cases} 0, & \text{if } n \leqslant 1; \\ 1 + \alpha_k(\alpha_{k-1}(n)), & \text{otherwise.} \end{cases}$$

Thus $\alpha_2(n) = \lceil \log_2 n \rceil$, and $\alpha_3(n) = \log^* n$.

Now fix n and consider the sequence $\alpha_1(n), \alpha_2(n), \alpha_3(n), \ldots$. The sequence decreases rapidly until it drops below 3. We define the inverse Ackermann function $\alpha(n)$ by

$$\alpha(n) = \min\{k : \alpha_k(n) \leqslant 3\}.$$

Exercise 9.33.

(a) Show that for every fixed $k \geqslant 2$ we have $\alpha_k(n) = o(\alpha_{k-1}(n))$; in fact, for every fixed k and t we have $\alpha_k(n) = o(\alpha_{k-1}(\alpha_{k-1}(\cdots \alpha_{k-1}(n) \cdots)))$, with t applications of α_{k-1}.

(b) Show that for every fixed k we have $\alpha(n) = o(\alpha_k(n))$.

Coming back to the bounds for Davenport-Schinzel sequences, for $\lambda_3(n)$ it is known that $\lambda_3(n) = \Theta(n\alpha(n))$ [12]. In fact we even know $\lambda_3(n) = 2n\alpha(n) \pm O(n\sqrt{\alpha(n)})$ [16, 20]. For $\lambda_4(n)$ we have $\lambda_4(n) = \Theta(n \cdot 2^{\alpha(n)})$ [3].

For higher-order sequences the known upper and lower bounds are almost tight, and they are of the form $\lambda_s(n) = n \cdot 2^{\mathrm{poly}(\alpha(n))}$, where the degree of the polynomial in the exponent is roughly $s/2$ [3, 20].

Realizing Davenport-Schinzel sequences as lower envelopes. There exists a construction of a set of n segments in the plane whose lower-envelope sequence has length $\Omega(n\alpha(n))$. (In fact, the lower-envelope sequence has length $n\alpha(n) - O(n)$, with a leading coefficient of 1; it is an open problem to get a leading coefficient of 2, or prove that this is impossible.)

It is an open problem to construct a set of n parabolic arcs in the plane whose lower-envelope sequence has length $\Omega(n \cdot 2^{\alpha(n)})$.

Exercise 9.34. *Show that every (n, s)-Davenport-Schinzel sequence can be realized as the lower envelope of n continuous functions from $\mathbb{R}$ to $\mathbb{R}$, every pair of which intersect at most s times.*

Exercise 9.35. *Show that every $(n, 2)$-Davenport-Schinzel sequence can be realized as the lower envelope of n parabolas.*

Generalizations of Davenport-Schinzel sequences. Generalized Davenport-Schinzel sequences have also been studied, for instance, where arbitrary subsequences (not necessarily an alternating pattern) are forbidden. For a pattern π and $n \in \mathbb{N}$ define $\mathrm{Ex}(\pi, n)$ to be the maximum length of a sequence over $\{1, \ldots, n\}$ that does not contain a subsequence of the form π. For example, $\mathrm{Ex}(ababa, n) = \lambda_3(n)$. If π contains two meta-symbols only, say a and b, then $\mathrm{Ex}(\pi, n)$ is super-linear if and only if π contains $ababa$ as a subsequence [1]. This highlights that the alternating forbidden pattern is of particular interest.

9.11 Constructing Lower Envelopes

Theorem 9.36. *Let $\mathcal{F}$ be a collection of n real-valued continuous functions defined on a common closed interval. Assume that any two functions intersect at most s times, and that each intersection point can be constructed in constant time. Then the lower envelope $\mathcal{L}_{\mathcal{F}}$ can be constructed in $O(\lambda_s(n) \log n)$ time.*

Proof. Divide and conquer. For simplicity, assume that n is a power of two. Split $\mathcal{F}$ into two subsets $\mathcal{F}_1$ and $\mathcal{F}_2$ of $n/2$ functions each. Construct their lower envelopes $\mathcal{L}_{\mathcal{F}_1}$ and $\mathcal{L}_{\mathcal{F}_2}$ recursively. After they return, we merge them by a scan from left to right. At coordinate x, the defining function of $\mathcal{L}_{\mathcal{F}}$ can change only when (1) the defining function of $\mathcal{L}_{\mathcal{F}_1}$ changes; (2) the defining function of $\mathcal{L}_{\mathcal{F}_2}$ changes; or (3) $\mathcal{L}_{\mathcal{F}_1}$ and $\mathcal{L}_{\mathcal{F}_2}$ intersect. So the scan concerns with discrete events only.

There are at most $\lambda_s(n/2)$ events of type (1); and the same holds for type (2). (Note, however, that they do not necessarily guarantee a change for $\mathcal{L}_{\mathcal{F}}$.) Events of type (3) are different: every occurrence does contribute to a change for $\mathcal{L}_{\mathcal{F}}$, thus we can upper bound the number of occurrences by the complexity of $\mathcal{L}_{\mathcal{F}}$, namely $\lambda_s(n)$. So the total number of events is at most $2\lambda_s(n/2)+\lambda_s(n) \leqslant 2\lambda_s(n)$ where we used Proposition 9.29.

Now that the scan costs time linear in $\lambda_s(n)$, we obtain a recursion for the runtime $T(n)$ of the algorithm: $T(n) \leqslant 2T(n/2)+c\lambda_s(n)$ for some constant c. Therefore, $T(n) \leqslant c\sum_{i=1}^{\log n} 2^i\lambda_s(n/2^i) \leqslant c\sum_{i=1}^{\log n}\lambda_s(n) = O(\lambda_s(n)\log n)$ where the second inequality is by Proposition 9.29. □

9.12 Complexity of a Single Cell (not covered in HS 2025)

The section was not covered in this year's course $\Longrightarrow$ not relevant for the exam.

With all tools gathered, we can now go back to the initial question: What is the complexity of a single cell in an arrangement of n simple curves?

Theorem 9.37. *Let $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$ be a collection of simple curves in $\mathbb{R}^2$ such that each pair intersects at most s times. Then the combinatorial complexity of any single cell in the arrangement $\mathcal{A}(\Gamma)$ is $O(\lambda_{s+2}(n))$.*

Proof. Consider a cell f of $\mathcal{A}(\Gamma)$. In general, ∂f might consist of multiple connected components. But as every γ_i can appear in at most one component, we may deal with each component separately and sum up their complexity by Proposition 9.29. Hence it is no loss of generality to assume that ∂f is connected.

Replace each γ_i by two opposite directed curves γ_i^+ and γ_i^- that together form a closed curve that is infinitesimally close to γ_i. Denote by S the circular order that these directed curves appear along a clockwise traversal of ∂f. See Figure 9.16 for an illustration.

> **Consistency Lemma.** Fix $i \in \{1, \ldots, n\}$, and let ξ be one of the directed curves γ_i^+ or γ_i^-. The order of portions of ξ that appear in S is consistent with their order along ξ. Hence we can break the circular order S into a linear sequence $S(\xi)$ that, as we scan it from left to right, the portions of ξ appear in their order along ξ.

Consider two portions ξ_1 and ξ_2 of ξ that appear consecutively in S (that is, there is no other portion of ξ in between). Choose points $x_1 \in \xi_1$ and $x_2 \in \xi_2$ and connect them via two curves: a curve α following ∂f clockwise, and a curve β sandwiched between γ_i^+ and γ_i^-. Then α and β do not intersect internally and they are both contained in $\mathbb{R}^2 \setminus f^\circ$. In other words, $\alpha \cup \beta$ forms a closed Jordan curve and f lies either in its interior or in its exterior (Figure 9.17). In either case, the part of ξ between ξ_1 and ξ_2 is shielded away from f by $\alpha \cup \beta$ and, therefore, no point from this part can appear anywhere along ∂f. In other words, the boundary portions ξ_1 and ξ_2 are also consecutive along ξ, which proves the lemma.

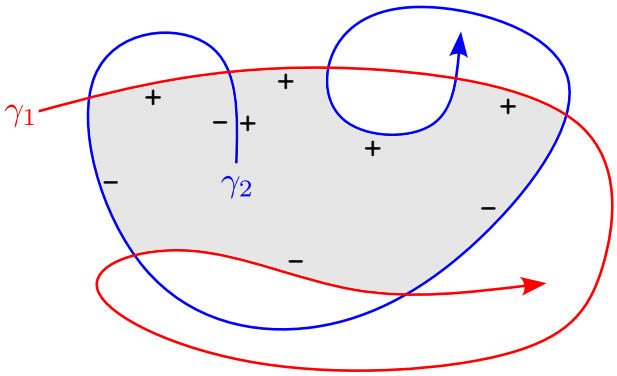
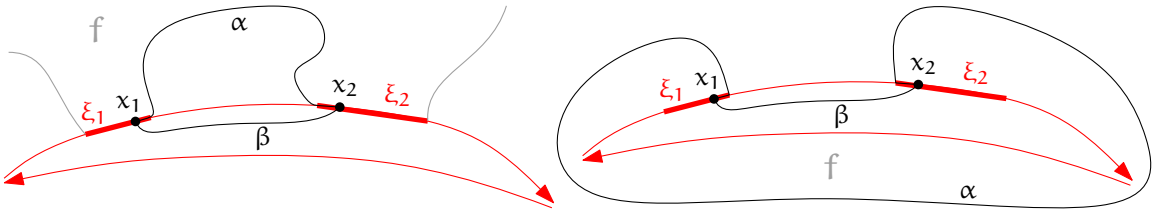


Figure 9.16: *An arrangement of two simple curves γ_1, γ_2. The arrows indicate the positive curves γ_1^+, γ_2^+, and their reversals are the negative curves γ_1^-, γ_2^-. Traversing the boundary of the shaded cell clockwise from the top-left corner, we encounter $S = (\gamma_1^+, \gamma_2^-, \gamma_2^+, \gamma_1^+, \gamma_2^+, \gamma_1^+, \gamma_2^-, \gamma_1^-, \gamma_2^-)$.*



(a) f lies in the exterior of $\alpha \cup \beta$. **(b)** f lies in the interior of $\alpha \cup \beta$.

Figure 9.17: *Two cases in the proof of Consistency Lemma.*

Let us break the circular order S into a linear sequence $S' = (s_1, \ldots, s_t)$ arbitrarily. Now we traverse a directed curve ξ, paying attention to its portions that appear in ∂f. By the Consistency Lemma, these portions correspond to $(s_{i_1}, \ldots, s_{i_\ell}, s_{i_{\ell+1}}, \ldots, s_{i_r})$, where $1 \leqslant i_{\ell+1} < \cdots < i_r < i_1 < \cdots < i_\ell \leqslant t$. This "wrap-around" issue is caused by our breaking the circular S into a linear S'. As a precaution, we replace all occurrences of ξ in S' after index i_ℓ by a brand new symbol ξ'. Doing so for all directed curves results in a sequence S^* over $4n$ symbols. With this trick, any subsequence $\xi\xi\ldots\xi$ (as well as $\xi'\xi'\ldots\xi'$) of S^* will agree with the traversal order of directed curve ξ.

Claim. *S^* is a $(4n, s+2)$-Davenport-Schinzel sequence.*

Clearly no two adjacent symbols in S^* are the same. Suppose S^* contains a subsequence σ of length $s+4$ that alternates between ξ and η. For any occurrence of ξ in σ, choose a point from the corresponding part of ∂f. This gives a sequence $x_1, \ldots, x_{\lceil (s+4)/2 \rceil}$ of points on ∂f. Connect them in this order by a curve $C(\xi)$ sandwiched between ξ and its counterpart. (This is possible thanks to our precaution.) Similarly we may choose points $y_1, \ldots, y_{\lfloor (s+4)/2 \rfloor}$ on ∂f that correspond to the occurrences of η in σ, and connect them in this order by a curve $C(\eta)$ sandwiched between η and its counterpart.

Now consider any quadruple $x_i, y_i, x_{i+1}, y_{i+1}$. (The case $y_i, x_i, y_{i+1}, x_{i+1}$ is symmetric.) They must be appearing in this order along ∂f. In addition, the pair $x_i\, x_{i+1}$ is connected by part of $C(\xi)$, and similarly the pair $y_i\, y_{i+1}$ is connected by part of $C(\eta)$; both curves, except for their endpoints, lie in the exterior of f. Finally, we can place a point u at the interior of f, and connect it to x_i, y_i, x_{i+1} and y_{i+1} via internally disjoint curves inside the interior of f (Figure 9.18). By construction, no two of these curves cross, except possibly for the curves $x_i\, x_{i+1}$ and $y_i\, y_{i+1}$ in the exterior of f. In fact, these two curves must intersect, because otherwise we are facing a plane embedding of K_5 which does not exist.

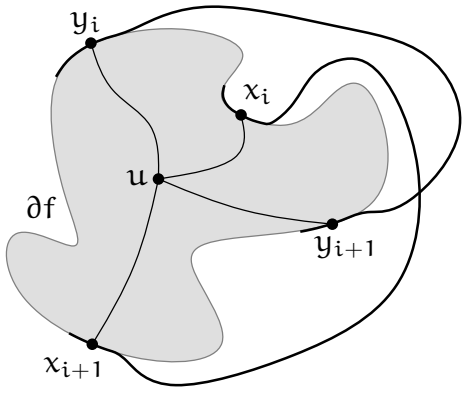


Figure 9.18: *Every quadruple x_i, y_i, x_{i+1}, y_{i+1} generates an intersection between the curves ξ and η.*

In other words, any quadruple of consecutive elements from the subsequence σ induces an intersection between $C(\xi)$ and $C(\eta)$. Clearly these intersections are distinct for all

quadruples, which altogether provide $s + 4 - 3 = s + 1$ intersections between $C(\xi)$ and $C(\eta)$, in contradiction to the fact that these curves can intersect at most s times. □

Corollary 9.38. *The combinatorial complexity of a single cell in an arrangement of* n *line segments in* $\mathbb{R}^2$ *is* $O(\lambda_3(n)) = O(n\alpha(n))$.

When counting the number of Davenport-Schinzel sequences of a certain type we want to count *essentially distinct* sequences only. Therefore we call two sequences over a common alphabet A *equivalent* if and only if one can be obtained from the other by a permutation of A. Then two sequences are *distinct* if and only if they are not equivalent. A typical way to select a representative from each equivalence class is to order the alphabet and demand that the first appearance of a symbol in the sequence follows that order. For example, ordering $A = \{a, b, c\}$ alphabetically demands that the first occurrence of a precedes the first occurrence of b, which in turn precedes the first occurrence of c.

Exercise 9.39. *Let* P *be a convex polygon with* $n+1$ *vertices. Find a bijection between the triangulations of* P *and the set of pairwise distinct* $(n, 2)$*-Davenport-Schinzel sequences of maximum length* $(2n - 1)$*. It follows that the number of distinct maximum* $(n, 2)$*-Davenport-Schinzel sequences is exactly* $C_{n-1} = \frac{1}{n}\binom{2n-2}{n-1}$*, which is the* $(n-1)$*-st Catalan number.*

Questions

40. *How can one construct an arrangement of lines in* $\mathbb{R}^2$*?* Describe the incremental algorithm and prove that its time complexity is quadratic in the number of lines (incl. statement and proof of the Zone Theorem).
41. *How can one test whether there are three collinear points in a set of* n *given points in* $\mathbb{R}^2$*?* Describe an $O(n^2)$ time algorithm.
42. *How can one compute the minimum area triangle spanned by three out of* n *given points in* $\mathbb{R}^2$*?* Describe an $O(n^2)$ time algorithm.
43. *What is a ham sandwich cut? Does it always exist? How to compute it?* State and prove the theorem about the existence of a ham sandwich cut in $\mathbb{R}^2$ and describe an $O(n^2)$ algorithm to compute it and sketch the $O(n \log n)$ algorithm by Edelsbrunner and Waupotitsch.
44. *What is the endpoint visibility graph for a set of disjoint line segments in the plane and how can it be constructed?* Give the definition and explain the relation to shortest paths. Describe the $O(n^2)$ algorithm by Welzl, including a proof of Theorem 9.15.
45. *Is there a subquadratic algorithm for General Position?* Explain the term 3-Sum hard and its implications and give the reduction from 3-Sum to General Position.

46. *Which problems are known to be 3-Sum-hard?* List at least three problems (other than 3-Sum) and briefly sketch the corresponding reductions.

47. *What is an* (n, s) *Davenport-Schinzel sequence and how does it relate to the lower envelope of real-valued continuous functions?* Give the precise definitions and some examples. Explain in particular how to apply the machinery to line segments.

48. *What is the value of* $\lambda_s(n)$*, for* $1 \leqslant s \leqslant 3$*?* Derive the precise value for $s \in \{1, 2\}$ and prove an $O(n \log n)$ upper bound for $\lambda_3(n)$.

49. *What is the combinatorial complexity of the lower envelope of a set of* n *lines/parabolas/line segments?*

50. **(This topic was not covered in HS25 and therefore the question will not be asked in the exam.)** *What is the combinatorial complexity of a single cell in an arrangement of* n *line segments?* State the result and sketch the proof (Theorem 9.37).

References

[1] Radek Adamec, Martin Klazar, and Pavel Valtr, Generalized Davenport-Schinzel sequences with linear upper bound. *Discrete Math.*, 108, (1992), 219–229.

[2] Pankaj K. Agarwal and Micha Sharir, *Davenport-Schinzel sequences and their geometric applications*, Cambridge University Press, New York, NY, 1995.

[3] Pankaj K. Agarwal, Micha Sharir, and Peter W. Shor, Sharp upper and lower bounds on the length of general Davenport-Schinzel sequences. *J. Combin. Theory Ser. A*, 52/2, (1989), 228–274.

[4] Manuel Blum, Robert W. Floyd, Vaughan Pratt, Ronald L. Rivest, and Robert E. Tarjan, Time bounds for selection. *J. Comput. Syst. Sci.*, 7/4, (1973), 448–461.

[5] Herbert Edelsbrunner and Roman Waupotitsch, Computing a ham-sandwich cut in two dimensions. *J. Symbolic Comput.*, 2, (1986), 171–178.

[6] David Eppstein, Happy endings for flip graphs. *J. Comput. Geom.*, 1/1, (2010), 3–28.

[7] Jeff Erickson, Lower bounds for linear satisfiability problems. *Chicago J. Theoret. Comput. Sci.*, 1999/8.

[8] Anka Gajentaan and Mark H. Overmars, On a class of $O(n^2)$ problems in computational geometry. *Comput. Geom. Theory Appl.*, 5, (1995), 165–185.

[9] Allan Grønlund and Seth Pettie, Threesomes, degenerates, and love triangles. *J. ACM*, 65/4, (2018), 22:1–22:25.

[10] Branko Grünbaum, Hamiltonian polygons and polyhedra. *Geombinatorics*, 3/3, (1994), 83–89.

[11] Heiko Harborth, Konvexe Fünfecke in ebenen Punktmengen. *Elem. Math.*, 33, (1979), 116–118.

[12] Sergiu Hart and Micha Sharir, Nonlinearity of Davenport-Schinzel sequences and of generalized path compression schemes. *Combinatorica*, 6, (1986), 151–177.

[13] Michael Hoffmann, *On the existence of paths and cycles*. Ph.D. thesis, ETH Zürich, 2005.

[14] Michael Hoffmann and Csaba D. Tóth, Segment endpoint visibility graphs are Hamiltonian. *Comput. Geom. Theory Appl.*, 26/1, (2003), 47–68.

[15] Daniel M. Kane, Shachar Lovett, and Shay Moran, Near-optimal linear decision trees for k-SUM and related problems. *J. ACM*, 66/3, (2019), 16:1–16:18.

[16] Martin Klazar, On the maximum lengths of Davenport-Schinzel sequences. In R. Graham et al., ed., *Contemporary Trends in Discrete Mathematics*, vol. 49 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pp. 169–178, Amer. Math. Soc., Providence, RI, 1999.

[17] Christian Knauer, Hans Raj Tiwary, and Daniel Werner, On the computational complexity of ham-sandwich cuts, Helly sets, and related problems. In *Proc. 28th Sympos. Theoret. Aspects Comput. Sci.*, vol. 9 of *LIPIcs*, pp. 649–660, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2011.

[18] Chi-Yuan Lo, Jiří Matoušek, and William L. Steiger, Algorithms for ham-sandwich cuts. *Discrete Comput. Geom.*, 11, (1994), 433–452.

[19] Jiří Matoušek, *Using the Borsuk–Ulam theorem*, Springer, Berlin, 2003.

[20] Gabriel Nivasch, Improved bounds and new techniques for Davenport-Schinzel sequences and their generalizations. *J. ACM*, 57/3, (2010), Article No. 17.

[21] Seth Pettie, Splay trees, Davenport-Schinzel sequences, and the deque conjecture. In *Proc. 19th ACM-SIAM Sympos. Discrete Algorithms*, pp. 1115–1124, 2008.

[22] Masatsugu Urabe and Mamoru Watanabe, On a counterexample to a conjecture of Mirzaian. *Comput. Geom. Theory Appl.*, 2/1, (1992), 51–53.

[23] Emo Welzl, Constructing the visibility graph for n line segments in $O(n^2)$ time. *Inform. Process. Lett.*, 20, (1985), 167–171.

Chapter 10

Convex Polytopes

Recall from Definition 5.8 that a **convex polytope** is the convex hull of a finite point set $P \subset \mathbb{R}^d$. In this chapter, we take a closer look at their structures and reveal their links to high-dimensional Delaunay triangulations and Voronoi diagrams. For convenience, we shall omit the attribute *convex* and refer to them simply as *polytopes*. In the sequel we will be borrowing a lot of material from Ziegler's classical book *Lectures on Polytopes* [12], sometimes without proofs as they would take us too far from geometry.

We are already familiar with polytopes in dimension $d = 2$, which are just convex polygons; see Figure 10.1. These are boring in the combinatorial sense: the vertex-edge graph is always a cycle.

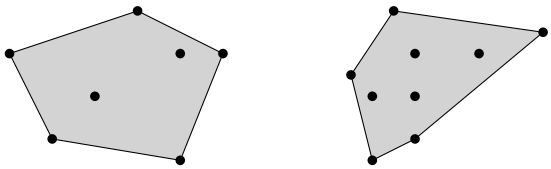
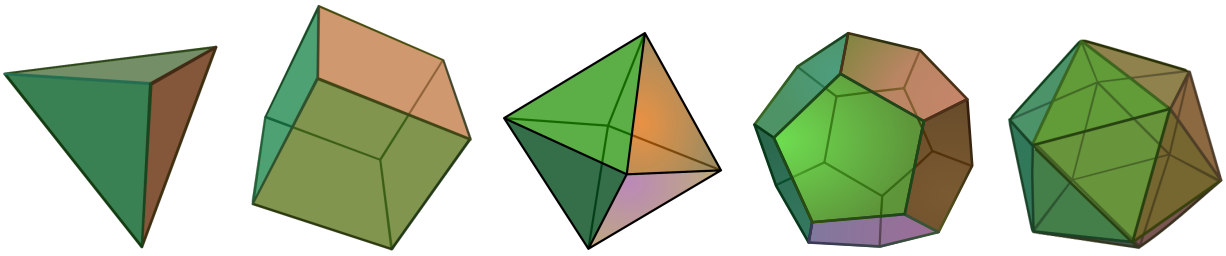


Figure 10.1: *In $\mathbb{R}^2$, convex polytopes are convex polygons.*

Polytopes in dimension $d = 3$ are more interesting, as they own a richer combinatorial structure. The most popular examples are the five platonic solids; see Figure 10.2.



(a) tetrahedron **(b)** cube **(c)** octahedron **(d)** dodecahedron **(e)** icosahedron

Figure 10.2: *The five platonic solids. (Images from Wikipedia [3, 2, 10, 1, 4])*

Despite this diversity, the vertex-edge graphs of 3-dimensional polytopes are well-understood, due to the following classical result. (See Lecture 4 in Ziegler's book [12] for a thorough treatment.)

Theorem 10.1 (Steinitz). *A graph* G *is the vertex-edge graph of a 3-dimensional polytope if and only if* G *is planar and 3-connected.*

We have already encountered 3-connected planar graphs in Chapter 2. Recall that Whitney's Theorem 2.27 states that every such graph has a unique combinatorial embedding in the plane. Here, Steinitz's theorem says that it also admits a geometric embedding as the skeleton of some polytope in $\mathbb{R}^3$. One can easily verify the theorem on the five platonic solids; for example, Figure 10.3 shows the vertex-edge graph of the octahedron, which is clearly planar and 3-connected.

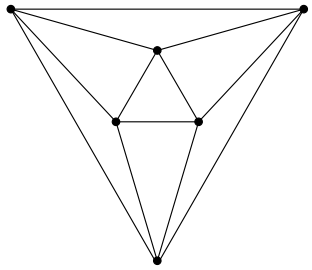


Figure 10.3: *The vertex-edge graph of the octahedron*

The theorem implies that a polytope in $\mathbb{R}^3$ with n vertices has at most $3n-6$ edges and $2n-4$ faces, by Corollary 2.5. What happens in higher dimensions? In particular, we want to understand how complicated a polytope in $\mathbb{R}^d$ can be. For example, how many edges can a 4-dimensional polytope with n vertices have? Is it still $O(n)$ as for $d = 2, 3$? To discuss this, we first have to define "vertices" and "edges" formally—our intuition unfortunately stops in $\mathbb{R}^3$. In fact, it is useful to define the more general notion of *faces* which subsumes vertices and edges.

10.1 Faces of a Polytope

In studying general dimension d, linear algebra tools are prominent. For a quick refresher we refer the reader to Chapter 5. Also recall that the dimension of a linear space is the maximum size of its linear independent subset. The dimension of an affine space is the maximum size of its affinely independent subset.

Let $\mathcal{P} = \text{conv}(P)$ be a polytope. Its **dimension** $\dim(\mathcal{P})$ is the dimension of its affine hull. The polytope is **full-dimensional** if $\dim(\mathcal{P}) = d$. Many results are stated for full-dimensional polytopes only, but this is not really a restriction: If $\dim(\mathcal{P}) < d$ then we can study it in the affine subspace $\text{aff}(P) \cong \mathbb{R}^{\dim(\mathcal{P})}$ where $\mathcal{P}$ becomes full-dimensional.

Definition 10.2. *Let $\mathcal{P} \subset \mathbb{R}^d$ be a polytope. We call $F \subseteq \mathcal{P}$ a* **face** *of $\mathcal{P}$ if there is a hyperplane*

$$h = \left\{ x \in \mathbb{R}^d : \sum_{i=1}^{d} h_i x_i = h_{d+1} \right\}$$

such that $F = \mathcal{P} \cap h$ and $\mathcal{P} \subset h^+$, where

$$h^+ = \left\{ x \in \mathbb{R}^d : \sum_{i=1}^{d} h_i x_i \geqslant h_{d+1} \right\}$$

is the closed positive halfspace bounded by h.[1] *We say that the h* **supports** *face F.*

You should think of a face as the intersection of $\mathcal{P}$ with a hyperplane "tangent" to $\mathcal{P}$. Figure 10.4 illustrates this notion. The **dimension** of a face is the dimension of its affine hull. A face of dimension k is called a k-face. Conventionally,

- 0-faces are called *vertices*,
- 1-faces are called *edges*,
- $(\dim(\mathcal{P}) - 2)$-faces are called *ridges*, and
- $(\dim(\mathcal{P}) - 1)$-faces are called *facets*.

For example, the octahedron in Figure 10.2(c) has 8 facets, 12 edges (which are also ridges), and 6 vertices. The dodecahedron in Figure 10.2(d) has 12 facets, 30 edges(=ridges), and 20 vertices.

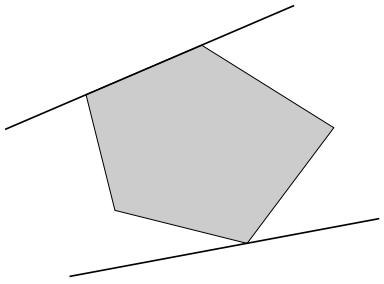


Figure 10.4: *Two faces (an edge and a vertex) with supporting hyperplanes.*

Degeneracy occurs if we set $h_1 = \cdots = h_d = 0$ in the definition.[2] If $h_{d+1} = 0$ then $h = h^+ = \mathbb{R}^d$, so this hyperplane supports $\mathcal{P}$. If $h_{d+1} < 0$ then $h = \emptyset, h^+ = \mathbb{R}^d$, so this hyperplane supports $\emptyset$. These two are called degenerate faces of $\mathcal{P}$; the others are called proper faces.

[1] Note that an inequality $\sum_{i=1}^{d} h_i x_i \leqslant h_{d+1}$ is equivalent to $\sum_{i=1}^{d} (-h_i) x_i \geqslant -h_{d+1}$, so sticking to positive halfspaces in the definition is no loss of generality.

[2] In Section 1.2 we did not allow such, but here we need it.

Exercise 10.3. *Show that every facet of a full-dimensional polytope has a unique supporting hyperplane: its affine hull.*

The definition of a face, in its current form, is cumbersome to work with. In particular, to verify a supporting hyperplane h, we have to reason about its interaction with the continuous mass $\mathcal{P}$. Thankfully, the lemma below reduces the verification to a finite set.

Lemma 10.4. *Let $\mathcal{P} = \mathrm{conv}(P) \subset \mathbb{R}^d$ be a polytope. If a hyperplane h satisfies $P \subset h^+$, then $\mathcal{P} \subset h^+$ and $\mathcal{P} \cap h = \mathrm{conv}(P \cap h)$.*

To get the intuition for the lemma as well as its proof, let us rephrase it: Imagine a hyperplane h and several points, some sitting on h while others living strictly on one side of h. Then the convex hull of the points should also dwell in that side and, moreover, intersect h in the zone that "fills between" the points on h.

Proof. As $P \subset h^+$ and h^+ is convex, the first claim follows immediately. The intersection $\mathcal{P} \cap h$ is convex as both $\mathcal{P}$ and h are convex, therefore $\mathrm{conv}(P \cap h) \subseteq \mathcal{P} \cap h$.

It remains to show $\mathcal{P} \cap h \subseteq \mathrm{conv}(P \cap h)$. Assume $h = \{x \in \mathbb{R}^d : \sum_{i=1}^d h_i x_i = h_{d+1}\}$. Let $p \in (\mathcal{P} \setminus P) \cap h$. Since $p \in \mathcal{P}$ we can express $p = \sum_{q \in Q} \lambda_q q$ as a convex combination of some other points $Q \subseteq P$, where $\lambda_q > 0$ for all $q \in Q$. Since $p \in h$ we know

$$h_{d+1} = \sum_{i=1}^d h_i p_i = \sum_{i=1}^d h_i \sum_{q \in Q} \lambda_q q_i = \sum_{q \in Q} \lambda_q \sum_{i=1}^d h_i q_i \geqslant \sum_{q \in Q} \lambda_q h_{d+1} = h_{d+1}.$$

where the "$\geqslant$" uses the fact that $P \subset h^+$. As the two ends are equal, the inequality is in fact an equality. But recall $\lambda_q > 0$ for all $q \in Q$, so we must have $\sum_{i=1}^d h_i q_i = h_{d+1}$ for all $q \in Q$. In other words, $Q \subseteq P \cap h$. Therefore, $p \in \mathrm{conv}(Q) \subseteq \mathrm{conv}(P \cap h)$. □

As an immediate consequence, every face of $\mathrm{conv}(P)$ is a convex hull of some points in P. In particular, there can be at most $2^{|P|}$ faces—a finite number as one would expect, but not at all obvious from the definition!

Lemma 10.4 is central to many proofs in this chapter; let us see an application right away. You might have speculated that the extreme points of a set P (cf. Definition 5.8) coincide with the vertices of polytope $\mathrm{conv}(P)$. Indeed this is true, up to the formal subtlety that a vertex is a singleton set rather than a point (we will later ignore this nuance, but it is good to have talked about it once).

Lemma 10.5. *Let $\mathcal{P} = \mathrm{conv}(P) \subset \mathbb{R}^d$ be a polytope. Then p is an extreme point of P if and only if $\{p\}$ is vertex of $\mathcal{P}$.*

Proof. If $p = (p_1, \ldots, p_d)$ is an extreme point of P, then the compact convex sets $\{p\}$ and $\mathrm{conv}(P \setminus \{p\})$ are disjoint. By the Separation Theorem 5.19, there is a hyperplane h that

strictly separates them. In formulas, there exist non-degenerate hyperplane parameters $h_1, \dots, h_{d+1} \in \mathbb{R}$ such that

$$\sum_{i=1}^{d} h_i p_i < h_{d+1} \qquad \text{and} \qquad \sum_{i=1}^{d} h_i q_i > h_{d+1} \quad \forall q \in \operatorname{conv}(P \setminus \{p\}).$$

Let us decrease h_{d+1} until the first inequality becomes tight. At that moment we have in particular

$$\sum_{i=1}^{d} h_i p_i = h_{d+1} \qquad \text{and} \qquad \sum_{i=1}^{d} h_i q_i > h_{d+1} \quad \forall q \in P \setminus \{p\},$$

meaning that $P \subset h^+$ and $P \cap h = \{p\}$. Applying Lemma 10.4 we obtain $\mathcal{P} \subseteq h^+$ and $\mathcal{P} \cap h = \{p\}$, so the hyperplane h supports $\{p\}$.

For the other direction, let p be a vertex supported by some hyperplane h. Consider the set $P' := P \setminus \{p\}$. Clearly $P' \subset h^+$ and $P' \cap h = \emptyset$. Applying Lemma 10.4 on P', we derive $\operatorname{conv}(P') \cap h = \emptyset$. But $p \in h$, so $p \notin \operatorname{conv}(P')$, namely p is an extreme point. □

By $V(\mathcal{P})$ we denote the set of vertices of a polytope $\mathcal{P}$. Then Proposition 5.10 with Lemma 10.5 imply the following:

Corollary 10.6. *$\mathcal{P} = \operatorname{conv}(V(\mathcal{P}))$; moreover, $V(\mathcal{P}) = \bigcap_{P:\operatorname{conv}(P)=\mathcal{P}} P$.*

In words, $V(\mathcal{P})$ is the minimal description of a polytope $\mathcal{P}$ as a convex hull of points. With little extra work we can relate vertices with arbitrary faces.

Lemma 10.7. *Every face F of a polytope $\mathcal{P}$ is a polytope itself with $V(F) = V(\mathcal{P}) \cap F$.*

Proof. Let F be a face supported by some hyperplane h. By Lemma 10.4, $F = \mathcal{P} \cap h = \operatorname{conv}(V(P) \cap h) = \operatorname{conv}(V(P) \cap F)$. So F is a polytope. Moreover, $V(F) \subseteq V(P) \cap F$ due to Corollary 10.6. The converse inclusion is clear, as any hyperplane supporting a vertex $v \in V(P) \cap F$ in the polytope $\mathcal{P}$ is also supporting v in the face F. □

Let us consider some concrete examples. Each facet of the octahedron is a triangle (which is a polytope); its three vertices are exactly the vertices of the octahedron that lie on the triangle. Similarly, each edge is a line segment (which is a polytope as well); its two vertices are those of the octahedron that lie on the segment.

In view of Corollary 10.6 and Lemma 10.7, every face F can be encoded by $V(\mathcal{P}) \cap F$ and later restored by taking convex hull. Namely, $F \mapsto V(\mathcal{P}) \cap F$ is an injection from faces of $\mathcal{P}$ to subsets of $V(\mathcal{P})$. In particular, if $|V(\mathcal{P})| = n$ then $\mathcal{P}$ has at most 2^n faces.

Exercise 10.8. *Let $\mathcal{P}$ be a polytope with n vertices. Show that $\mathcal{P}$ has at most $\binom{n}{k+1}$ many k-faces, for every $0 \leqslant k < \dim(\mathcal{P})$.*

Specializing for $k = 2$, a polytope with n vertices can have at most $\binom{n}{2}$ edges which doesn't surprise us: the vertex-edge graph cannot be more than complete. For $d = 2, 3$ this is a gross overestimate as we know there can be only $O(n)$ many edges; nevertheless, we can use Exercise 10.8 to upper bound the total number of proper faces by

$$\sum_{k=0}^{\dim(\mathcal{P})-1} \binom{n}{k+1} = O\left(n^{\dim(\mathcal{P})}\right),$$

which substantially improves the previous bound 2^n (for $n \to \infty$ and constant d).

Lemma 10.9. *Let F, G be two faces of a polytope $\mathcal{P}$. Then $F \cap G$ is also a face of $\mathcal{P}$. It has vertex set $V(F \cap G) = V(F) \cap V(G)$.*

Proof Sketch. Assume that F and G are supported by hyperplanes $\sum_{i=1}^{d} a_i x_i = a_{d+1}$ and $\sum_{i=1}^{d} b_i x_i = b_{d+1}$, respectively. Consider their "mixture"

$$h := \left\{ x \in \mathbb{R}^d : \sum_{i=1}^{d} (a_i + b_i) x_i = a_{d+1} + b_{d+1} \right\}.$$

It is not hard to check that

- every point $p \in F \cap G$ is lying on h;
- every point $p \in V(\mathcal{P}) \setminus (V(F) \cap V(G))$ is strictly contained in h^+.

It follows from Lemma 10.4 that $\mathcal{P} \subset h^+$ and $\mathcal{P} \cap h = \mathrm{conv}(F \cap G) = F \cap G$. So $F \cap G$ is a face supported by h. By Lemma 10.7, its vertex set is exactly $\mathcal{P} \cap (F \cap G) = V(F) \cap V(G)$. □

Exercise 10.10. *Show that every ridge is incident to exactly two facets.*

For polytopes in $\mathbb{R}^3$, Euler's formula gives us a relation between the number of vertices, edges and facets. In higher dimension this is generalized by the Euler-Poincaré formula. Let us denote by f_k the number of k-faces of a polytope $\mathcal{P}$.

Theorem 10.11 (Euler-Poincaré formula). *For every d-dimensional polytope we have*

$$\sum_{k=0}^{d-1} (-1)^k f_k = 1 - (-1)^d.$$

When specializing to $d = 3$ we recover the familiar $f_0 - f_1 + f_2 = 2$. We will see an elegant proof of the formula later in Chapter 11; but now let us explore one of its consequences:

Exercise 10.12. *Let $P \subset \mathbb{R}^4$ be a finite set of points in general position and let $\mathcal{P}$ be the polytope defined by the convex hull of P. Show that $f_3 \geqslant f_0$.*

10.2 The Main Theorem

We already know from Theorem 5.22 that a polytope can be written as the intersection of infinitely many halfspaces. But it seems that most of them are redundant; at least in dimension $d \leqslant 3$ finitely many halfspaces suffice. Is it true for higher dimensions? This motivates the following definition.

Definition 10.13. *A* **polyhedron** *is the intersection of* finitely *many halfspaces in* $\mathbb{R}^d$.

Unlike polytopes, polyhedra may be unbounded. For example, the whole space $\mathbb{R}^d$ is a polyhedron (the intersection of no halfspaces), and every halfspace is also a polyhedron (the intersection of one halfspace). Figure 10.5 gives another example in $\mathbb{R}^2$.

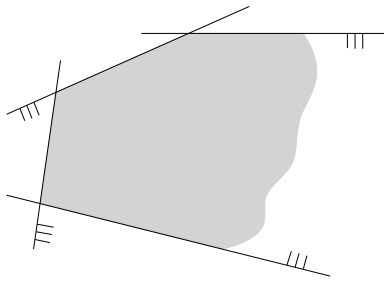


Figure 10.5: *An (unbounded) polyhedron in* $\mathbb{R}^2$ *(intersection of four halfspaces)*

The faces of a polyhedron $\mathcal{P}$ can be defined in the same way as for polytopes: F is a face if there exists a hyperplane h such that $F = \mathcal{P} \cap h$ and $\mathcal{P} \subset h^+$. For example, the polyhedron in Figure 10.5 has 3 vertices and 4 edges (= facets), two of which unbounded.

By extrapolating from the case $d = 2$ (which is always a bit dangerous, but let's try anyway), it seems that the only thing that can stop a polyhedron from being a polytope is its unboundedness. Conversely, it seems that a polytope is always a (bounded) polyhedron. These are indeed true in any dimension! So polytopes and bounded polyhedra is the same object. This is arguably the most fundamental result in polytope theory, and for this reason, Ziegler calls it the *Main Theorem* [12, Theorem 1.1].

Theorem 10.14 (Main Theorem). *A subset* $\mathcal{P} \subset \mathbb{R}^d$ *is the convex hull of a finite set of points if and only if* $\mathcal{P}$ *is a bounded intersection of finitely many halfspaces.*

People usually use the attributes $\mathcal{V}$-polytope and $\mathcal{H}$-polytope to mean a polytope *represented* as a convex hull of points or an intersection of hyperspaces, respectively.

Exercise 10.15. *Let* $\mathcal{P} = \bigcap_{i=1}^{m} h_i^+$ *be a full-dimensional polytope, represented as the intersection of* m *halfspaces* $h_1^+, \ldots, h_m^+$. *Prove that each facet of* $\mathcal{P}$ *is supported by one of the* m *hyperplanes* h_i. *(As a hyperplane can by definition support only one facet,* $\mathcal{P}$ *has at most* m *facets.)*

It can also be shown [12, Theorem 2.15 (7)] that hyperplanes not supporting a facet are redundant, meaning that we can always write a full-dimensional polytope with m facets in the form $\mathcal{P} = \bigcap_{i=1}^{m} h_i^+$, where each h_i supports one of the facets. Hence, in the same way that non-extreme points are redundant in representing a $\mathcal{V}$-polytope, hyperplanes not supporting facets are redundant in representing an $\mathcal{H}$-polytope.

Corollary 10.16. *Let $\mathcal{P}$ be a full-dimensional polytope. Then every point $p \in \partial\mathcal{P}$ is contained in some facet.*

Proof. Represent $\mathcal{P} = \bigcap_{i=1}^{m} h_i^+$ as an intersection of facet-supporting hyperplanes. If $p \in \mathcal{P}$ is not contained in any facet, then it is not contained in any of the hyperplanes. So a sufficiently small ball around p would still be in $\mathcal{P}$, meaning that $p \notin \partial\mathcal{P}$. □

10.3 Two Examples

Let's look at two families of higher-dimensional polytopes, called *hypercubes* and *simplices*, that naturally generalize the cube and the tetrahedron, respectively. The standard d-dimensional **hypercube** is the set

$$\left\{x \in \mathbb{R}^d : -1 \leqslant x_i \leqslant 1 \quad \forall i \in \{1, \ldots, d\}\right\}.$$

Formally this is a polyhedron, described as the intersection of $2d$ halfspaces. But as the boundedness is clear, the Main Theorem guarantees that it is a polytope. It has at most $2d$ facets by Exercise 10.15; but one can easily show that the number is precisely $2d$ (try to make the argument!). The next exercise is about its vertices.

Exercise 10.17. *Prove that the standard d-dimensional hypercube has 2^d vertices. What are they?*

A d-dimensional simplex, or simply d-**simplex**, is the convex hull of $d+1$ affinely independent points.

Exercise 10.18. *Prove that any d-simplex has 2^{d+1} faces. Specifically, for every subset Q of its defining points, show that there is a face F with $V(F) = Q$. (This count is maximum possible for polytopes with $d+1$ vertices, by Lemma 10.7.)*

10.4 Polytope Structure

In this section, we will summarize some more advanced properties of polytopes. All of these classical material can be found in full detail in Ziegler's book [12].

10.4.1 The Graph of a Polytope

For any d-dimensional polytope $\mathcal{P}$, its vertices and edges form a graph $G(\mathcal{P})$, sometimes also called the 1-skeleton of $\mathcal{P}$. As we discussed in the beginning of this chapter, these graphs are just cycles for dimension $d = 2$, and triconnected planar graphs for dimension $d = 3$ (Steinitz's Theorem 10.1). It turns out that for higher dimensions d, the graphs are also d-connected, as we will soon show.

Why do we care about these graphs? From a computational viewpoint they are very relevant to *linear programming*, a cornerstone in optimization theory. We will briefly explain the connection here without going into details. In a **linear program** we want to maximize a linear function $c^\top x$ where the variable x is subject to a system of linear inequalities $Ax \leqslant b$. Each row in $Ax \leqslant b$ specifies a halfspace, so all the rows together define a polyhedron $\mathcal{P}$. Let us assume for simplicity that it is non-empty and bounded, hence a polytope by the Main Theorem. Let $\zeta_{\max} := \max_{x\in\mathcal{P}} c^\top x$ be the optimal value of the linear program. Then $c^\top x = \zeta_{\max}$ is a hyperplane whose intersection with $\mathcal{P}$ is the set of optimal solutions. In particular, the set of optimal solutions is a face of $\mathcal{P}$. Let us orient every edge $\{v, w\}$ of $G(\mathcal{P})$ so that $v \to w$ if and only if $c^\top v < c^\top w$. Clearly the oriented graph is acyclic. Further, Proposition 10.19 below implies that every sink is an optimal solution. Thus one way to find an optimal solution is to start from any vertex and follow the directed edges, until we reach a sink. This is the main idea of an entire family of algorithms for linear programming, called the *simplex method*.

Proposition 10.19 (see [12]). *Let $\mathcal{P}$ be a polytope. Orient the graph $G(\mathcal{P})$ as above, according to the linear function $c^\top x$. Show that if vertex $v \in V(\mathcal{P})$ is suboptimal, that is if $c^\top v < \max_{x\in\mathcal{P}} c^\top x$, then there is an edge going out of v.*

In order for the simplex method to work efficiently, the graph $G(\mathcal{P})$ needs to have small diameter. Warren M. Hirsch made the following conjecture in 1957, known as the *Hirsch conjecture*: For any d-dimensional polytope $\mathcal{P}$ with m facets, the diameter of graph $G(\mathcal{P})$ is at most $m - d$.

This conjecture was disproven in 2010 by Francisco Santos [8], who constructed a 43-dimensional polytope with 86 facets whose graph has diameter larger than 43. However, the weaker *polynomial Hirsch conjecture*, which states that the graph of a polytope with m facets has diameter polynomial in m, is still open.

We conclude this section with Balinski's theorem about the connectivity of $G(\mathcal{P})$.

Theorem 10.20 (Balinski). *For any d-dimensional polytope $\mathcal{P}$, its graph $G(\mathcal{P})$ is d-connected.*

Proof. Let $\mathcal{P} = \operatorname{conv}(V) \subseteq \mathbb{R}^d$ where V is the vertex set of $\mathcal{P}$, with $|V| \geqslant d+1$. We want to show that deleting any subset $S \in \binom{V}{d-1}$ does not disconnect $G(\mathcal{P})$.

Let us fix a vertex $v_0 \in V \setminus S$ and a hyperplane $h : c^\top x = \zeta$ that goes through $S \cup \{v_0\}$. Such a plane must exist because any d points in $\mathbb{R}^d$ is contained in some hyperplane. Let $\zeta_{\min}$ and $\zeta_{\max}$ be the minimum and maximum values that the linear function $c^\top x$ can attain on $\mathcal{P}$, respectively; note that $\zeta_{\min} \leqslant \zeta \leqslant \zeta_{\max}$.

Let $F_{\min}$ and $F_{\max}$ be the faces supported by the hyperplanes $c^\top x = \zeta_{\min}$ and $c^\top x = \zeta_{\max}$, respectively. Now consider an arbitrary vertex $v \in V \setminus S$.

- If $c^\top v \geqslant \zeta$, then by Proposition 10.19, either $v \in V(F_{\max})$ or there is a path from v to $V(F_{\max})$ such that the function value $c^\top x$ strictly increases in each hop. In particular, the path avoids the set S because $c^\top x = \zeta$ for all $x \in S$.
- If $c^\top v \leqslant \zeta$, then by a symmetric argument, either $v \in F_{\min}$ or there is a path from v to $V(F_{\min})$ that avoids S.

Moreover, since $c^\top v_0 = \zeta$ we know that v_0 connects to *both* $F_{\min}$ and $F_{\max}$ without going through set S.

Finally, observe that $F_{\min}$ and $F_{\max}$ are lower dimensional polytopes, so by induction both $G(F_{\min})$ and $G(F_{\max})$ are connected. Therefore we may conclude that all vertices in $V \setminus S$ are connected without going through S. □

10.4.2 The Face Lattice

The graph of a polytope concerns only the 0-faces (vertices) and 1-faces (edges). More generally, we can collect all the faces of a polytope $\mathcal{P}$ and order them by inclusion. That is, $F \leqslant G$ if $F \subseteq G$; and $F < G$ if $F \subset G$. This partially ordered set (or poset) is called the **face lattice** of $\mathcal{P}$. Posets are usually drawn as *Hasse diagrams* where larger elements appear higher up. Two elements $F < G$ are linked by a line if there is no $H : F < H < G$. For example, the face lattice of the 3-dimensional cube is depicted in Figure 10.6.

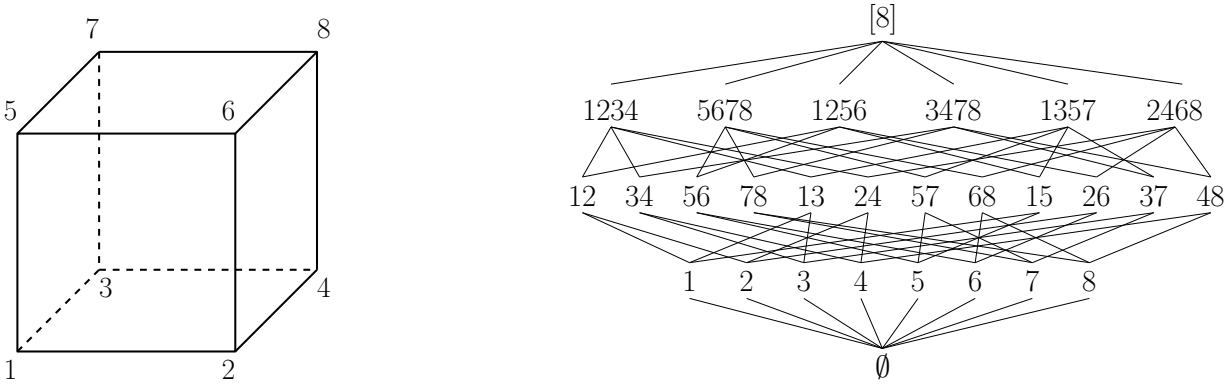


Figure 10.6: *The cube (left) and its face lattice (right). Faces are named with the labels of their vertices.*

What makes this poset a *lattice* is the following property. For any two faces F and G, there is

- a unique maximal face E such that $E \leqslant F, G$ (called their *meet*); and
- a unique minimal face H such that $H \geqslant F, G$ (called their *join*).

Clearly the meet of F and G is exactly $F \cap G$ (which is indeed a face by Lemma 10.9, with vertex set $V(F) \cap V(G)$). It may be tempting to think that the join of F and G is $\operatorname{conv}(V(F) \cup V(G))$, but that is not a face in general. The join turns out to be something more intricate. We can already observe this in the face lattice of a cube (Figure 10.6). The join of the edges 12 and 13, for example, is the face with four vertices 1234. The following exercise asks you to prove the existence of a join, implicitly.

Exercise 10.21. *In general, a* poset *is a pair* $(\mathcal{F}, \leqslant)$. *Here* $\leqslant$ *is a partial order over* $\mathcal{F}$, *meaning that it is reflexive* ($F \leqslant F$ *always holds*), *antisymmetric* ($F \leqslant G$ *and* $G \leqslant F$ *implies* $F = G$) *and transitive* ($F \leqslant G$ *and* $G \leqslant H$ *implies* $F \leqslant H$).

An element $F \in \mathcal{F}$ *is* maximal *if there is no element* $G > F$. *Similarly, it is* minimal *if there is no element* $G < F$.

An element E *is a* maximal lower bound *of* F *and* G *if* $E \leqslant F, G$ *but no element* $E' > E$ *has this property. If there is only one such* E, *then we call it the* meet *of* F *and* G. *Similarly, an element* H *is a* minimal upper bound *of* F *and* G *if* $F, G \leqslant H$ *but no element* $H' < H$ *has this property. If there is only one such* H *then we call it the* join *of* F *and* G.

Now for the actual exercise: Let $(\mathcal{F}, \leqslant)$ *be a finite poset with a unique maximal element* 1. *Further suppose that every two elements* F *and* G *have a meet. Prove that then also every two elements* F *and* G *have a join!*

For other fine-grained properties of the face lattice, see [12, Theorem 2.7].

The face lattice stores the combinatorial information of a polytope. Two polytopes are called *combinatorially equivalent* if they have isomorphic face lattices [12, Section 2.2]. Combinatorially equivalent polytopes may geometrically look different. For example, all triangles in the plane are combinatorially equivalent, but some are equilateral while others can be long and skinny.

10.4.3 Polarity

For every polytope $\mathcal{P} \ni 0$, there is a so-called **polar polytope** $\mathcal{P}^\triangle \ni 0$ whose face lattice is that of $\mathcal{P}$ but turned upside down [12, Theorem 2.11]. This means that the vertices of $\mathcal{P}$ correspond to facets of $\mathcal{P}^\triangle$, edges of $\mathcal{P}$ to ridges of $\mathcal{P}^\triangle$, and so on.

If $\mathcal{P} = \operatorname{conv}(P)$, then its polar polytope can be constructed as

$$\mathcal{P}^\triangle = \bigcap_{p \in P} h_p^- \quad \text{where} \quad h_p^- := \left\{ x \in \mathbb{R}^d : \sum_{i=1}^{d} p_i x_i \leqslant 1 \right\}.$$

Geometrically, going to the polar polytope corresponds to replacing a point (part of the description of the $\mathcal{V}$-polytope $\mathcal{P}$) with a halfspace (part of the description of the $\mathcal{H}$-polytope $\mathcal{P}^\triangle$). This operation is called *inversion at the unit sphere*; see Figure 10.7. It can be shown that $\mathcal{P}^{\triangle\triangle} = \mathcal{P}$.

We can also "polarize" $\mathcal{P}$ even if it does not contain the origin: simply choose the center of inversion at some interior point of $\mathcal{P}$. Depending on which point we choose,

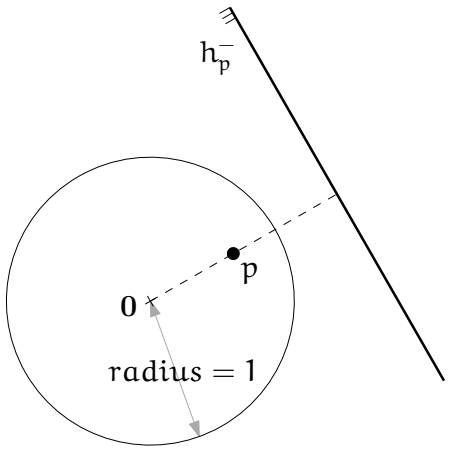


Figure 10.7: *The polar halfspace* h_p^- *has distance* $1/\|p\|$ *from the origin* 0 *and is perpendicular to the vector* p.

$\mathcal{P}^\triangle$ can look different geometrically, but the combinatorial structure (i.e. face lattice) is nevertheless the same.

We have already seen some pairs of polar polytopes: In fact, each platonic solid is polar to another one (Figure 10.8). As a sanity check, the dodecahedron has 12 facets (hence its name), 30 edges and 20 vertices; its polar, the icosahedron, has 20 facets (hence its name), 30 edges and 12 vertices.

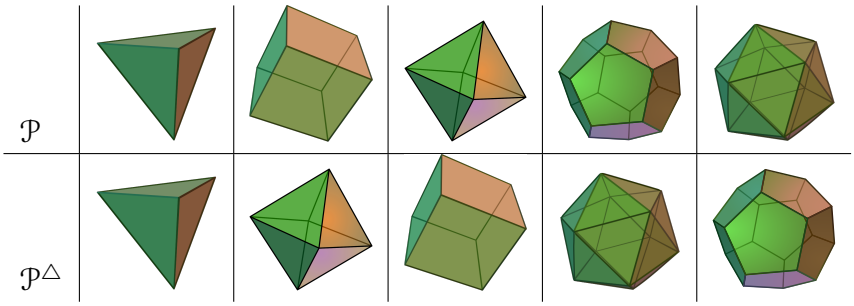


Figure 10.8: *Polarities among the platonic solids: the tetrahedron is polar to itself (first column); cube and octahadreon are polar to each other (second and third columns); dodecahedron and icosahedron are polar to each other (fourth and fifth columns).*

Three of the platonic solids generalize to polytopes in arbitrary dimension d, and we have already encountered two of these in Section 10.3: simplices and hypercubes. Exercise 10.22 below asks you to show that simplices are polar to simplices. What polytopes are polar to hypercubes? These are called *cross-polytopes* which generalizes the octahedron. The standard d-dimensional cross-polytope is the convex hull of the $2d$ points

$(\pm 1, 0, \ldots, 0), \ldots, (0, 0, \ldots, \pm 1)$. Equivalently, we may represent it as the intersection of 2^d halfspaces $\{x \in \mathbb{R}^d : \sum_{i=1}^d h_i x_i \leqslant 1\}$ for $h \in \{-1, 1\}^d$.

Exercise 10.22. *Argue that the face lattice of a d-simplex is isomorphic to the Boolean cube $(\{0,1\}^{d+1}, \leqslant)$. Conclude that d-simplex is polar to itself. (Hint: Exercise 10.18.)*

Polarity sometimes yields surprisingly short proofs that would otherwise require a non-trivial argument. Below is an example.

Lemma 10.23. *Every proper face is contained in some facet.*

Proof. Via polarity, the statement for $\mathcal{P}$ translates to "every proper face in $\mathcal{P}^\triangle$ contains one or more vertices". The latter follows from Lemma 10.7. □

10.5 Simplicial and Simple Polytopes

Let us return to the important question that we asked earlier in this chapter:

How many facets can a d-dimensional polytope with n vertices have?

As we have discussed, the answer is $\Theta(n)$ for $d = 2, 3$. For general d, we have an upper bound of $O(n^d)$ from Exercise 10.8, which is an overestimate for $d = 2, 3$ already. The full answer will come later in Chapter 11, but let us make a general observation right now. To address the question, we can actually restrict our attention to **simplicial polytopes**. These are d-dimensional polytopes whose facets are all simplices (or more specifically, $(d-1)$-simplices). For example, the octahedron is simplicial since all its facets are triangles (2-simplices), whereas the dodecahedron is not since its facets are pentagons.

Fixing dimension d and the number n of vertices, the number of facets can only be maximized by a simplicial polytope. The reason is that a non-simplicial polytope can be "made simplicial" by slightly and randomly perturbing its vertices. Intuitively, each non-simplex facet "breaks apart" and gets replaced by several simplex facets. More formally, with probability 1, all subsets of $d+1$ vertices become affinely independent, thus forcing every facet to contain exactly d vertices (otherwise its dimension would not be $d-1$). Hence each facet is a $(d-1)$-simplex now. One can show that the original facets injectively maps to the new facets, so the number of facets cannot decrease.[3]

Let's illustrate this in the cube $[-1, 1]^3$. Suppose that we push the two vertices $(-1, -1, -1)$ and $(1, 1, 1)$ "slightly inwards" so that they become $(-1+\varepsilon, -1+\varepsilon, -1+\varepsilon)$ and $(1-\varepsilon, 1-\varepsilon, 1-\varepsilon)$, respectively, for some small $\varepsilon > 0$, then we obtain the simplicial polytope in Figure 10.9. Similarly, for the dodecahedron, each pentagon facet gets replaced by three triangles when we slightly perturb the vertices.

[3] In fact, it will strictly increase. See [12, Lemma 8.24] for a formal statement and reference to a proof.

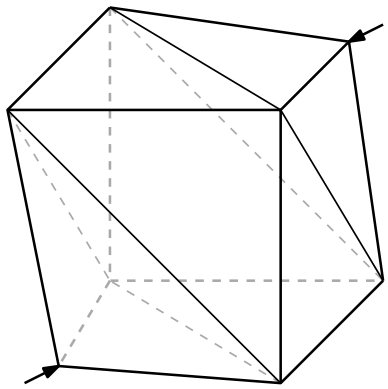


Figure 10.9: *Perturbing the cube vertices: by pushing the two diagonal vertices slightly inwards, each square facet breaks up into two triangles, and the resulting polytope is simplicial.*

Exercise 10.24. *What happens if we move the two vertices "slightly outwards" so that they become* $(-1-\varepsilon,-1-\varepsilon,-1-\varepsilon)$ *and* $(1+\varepsilon,1+\varepsilon,1+\varepsilon)$*, respectively? Draw the resulting simplicial polytope!*

Simplicial polytopes not only maximizes the number of facets; they are also very nice structurally. In fact, not only their facets but also *all* their faces are simplices!

Proposition 10.25. *A* d*-dimensional polytope is simplicial if and only if every* k*-face is a* k*-simplex, for* $0 \leqslant k \leqslant d-1$*.*

Proof. The ($\Leftarrow$) direction is trivial. For the ($\Rightarrow$) direction, Lemma 10.23 states that every k-face F is contained in some facet F', which is a simplex by assumption. In particular, $V(F) \subseteq V(F')$ is a set of affinely independent points. So F is a k-simplex. □

We can also define the polar notion of simplicial polytopes. A polytope is **simple** if every vertex is incident to d edges. As polarity transform turns the face lattice upside down, a polytope is simple if and only if its polar polytope is simplicial. Checking Figure 10.8 again, the tetrahedron is both simple and simplicial, the octahedron as well as the icosahedron are simplicial, and their polars—the cube and the dodecahedron—are simple. Via polarity, an alternative way to phrase our initial question is:

How many vertices can a d-dimensional polytope with n facets have?

The count is maximized only by the simple polytopes.

Exercise 10.26. *Characterize all polytopes in* $\mathbb{R}^3$ *that are both simple and simplicial.*

10.6 High-Dimensional Delaunay Triangulations

In discussing Delaunay triangulations and proving the termination of the Lawson flip algorithm in Section 6.3, we have argued that every triangulation in the plane gives rise to a "lifted surface" that can pointwise only decrease in height under a Lawson flip, so that eventually no Lawson flip is possible any more. In this section we discuss more systematically what the lifted surface actually is after the algorithm terminates, that is, when the triangulation has become Delaunay. In fact, we want to generalize this to arbitrary dimension d.

We will give the big picture upfront, borrowing the very neat Figure 10.10 from Hang Si [9]. Let us consider a planar point set in general position (no three points on a line, no four points on a circle), so its Delaunay triangulation is unique by Corollary 6.18. Imagine lifting the points to the unit paraboloid in $\mathbb{R}^3$ and consider the convex hull of the lifted points—a polytope in $\mathbb{R}^3$. Its lower facets (triangles by general position), when projected back to $\mathbb{R}^2$, must satisfy the empty circle property (cf. Lemma 6.12) and thus yield the Delaunay triangulation. See the lower right part in Figure 10.10.

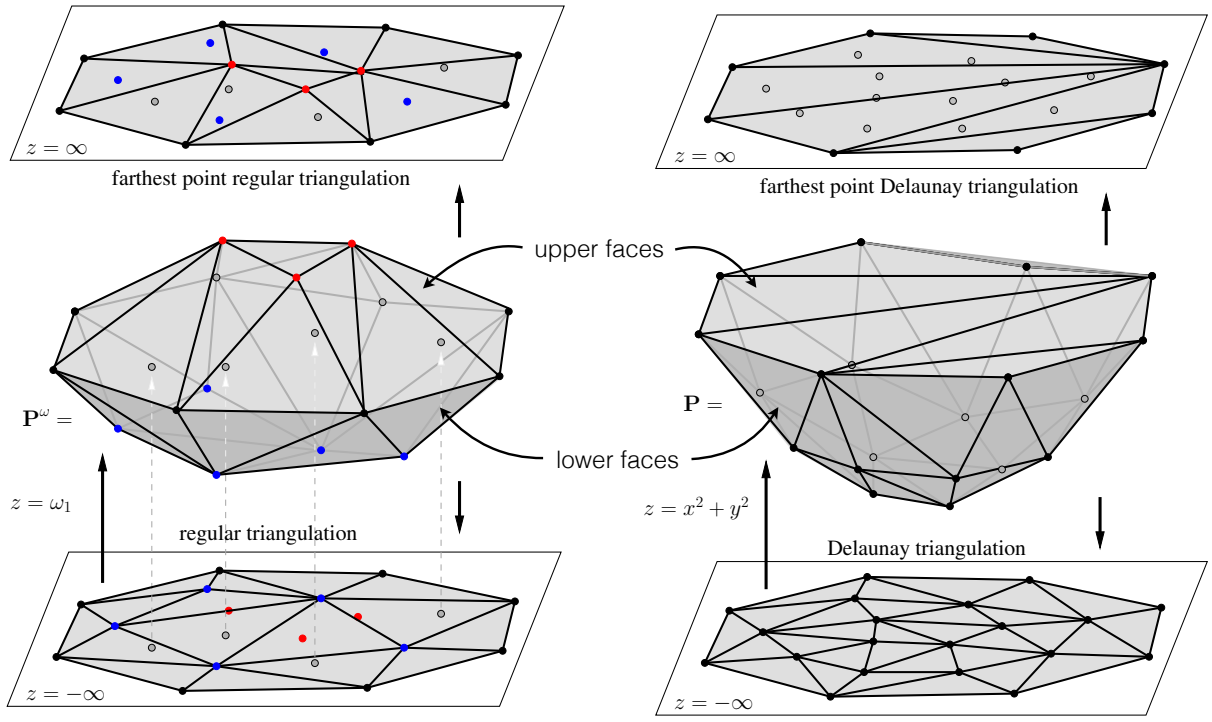


Figure 10.10: *Triangulations in $\mathbb{R}^d$ as projections of polytopes in $\mathbb{R}^{d+1}$*

So the "lifted surface" when the Lawson flip algorithm terminates is exactly the *lower convex hull* of the lifted points. This also means that we can reduce the computation of the Delaunay triangulation to computing a convex hull in $\mathbb{R}^3$. We will formally state and prove this relation for general dimension soon.

Figure 10.10 shows more. In the upper right part, we see what happens when we project the *upper* facets back to $\mathbb{R}^2$. The result is called the *farthest-point Delaunay triangulation*. It is generally not a triangulation of the point set, but only of the extreme points. Each triangle in this triangulation is "anti-Delaunay" in the sense that its circumcircle contains *all* other points; see Exercise 10.41 below. The left part of Figure 10.10 shows what happens if we lift the points not onto the unit paraboloid but in some arbitrary way. The convex hull of the lifted points is still a polytope, and if it is simplicial, we can recover two triangulations in the plane by projecting the lower and upper facets back to $\mathbb{R}^2$, respectively. Such triangulations are called *regular*; the (farthest-point) Delaunay triangulation is a specific regular triangulation.

After the pictorial outline, we will now formalize the intuition. In Definition 6.1, we have introduced triangulations of point sets in the plane. We can generalize it to higher dimensions in a straightforward way, replacing "triangles" by "d-simplices". We still call it a triangulation, for lack of a better name derived from the word "simplex".

Definition 10.27. *A* **triangulation** *of a finite point set $P \subset \mathbb{R}^d$ is a collection $\mathcal{T}$ of d-simplices, such that*

(1) $\operatorname{conv}(P) = \bigcup_{T \in \mathcal{T}} T$*;*

(2) $P = \bigcup_{T \in \mathcal{T}} V(T)$*; and*

(3) for every distinct pair $T, T' \in \mathcal{T}$, the intersection $T \cap T'$ is a face of both.[4]

For $d = 2$ we recover Definition 6.1. Also for $d = 1$ the definition makes sense, as a point set $\{a_1, a_2, \ldots, a_n\}$ in $\mathbb{R}^1$ (assuming $a_1 < a_2 < \cdots < a_n$) has a unique triangulation $\mathcal{T} = \{[a_i, a_{i+1}] : 1 \leqslant i < n\}$.

However, at the moment it is not clear whether every point set in $\mathbb{R}^d$ has a triangulation, for $d \geqslant 3$. Anyway, we go ahead and define Delaunay triangulations in the same way as before.

Definition 10.28. *A triangulation $\mathcal{T}$ of a finite point set $P \subset \mathbb{R}^d$ is a* **Delaunay triangulation***, if the circumsphere of every d-simplex $T \in \mathcal{T}$ is empty of points from P.*

What is the circumsphere of a d-simplex? This is the unique sphere that contains all its $d + 1$ vertices. Before you can even question about its existence and uniqueness, let us prove it.

Lemma 10.29. *Let $S \subset \mathbb{R}^d$ be a set of $d + 1$ affinely independent points. Then there exists a unique sphere containing S.*

Proof. Recall that a sphere with center $c \in \mathbb{R}^d$ and radius $r \geqslant 0$ is formally defined as the set $\{x \in \mathbb{R}^d : \|x - c\| = r\}$. Squaring the condition, we are looking for a (unique) point $c \in \mathbb{R}^d$ and a (unique) number $r \geqslant 0$ such that

$$\|q - c\|^2 = r^2, \quad \forall q \in S \tag{10.30}$$

[4]Note that this also allows for $T \cap T' = \emptyset$, since $\emptyset$ is a face of every polytope.

As usual, we understand a point $x \in \mathbb{R}^d$ as a column vector. Then $x^\top y$ is the scalar product $\sum_{i=1}^d x_i y_i$ of two points $x, y \in \mathbb{R}^d$. With this, the previous system of equations is equivalent to

$$\|q\|^2 = 2q^\top c + \underbrace{r^2 - \|c\|^2}_{=:\alpha}, \quad \forall q \in S. \tag{10.31}$$

In still other words,

$$\|q\|^2 = (q^\top, 1) \begin{pmatrix} 2c \\ \alpha \end{pmatrix}, \quad \forall q \in S.$$

Stacking the $d+1$ equations row by row, this is a linear system of the form $b = B \begin{pmatrix} 2c \\ \alpha \end{pmatrix}$ where $b \in \mathbb{R}^{d+1}$ and $B \in \mathbb{R}^{(d+1)\times(d+1)}$, one row for each $q \in S$. As the points $q \in S$ are affinely independent, the rows of B are lineary independent (Proposition 5.4) and so B is invertible. So there is a unique $c \in \mathbb{R}^d$ and a unique $\alpha \in \mathbb{R}$ satisfying (10.31), which uniquely determine $r^2 := \alpha + \|c\|^2$ and satisfy (10.30). Note that such r^2 must be positive because the left hand side of this satisfied equation (10.30) is always positive. □

Next we want to show that there is a unique Delaunay triangulation, assuming sufficiently general position. This means that no $d+1$ points lie on a common hyperplane, and no $d+2$ points lie on a common sphere.

As a preparation, we first define the concept of a Delaunay simplex.

Definition 10.32. *Let $P \subset \mathbb{R}^d$ be a set of points in general position. A simplex $\operatorname{conv}(S)$ where $S \in \binom{P}{d+1}$ is called a* **Delaunay simplex** *for P if the circumsphere of S is empty of points from P.*

Next comes the crucial insight. Generalizing Section 6.3, we show that Delaunay simplices correspond to "lower" facets of a polytope in one dimension higher, namely the convex hull of the lifted points. For $p = (p_1, \dots, p_d) \in \mathbb{R}^d$, we define the *lifted point*

$$\ell(p) := (p, \|p\|^2) = (p_1, \dots, p_d, \|p\|^2) \in \mathbb{R}^{d+1}. \tag{10.33}$$

For $d = 2$, this is the standard lifting map that raises points in the plane to the unit paraboloid in $\mathbb{R}^3$. The following lemma naturally extends Lemma 6.12.

Lemma 10.34. *For any sphere $S \subset \mathbb{R}^d$, there is an upward hyperplane[5] $h \subset \mathbb{R}^{d+1}$ such that the following property holds. A point $q \in \mathbb{R}^d$ is on/outside/inside S if and only if the lifted point $\ell(q) \in \mathbb{R}^{d+1}$ is on/above/below h.*

Conversely, for any upward hyperplane $h \subset \mathbb{R}^{d+1}$ that intersects the unit paraboloid, there is a sphere $S \subset \mathbb{R}^d$ such that the aforementioned property holds.

[5]Being upward means that the coefficient for the last coordinate is positive. By scaling the equation appropriately, we may assume that the coefficient is exactly one. Geometrically, the normal vector of such hyperplane is pointing upward.

Proof. We have already done most of the work in the proof of Lemma 10.29. Given a sphere $\mathcal{S} \subset \mathbb{R}^d$ with center c and radius r, let us denote $\alpha := r^2 - \|c\|^2$. Along the same lines of deriving (10.31), we have

$$\begin{aligned} \|q\|^2 &= 2q^\top c + \alpha \iff q \text{ on } \mathcal{S}, \\ \|q\|^2 &> 2q^\top c + \alpha \iff q \text{ outside } \mathcal{S}, \\ \|q\|^2 &< 2q^\top c + \alpha \iff q \text{ inside } \mathcal{S}. \end{aligned} \tag{10.35}$$

Recall that $\ell(q) = (q, \|q\|^2)$, so the formulas may be rephrased as

$$\begin{aligned} (-2c^\top, 1)\ell(q) &= \alpha \iff q \text{ on } \mathcal{S}, \\ (-2c^\top, 1)\ell(q) &> \alpha \iff q \text{ outside } \mathcal{S}, \\ (-2c^\top, 1)\ell(q) &< \alpha \iff q \text{ inside } \mathcal{S}. \end{aligned} \tag{10.36}$$

In other words, the lifted point $\ell(q)$ is on/above/below the upward hyperplane $h := \{x \in \mathbb{R}^{d+1} : (-2c^\top, 1)x = \alpha\}$ in the respective cases.

Conversely, given an upward hyperplane $h \subset \mathbb{R}^{d+1}$, let $h_1, \ldots, h_d, 1, h_{d+2}$ be its parameters. Define $c_i := -h_i/2$, $\alpha := h_{d+2}$ and $r^2 := \alpha + \|c\|^2$, and consider the *formal* sphere $\mathcal{S} := \{x \in \mathbb{R}^d : \|x - c\| = r\}$. It is not yet clear that $r^2 \geqslant 0$, or $\mathcal{S} \neq \emptyset$, but the derivations (10.35) and (10.36) hold anyway, so the desired property definitely hold. We just need to show $\mathcal{S} \neq \emptyset$. To this end, recall from assumption that h intersects the unit paraboloid, thus the "=" case in (10.36) does happen, which certifies that there is *some* point on $\mathcal{S}$. □

Corollary 10.37. *Let $P \subset \mathbb{R}^d$ be a finite point set. Denote by $\ell(P) = \{\ell(p) : p \in P\}$ the set of lifted points. Then the polytope $\mathcal{P} := \operatorname{conv}(\ell(P))$ has vertex set $\ell(P)$.*

Proof. By definition $V(\mathcal{P}) \subseteq \ell(P)$, so it remains to show that $\ell(p)$ is vertex for all $p \in P$. To this end, apply Lemma 10.34 to the singleton $\mathcal{S} = \{p\}$ (a sphere with center p and radius 0!) and get a hyperplane h. Every point $q \in P \setminus p$ is outside the sphere $\mathcal{S}$, so its lifting $\ell(q)$ is above the hyperplane h. Hence h supports $\{\ell(p)\}$ by Lemma 10.4. □

Lemma 10.38. *Let $P \subset \mathbb{R}^d$ be a finite point set in general position. Then $\mathcal{P} = \operatorname{conv}(\ell(P))$ is a simplicial polytope in $\mathbb{R}^{d+1}$. Moreover, for any subset $S \subseteq P$, the following two statements are equivalent.*

- *$\operatorname{conv}(S)$ is a Delaunay simplex for P.*
- *$\operatorname{conv}(\ell(S))$ is a lower facet of $\mathcal{P}$, meaning that it is a facet supported by some upward hyperplane.*

Proof. That $\mathcal{P}$ is simplicial follows from general position. Indeed, every facet of $\mathcal{P}$ is a d-face, so it contains at least $d+1$ (affinely independent) vertices. But it cannot contain more: All the vertices, necessarily in the form $\ell(p)$, $p \in P$ by Corollary 10.37, are lying on a common hyperplane in $\mathbb{R}^{d+1}$, so their projections onto $\mathbb{R}^d$ are on a common sphere by Lemma 10.34. General position requires the number to be less than $d+2$.

Now we proceed to the "moreover" part. Let $\mathrm{conv}(S)$ be a Delaunay simplex, so S consists of $d+1$ affinely independent points whose circumsphere is empty of points from P. Applying Lemma 10.34 on this sphere, there is an upward hyperplane h such that $\ell(P) \cap h = \ell(S)$ and $\ell(P) \subset h^+$. So h supports $\mathrm{conv}(\ell(S))$ by Lemma 10.4. Note that $\ell(S)$ consists of $d+1$ affinely independent points, so the face $\mathrm{conv}(\ell(S))$ has dimension d and is a (lower) facet, indeed.

Conversely, assume that $\mathrm{conv}(\ell(S))$ is a lower facet (a d-simplex since the polytope is simplicial), supported by some upward hyperplane h. This time we apply Lemma 10.34 on h, and obtain a sphere that goes through S and satisfies the empty property. Finally, S is indeed a simplex by general position (no $d+1 = |S|$ points on a common hyperplane). □

From this correspondence, we may obtain the existence of a unique Delaunay triangulation for a finite point set $P \subset \mathbb{R}^d$ in general position.

Theorem 10.39. *Let $P \subset \mathbb{R}^d$ be a finite point set in general position. Then the collection $\mathcal{T}$ of all Delaunay simplices for P is the unique Delaunay triangulation of P.*

Proof. Suppose $\mathcal{T}$ is a triangulation. Then it would be a Delaunay triangulation by definition. The uniqueness also follows: Any other Delaunay triangulation consists of Delaunay simplices, thus a proper subset of $\mathcal{T}$; but then it cannot cover $\mathrm{conv}(P)$ in full, a contradiction.

It remains to prove that $\mathcal{T}$ is a triangulation, so let's look at the three properties in Definition 10.27. Denote by $\mathcal{P} = \mathrm{conv}(\ell(P))$ the convex hull of the lifted points (a polytope in $\mathbb{R}^{d+1}$).

(1) $\mathrm{conv}(P) = \bigcup_{T \in \mathcal{T}} T$. Let $q \in \mathrm{conv}(P)$, and our goal is to find a simplex $T \in \mathcal{T}$ that contains q. Consider a vertical[6] line in $\mathbb{R}^{d+1}$ from $(q, -\infty)$ to (q, ∞). Since $q \in \mathrm{conv}(P)$, the line must intersect $\mathrm{conv}(\ell(P)) = \mathcal{P}$ in a non-empty closed interval (it is an interval since $\mathcal{P}$ is convex and compact). So let us choose the minimum height $t \in \mathbb{R}$ such that $(q, t) \in \mathcal{P}$. Then (q, t) is on the boundary of $\mathcal{P}$ and hence contained in one or more facets by Corollary 10.16; one of these must be a lower facet $\mathrm{conv}(\ell(S))$ by Exercise 10.40 below. So $q \in \mathrm{conv}(S)$. But we know $\mathrm{conv}(S) \in \mathcal{T}$ by Lemma 10.38, and this is the simplex we are looking for.

We have shown $\mathrm{conv}(P) \subseteq \bigcup_{T \in \mathcal{T}} \mathrm{conv}(T)$. The reverse inclusion follows from $\mathrm{conv}(S) \subseteq \mathrm{conv}(P)$ for all $S \in \binom{P}{d+1}$.

(2) $P = \bigcup_{T \in \mathcal{T}} V(T)$. The inclusion $\supseteq$ is trivial. Now for the other inclusion, let $p \in P$. We claim that $\ell(p)$ is the vertex of some lower facet of $\mathcal{P}$. Via Lemma 10.38 this implies that p is the vertex of some $T \in \mathcal{T}$.

[6] In high dimension, the word "vertical" should read "along the last axis". Similarly, the word "height" should read "the last coordinate".

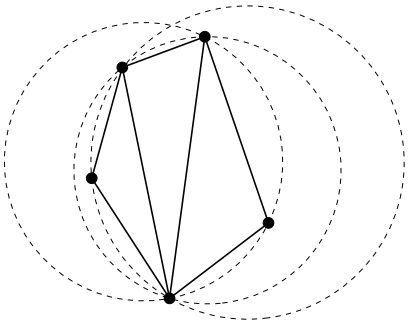
Here is the key observation: $\min\{t \in \mathbb{R} : (p,t) \in \mathcal{P}\} = \|p\|^2$. Indeed, for $t = \|p\|^2$ we have $(p,t) = \ell(p) \in \mathcal{P}$. But if $t < \|p\|^2$ then (p,t) is outside the convex "bowl" $\mathcal{U} = \{x \in \mathbb{R}^{d+1} : x_{d+1} \geqslant \sum_{i=1}^{d} x_i^2\}$, whereas $\mathcal{P} = \mathrm{conv}(\ell(P)) \subset \mathcal{U}$.

By the argument for (1), vertex $\ell(p)$ is hence contained in some lower facet $\mathrm{conv}(\ell(S))$ of $\mathcal{P}$ and is then also a vertex of this facet. So the claim is proved.

(3) The intersection of any two simplices $T, T' \in \mathcal{T}$ is a face of both. This follows from the general structure of polytopes. Let F and F' be the lower facets of $\mathcal{P}$ corresponding to the Delaunay simplices T and T'. By Lemma 10.9, the intersection $F \cap F'$ is a face of $\mathcal{P}$ with vertex set $\ell(V(T)) \cap \ell(V(T')) = \ell(V(T) \cap V(T'))$, hence $F \cap F' = \mathrm{conv}(\ell(V(T) \cap V(T')))$. Projecting back onto $\mathbb{R}^d$ we see $T \cap T' = \mathrm{conv}(V(T) \cap V(T'))$. This is a face of both simplices T and T', since every subset of vertices of a simplex defines a face (Exercise 10.18). □

Exercise 10.40. *Let $\mathcal{P} \subset \mathbb{R}^{d+1}$ be a polytope and $(q,t) \in \mathbb{R}^{d+1}$ such that $(q,t) \in \mathcal{P}$ but $(q,t') \notin \mathcal{P}$ for $t' < t$. Prove that (q,t) is contained in some lower facet of $\mathcal{P}$.*

Exercise 10.41. *Let $P \subset \mathbb{R}^d$ be a finite set of points in convex position (every point is extreme), and in general position (no $d+1$ points on a hyperplane, no $d+2$ on a sphere). A* farthest-point Delaunay triangulation *of P is a triangulation $\mathcal{T}$ of P with the property that the circumsphere of every d-simplex T in $\mathcal{T}$ contains all points $P \setminus V(T)$:*



Prove that P has a unique farthest-point Delaunay triangulation; Figure 10.10 provides the intuition. The name comes from the fact that in the plane, the farthest-point Delaunay triangulation is dual to the farthest-point Voronoi diagram*, the subdivision of the plane into regions with the same* farthest *point.*

10.7 Complexity of 4-Dimensional Polytopes

The **complexity** of a polytope is defined as the number of faces. Indeed, if we talk about computing a polytope, we typically mean that we want to compute its face lattice. In dimensions $d = 2, 3$, each polytope with n vertices has complexity $\Theta(n)$. We have also seen that for $d = 4$, the complexity is bounded by $O(n^4)$ (Exercise 10.8). But can we

actually have superlinear complexity for $d = 4$, or does the linear behavior in dimensions $d = 2, 3$ continue?

Using the previously derived connection to 3-dimensional Delaunay triangulations, we can answer this question.

Theorem 10.42. *For every even natural number $n \geqslant 4$, there exists a 4-dimensional simplicial polytope with n vertices and at least $(\frac{n}{2} - 1)^2 = \Theta(n^2)$ facets.*

Moreover, this polytope also has $\Theta(n^2)$ edges which is asymptotically maximal since Exercise 10.8 implies that there are $O(n^2)$ edges. In particular, vertex-edge graphs of 4-dimensional polytopes can be dense and highly non-planar. They can even be complete [12, Corollary 0.8]. This may be somewhat counter-intuitve, as it seems to require the edges "cutting through" the polytope which they obviously cannot. On the other hand, 4 dimensions are counterintuitive *per se*, so let's not worry to much about intuition here.

Proof. We construct a point set $P \subset \mathbb{R}^3$ in general position, $|P| = n$, for which there are at least $(\frac{n}{2} - 1)^2$ Delaunay simplices. By Lemma 10.38, the convex hull of the lifted point set $\ell(P)$ is a 4-dimensional simplicial polytope with at least $(\frac{n}{2} - 1)^2$ (lower) facets.

Let ℓ_1, ℓ_2 be two skew (non-parallel, non-intersecting) lines in $\mathbb{R}^3$. We choose a set P_1 of $n/2$ points on ℓ_1, and another set P_2 of $n/2$ points on ℓ_2. Then we set $P = P_1 \cup P_2$, after slightly perturbing all points to ensure general position.

The claim is that for any points $p, q \in P_1$ consecutive along ℓ_1, and points $r, s \in P_2$ consecutive along ℓ_2, their convex hull $\operatorname{conv}\{p, q, r, s\}$ is a Delaunay simplex. (See the cartoonish Figure 10.11.) As there are $(\frac{n}{2} - 1)^2$ ways to choose such p, q, r, s, there are at least this many Delaunay simplices.

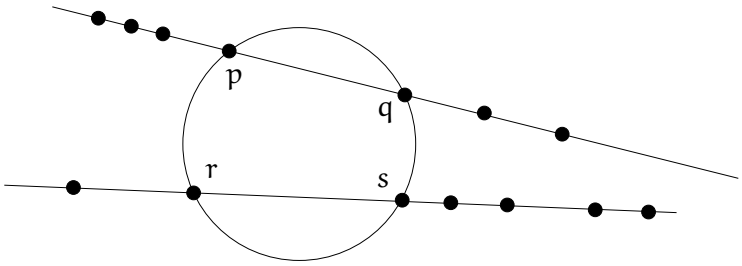


Figure 10.11: *Proof of Theorem 10.42*

It remains to prove the claim. The points p, q, r, s are affinely independent (by general position) and hence have a unique circumsphere. The line ℓ_1 intersects this sphere in exactly the line segment $\overline{pq}$; but p, q are consecutive along ℓ_1, so no point of P_1 lies inside the sphere. For the same reason, no point of P_2 lies inside. As the sphere is empty, $\operatorname{conv}\{p, q, r, s\}$ is a Delaunay simplex. □

It is actually the case that a 4-dimensional polytope with n vertices has $O(n^2)$ facets, so the lower bound provided by Theorem 10.42 is asymptotically best possible. We will

postpone the full account to the later Chapter 11, where we give a tight upper bound on the number of facets that a d-dimensional polytope with n vertices can have.

10.8 High Dimensional Voronoi Diagrams (not exam-relevant in HS2025)

Using lifting map, we can also relate the Voronoi diagram of a finite point set $P \subset \mathbb{R}^d$ with the facets of some *polyhedron* in $\mathbb{R}^{d+1}$. In fact this is what Theorem 8.16 did for $d = 2$, without explicitly mentioning polyhedra. Here we simply reprove this theorem for general d. No new idea appear, so the reader is invited to consider it as a repetition of Section 8.4, but formulated in the more abstract language of polyhedra and replacing "2" by "d".

Let us start by generalizing Voronoi cells to higher-dimensions which is a straightforward adaptation of Definition 8.3.

Definition 10.43. *Let $P \subset \mathbb{R}^d$ be a finite point set. The* **Voronoi cell** *of point $p \in P$ is defined as*

$$V_P(p) := \left\{ q \in \mathbb{R}^2 : \|q - p\| \leqslant \|q - p'\| \text{ for all } p' \in P \right\}.$$

In words, $V_P(p)$ is the set of points in $\mathbb{R}^d$ for which p is a (not necessarily unique) closest point among all points in P.

Theorem 10.44. *Let $P \subset \mathbb{R}^d$ be a finite point set. For each $p \in P$, we define a hyperplane*

$$h_p = \left\{ x \in \mathbb{R}^{d+1} : x_{d+1} - \sum_{i=1}^{d} 2p_i x_i = -\|p\|^2 \right\}.$$

Consider the polyhedron $\mathcal{P} := \bigcap_{p \in P} h_p^+$ in $\mathbb{R}^{d+1}$. Then every h_p supports a lower facet of $\mathcal{P}$. Moreover, for all $q \in \mathbb{R}^d$, the following two statements are equivalent.

(i) $q \in V_P(p)$.

(ii) $(q, t) \in h_p$, *where* $t \in \mathbb{R}$ *is the minimum value such that* $(q, t) \in \mathcal{P}$.

Pictorially, condition (ii) means that the vertical ray emanating up from $(q, -\infty)$ hits the polytope at the lower facet $\mathcal{P} \cap h_p$. Hence the theorem says that the Voronoi cell $V_P(p)$ is simply the vertical projection of the facet $\mathcal{P} \cap h_p$ back to $\mathbb{R}^d$. If we project all the facets of $\mathcal{P}$ to $\mathbb{R}^d$, we obtain the Voronoi diagram of P. Figure 10.12, borrowed from the book by Joswig and Theobald [7, Page 87], visualizes this for $d = 3$.

Proof. We first show that all h_p are actually facet-supporting hyperplanes. For this, it suffices to show that none of the halfspaces h_p^+ is redundant; see the remark above Corollary 10.16.

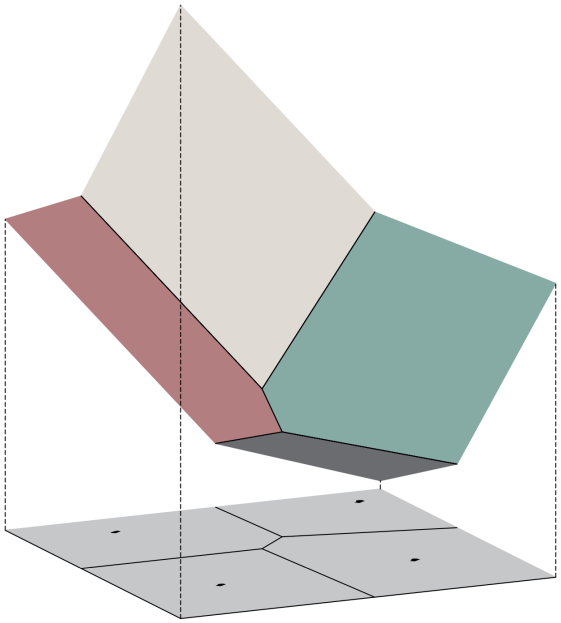


Figure 10.12: *A part of the polyhedron $\mathcal{P} \subset \mathbb{R}^3$ in Theorem 10.44, and the Voronoi diagram as the projections of its facets to $\mathbb{R}^2$.*

The mysterious-looking hyperplanes h_p are actually our old friends! They appeared implicitly in the proof of Corollary 10.37. The hyperplane h_p is obtained exactly by applying Lemma 10.34 on the singleton sphere $\{\ell(p)\}$, where ℓ denotes the lifting map. Hence $\ell(p)$ is on h_p and strictly above other h_q. So it is in the interior of the polyhedron $\bigcap_{q \in P \setminus \{p\}} h_q^+$ but on the boundary of $\bigcap_{q \in P} h_q^+$, thus the hyperplane h_p is not redundant.

Next we claim that the vertical distance from $\ell(q)$ to h_p is precisely $\|q - p\|^2$ (cf. Lemma 8.14 and Figure 8.7). Indeed, $\ell(q) = (q, \|q\|^2)$ has height $\|q\|^2$, and its vertical projection onto h_p has height

$$x_{d+1} = \sum_{i=1}^{d} 2p_i q_i - \|p\|^2 = 2p^\top q - \|p\|^2.$$

So the vertical distance is $\|q\|^2 - 2p^\top q + \|p\|^2 = \|q - p\|^2$.

Now we can show the equivalence of (i) and (ii). Given any point $q \in \mathbb{R}^d$, $\ell(q)$ is on or above the hyperplanes h_p. So by the claim we have the following chain of equivalence:

- $q \in V_P(p)$.
- The vertically closest hyperplane to $\ell(q)$ is h_p.
- Projecting $\ell(q)$ vertically onto the hyperplanes, the highest point is on h_p.
- Raising $(q, -\infty)$ vertically until we hit $\mathcal{P}$, the hitting point is on h_p.
- $(q, t) \in h_p$. □

10.9 Regular Triangulations and the Secondary Polytope

We have seen in Section 10.6 that the Delaunay triangulation of a set P of points in $\mathbb{R}^d$ can be constructed as the projection of the lower convex hull of the polytope in $\mathbb{R}^{d+1}$ defined by lifting the points in P to the (generalized) unit paraboloid. As already hinted at in Section 10.6, we can get other (partial) triangulations through other liftings. In order to make this precise, consider a set $P = \{p_1, \ldots, p_n\}$ of points in $\mathbb{R}^d$ and let $\omega : P \to \mathbb{R}$ be a *height function*. Denote by $P^\omega := \{(p_1, \omega(p_1), \ldots, (p_n, \omega(p_n))\}$ the *lifted point set*. Consider now the polytope $\mathcal{P} = \text{conv}(P^\omega)$ and let $Q^\omega \subseteq P^\omega$ be the vertices of $\mathcal{P}$ that lie on a lower face of $\mathcal{P}$. Projecting the lower convex hull of $\mathcal{P}$ back down to $\mathbb{R}^d$ then gives a convex subdivision on Q and thus a partial triangulation of P. Note that if Q^ω is in general position then the lower convex hull is simplicial and we get a triangulation of Q. If additionally we have that $Q^\omega = P^\omega$, that is, all vertices of $\mathcal{P}$ lie on the lower convex hull (as is the case when lifting to the generalized unit paraboloid), then we get a triangulation of P.

Definition 10.45. *A (partial) triangulation of a set P of points in $\mathbb{R}^d$ is* regular *if it can be obtained by projecting the lower convex hull of a lifting of P to $\mathbb{R}^{d+1}$.*

From Section 10.6 we can deduce that every point set admits a regular triangulation, for example the Delaunay triangulation. On the other hand, we may ask whether every triangulation is regular? While this is true for point sets in $\mathbb{R}^1$ (see Exercise 10.47), it fails already in the plane.

Theorem 10.46. *There is a triangulation of six points in $\mathbb{R}^2$ that is not regular.*

Proof. Consider the following point set, which is also called the *mother of all examples* (see Figure 10.13). It consists of six points that lie on two nested equilateral triangles p_1, p_2, p_3 and p_4, p_5, p_6, where the lines p_1p_4, p_2p_5 and p_3p_6 all intersect in a single point. Consider the triangulation $\mathcal{T}$ obtained by drawing the edges of these two triangles, together with the edges p_1p_5, p_2p_6 and p_3p_4. We claim that this triangulation is not regular.

Assume for the sake of contradiction that $\mathcal{T}$ is regular, that is, there is a lifting map ω such that $\mathcal{T}$ is the lower convex hull of the polytope defined by the lifted points. Without loss of generality we may assume that $\omega(p_4) = \omega(p_5) = \omega(p_6) = 0$. Now, in order to get the edge p_1p_5 we need $\omega(p_2) > \omega(p_1)$. Similarly, from the edges p_2p_6 and p_3p_4 we get $\omega(p_3) > \omega(p_2)$ and $\omega(p_1) > \omega(p_3)$. Combining these three inequalities, we get $\omega(p_1) > \omega(p_1)$, which is a contradiction. □

Exercise 10.47. *Show that every partial triangulation of a set P of points in $\mathbb{R}^1$ is regular.*

While not every triangulation is regular, the set of all regular triangulations of a point set have a very nice structure: they again form a polytope, as discovered by Gelfand, Karpanov and Zelevinsky in 1989 [5, 6].

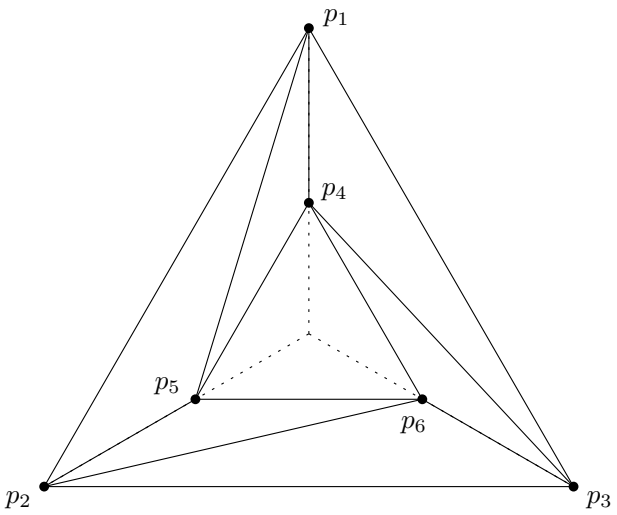


Figure 10.13: *A non-regular triangulation of the mother of all examples.*

Definition 10.48. *Let* P *be a set of points in* $\mathbb{R}^d$ *and let* $\mathcal{T}$ *be a triangulation of* P*. The* GKZ-vector $\Phi_P(\mathcal{T})$ *is defined as*

$$\Phi_P(\mathcal{T}) := \sum_{p \in P} \sum_{F \in \mathcal{T} : p \in F} \mathrm{vol}(F) e_p \subset \mathbb{R}^{|P|},$$

where $\mathrm{vol}()$ *denotes the volume and* e_i *is the standard unit vector in dimension* i*.*

The polytope

$$\Sigma(P) := \mathrm{conv}\{\Phi_P(\mathcal{T}) \mid \mathcal{T} \text{ is a triangulation of } P\} \subset \mathbb{R}^{|P|}$$

is the secondary polytope *of* P*.*

While the secondary polytope is defined as a subset of $\mathbb{R}^{|P|}$, its dimension is smaller

Theorem 10.49. *Let* P *be a set of* n *points in* $\mathbb{R}^d$*, some* $d+1$ *of which are affinely independent. Then the dimension of* $\Sigma(P)$ *is* $n-d-1$*.*

By definition, the secondary polytope contains all triangulations of P, but a priori they might all lie in the interior. As it turns out, it is exactly the regular triangulations that do not lie in the relative interior of any face:

Theorem 10.50 ([6])**.** *The vertices of the secondary polytope* $\Sigma(P)$ *are in one-to-one correspondence with the regular triangulations of* P*. In other words, the GKZ-coordinates* $\Phi_P(\mathcal{T})$ *of a triangulation* $\mathcal{T}$ *are extremal in the set of all GKZ-coordinates of triangulations of* P *if and only if* $\mathcal{T}$ *is regular.*

Now that we understand the vertices of the secondary polytope, we might ask ourselves whether the edges also have a nice interpretation. Indeed they correspond exactly

to *flips*. We have seen flips of triangulations in the plane in Chapter 6, where any four points in convex position defined a flip that replaced two triangles with two other triangles, flipping between the two possible triangulations of the four points. In higher dimension the situation is analogous, but slightly more intricate.

Lemma 10.51. *Let* $P = (p_1, \dots, p_{d+2})$ *be a set of* $d+2$ *points in convex position in* $\mathbb{R}^d$. *Then* P *has exactly two triangulations, both of which are regular.*

Proof. By Radon's theorem, P can be partitioned in a unique way into two parts P^+ and P^- such that $\mathrm{conv}P^+ \cap \mathrm{conv}(P^-) \neq \emptyset$. Note that as P is in convex position, each part contains at least two points. Consider now the two sets of simplices

$$\mathcal{T}^+ := \{F \subset P \mid P^+ \nsubseteq F\} \text{ and } \mathcal{T}^- := \{F \subset P \mid P^- \nsubseteq F\}.$$

As $\mathrm{conv}P^+ \cap \mathrm{conv}(P^-) \neq \emptyset$, any triangulation of P is contained either in $\mathcal{T}^+$ or in $\mathcal{T}^-$, but not in both. It thus remains to show that both $\mathcal{T}^+$ and$\mathcal{T}^-$ are regular triangulations.

Recall that by Radon's theorem there are nonzero coefficients λ_i such that

$$\sum_{p_i \in P^+} \lambda_i p_i = \sum_{p_i \in P^-} \lambda_i p_i,$$

where $\lambda_i > 0$ for $p_i \in P^+$ and $\lambda_i < 0$ for $p_i \in P^-$. Consider now some height function $\omega : P \to \mathbb{R}$ and let

$$c = \sum_{p_i \in P} \lambda_i \omega(p_i).$$

If $c = 0$ then the lifted points lie on a hyperplane. If $c > 0$ then each point in P^+ is above the unique hyperplane spanned by the other $d+1$ lifted points, which implies that the set of lower facets is

$$\{P \setminus \{p_i\} \mid p_i \in P^+\} = \{F \subset P \mid P^+ \nsubseteq F\},$$

which shows that $\mathcal{T}^+$ is regular. The analogous argument for $c < 0$ shows that $\mathcal{T}^-$ is regular. □

Lemma 10.51 now allows us to define a flip in general dimension analogously to the planar case: a flip on a triangulation $\mathcal{T}$ of a set of points P in $\mathbb{R}^d$ takes a triangulated convex subset of P' of P of size $d+2$ and replaces the triangulation on P' with the unique other one. With this we can again define the *flip graph* of triangulations of a point set.

Theorem 10.52. *Let* P *be a set of points in* $\mathbb{R}^d$. *Then the edge graph of the secondary polytope* $\Sigma(P)$ *is contained in the flip graph of* regular *triangulations of* P.

Using Balinski's theorem, we can thus conclude the following

Corollary 10.53. *Let P be a set of n points in $\mathbb{R}^d$, some $d+1$ of which are affinely independent. Then the flip graph of regular triangulations of P is $(n-d-1)$-connected.*

Note that we have restricted our attention to the flip graph of *regular* triangulations. If we also include non-regular triangulations, the situation becomes much more difficult. Even in the plane, it was only shown in 2020 by Wagner and Welzl [11] that the flip graph of *all* triangulations of a set of n points in $\mathbb{R}^2$ is $(n/2-2)$-connected (compared to the $(n-3)$-connectedness for regular triangulations). In dimensions 5 and above there are point sets where the flip graph of all triangulations is not connected. In dimensions 3 and 4 it is still an open problem whether the flip graph of all triangulations is connected, even for convex point sets.

Questions

51. *What is a polytope?* Give a definition and provide a few examples.
52. *What is a face of a polytope? What is a vertex, an edge, a ridge, a facet?* Give precise definitions!
53. *Can you characterize vertex-edge graphs of 3-dimensional polytopes?* Explain Steinitz' Theorem.
54. *What is a hypercube? What is a simplex?* Define these polytope and explain what their faces are.
55. *How many k-faces can a d-dimensional polytope with n vertices have?* Prove a nontrivial upper bound.
56. *What is the face lattice of a polytope?* Give a precise definition, explain what the lattice property is, and why it holds for the face lattice of a polytope.
57. *What is the polar of a given polytope?* Explain the polarity transform and how face lattices of the original polytope and its polar relate to each other. Show a pair of mutually polar polytopes and interpret the aforementioned relation in the example.
58. *What are simple and simplicial polytopes?* Explain why they are relevant with respect to counting the maximal number of facets (or vertices) that a d-dimensional polytope with n vertices (or facets) can have.
59. *How connected is the graph of a polytope?* State and prove Balinski's theorem.
60. *What is a d-dimensional (Delaunay) triangulation?* Give a precise definition.
61. *Does every point set $P \subseteq \mathbb{R}^d$ have a Delaunay triangulation?* Explain why the answer is yes under general position, why the Delaunay triangulation is unique in this case, and how you can obtain it from a polytope in one dimension higher.

62. *How many facets can a* 4*-dimensional polytope with* n *vertices have?* Prove a lower bound of $\Omega(n^2)$.

63. **(This topic was not covered in HS25 and therefore the question will not be asked in the exam.)** *What is a* d*-dimensional Voronoi diagram?* Give a definition and explain how the Voronoi diagram relates to a polyhedron in one dimension higher!

64. *What is a regular triangulation?* Give the definition and show that not all triangulations are regular.

65. *What is the secondary polytope?* Describe the construction and explain its connection to the flip graph of regular triangulations.

References

[1] Kjell André and DTR, Dodecahedron.svg. https://commons.wikimedia.org/w/index.php?curid=2231561, licensed under CC BY-SA 3.0.

[2] Kjell André and DTR, Hexahedron.svg. https://commons.wikimedia.org/w/index.php?curid=2231470, licensed under CC BY-SA 3.0.

[3] Kjell André and DTR, Tetrahedron.svg. https://commons.wikimedia.org/w/index.php?curid=2231463, licensed under CC BY-SA 3.0.

[4] Cyp and DTR, Icosahedron.svg. https://commons.wikimedia.org/w/index.php?curid=2231553, licensed under CC BY-SA 3.0.

[5] I.M. Gelfand, M.M. Kapranov, and A.V. Zelevinsky, Newton polyhedra of principal a-determinants. *Soviet Math. Dokl.*, 40, (1990), 278–281.

[6] I.M. Gelfand, M.M. Kapranov, and A.V. Zelevinsky, Discriminants of polynomials in several variables and triangulations of Newton polytopes. *Leningrad Math. J.*, 2, (1991), 449–505.

[7] Michael Joswig and Thorsten Theobald, *Polyhedral and Algebraic Methods in Computational Geometry*, Universitext, Springer, 2013.

[8] Francisco Santos, A counterexample to the Hirsch Conjecture. *Annals of Math.*, 176/1, (2012), 383–412.

[9] Hang Si, On Monotone Sequences of Directed Flips, Triangulations of Polyhedra, and Structural Properties of a Directed Flip Graph. *CoRR*, abs/1809.09701, (2018), 1–40.

[10] Stannered, Octahedron.svg. https://commons.wikimedia.org/w/index.php?curid=1742116, licensed under CC BY-SA 3.0.

[11] Uli Wagner and Emo Welzl, Connectivity of triangulation flip graphs in the plane. *Discrete & Computational Geometry*, 68/4, (2022), 1227–1284.

[12] Günter M. Ziegler, *Lectures on Polytopes*, vol. 152 of *Graduate Texts in Mathematics*, Springer, Heidelberg, 1994.

Chapter 11

Counting

We consider the problems of counting (i) simplices spanned by $d+1$ out of n points in $\mathbb{R}^d$ that contain a query point; and (ii) facets of the convex hull of n points in $\mathbb{R}^d$. These two problems are closely related by yet another duality called *Gale Duality*.

Counting refers to *extremal counting* (given only the parameters, what is the maximum/minimum possible number of the considered object), and to *algorithmic counting* (given a concrete input, compute the number of the considered object). Sometimes we are also interested in *enumeration* (given a concrete input, produce all objects under consideration).

Here are a few notational conventions: $0 := (0, 0, \ldots, 0)$ is the origin in the considered ambient space $\mathbb{R}^d$. $\mathbb{N}$ is the set of positive integers, $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$, and $[n] := \{1, 2, \ldots, n\}$. $\binom{S}{k}$ denotes the set of all k-element subsets of a given set S. Finally, **Checkpoints** are usually simple quizzes to check your understanding of definitions or notions, to be answered perhaps in a minute or two if you truly absorbed the material.

It will be useful to remember

$$\sum_{i=0}^{n-1} \binom{i}{k-1} = \binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k} .$$

It does not hurt to recapitulate the combinatorial interpretation of these identities. Recall that $\binom{n}{k} = \left|\binom{[n]}{k}\right|$ is the number of k-element subsets of $[n]$. Every subset $A \in \binom{[n]}{k}$ has a unique maximum element, and we *charge* A to that element.[1] Conversely, if $j \in [n]$ is charged by $A = \binom{[n]}{k}$, then the set A has to consist of j together with a $(k-1)$-element subset of $[j-1]$. In other words, j is charged exactly $\binom{j-1}{k-1}$ times. Therefore,

$$\binom{n}{k} = \sum_{j=1}^{n} \binom{j-1}{k-1} ,$$

[1] At the end of the day, "charging" here means nothing but the mapping $A \mapsto \max(A)$. We also say that $\max(A)$ is "charged by" or "witnesses" A. These are established counting jargon. Often, an object is charged multiple times.

which proves the first identity. For the second identity, we discriminate sets $A \in \binom{[n]}{k}$ by whether n is selected or not. (Rephrasing in our counting jargon, we charge A to "true" if $n \in A$, and to "false" otherwise.) There are $\binom{n-1}{k-1}$ sets where n is selected, and $\binom{n-1}{k}$ sets where it is not. This shows

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}.$$

11.1 Introduction

Consider a set $P \subseteq \mathbb{R}^d$ and a point $q \in \mathbb{R}^d$. We call a set $A \in \binom{P}{d+1}$ a q-**embracing simplex** if $q \in \text{conv}(A)$. The **simplicial depth** of point q is the number of q-embracing simplices; that is

$$\text{sd}_q(P) := \left| \left\{ A \in \binom{P}{d+1} \ : \ q \in \text{conv}(A) \right\} \right|.$$

Note that when specialized to $\mathbb{R}^1$, a median of P is exactly a point of maximum simplicial depth. So this notion, among others, is a possible response to the search for a higher-dimensional counterpart of medians. We will investigate questions like:

- *How large can the simplicial depth of q be, in any set of n points in general position?*
- *How efficiently can we compute the simplicial depth of a point?*

For these questions, we may assume without loss of generality that $q = 0$, as we could translate all the points rigidly with q while preserving the embracing property.

A second direction we want to explore is the complexity of polytopes in general dimension d:

- *How many facets can a polytope defined by n points have? How few?*
- *Given n points, how efficiently can we compute the number of facets of its convex hull? Can we do that asymptotically faster than enumerating these facets (which is a hard enough problem per se)?*

A small reminder as has been reiterated in Chapter 10: The number of facets are linear in n for $d = 2, 3$, and can be quadratic in n for $d = 4$. For general dimension the superlinear growth continues, and we will see what the right bounds are.

These two directions about simplicial depth and number of facets of a polytope are very closely related; in a sense that we will make very explicit later (via the so-called Gale Duality), it is the same question. But for the moment, let us focus on the simplicial depth view.

11.2 Embracing Sets in the Plane

In this section we investigate simplicial depth in the plane $\mathbb{R}^2$. As we mentioned, we may assume $q = 0$. First we generalize embracing simplices (which are triangles in $\mathbb{R}^2$) to *embracing sets*, relaxing the constraint on cardinality. This is not only a natural step to take, but also integral to the argument even if we were interested in simplicial depth only.

Consider a set P of n points in $\mathbb{R}^2$, with $0 \notin P$ and $P \dot\cup \{0\}$ in general position (no three collinear points). This setting will be implicitly assumed throughout the section. For $k \in \mathbb{N}_0$, we define

$$e_k = e_k(P) := \left| \left\{ A \in \binom{P}{k} \,:\, 0 \in \operatorname{conv}(A) \right\} \right|.$$

We call a set $A \in \binom{P}{k}$ with $0 \in \operatorname{conv}(A)$ an **embracing** k**-set**. When $|A| = 3$, it is called an **embracing triangle**.

Checkpoint 11.1. $e_0 = e_1 = e_2 = 0$ *by general position.* $e_3 = \mathrm{sd}_0(P)$ *is the simplicial depth of* 0 *in* P. *And* $e_n \in \{0, 1\}$ *indicates whether* $0 \in \operatorname{conv}(P)$.

We start a general investigation of the vector $\vec{e} = (e_0, e_1, \ldots, e_n) \in \mathbb{N}_0^{n+1}$. Bounds and algorithms will follow easily, but you need to be patient, until it becomes apparent how everything fits together nicely—reward will come. As a preparation consider real vectors $\vec{x}_{0..n-3} = (x_0, x_1, \ldots, x_{n-3})$, $\vec{y}_{0..n-2}$ and $\vec{z}_{0..n-1}$ satisfying

$$e_k = \sum_{i=0}^{n-3} \binom{i}{k-3} x_i \quad \text{for all } k \geqslant 3, \tag{11.2}$$

$$e_k = \sum_{i=0}^{n-2} \binom{i}{k-2} y_i \quad \text{for all } k \geqslant 2, \tag{11.3}$$

$$e_k = \sum_{i=0}^{n-1} \binom{i}{k-1} z_i \quad \text{for all } k \geqslant 1. \tag{11.4}$$

Observe that $\vec{x}_{0..n-3}$ exists and is uniquely determined by $\vec{e}_{3..n}$, since

$$\begin{aligned}
e_n &= \tbinom{n-3}{n-3} x_{n-3} && \Longrightarrow\ x_{n-3} = e_n \\
e_{n-1} &= \tbinom{n-4}{n-4} x_{n-4} + \tbinom{n-3}{n-4} x_{n-3} && \Longrightarrow\ x_{n-4} = e_{n-1} - (n-3)\underbrace{x_{n-3}}_{e_n} \\
&\vdots && \vdots
\end{aligned}$$

Similarly, this works for $\vec{y}_{0..n-2}$ and $\vec{z}_{0..n-1}$. Thus we have

$$\vec{e}_{3..n} \xleftrightarrow[\text{each other}]{\text{determine}} \vec{x}_{0..n-3}, \qquad \vec{e}_{2..n} \xleftrightarrow[\text{each other}]{\text{determine}} \vec{y}_{0..n-2}, \qquad \vec{e}_{1..n} \xleftrightarrow[\text{each other}]{\text{determine}} \vec{z}_{0..n-1}\,.$$

Note that these facts hold for any vector $\vec{e}$, as we have not used any property of the specific $\vec{e}$ we are interested in. They simply describe some possible transformations for any given vector, although it is by no means clear how they should help...

11.2.1 Adding a Dimension

Another step that comes across unmotivated: Lift the point set P vertically to a set P' in space, *arbitrarily*, with the only requirement being general position (no four points from P' on a common plane). For example, we *may* choose the parabolic lifting map $(x, y) \mapsto (x, y, x^2 + y^2)$; but stay flexible! Let us denote the lifting by

$$P \ni \quad q = (x, y) \mapsto q' = (x, y, z(q)) \quad \in P'.$$

For an embracing triangle $\Delta = \{p, q, r\}$ in the plane, let β_Δ be the number of lifted points in P' strictly below the plane containing $\Delta' = \{p', q', r'\}$. (Just to avoid confusion: β_Δ clearly depends on the choice of the lifting.) Let

$$h_i := \text{ the number of embracing triangles } \Delta \text{ with } \beta_\Delta = i.$$

Checkpoint 11.5. $\sum_{i=0}^{n-3} h_i = e_3$.

Let us recall here that we are assuming general position for $P \dot\cup \{0\}$.

Lemma 11.6. $0 \in \mathrm{conv}(P) \iff h_0 = h_{n-3} = 1$.

Proof. ($\Leftarrow$) That's obvious, since $h_0 = 1$ says that there exists an embracing triangle, and in particular 0 is in the convex hull of P.

($\Rightarrow$) If $0 \in \mathrm{conv}(P)$ then the z-axis (i.e. the vertical line through 0 in $\mathbb{R}^3$) intersects $\mathrm{conv}(P')$. Due to general position, it intersects exactly two facets, both of which are triangles. The bottom one Δ_0' has no point in P' below its supporting plane, thus $\beta_{\Delta_0} = 0$. The top one Δ_1' has no point in P' above its supporting plane, hence all but the three points defining Δ_1' are below, that is $\beta_{\Delta_1} = n - 3$. Clearly both Δ_0 and Δ_1 are embracing, so $h_0, h_{n-3} \geqslant 1$.

On the other hand, any embracing triangle $\Delta \in \binom{P}{3}$ counted by h_0 (respectively h_{n-3}) has all other points in P' above (respectively below) Δ', hence Δ' must give rise to a facet. In addition it must be hit by the z-axis by the embracing property. But Δ_0' and Δ_1' are the only candidates, so $h_0 = h_{n-3} = 1$. □

Consider an embracing k-set $A \in \binom{P}{k}$ and its lifting A'. As observed before, the z-axis intersects the boundary of $\mathrm{conv}(A')$ in two facets. Consider the top facet—it is given by the lifting of some embracing triangle $\Delta \in \binom{P}{3}$. We say that this Δ *witnesses* (the embracing property of) A. How many embracing k-sets does Δ witness?

For Δ to witness an embracing k-set B, we must have $\Delta \subseteq B$ and the remaining $k-3$ points in $B \setminus \Delta$ are chosen so that $B' \setminus \Delta'$ is below the plane spanned by Δ'. Hence Δ witnesses exactly $\binom{\beta_\Delta}{k-3}$ embracing k-sets. It follows that

$$e_k = \sum_{\Delta \in \binom{P}{3} \text{ embracing}} \binom{\beta_\Delta}{k-3} = \sum_{i=0}^{n-3} \binom{i}{k-3} h_i. \tag{11.7}$$

Note that this is exactly the relation (11.2) (with h_i replacing x_i). We thus have

$$\vec{e}_{3..n} \xleftrightarrow[\text{each other}]{\text{determine}} \vec{h}_{0..n-3} := (h_0, h_1, \ldots, h_{n-3})$$

and therefore the vector $\vec{h}_{0..n-3}$ is independent of the lifting we chose, i.e. $h_i = h_i(P)$.

A few properties emerge. First, note that $\vec{h}$ (consisting of nonnegative integers no larger than $\binom{n}{3} = O(n^3)$, or $O(\log n)$ bits) is a compact way of representing $\vec{e}$ (consisting of integers potentially exponential in n, or $\Omega(n)$ bits). Also, since it is easy to compute the vector $\vec{h}$ in $O(n^4)$ time[2], we can compute each entry of e_k in $O(n^4)$ time.

Second, the independence of the vector $\vec{h}$ from the chosen lifting allows quite simple proofs of properties of $\vec{h}$: You can choose the lifting! If you can make a property of $\vec{h}$ hold for a chosen lifting, then it will be true for all liftings. Keep this in mind when solving the following exercise.

Exercise 11.8. *Show that*

$$h_0 = 1 \iff 0 \in \operatorname{conv}(P) \iff h_i \geqslant 1 \text{ for } 0 \leqslant i \leqslant n-3.$$

Now, don't hesitate to use the assertion of this exercise and relation (11.7) for the following exercise.

Exercise 11.9. *Assume $0 \in \operatorname{conv}(P)$. What is the minimal possible value of e_3 in terms of $n := |P|$? (Note that this is a quantified version of Carathéodory's Theorem for $\mathbb{R}^3$.) Generally, what is the minimal possible value of e_k, $3 \leqslant k \leqslant n$?*

Exercise 11.10. *Let $P \subset \mathbb{R}^2$ be a set of n points in general position (with the origin). What does $\sum_{i=0}^{n-3} 2^i h_i$ count?*

Exercise 11.11. *Let $P \subset \mathbb{R}^2$ be a set of n points in general position (with the origin) and assume $0 \in \operatorname{conv}(P)$. Recall that e_k denotes the number of embracing k-sets. Show that $\sum_{k=3}^{n} (-1)^k e_k = -1$. (Hint: Plug in the relation $e_k = \sum_{i=0}^{n-3} \binom{i}{k-3} h_i$ in this sum and simplify.)*

In a next step we show that the vector $\vec{h}$ is symmetric.

Lemma 11.12. $h_i = h_{n-3-i}$.

Proof. Define $\widehat{h}_i$ in the same way as h_i, except that you count the points *above* (instead of below) the plane through the lifting of an embracing triangle. Note that $\widehat{h}_i = h_{n-3-i}$ by definition. On the other hand, with the same witness argument as before we derive

$$e_k = \sum_{i=0}^{n-3} \binom{i}{k-3} \widehat{h}_i,$$

and therefore $h_i = \widehat{h}_i = h_{n-3-i}$. □

[2] With some tricks from computational geometry, in $O(n^3)$ time.

Hence the vector $\vec{h}_{0..n-3}$ is determined by its first half $h_0, h_1, \ldots, h_{\lfloor (n-3)/2 \rfloor}$.

Exercise 11.13. *Let $P \subset \mathbb{R}^2$ be a set of n points in general position (with the origin). Show that e_3 (the number of embracing triangles) and e_4 (the number of embracing 4-sets) are related by $(n-3)e_3 = 2e_4$.*

Exercise 11.14. *Show that if $|P| = 6$, then e_3 determines $e_{3..6}$. How?*

Exercise 11.15. *Show that if $|P|$ is even then e_3 is even.*

11.2.2 The Upper Bound

We have seen in one of the exercises how the relation between $\vec{e}$ and $\vec{h}$ can be useful in proving lower bounds on the e_k's. We need two lemmas towards a proof of upper bounds. The first lemma states that removing a point from P cannot increase h_j.

Lemma 11.16. *For all $j \in \mathbb{N}_0$ and all $q \in P$, we have $h_j(P \setminus \{q\}) \leqslant h_j(P)$.*

Proof. What changes happen to h_j as we remove a point q from P?

- We *lose* those embracing triangles Δ with $\beta_\Delta = j$ (before removal) such that q' is in or below Δ'.
- We *keep* those embracing triangles Δ with $\beta_\Delta = j$ such that q' is above Δ'.
- We *gain* those embracing triangles Δ with $\beta_\Delta = j+1$ such that q' is below Δ'.

Now lift q' vertically above all planes defined by three points in $P' \setminus \{q'\}$. It does not change the values h_i as $\vec{h}$ is independent of the lifting, but eliminates the "gain" case. This gives the lemma. □

Lemma 11.17. *For all $j \in \mathbb{N}_0$ we have*

$$\sum_{q \in P} h_j(P \setminus \{q\}) \;=\; (n-j-3)\, h_j(P) \;+\; (j+1)\, h_{j+1}(P).$$

Proof. Fix an arbitrary lifting. A contribution to $\sum_{q \in P} h_j(P \setminus \{q\})$ can come only from triangles Δ with $\beta_\Delta = j$ or $\beta_\Delta = j+1$ (relative to the complete point set P).

- If $\beta_\Delta = j$, then Δ' remains a triangle with j points below after removing q iff q is one of the $(n-3-j)$ points above.
- If $\beta_\Delta = j+1$, then Δ' turns into a triangle with j points below after removing q iff q is one of the $(j+1)$ points below.

Hence the lemma. □

Now we apply the previous Lemma 11.16 to bound

$$\sum_{q\in P} h_j(P\setminus\{q\}) \leqslant n\cdot h_j(P) \;=: n\cdot h_j,$$

and with Lemma 11.17 we can derive

$$\begin{aligned}(n-j-3)h_j+(j+1)h_{j+1} &\leqslant n\cdot h_j\\ (j+1)h_{j+1} &\leqslant (j+3)h_j\\ h_{j+1} &\leqslant \frac{j+3}{j+1}h_j\ .\end{aligned}$$

This bound can be iterated until we reach h_0:

$$h_{j+1}\leqslant \frac{j+3}{j+1}h_j\leqslant \frac{j+3}{j+1}\frac{j+2}{j}h_{j-1}\leqslant\cdots\leqslant \underbrace{\frac{j+3}{j+1}\frac{j+2}{j}\cdots\frac{3}{1}}_{=\binom{j+3}{2}}\underbrace{h_0}_{\leqslant 1}\leqslant\binom{j+3}{2}\ .$$

Theorem 11.18. *Let P be a set of n points in general position. Then for $0\leqslant j\leqslant n-3$ we have $h_j=h_{n-3-j}$ and $h_j\leqslant\binom{j+2}{2}$. Consequently $h_j\leqslant\min\left\{\binom{j+2}{2},\binom{n-1-j}{2}\right\}$. Moreover,*

$$e_3\leqslant\begin{cases}2\binom{n/2+1}{3}=\frac{n(n^2-4)}{24} & \text{for } n \text{ even,}\\ 2\binom{(n+1)/2}{3}+\binom{(n+1)/2}{2}=\frac{n(n^2-1)}{24} & \text{for } n \text{ odd.}\end{cases}$$

Proof. The first part is just a summary of what we have derived so far. For the "moreover" part, we simply plug them into relation (11.7). Suppose first that n is even. Then

$$(h_0,h_1,\ldots,h_{n/2-2})=(h_{n-3},h_{n-4},\ldots,h_{n/2-1})$$

and, therefore,

$$e_3=\sum_{i=0}^{n-3}h_i=2\sum_{i=0}^{n/2-2}h_i\leqslant 2\sum_{i=0}^{n/2-2}\binom{i+2}{2}=2\binom{n/2+1}{3}\ .$$

Second, if n is odd then

$$(h_0,h_1,\ldots,h_{(n-3)/2})=(h_{n-3},h_{n-2},\ldots,h_{(n-3)/2})$$

with $h_{(n-3)/2}$ appearing on both sides. So

$$\begin{aligned}e_3=\sum_{i=0}^{n-3}h_i &= 2\sum_{i=0}^{(n-3)/2-1}h_i \;+h_{(n-3)/2}\\ &\leqslant 2\sum_{i=0}^{(n-3)/2-1}\binom{i+2}{2}\;+\binom{(n+1)/2}{2}\\ &= 2\binom{(n+1)/2}{3}+\binom{(n+1)/2}{2}\ .\end{aligned}$$

□

There are sets where all these bounds are tight, simultaneously. We find it more convenient to substantiate this claim after further considerations.

Exercise 11.19. *Show* $e_3 \leqslant \frac{1}{4}\binom{n}{3} + O(n^2)$. *(That is, asymptotically, at most* $1/4$ *of all triangles embrace the origin.)*

Exercise 11.20. *Try to understand the independence of* $\vec{h}$ *of the actual lifting by observing what happens as you move a single point vertically.*

We have obtained lower and upper bounds in the plane. Before proceeding to better methods for computing the e_k's, we generalize to arbitrary dimension d.

11.3 Embracing Sets in Higher Dimension

It has been announced that our methods easily carry over to higher dimensions. So let us do a quick tour of deriving the bounds analogous to Theorem 11.18. The reader should make sure that indeed all arguments can be generalized. It is a good exercise to recapitulate the proofs.

Let us now assume that $P \subset \mathbb{R}^d$ is a set of n points in general position with the origin, that is, $0 \notin P$ and no $d+1$ points in $P \cup \{0\}$ lie on a common hyperplane. There is no change in the notion of an embracing k-set and of the vector $\vec{e}$, but let us still repeat:

For $k \in \mathbb{N}_0$, we define $e_k = e_k(P) := \left|\left\{A \in \binom{P}{k} : 0 \in \operatorname{conv}(A)\right\}\right|$. We call a set $A \in \binom{P}{k}$ with $0 \in \operatorname{conv}(A)$ an **embracing k-set**. When $|A| = d+1$, it is called an **embracing simplex**. We will still use symbol Δ for embracing simplices. Observe that $e_0 = e_1 = \cdots = e_d = 0$ by general position, and that $e_{d+1} = \mathrm{sd}_0(P)$.

We consider a generic vertical lifting from P to $\mathbb{R}^{d+1}$, denoted by $p \mapsto p'$. "Vertical" means we lift along the new dimension; "generic" means that no $d+2$ points in P' lie in a common hyperplane.

If $\Delta \in \binom{P}{d+1}$ is an embracing simplex, then its lifting Δ' affinely spans a hyperplane. We use β_Δ for the number of P' strictly below this hyperplane. We emphasize that β_Δ depends on the lifting chosen.

As before, we define the vector $\vec{h}$ with

$$h_i := \text{ the number of embracing simplices } \Delta \text{ with } \beta_\Delta = i.$$

with the only difference that we now consider embracing simplices rather than triangles.

Checkpoint 11.21. $\sum_{i=0}^{n-(d+1)} h_i = e_{d+1}$.

In the plane, our next lemma was $0 \in P \iff h_0 = h_{n-3} = 1$, where h_{n-3} counted all embracing triangles Δ with all other points below (in the lifting). This number is now $n-(d+1)$, so we get

Lemma 11.22. $0 \in P \iff h_0 = h_{n-(d+1)} = 1$

We can take over the proof we have seen for Lemma 11.6. Essential ingredients were that the x_{d+1}-axis intersects the convex polytope $\mathrm{conv}(P')$ in a non-empty interval. Each of its endpoints is on a facet of the polytope. The supporting hyperplane of one facet has no point in P' below, and the supporting hyperplane of the other facet has no point above (here we use the fact that the intersecting line, the x_{d+1}-axis, is vertical). Hence, these facets are liftings of embracing simplices Δ_0 and Δ_1, respectively, with $\beta_{\Delta_0} = 0$ and $\beta_{\Delta_1} = n - (d+1)$. Via the notion of a witness embracing simplex $\Delta \subseteq A$ of an embracing k-set A, the counterpart of (11.7) reads now

$$e_k = \sum_{\Delta \in \binom{P}{d+1} \text{ embracing}} \binom{\beta_\Delta}{k-(d+1)} = \sum_{i=0}^{n-(d+1)} \binom{i}{k-(d+1)} h_i \ , \tag{11.23}$$

and thus

$$\vec{e}_{d+1..n} \xleftrightarrow[\text{each other}]{\text{determine}} \vec{h}_{0..n-(d+1)}.$$

Hence, h_i is independent of the lifting chosen and we can write $h_i = h_i(P)$. Symmetry of $\vec{h}$ follows readily, as before, by looking at points above instead of below a lifted embracing simplex.

Lemma 11.24. $h_i = h_{n-(d+1)-i}$.

The two lemmas towards the upper bound carry over, with the first identical to what we have seen before, and the second with the constants adapted to the dimension.

Lemma 11.25. *For all* $j \in \mathbb{N}_0$ *and all* $q \in P$*, we have* $h_j(P \setminus \{q\}) \leqslant h_j(P)$.

Lemma 11.26. *For all* $j \in \mathbb{N}_0$ *we have*

$$\sum_{q \in P} h_j(P \setminus \{q\}) \ = \ \big(n - j - (d+1)\big)\, h_j(P) \ + \ (j+1)\, h_{j+1}(P).$$

Proof. Fix an arbitrary generic lifting. A contribution to $\sum_{q\in P} h_j(P\setminus\{q\})$ can come only from simplices Δ with $\beta_\Delta = j$ or $\beta_\Delta = j+1$ (relative to the complete set P).

- If $\beta_\Delta = j$, then Δ' remains a simplex with j points below after removing q, iff q is one of the $(n-(d+1)-j)$ points above.
- If $\beta_\Delta = j+1$, then Δ' turns into a simplex with j points below after removing q, iff q is one of the $(j+1)$ points below. □

Again, for the upper bound on the h_i's, just like in the plane, we start with

$$\sum_{q \in P} h_j(P \setminus \{q\}) \leqslant n \cdot h_j(P)$$

by Lemma 11.25, and then continue

$$\begin{aligned}(n-j-(d+1))h_j + (j+1)h_{j+1} &\leqslant n \cdot h_j \\ (j+1)h_{j+1} &\leqslant (j+d+1)h_j \\ h_{j+1} &\leqslant \frac{j+d+1}{j+1}\, h_j\end{aligned}$$

by Lemma 11.26. Then we iterate this until we reach h_0:

$$h_{j+1} \leqslant \frac{j+d+1}{j+1}\, h_j \leqslant \cdots \leqslant \underbrace{\frac{j+d+1}{j+1}\,\frac{j+d}{j}\,\cdots\,\frac{d+1}{1}}_{=\binom{j+d+1}{d}}\,\underbrace{h_0}_{\leqslant 1} \leqslant \binom{j+d+1}{d}.$$

Theorem 11.27. *Let $P \subset \mathbb{R}^d$ be a set of n points in general position. Then for $0 \leqslant j \leqslant n-(d+1)$ we have $h_j = h_{n-(d+1)-j}$ and $h_j \leqslant \binom{j+d}{d}$. Consequently $h_j \leqslant \min\left\{\binom{j+d}{d}, \binom{n-1-j}{d}\right\}$. Moreover,*

$$e_{d+1} \leqslant \begin{cases} 2\binom{(n+d)/2}{d+1} & \text{for } n-d \text{ even,} \\ 2\binom{(n+d-1)/2}{d+1} + \binom{(n+d-1)/2}{d} & \text{for } n-d \text{ odd.} \end{cases}$$

Proof. The first part is just a summary of what we have derived so far. For the "moreover" part, we simply plug them into relation (11.23). For $n-d$ even we have

$$(h_0, h_1, \ldots, h_{(n-d)/2-1}) = (h_{n-d-1}, h_{n-d-2}, \ldots, h_{(n-d)/2})$$

and, therefore,

$$e_{d+1} = \sum_{i=0}^{n-(d+1)} h_i = 2\sum_{i=0}^{(n-d)/2-1} h_i \leqslant 2\sum_{i=0}^{(n-d)/2-1} \binom{i+d}{d} = 2\binom{(n+d)/2}{d+1}.$$

If $n-d$ is odd then

$$(h_0, h_1, \ldots, h_{(n-(d+1))/2}) = (h_{n-3}, h_{n-2}, \ldots, h_{(n-(d+1))/2})$$

with $h_{(n-(d+1))/2}$ appearing on both sides. So

$$\begin{aligned} e_{d+1} = \sum_{i=0}^{n-(d+1)} h_i &= 2\sum_{i=0}^{(n-(d+1))/2-1} h_i \quad + h_{(n-(d+1))/2} \\ &\leqslant 2\sum_{i=0}^{(n-(d+1))/2-1} \binom{i+d}{d} \quad + \binom{(n+d-1)/2}{2} \\ &= 2\binom{(n+d-1)/2}{d+1} + \binom{(n+d-1)/2}{d}. \end{aligned}$$

□

11.4 Embracing Sets vs. Faces of Polytopes

This section exhibits a duality between points sets of size n. It is very different from polarity and projective duality that you have learnt in previous chapters. Roughly speaking, if $P \subset \mathbb{R}^d$ is dual to $Q \subset \mathbb{R}^{n-d-1}$, then the faces of $\mathrm{conv}(P)$ one-one correspond to the embracing sets of Q.

In order to describe this duality, and then to handle it, we need some handy linear algebra terminology as well as the algebraic rephrasing of our target notions "embracing" and "supporting hyperplane" (for polytope faces). We will approach this smoothly, and I apologize to those who have these matters on top of their head anyway.[3]

11.4.1 Warm-up

Point sequences and matrices. For integers $d, n \in \mathbb{N}_0$, consider a matrix $A \in \mathbb{R}^{n\times d}$. The sequence $S_A = (p_1, p_2, \ldots, p_n)$ of row vectors of A can be interpreted as a sequence of points in $\mathbb{R}^d$ (or strictly speaking $\mathbb{R}^{1\times d}$, if we want to emphasize that they are row vectors). Vice versa, every sequence of n points in $\mathbb{R}^d$ can be thought of as a matrix $A \in \mathbb{R}^{n\times d}$. Let us say right away that we abandon the general position assumption, at least for the time being. In particular, we allow repetitions in a sequence of points.

We write $\vec{1}$ and $\vec{0}$ for the vector of all 1's and all 0's, respectively. Their dimensions and their being a row or a column will be clear from the context. Hence $\vec{0}$ also represents the origin in the ambient space. Given a vector $u = (u_1, \ldots, u_m) \in \mathbb{R}^m$ (row or column), we write $u \geqslant \vec{0}$ if $u_i \geqslant 0$ for all $i = 1, \ldots, m$.

Linear and convex combinations. A linear combination $\lambda_1 p_1 + \cdots + \lambda_n p_n$ of the rows of $A \in \mathbb{R}^{n\times d}$ with coefficients $\lambda = (\lambda_1, \ldots, \lambda_n) \in \mathbb{R}^{1\times n}$ can be compactly written as matrix multiplication $\lambda \cdot A \in \mathbb{R}^{1\times d}$. Here are a few simple observations.

(1) $\frac{1}{n}(\vec{1} \cdot A)$ is the centroid[4] of S_A.

(2) $\vec{1} \cdot A = \vec{0}$ iff $\vec{0}$ is the centroid of S_A. Another way of interpreting $\vec{1} \cdot A = \vec{0}$ is that $\vec{1}$ is orthogonal to all column vectors of A.

(3) $\lambda \cdot A$, with $\lambda \geqslant \vec{0}$ and $\vec{1} \cdot \lambda = 1$, is a convex combination of S_A.

(4) If $\lambda \cdot A = \vec{0}$ with $\vec{0} \neq \lambda \geqslant \vec{0}$, then $\vec{0} \in \mathrm{conv}(S_A)$. The reason is that we can scale such λ to convex coefficients $\lambda' := \frac{1}{\vec{1}\cdot\lambda}\lambda$ which also satisfies $\lambda' \cdot A = \vec{0}$.

Just like the left product $\lambda \cdot A$ denotes a linear combination of the rows of A, the right product $A \cdot \mu$, for $\mu \in \mathbb{R}^d$, denotes a linear combination of the columns of A.

[3] Think of it as a warm-up of your linear-algebra-muscles.

[4] Center of gravity, or the average of the points.

Hyperplanes. An oriented hyperplane in $\mathbb{R}^d$ is represented by a column vector $\nu \in \mathbb{R}^{d+1}$:

$$H_\nu := \left\{ x \in \mathbb{R}^d \;:\; (x,-1)\cdot\nu = \sum_{i=1}^{d} \nu_i x_i - \nu_{d+1} = 0 \right\} \text{ for } \nu = \begin{pmatrix} \nu_1 \\ \vdots \\ \nu_{d+1} \end{pmatrix} \neq \vec{0}.$$

Here $(x,-1)$ is the row vector x extended by an extra dimension with coordinate -1. Denote by H_ν^+ the closed positive halfspace bounded by H_ν. Recall that H_ν is a supporting hyperplane of some face of $\mathcal{P}$ if $\mathcal{P} \subseteq H_\nu^+$.

(1) The vector $\sigma := (A,-\vec{1})\cdot\nu \in \mathbb{R}^n$ indicates the relations between the points p_i and the hyperplane H_ν:

$$\begin{aligned} \sigma_i = 0 &\iff p_i \in H_\nu, \\ \sigma_i \geqslant 0 &\iff p_i \in H_\nu^+. \end{aligned}$$

Here $(A,-\vec{1})$ denotes the matrix in $\mathbb{R}^{n\times(d+1)}$ obtained from A by extending it by an extra column $-\vec{1}$.

(2) $(A,-\vec{1})\cdot\nu = \vec{0}$ iff H_ν contains all points from S_A.

(3) $(A,-\vec{1})\cdot\nu \geqslant \vec{0}$ iff H_ν is a supporting hyperplane of some face of $\operatorname{conv}(S_A)$.

We recall that matrix $A \in \mathbb{R}^{n\times d}$ has full rank d iff its columns are independent; that is, there is no $\mu \neq \vec{0}$ with $A\cdot\mu = \vec{0}$. Rephrased geometrically, $\operatorname{rank}(A) = d$ iff there is no hyperplane $H_{(\mu,0)}$ through the origin that contains all points from S_A.

11.4.2 Gale Duality

Assume $0 \leqslant d < n$. We are now ready to describe a duality between sequences of n points in $\mathbb{R}^d$ and $\mathbb{R}^{n-d-1}$.

We call a matrix $A \in \mathbb{R}^{n\times d}$ **legal** if $\vec{1}\cdot A = \vec{0}$ and $\operatorname{rank}(A) = d$. What is the geometric interpretation of legality? The first condition says that the origin is the centroid of S_A. In particular, $\operatorname{conv}(S_A)$ contains the origin. Hence $\operatorname{conv}(S_A)$ is a full dimensional polytope by the second condition: otherwise $\operatorname{conv}(S_A)$ is entirely contained in some hyperplane (which has to go through the origin), contradicting $\operatorname{rank}(A) = d$.

Vice versa, if S_A has centroid $\vec{0}$ and $\operatorname{conv}(S_A)$ is full-dimensional, then A is legal. Hence legality is a much weaker assumption than general position!

Given legal matrices $A \in \mathbb{R}^{n\times d}$ and $B \in \mathbb{R}^{n\times(n-d-1)}$, we call B an **orthogonal dual** of A, in symbols $A \perp B$, if $A^\top B = 0^{d\times(n-d-1)}$. In other words, all columns of A are orthogonal to all columns of B; as a result, the columns of A span a linear space of dimension d orthogonal to the linear space of dimension $n-d-1$ spanned by the columns of B, and both spaces are orthogonal to $\vec{1}$ (by the legality condition). Hence for any legal matrix A, we may always find an orthogonal dual B, and it is unique up to linear transformations.

Clearly, $A \perp B \iff B \perp A$.[5] See Figures 11.1 and 11.2 for examples of orthogonal duals and their point sequences.

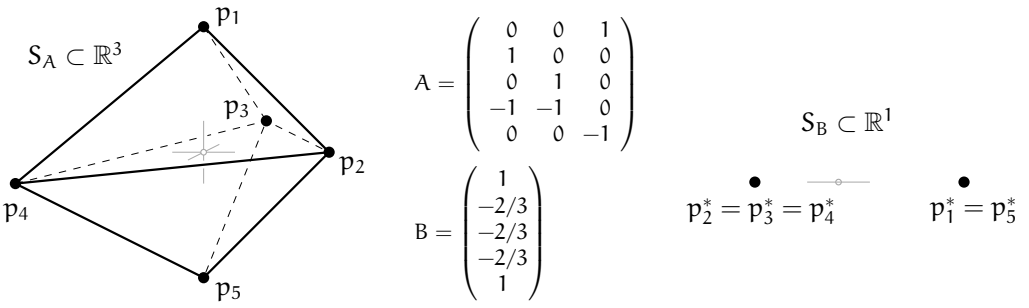


Figure 11.1: *Point sequences* S_A *and* S_B *from orthogonal duals* $\mathbb{R}^{5\times 3} \ni A \perp B \in \mathbb{R}^{5\times 1}$.

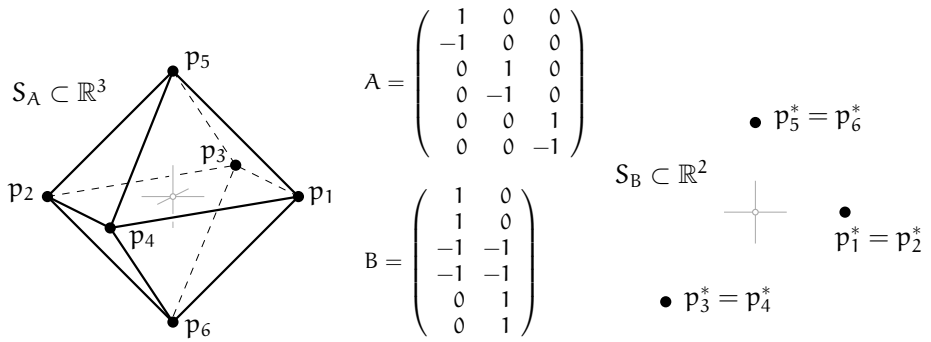


Figure 11.2: *Point sequences* S_A *and* S_B *from orthogonal duals* $\mathbb{R}^{6\times 3} \ni A \perp B \in \mathbb{R}^{6\times 2}$.

Lemma 11.28 (Gale Duality). *Let* $\mathbb{R}^{n\times d} \ni A \perp B \in \mathbb{R}^{n\times(n-d-1)}$ *be legal matrices that are orthogonal duals to each other, and denote* $S_A = (p_1, \ldots, p_n)$ *and* $S_B = (p_1^*, \ldots, p_n^*)$. *For any given* $I \subseteq [n]$, *consider* $F := \{p_i : i \in I\}$ *and* $\overline{F^*} := \{p_i^* : i \notin I\}$. *Then* F *is contained in a supporting hyperplane of* $\operatorname{conv}(S_A)$ *if and only if* $\vec{0} \in \operatorname{conv}(\overline{F^*})$.

Proof. Let F lie in a supporting hyperplane. That is, there is a vector $\nu \in \mathbb{R}^{d+1}$, $\nu_{1..d} \neq \vec{0}$, such that $\sigma := (A, -\vec{1}) \cdot \nu \geqslant \vec{0}$ and $\sigma_i = 0$ for all $i \in I$. Note also $\sigma \neq \vec{0}$ since A has full rank. Moreover,

$$\sigma^\top \cdot B = \nu^\top \cdot \underbrace{(A, -\vec{1})^\top \cdot B}_{0^{(d+1)\times(n-d-1)}} = \vec{0}$$

which implies that $\vec{0}$ can be written as a convex combination of points in S_B. Since only those points in $\overline{F^*}$ contribute positively, we conclude that $\vec{0} \in \operatorname{conv}(\overline{F^*})$.

[5] This convenient symmetry, enforced by the condition $\vec{1} \cdot A^\top = \vec{0}$, is the only difference to the standard Gale transform—apart from expository details.

For the reverse direction, let vector $\lambda \in \mathbb{R}^n$ certify the fact that $\vec{0} \in \text{conv}(\overline{F^*})$. That is, $\lambda^\top \cdot B = \vec{0}$ with $\vec{0} \neq \lambda \geqslant \vec{0}$ and $\lambda_i = 0$ for $i \in I$. Since λ is orthogonal to the columns of B (that's what $\lambda^\top \cdot B = \vec{0}$ says), it is in the linear space spanned by the columns of $(A, -\vec{1})$, and so there is a vector $\nu \in \mathbb{R}^{d+1}$ with $(A, -\vec{1}) \cdot \nu = \lambda$. Hence, ν represents a supporting hyperplane of $\text{conv}(S_A)$ that passes through $\{p_i : \lambda_i = 0\} \supseteq F$. □

Faces in simplicial polytopes. At this point recall (and you probably already did) the discussion about simplicial polytopes. We have seen that they maximize the number of facets for a given number n of vertices. In such a polytope, every k-face where $0 \leqslant k \leqslant d-1$ is a k-simplex and hence has exactly $k+1$ vertices. The convex hull of any subset of these vertices produces a face of the polytope. Therefore, F is the vertex set of an i-face iff $|F| = i+1$ and F is contained in some supporting hyperplane.[6]

Given a d-dimensional polytope $\mathcal{P}$, define the f-**vector** of $\mathcal{P}$ by

$$\vec{f} = \vec{f}(\mathcal{P}) = (f_{-1}, f_0, f_1, \ldots, f_{d-1}) \in \mathbb{N}_0^{d+1}$$

where f_i is the number of i-faces. (Recall that there is the empty face, which we agree to be -1-dimensional; but we ignore the d-dimensional face, the whole polytope itself). Hence $f_{-1} = 1$, f_0 is the number of vertices of $\mathcal{P}$, and f_{d-1} is the number of facets of $\mathcal{P}$.

Observation 11.29. *If $\mathcal{P}$ is a d-dimensional simplicial polytope with vertex set $V(\mathcal{P})$, then f_i counts the number of $(i+1)$-element subsets of $V(\mathcal{P})$ that are contained in a supporting hyperplane of $\mathcal{P}$.*

We are ready to employ Gale Duality.

Lemma 11.30. *Let $A \in \mathbb{R}^{n \times d}$ be a legal matrix, whose rows S_A encode n points in $\mathbb{R}^d$ in general position, so that $\text{conv}(S_A)$ is a d-dimensional simplicial polytope with f-vector $(f_{-1}, f_0, \ldots, f_{d-1})$. Suppose that the legal matrix $B \in \mathbb{R}^{n \times (n-d-1)}$ is an orthogonal dual of A, whose rows S_B encode n points in $\mathbb{R}^{n-d-1}$ in general position with the origin. Then*

$$f_{i-1} = e_{n-i}(S_B)$$

for all $0 \leqslant i \leqslant d$. In particular, $f_{-1} = e_n = 1$, the number of vertices is $f_0 = e_{n-1}$, and the number of facets is $f_{d-1} = e_{n-d} = \text{sd}_0(S_B)$.

Proof. Denote $S_A = (p_1, \ldots, p_n)$ and $S_B = (p_1^*, \ldots, p_n^*)$. Since $\text{conv}(S_A)$ is simplicial, f_{i-1} counts the number of $I \in \binom{[n]}{i}$ such that the points $\{p_i : i \in I\}$ are contained in a supporting hyperplane (Observation 11.29). By Gale duality Lemma 11.28, this happens iff $\vec{0} \in \text{conv}\{p_i^* : i \notin I\}$, namely $\{p_i^* : i \notin I\}$ is an embracing $(n-i)$-set of S_B. □

[6] For all of this it is important that the polytope is simplicial. Think of a 3-dimensional cube: The facets, which are 2-faces, have vertex sets of size 4, and 3-element subsets of these 4-element sets are *not* vertex sets of any face.

In order to carry the upper bounds for $\vec{e}$ over $\vec{f}$, it remains to ensure that for every simplicial polytope, the conditions of the lemma can be achieved effectively. To this end, we start with a simplicial polytope, translate its vertices rigidly so that $\vec{0}$ becomes the centroid, and then perturb the vertices into general position with $\vec{0}$, without changing the face lattice and, in particular, the f-vector. That this is possible needs a not too difficult careful argument, which we sweep under the rug here. In fact, under these assumptions, an orthogonal dual of the vertices is also a set in general position with the origin (see Exercise 11.34).

Finally, it comes the bound on the number of faces, first shown by McMullen in 1970.

Theorem 11.31 (Upper Bound Theorem). *Let $\mathcal{P}$ be a simplicial d-dimensional polytope with n vertices and f-vector $(f_{-1}, f_0, \ldots, f_{d-1})$. Then there is a vector $\vec{h} = (h_0, h_1, \ldots, h_d) \in \mathbb{N}_0^{d+1}$ such that*

$$f_{i-1} = \sum_{j=0}^{d} \binom{d-j}{i-j} h_j \quad \textit{with} \quad h_j = h_{d-j} \leqslant \binom{j+n-d-1}{j} \textit{ for all } j.$$

In particular,

$$f_{d-1} \leqslant \left.\begin{cases} 2\binom{n-(d+1)/2}{(d-1)/2} & \textit{for } d \textit{ odd} \\ 2\binom{n-d/2-1}{d/2-1} + \binom{n-d/2-1}{d/2} & \textit{for } d \textit{ even} \end{cases}\right\} = O\left(n^{\lfloor d/2 \rfloor}\right)$$

The proof of the theorem is just a transformation of Theorem 11.27 via Lemma 11.30. Let us first check the bounds for f_{d-1} for the values we are familiar with: For $d = 2$ we get $f_1 \leqslant 2\binom{n-2}{0} + \binom{n-2}{1} = 2 + n - 2 = n$, and for $d = 3$ we get $f_2 \leqslant 2\binom{n-2}{1} = 2n - 4$, which are both the values to be expected. For $d = 4$, we see that the upper bound $f_3 \leqslant 2\binom{n-3}{1} + \binom{n-3}{2} = \frac{n(n-3)}{2}$ grows quadratically, which confirms that the lower bound in Section 10.7 is asymptotically tight.

Proof. With the translation and perturbation mentioned earlier, we may assume that $V(\mathcal{P})$ has centroid $\vec{0}$ and is in general position with $\vec{0}$. Let $A \perp B$ with S_A an ordering of $V(\mathcal{P})$. Denote $d^* := n - d - 1$. Applying the theory of embracing sets on S_B in the dual space $\mathbb{R}^{d^*}$, we know there is a vector $\vec{h} = (h_0, h_1, \ldots, h_{n-(d^*+1)})$ such that

$$e_k(S_B) = \sum_{j=0}^{n-(d^*+1)} \binom{j}{k-(d^*+1)} h_j \quad \text{and} \quad h_j = h_{n-(d^*+1)-j} = h_{d-j}.$$

and $h_j \leqslant \binom{j+d^*}{d^*} = \binom{j+n-d-1}{n-d-1} = \binom{j+n-d-1}{j}$. Hence via Lemma 11.30,

$$\begin{aligned} f_{i-1} = e_{n-i} &= \sum_{j=0}^{n-(d^*+1)} \binom{j}{n-i-(d^*+1)} h_j \\ &= \sum_{j=0}^{d} \binom{j}{d-i} h_j = \sum_{j=0}^{d} \binom{j}{d-i} h_{d-j} = \sum_{j=0}^{d} \binom{d-j}{d-i} h_j = \sum_{j=0}^{d} \binom{d-j}{i-j} h_j. \end{aligned}$$

In particular, with the bound in Theorem 11.27, we obtain

$$f_{d-1} = e_{n-d} = e_{d^*+1} \leqslant \left\{ \begin{matrix} 2\binom{(n+d^*)/2}{d^*+1} & \text{for } n-d^* \text{ even} \\ 2\binom{(n+d^*-1)/2}{d^*+1} + \binom{(n+d^*-1)/2}{d^*} & \text{for } n-d^* \text{ odd} \end{matrix} \right\}$$
$$= \left\{ \begin{matrix} 2\binom{n-(d+1)/2}{(d-1)/2} & \text{for } d \text{ odd} \\ 2\binom{n-d/2-1}{d/2-1} + \binom{n-d/2-1}{d/2} & \text{for } d \text{ even} \end{matrix} \right\} = O\left(n^{\lfloor d/2 \rfloor}\right) \qquad \square$$

Tightness. The bounds in the Upper Bound Theorem are tight, for the whole f-vector. A family of polytopes that attain this bound are quite easy to describe: the so-called *cyclic polytope* is the convex hull of $\{(t, t^2, \ldots, t^d) \in \mathbb{R}^d : t = 1, 2, \ldots, n\}$. These polytopes have the property that for all $i \leqslant \lfloor \frac{d}{2} \rfloor$, every i-element subset of the vertices form an $(i-1)$-face. For example, when $d = 4$, all pairs of its vertices are connected by an edge. But we skip the proof that such polytopes have the prescribed number of faces in various dimensions.

The beauty of the theorem goes much beyond supplying an upper bound. Many facts known about polytopes follow now quite naturally.

Dehn-Sommerville relations. The symmetry $h_j = h_{d-j}$ for $0 \leqslant j \leqslant d$ is also called *Dehn-Sommerville relations*. Originally they are formulated in terms of the f-vector, but Sommerville later restated them in the current compact form in terms of the h-vector.

To recover the original form, we recall from the Upper Bound Theorem that $f_{i-1} = \sum_{j=0}^{d} \binom{d-j}{i-j} h_j$. It is not hard to derive an inversion formula $h_j = \sum_{i=0}^{j} (-1)^{j-i} \binom{d-i}{d-j} f_{i-1}$ just like what we did in Exercise 11.11. The original Dehn-Sommerville relations simply replace both sides of $h_j = h_{d-j}$ by the f-expressions.

Let us discuss an important consequences. Specializing with $j := d$ we get

$$1 = h_d = \sum_{i=0}^{d} (-1)^{d-i} f_{i-1}$$
$$= f_{d-1} - f_{d-2} + f_{d-3} - \cdots + (-1)^{d-1} f_0 + (-1)^d,$$

which is exactly the Euler-Poincaré Formula that we saw in Chapter 10.

More formulas of the type are $2f_{d-2} = df_{d-1}$, which can be easily obtained directly by double-counting.

The usual proof. The "usual proof" of the Upper Bound Theorem does not take the detour to the Gale dual. Instead, the h-vector is defined directly for a simplicial polytope $\mathcal{P} \subset \mathbb{R}^d$. The ingredients of the proof are similar, actually the same as we saw translated to the Gale Dual. Apart from the original paper by McMullen, see for example the book by Ziegler [1] for this version of the proof.

Exercise 11.32. *Let $A \in \mathbb{R}^{n\times d}$ and $B \in \mathbb{R}^{n\times(n-d-1)}$ be legal matrices with $A \perp B$, such that S_A and S_B are in general position with the origin (in particular, all points are distinct).*

(i) Suppose that all elements in S_A are extreme, i.e. vertices of $\mathrm{conv}(S_A)$. What does this translate to for the embracing sets of S_B?

(ii) Suppose $f_{i-1}(\mathrm{conv}(S_A)) = \binom{n}{i}$. What does this translate to for the embracing sets of S_B?

Exercise 11.33. *Show that a simplicial d-dimensional polytope $\mathcal{P}$ with $d+2$ vertices has always $k(d+2-k)$ facets, for some $2 \leqslant k \leqslant d$.*

Exercise 11.34. *Let $A \in \mathbb{R}^{n\times d}$ and $B \in \mathbb{R}^{n\times(n-d-1)}$ be legal matrices with $A \perp B$. Suppose $S_A = (p_1,\ldots,p_n)$, $S_B = (p_1^*,\ldots,p_n^*)$ and let $I \subseteq [n]$. Note that we do not assume general position beyond the legality of A and B; in particular, S_A and S_B may contain repeated points.*

(i) Suppose $|I| = d+1$, and points $\{p_i\}_{i\in I}$ lie in a common hyperplane. What does this translate to for S_B?

(ii) Suppose $|I| = d$, and points $\{p_i\}_{i\in I}$ lie in a common hyperplane with the origin $\vec{0} \in \mathbb{R}^d$. What does this translate to for S_B?

(iii) Show that S_A is generic iff S_B is generic. Here we call a sequence $(p_i)_{i=1}^n$ of points in $\mathbb{R}^d$ generic *if it does not contain $\vec{0}$ and no $d+1$ points in $\{p_i\}_{i=1}^n \cup \{\vec{0}\}$ lie in a common hyperplane.*

Exercise 11.35. *Let $A \in \mathbb{R}^{n\times d}$ and $B \in \mathbb{R}^{n\times(n-d-1)}$ be legal matrices with $A \perp B$, such that S_A and S_B are in general position with the origin (in particular, all points are distinct). Suppose n is even.*

We call a vector $\lambda \in \mathbb{R}^n$ balanced, *if no entry of λ is 0, and there is the same number of positive and negative entries in λ. We call (Q^+, Q^-) a feasible equipartition of $S_A = (p_1,\ldots,p_n)$ if there is a balanced vector λ such that $\lambda \cdot A = \vec{0}$ and $Q^+ = \{p_i : \lambda_i > 0\}$ and $Q^- = \{p_i : \lambda_i < 0\}$. What do these feasible equipartitions translate to for the points of S_B?*

11.5 Faster Counting in the Plane (not exam-relevant in HS 2025)

For $q \in P$, call the directed segment $\overrightarrow{0q}$ an i-*edge* if there are i points from P lying to its left. Let $\ell_i = \ell_i(P)$ be the number of i-edges of P.

Checkpoint 11.36. $\sum_i \ell_i = n$. *What is the vector $\vec{\ell} = (\ell_0, \ell_1, \ldots, \ell_{n-1})$ for the case $0 \notin \mathrm{conv}(P)$?*

For every nonempty subset $A \subseteq P$ with $0 \notin \mathrm{conv}(A)$, there is a left tangent and a right tangent to $\mathrm{conv}(A)$ from 0. We charge A to that right tangent point $q \in A$. How many sets $A \in \binom{P}{k}$ with $0 \notin \mathrm{conv}(A)$ charges to this particular point q?

Checkpoint 11.37. *The answer is* $\binom{i}{k-1}$ *if* $\overrightarrow{0q}$ *is an* i*-edge.*

Hence, for $1 \leqslant k \leqslant n$ we have

$$e_k = \underbrace{\binom{n}{k}}_{\text{all } k\text{-sets}} - \underbrace{\sum_{i=0}^{n-1} \binom{i}{k-1} \ell_i}_{\text{non-embracing } k\text{-sets}} = \sum_{i=0}^{n-1} \binom{i}{k-1} (1-\ell_i) \,. \tag{11.38}$$

As a remark, this fits in the relation (11.4) with $z_i = 1 - \ell_i$, so the numbers ℓ_i satisfying (11.38) are unique.

Exercise 11.39. *Show that* $\ell_i = \ell_{n-1-i}$*. (Hint: Wonder why "left" and not "right".)*

Theorem 11.40. *In the plane, the simplicial depth* $\mathrm{sd}_q(P)$ *can be computed in* $O(n \log n)$ *time, provided* $P \dot\cup \{q\}$ *is in general position.*

Proof. By translating the points appropriately, we may assume $q = 0$. Then we compute the vector $\vec{\ell}$ in $O(n \log n)$ time. For that we rotate a directed line around 0, starting with the horizontal line, say. We always maintain the number of points left of this line, and update this number whenever we sweep over a point $q \in P$. The q may lie ahead of 0 or behind it; depending on this the number increases or decreases by one, respectively. After a rotation by 180 degrees, we have collected the "number of points to the left" for every $q \in P$. The rotation can be implemented in discrete events; all we need is to sort the points by angle around 0, which takes $O(n \log n)$ time. The initialization of the "number to the left" costs $O(n)$ time, and each update costs $O(1)$ time. This gives $O(n \log n)$ altogether. Finally, from the vector $\vec{\ell}$, we recover the simplicial depth $\mathrm{sd}_q(P) = e_3$ via equation (11.38). □

Similarly, all entries e_k, $1 \leqslant k \leqslant n$, can be computed based on the vector $\vec{\ell}$ using (11.38). However, keep in mind that the binomial coefficients involved in the sum can be large (up to $\Theta(n)$-bit).

Given (11.38), showing that the upper bound in Theorem 11.18 is tight is actually easy. Consider the set of vertices P of a regular n-gon (n odd) centered at 0, then $\ell_{(n-1)/2} = n$ and all other ℓ_i's vanish. Therefore,

$$e_3 = \binom{n}{3} - \binom{(n-1)/2}{2} n = \frac{n(n^2-1)}{24} \,,$$

thus the case of n odd is tight in Theorem 11.18.

For n even, consider the vertices of a regular n-gon centered at 0, and let P be a slightly perturbed set of the vertices so that $P \cup \{0\}$ is in general position. For every $q \in P$, the directed segment $\overrightarrow{0q}$ is either an $(n/2-1)$-edge or an $(n/2)$-edge. Interestingly, because of the symmetry of the $\vec{\ell}$, we immediately know that $\ell_{n/2-1} = \ell_{n/2} = n/2$ and all other ℓ_i's vanish, independent of our perturbation. Now

$$e_3 = \binom{n}{3} - \left(\binom{n/2-1}{2} + \binom{n/2}{2} \right) \frac{n}{2} = \frac{n(n^2-4)}{24} \,,$$

and the tightness of Theorem 11.18 is proved also for n even.

11.6 Characterizing ℓ-Vectors (not exam-relevant in HS 2025)

A next step is to understand what possible ℓ-vectors for n points are, and to characterize and eventually count all possibilities for $\vec{\ell}$ and thus for $\vec{e}$. We start with two observations about $\vec{\ell}$.

Exercise 11.41. *Show that $\ell_{\lfloor (n-1)/2 \rfloor} \geqslant 1$. That is, there is always a bisecting edge.*

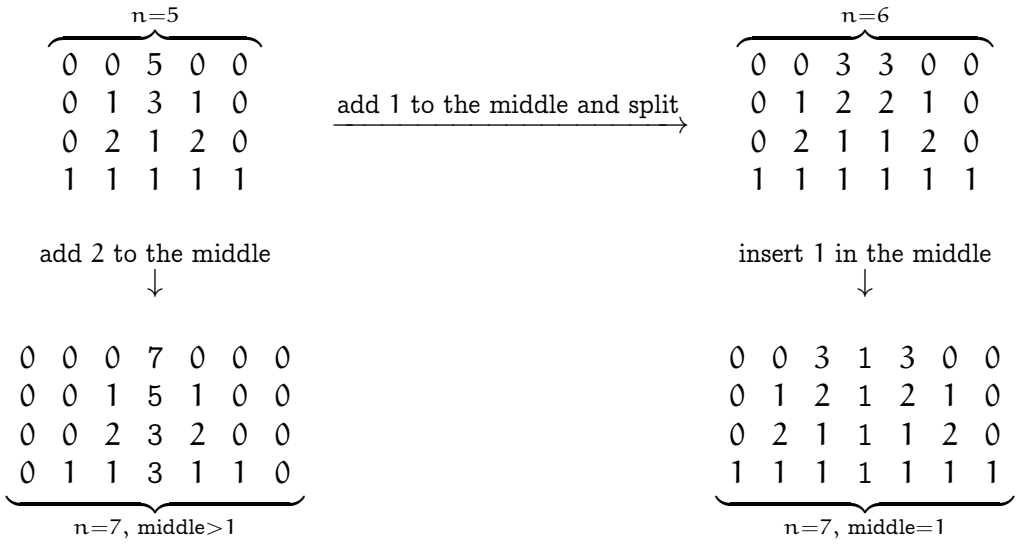
Exercise 11.42. *Show that if $\ell_i \geqslant 1$ for some $i \leqslant \lfloor (n-1)/2 \rfloor$, then $\ell_j \geqslant 1$ for all j, $i \leqslant j \leqslant \lfloor (n-1)/2 \rfloor$.*

We summarize our knowledge about $\vec{\ell}$.

Theorem 11.43. *For $n \in \mathbb{N}$, the vector $\vec{\ell} = (\ell_0, \ldots, \ell_{n-1})$ of an n-point set satisfies the following conditions.*

- *All entries are nonnegative integers.*
- $\sum_{i=0}^{n-1} \ell_i = n$.
- $\ell_i = \ell_{n-1-i}$, *namely the entries are symmetric.*
- *If $\ell_i \geqslant 1$ for some $i \leqslant \lfloor (n-1)/2 \rfloor$, then $\ell_j \geqslant 1$ for $i \leqslant j \leqslant \lfloor (n-1)/2 \rfloor$. That is, starting from the first positive entry, the subsequent entries remain positive towards the middle.*

Let us call a vector of length n a *legal n-vector* if the conditions of Theorem 11.43 are satisfied. Then the only legal 1-vector is (1), the only legal 2-vector is $(1,1)$, and the only legal 3-vectors are $(0,3,0)$ and $(1,1,1)$. The following scheme displays how we derive legal 6-vectors from legal 5-vectors, and how we can derive legal 7-vectors from legal 5- or 6-vectors.



Exercise 11.44. *Show that the scheme is complete when applied to odd n. That is, starting with all legal n-vectors, n odd, we can generate all legal $(n+1)$-vectors and all legal $(n+2)$-vectors this way.*

Exercise 11.45. *Show that the number of legal n-vectors is exactly $2^{\lfloor (n-1)/2 \rfloor}$.*

Exercise 11.46. *Show that every legal n-vector is the ℓ-vector of some set of n points in general position.*

With these exercises settled, we have given a complete characterization of all possible ℓ-vectors, thus of all possible e-vectors.

Theorem 11.47. *The number of different e-vectors (or ℓ-vectors) for n points is exactly $2^{\lfloor (n-1)/2 \rfloor}$.*

Exercise 11.48. *Show that $\sum_{i=0}^{j} \ell_i \leqslant j+1$ for all $0 \leqslant j \leqslant \lfloor (n-1)/2 \rfloor$. (Hint: Otherwise, we get into conflict with the "remains positive towards the middle" property).*

11.7 More Vector Identities (not exam-relevant in HS 2025)

We conclude the chapter with some additional identities that relate different vectors, many of which reveal illumimating combinatorial interpretations. The arguments are left as exercises. It is a good place for you to apply the mindset and methods from previous sections.

The first exercise gives an interpretation of the y_i's in relations (11.3).

Exercise 11.49. *For a set P of n points in general position, define a vector $(b_0, \ldots, b_{n-2})$ via the relations*

$$e_k = \binom{n}{k} - \sum_{i=0}^{n-2} \binom{i}{k-2} b_i = \sum_{i=0}^{n-2} \binom{i}{k-2} (n-i-1-b_i)\,,$$

for $2 \leqslant k \leqslant n$. Give a combinatorial interpretation of these numbers b_i.

Next let us investigate how the vectors $\vec{x}$, $\vec{y}$, and $\vec{z}$ from relations (11.2), (11.3), and (11.4) connect to each other. Clearly, with e_1 and e_2 given, they determine each other. But how? This will allow us to relate the vectors $\vec{h}$ and $\vec{\ell}$.

Exercise 11.50. *Consider the relations (11.2)–(11.4) on $\vec{x}_{0..n-3}, \vec{y}_{0..n-2}$, $\vec{z}_{0..n-1}$, and $\vec{e}_{1..n}$ (using $e_1 = e_2 = 0$). Prove that the y_i's are the forward differences of the x_i's, and the z_i's are the forward differences of the y_i's. More concretely, show that*

$$y_i = \begin{cases} -x_0 & i = 0 \\ x_{i-1} - x_i & 1 \leqslant i \leqslant n-3 \\ x_{n-3} & i = n-2 \end{cases}$$

or equivalently, $y_i = x_{i-1} - x_i$ for all $0 \leqslant i \leqslant n-2$, where $x_{-1} := x_{n-2} := 0$. Show that this entails $x_i = -\sum_{j=0}^{i} y_j$ for $0 \leqslant i \leqslant n-3$.

Exercise 11.51. *Prove for vectors $\vec{a}_{0..m}$ and $\vec{b}_{0..m}$,*

$$a_k = \sum_{i=0}^{m} \binom{i}{k} b_i \quad \textit{for all } 0 \leqslant k \leqslant m$$

$$\iff b_i = \sum_{k=0}^{m} (-1)^{i+k} \binom{k}{i} a_k \quad \textit{for all } 0 \leqslant i \leqslant m.$$

Exercise 11.52. *Employing the previous exercise, what does $h_0 = 1$ say about $\vec{e}_{3..n}$?*

The following facts can now be readily derived.

Theorem 11.53.

$$h_i = \binom{i+2}{2} - \sum_{j=0}^{i} (i+1-j)\ell_j$$

Exercise 11.54. *Prove Theorem 11.53.*

Note that this implies the upper bounds we proved for the h_i's in Theorem 11.18, since $\sum_{j=0}^{i}(i+1-j)\ell_j$ is always nonnegative. Moreover, a combinatorial interpretation of the slack becomes evident.

Theorem 11.55.

$$e_k = \sum_{i=0}^{n} \binom{i}{k} (\ell_i - \ell_{i-1}) \quad \textit{with } \ell_{-1} = \ell_n = 1$$

Exercise 11.56. *Prove Theorem 11.55.*

Let us point out some other counting problems that can be solved efficiently with the insights developed.

Exercise 11.57. *Given a ray r emanating from point q, and a point set $P = \{p_1, \ldots, p_n\}$ in the plane, design an efficient algorithm that counts the number of segments $\overline{p_i p_j}$ intersecting r. You may assume that $P \cup \{q\}$ is in general position and that r is disjoint from P.*

Exercise 11.58. *Let w be a line minus an interval on it (an infinite wall with a window). Given n points P in the plane, design an efficient algorithm that counts the number of pairs of points that can see each other, either because they are both on the same side of w or because they see each other through the window. You may assume general position.*

Exercise 11.59. *Recall that a point $c \in \mathbb{R}^2$ is a centerpoint of $P \subset \mathbb{R}^2$ if every halfplane containing c contains at least $|P|/3$ points from P. Identify the properties of $\vec{e}$, $\vec{h}$ and $\vec{\ell}$ which can certify that 0 is a centerpoint of P.*

Exercise 11.60. *Show that $y_i = -y_{n-2-i}$ and $y_i \leqslant 0$ for all $0 \leqslant i \leqslant \lfloor \frac{n-2}{2} \rfloor$. We refer here to the y_i's as defined by (11.3). (Hint: You may wish to use Exercises 11.48 and 11.50.)*

Exercise 11.61. *Show that $h_i \geqslant h_{i-1}$ for all $0 \leqslant \lfloor \frac{n-3}{2} \rfloor$.*

Questions

66. Explain how the h-vector of a planar point set is defined via a lifting. Give the relation between the e-vector (number of embracing k-sets) and the h-vector.
67. Argue why the h-vector is independent of the lifting.
68. Argue why the h-vector is symmetric.
69. Argue why for a given generic lifting $P' \subset \mathbb{R}^3$ of a point set $P \subset \mathbb{R}^2$ in general position, removing a point cannot increase h_j, namely $h_j(P \setminus \{p\}) \leqslant h_j(P)$ for all $j \in \mathbb{N}_0$ and all $p \in P$.
70. **(This topic was not covered in HS25 and therefore the question will not be asked in the exam.)**Show how the ℓ-vector can be computed in $O(n \log n)$ time.
71. **(This topic was not covered in HS25 and therefore the question will not be asked in the exam.)**Argue why the ℓ-vector is symmetric ($\ell_i = \ell_{n-1-i}$ for all $0 \leqslant i \leqslant n-1$).
72. Explain orthogonal duals (Gale Duality). How do embracing sets and faces of polytopes relate to each other?

References

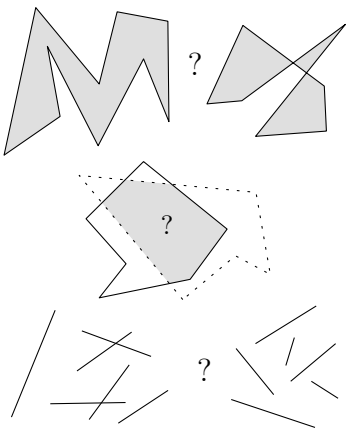
[1] Günter M. Ziegler, *Lectures on Polytopes*, vol. 152 of *Graduate Texts in Mathematics*, Springer, Heidelberg, 1994.

Appendix A

Line Sweep

In this chapter we will discuss a simple and widely applicable paradigm to design geometric algorithms: the so-called *Line-Sweep* (or Plane-Sweep) technique. It can be used to solve a variety of different problems, some examples are listed below. The first part may come as a reminder to many of you, because you should have heard something about line-sweep in one of the basic CS courses already. However, we will soon proceed and encounter a couple of additional twists that were most likely not covered there.

Consider the following geometric problems.



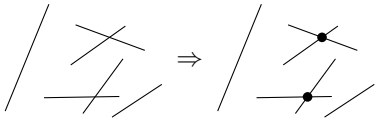
Problem A.1 (Simple Polygon Test). Given a sequence $P = (p_1, \ldots, p_n)$ of points in $\mathbb{R}^2$, does P describe the boundary of a simple polygon?

Problem A.2 (Polygon Intersection). Given two simple polygons P and Q in $\mathbb{R}^2$ as a (counterclockwise) sequence of their vertices, is $P \cap Q = \emptyset$?

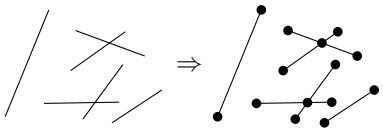
Problem A.3 (Segment Intersection Test). Given a set S of n closed line segments in $\mathbb{R}^2$, do any two of them intersect?

Remark: In principle it is clear what is meant by "two segments intersect". But there are a few special cases that one may have to consider carefully. For instance, does it count if an endpoint lies on another segment? What if two segments share an endpoint? What about overlapping segments and segments of length zero? In general, let us count all these as intersections. However, sometimes we may want to exclude some of these cases. For instance, in a simple polygon test, we do not want to consider the shared endpoint between two consecutive edges of the boundary as an intersection.

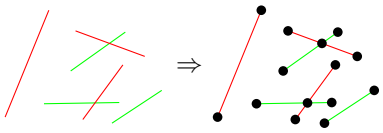
Problem A.4 (Segment Intersections). Given a set S of n closed line segments in $\mathbb{R}^2$, compute all pairs of segments that intersect.



Problem A.5 (Segment Arrangement). Given a set S of n closed line segments in $\mathbb{R}^2$, construct the arrangement induced by S, that is, the subdivision of $\mathbb{R}^2$ induced by S.



Problem A.6 (Map Overlay). Given two sets S and T of n and m, respectively, pairwise interior disjoint line segments in $\mathbb{R}^2$, construct the arrangement induced by $S \cup T$.



In the following we will use Problem A.4 as our flagship example.

Trivial Algorithm. Test all the $\binom{n}{2}$ pairs of segments from S in $O(n^2)$ time and $O(n)$ space. For Problem A.4 this is worst-case optimal because there may by $\Omega(n^2)$ intersecting pairs.

But in case that the number of intersecting pairs is, say, linear in n there is still hope to obtain a subquadratic algorithm. Given that there is a lower bound of $\Omega(n \log n)$ for *Element Uniqueness* (Given $x_1, \ldots, x_n \in \mathbb{R}$, is there an $i \neq j$ such that $x_i = x_j$?) in the algebraic computation tree model, all we can hope for is an output-sensitive runtime of the form $O(n \log n + k)$, where k denotes the number of intersecting pairs (output size).

A.1 Interval Intersections

As a warmup let us consider the corresponding problem in $\mathbb{R}^1$.

Problem A.7. Given a set I of n intervals $[\ell_i, r_i] \subset \mathbb{R}$, $1 \leqslant i \leqslant n$. Compute all pairs of intervals from I that intersect.

Theorem A.8. *Problem A.7 can be solved in* $O(n \log n + k)$ *time and* $O(n)$ *space, where* k *is the number of intersecting pairs from* $\binom{I}{2}$.

Proof. First observe that two real intervals intersect if and only if one contains the right endpoint of the other.

Sort the set $\{(\ell_i, 0) \mid 1 \leqslant i \leqslant n\} \cup \{(r_i, 1) \mid 1 \leqslant i \leqslant n\}$ in increasing lexicographic order and denote the resulting sequence by P. Store along with each point from P its origin (i). Walk through P from start to end while maintaining a list L of intervals that contain the current point $p \in P$.

Whenever $p = (\ell_i, 0)$, $1 \leqslant i \leqslant n$, insert i into L. Whenever $p = (r_i, 1)$, $1 \leqslant i \leqslant n$, remove i from L and then report for all $j \in L$ the pair $\{i, j\}$ as intersecting. □

A.2 Segment Intersections

How can we transfer the (optimal) algorithm for the corresponding problem in $\mathbb{R}^1$ to the plane? In $\mathbb{R}^1$ we moved a point from left to right and at any point resolved the situation locally around this point. More precisely, at any point during the algorithm, we knew all

intersections that are to the left of the current (moving) point. A point can be regarded a hyperplane in $\mathbb{R}^1$, and the corresponding object in $\mathbb{R}^2$ is a line.

General idea. Move a line ℓ (so-called *sweep line*) from left to right over the plane, such that at any point during this process all intersections to the left of ℓ have been reported.

Sweep line status. The list of intervals containing the current point corresponds to a list L of segments (sorted by y-coordinate) that intersect the current sweep line ℓ. This list L is called *sweep line status* (SLS). Considering the situation locally around L, it is obvious that only segments that are adjacent in L can intersect each other. This observation allows to reduce the overall number of intersection tests, as we will see. In order to allow for efficient insertion and removal of segments, the SLS is usually implemented as a balanced binary search tree.

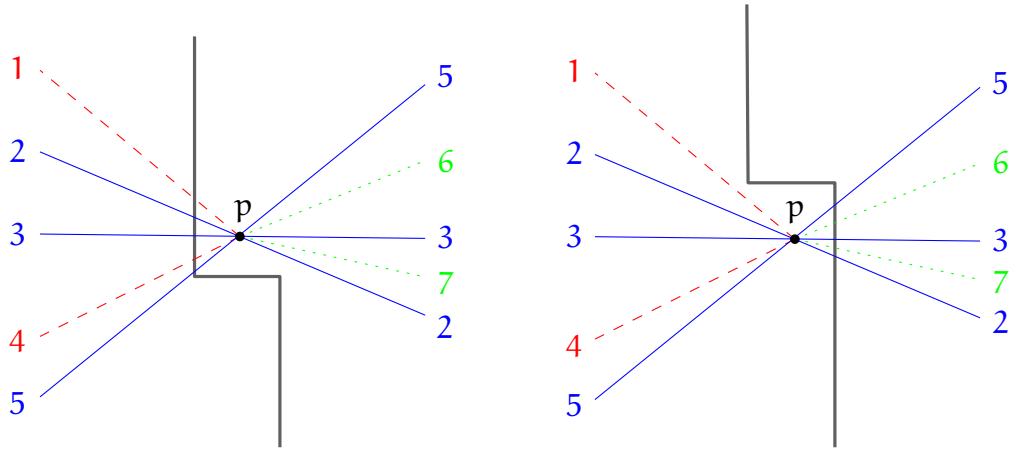
Event points. The order of segments in SLS can change at certain points only: whenever the sweep line moves over a segment endpoint or a point of intersection of two segments from S. Such a point is referred to as an *event point* (EP) of the sweep. Therefore we can reduce the conceptually continuous process of moving the sweep line over the plane to a discrete process that moves the line from EP to EP. This discretization allows for an efficient computation.

At any EP several events can happen simultaneously: several segments can start and/or end and at the same point a couple of other segments can intersect. In fact the sweep line does not even make a difference between any two event points that have the same x-coordinate. To properly resolve the order of processing, EPs are considered in lexicographic order and wherever several events happen at a single point, these are considered simultaneously as a single EP. In this light, the sweep line is actually not a line but an infinitesimal step function (see Figure A.1).

Event point schedule. In contrast to the one-dimensional situation, in the plane not all EP are known in advance because the points of intersection are discovered during the algorithm only. In order to be able to determine the next EP at any time, we use a priority queue data structure, the so-called *event point schedule* (EPS).

Along with every EP p store a list $\mathrm{end}(p)$ of all segments that end at p, a list $\mathrm{begin}(p)$ of all segments that begin at p, and a list $\mathrm{int}(p)$ of all segments in SLS that intersect at p a segment that is adjacent to it in SLS.

Along with every segment we store pointers to all its appearances in an $\mathrm{int}(\cdot)$ list of some EP. As a segment appears in such a list only if it intersects one of its neighbors there, every segment needs to store at most two such pointers.



(a) Before. **(b)** After.

Figure A.1: *Handling an event point* p. *Ending segments are shown red (dashed), starting segments green (dotted), and passing segments blue (solid).*

Invariants. The following conditions can be shown to hold before and after any event point is handled. (We will not formally prove this here.) In particular, the last condition at the end of the sweep implies the correctness of the algorithm.

1. L is the sequence of segments from S that intersect ℓ, ordered by y-coordinate of their point of intersection.
2. E contains all endpoints of segments from S and all points where two segments that are adjacent in L intersect to the right of ℓ.
3. All pairs from $\binom{S}{2}$ that intersect to the left of ℓ have been reported.

Event point handling. An EP p is processed as follows.

1. If $\text{end}(p) \cup \text{int}(p) = \emptyset$, localize p in L.
2. Report all pairs of segments from $\text{end}(p) \cup \text{begin}(p) \cup \text{int}(p)$ as intersecting.
3. Remove all segments in $\text{end}(p)$ from L.
4. Reverse the subsequence in L that is formed by the segments from $\text{int}(p)$.
5. Insert segments from $\text{begin}(p)$ into L, sorted by slope.
6. Test the topmost and bottommost segment in L from $\text{begin}(p) \cup \text{int}(p)$ for intersection with its successor and predecessor, respectively, and update EP if necessary.

Updating EPS. Insert an EP p corresponding to an intersection of two segments s and t. Without loss of generality let s be above t at the current position of ℓ.

1. If p does not yet appear in E, insert it.

2. If s is contained in an $\mathrm{int}(\cdot)$ list of some other EP q, where it intersects a segment t' from above: Remove both s and t' from the $\mathrm{int}(\cdot)$ list of q and possibly remove q from E (if $\mathrm{end}(q) \cup \mathrm{begin}(q) \cup \mathrm{int}(q) = \emptyset$). Proceed analogously in case that t is contained in an $\mathrm{int}(\cdot)$ list of some other EP, where it intersects a segment from below.

3. Insert s and t into $\mathrm{int}(p)$.

Sweep.

1. Insert all segment endpoints into $\mathrm{begin}(\cdot)/\mathrm{end}(\cdot)$ list of a corresponding EP in E.

2. As long as $E \neq \emptyset$, handle the first EP and then remove it from E.

Runtime analysis. Initialization: $O(n \log n)$. Processing of an EP p:

$$O(\#\text{intersecting pairs} + |\mathrm{end}(p)| \log n + |\mathrm{int}(p)| + |\mathrm{begin}(p)| \log n + \log n).$$

In total: $O(k + n \log n + k \log n) = O((n+k) \log n)$, where k is the number of intersecting pairs in S.

Space analysis. Clearly $|S| \leqslant n$. At begin we have $|E| \leqslant 2n$ and $|S| = 0$. Furthermore the number of additional EPs corresponding to points of intersection is always bounded by $2|S|$. Thus the space needed is $O(n)$.

Theorem A.9. *Problem A.4 and Problem A.5 can be solved in* $O((n+k) \log n)$ *time and* $O(n)$ *space.* □

Theorem A.10. *Problem A.1, Problem A.2 and Problem A.3 can be solved in* $O(n \log n)$ *time and* $O(n)$ *space.* □

Exercise A.11. *Flesh out the details of the sweep line algorithm for Problem A.2 that is referred to in Theorem A.10. What if you have to construct the intersection rather than just to decide whether or not it is empty?*

Exercise A.12. *You are given* n *axis–parallel rectangles in* $\mathbb{R}^2$ *with their bottom sides lying on the* x*–axis. Construct their union in* $O(n \log n)$ *time.*

A.3 Improvements

The basic ingredients of the line sweep algorithm go back to work by Bentley and Ottmann [2] from 1979. The particular formulation discussed here, which takes all possible degeneracies into account, is due to Mehlhorn and Näher [9].

Theorem A.10 is obviously optimal for Problem A.3, because this is just the 2-dimensional generalization of Element Uniqueness (see Section 1.1). One might suspect there is also a similar lower bound of $\Omega(n \log n)$ for Problem A.1 and Problem A.2. However, this is not the case: Both can be solved in $O(n)$ time, albeit using a very complicated algorithm of Chazelle [4] (the famous triangulation algorithm).

Similarly, it is not clear why $O(n \log n + k)$ time should not be possible in Theorem A.9. Indeed this was a prominent open problem in the 1980's that has been solved in several steps by Chazelle and Edelsbrunner in 1988. The journal version of their paper [5] consists of 54 pages and the space usage is suboptimal $O(n + k)$.

Clarkson and Shor [6] and independently Mulmuley [10, 11] described randomized algorithms with expected runtime $O(n \log n + k)$ using $O(n)$ and $O(n + k)$, respectively, space.

An optimal deterministic algorithm, with runtime $O(n \log n + k)$ and using $O(n)$ space, is known since 1995 only due to Balaban [1].

A.4 Algebraic degree of geometric primitives

We already have encountered different notions of complexity during this course: We can analyze the time complexity and the space complexity of algorithms and both usually depend on the size of the input. In addition we have seen examples of output-sensitive algorithms, whose complexity also depends on the size of the output. In this section, we will discuss a different kind of complexity that is the algebraic complexity of the underlying geometric primitives. In this way, we try to shed a bit of light into the bottom layer of geometric algorithms that is often swept under the rug because it consists of constant time operations only. Nevertheless, it would be a mistake to disregard this layer completely, because in most—if not every—real world application it plays a crucial role, by affecting efficiency as well as correctness. Regarding efficiency, the value of the constants involved can make a big difference. The possible effects on correctness will hopefully become clear in the course of this section.

In all geometric algorithms there are some fundamental geometric *predicates* and/or *constructions* on the bottom level. Both are operations on geometric objects, the difference is only in the result: The result of a construction is a geometric object (for instance, the point common to two non-parallel lines), whereas the result of a predicate is Boolean (*true* or *false*).

Geometric predicates and constructions in turn are based on fundamental arithmetic operations. For instance, we formulated planar convex hull algorithms in terms of an orientation predicate—given three points $p, q, r \in \mathbb{R}^2$, is r strictly to the right of the ori-

ented line pr— which can be implemented using multiplication and addition/subtraction of coordinates/numbers. When using limited precision arithmetic, it is important to keep an eye on the size of the expressions that occur during an evaluation of such a predicate.

In Exercise 5.31 we have seen that the orientation predicate can be computed by evaluating a polynomial of degree two in the input coordinates and, therefore, we say that

Proposition A.13. *The rightturn/orientation predicate for three points in $\mathbb{R}^2$ has algebraic degree two.*

The degree of a predicate depends on the algebraic expression used to evaluate it. Any such expression is intimately tied to the representation of the geometric objects used. Where not stated otherwise, we assume that geometric objects are represented as described in Section 1.2. So in the proposition above we assume that points are represented using Cartesian coordinates. The situation might be different, if, say, a polar representation is used instead.

But even once a representation is agreed upon, there is no obvious correspondence to an algebraic expression that describes a given predicate. In fact, it is not even clear why such an expression should exist in general. But if it does—as, for instance, in case of the orientation predicate—we prefer to work with a polynomial of smallest possible degree. Therefore we define the *(algebraic) degree* of a geometric predicate as the minimum degree of a polynomial that defines it (predicate true $\iff$ polynomial positive). So there still is something to be shown in order to complete Proposition A.13.

Exercise A.14. *Show that there is no polynomial of degree at most one that describes the orientation test in $\mathbb{R}^2$.*

Hint: Let $p = (0,0)$, $q = (x,0)$, and $r = (0,y)$ and show that there is no linear function in x and y that distinguishes the region where pqr form a rightturn from its complement.

The degree of a predicate corresponds to the size of numbers that arise during its evaluation. If all input coordinates are k-bit integers then the numbers that occur during an evaluation of a degree d predicate on these coordinates are of size about[1] dk. If the number type used for the computation can represent all such numbers, the predicate can be evaluated exactly and thus always correctly. For instance, when using a standard IEEE double precision floating point implementation which has a mantissa length of 53 bit then the above orientation predicate can be evaluated exactly if the input coordinates are integers between 0 and 2^{25}, say.

Let us now get back to the line segment intersection problem. It needs a few new geometric primitives: most prominently, constructing the intersection point of two line segments.

[1] It is only *about* dk because not only multiplications play a role but also additions. As a rule of thumb, a multiplication may double the bitsize, while an addition may increase it by one.

Two segments. Given two line segments $s = \lambda a + (1-\lambda)b, \lambda \in [0,1]$ and $t = \mu c + (1-\mu)d, \mu \in [0,1]$, it is a simple exercise in linear algebra to compute $s \cap t$. Note that $s \cap t$ is either empty or a single point or a line segment.

For $a = (a_x, a_y)$, $b = (b_x, b_y)$, $c = (c_x, c_y)$, and $d = (d_x, d_y)$ we obtain two linear equations in two variables λ and μ.

$$\begin{aligned} \lambda a_x + (1-\lambda)b_x &= \mu c_x + (1-\mu)d_x \\ \lambda a_y + (1-\lambda)b_y &= \mu c_y + (1-\mu)d_y \end{aligned}$$

Rearranging terms yields

$$\begin{aligned} \lambda(a_x - b_x) + \mu(d_x - c_x) &= d_x - b_x \\ \lambda(a_y - b_y) + \mu(d_y - c_y) &= d_y - b_y \end{aligned}$$

Assuming that the lines underlying s and t have distinct slopes (that is, they are neither identical nor parallel) we have

$$D = \begin{vmatrix} a_x - b_x & d_x - c_x \\ a_y - b_y & d_y - c_y \end{vmatrix} = \begin{vmatrix} a_x & a_y & 1 & 0 \\ b_x & b_y & 1 & 0 \\ c_x & c_y & 0 & 1 \\ d_x & d_y & 0 & 1 \end{vmatrix} \neq 0$$

and using Cramer's rule

$$\lambda = \frac{1}{D} \begin{vmatrix} d_x - b_x & d_x - c_x \\ d_y - b_y & d_y - c_y \end{vmatrix} \text{ and } \mu = \frac{1}{D} \begin{vmatrix} a_x - b_x & d_x - b_x \\ a_y - b_y & d_y - b_y \end{vmatrix}.$$

To test if s and t intersect, we can—after having sorted out the degenerate case in which both segments have the same slope—compute λ and μ and then check whether $\lambda, \mu \in [0,1]$.

Observe that both λ and D result from multiplying two differences of input coordinates. Computing the x-coordinate of the point of intersection via $b_x + \lambda(a_x - b_x)$ uses another multiplication. Overall this computation yields a fraction whose numerator is a polynomial of degree three in the input coordinates and whose denominator is a polynomial of degree two in the input coordinates.

In order to maintain the sorted order of event points in the EPS, we need to compare event points lexicographically. In case that both are intersection points, this amounts to comparing two fractions of the type discussed above. In order to formulate such a comparison as a polynomial, we have to cross-multiply the denominators, and so obtain a polynomial of degree $3 + 2 = 5$. It can be shown (but we will not do it here) that this bound is tight and so we conclude that

Proposition A.15. *The algebraic degree of the predicate that compares two intersection points of line segments lexicographically is five.*

Therefore the coordinate range in which this predicate can be evaluated exactly using IEEE double precision numbers shrinks down to integers between 0 and about $2^{10} = 1'024$.

Exercise A.16. *What is the algebraic degree of the predicate checking whether two line segments intersect? (Above we were interested in the actual intersection point, but now we consider the predicate that merely answers the question whether two segments intersect or not by yes or no).*

A.5 Red-Blue Intersections

Although the Bentley-Ottmann sweep appears to be rather simple, its implementation is not straightforward. For once, the original formulation did not take care of possible degeneracies—as we did in the preceding section. But also the algebraic degree of the predicates used is comparatively high. In particular, comparing two points of intersection lexicographically is a predicate of degree five, that is, in order to compute it, we need to evaluate a polynomial of degree five. When evaluating such a predicate with plain floating point arithmetic, one easily gets incorrect results due to limited precision roundoff errors. Such failures frequently render the whole computation useless. The line sweep algorithm is problematic in this respect, as a failure to detect one single point of intersection often implies that no other intersection of the involved segments to the right is found.

In general, predicates of degree four are needed to construct the arrangement of line segments because one needs to determine the orientation of a triangle formed by three segments. This is basically an orientation test where two points are segment endpoints and the third point is an intersection point of segments. Given that the coordinates of the latter are fractions, whose numerator is a degree three polynomial and the common denominator is a degree two polynomial, we obtain a degree four polynomial overall.

Motivated by the Map overlay application we consider here a restricted case. In the *red-blue intersection* problem, the input consists of two sets R (red) and B (blue) of segments such that the segments in each set are *interior-disjoint*, that is, for any pair of distinct segments their relative interior is disjoint. In this case there are no triangles of intersecting segments, and it turns out that predicates of degree two suffice to construct the arrangement. This is optimal because already the intersection test for two segments is a predicate of degree two.

Predicates of degree two. Restricting to degree two predicates has certain consequences. While it is possible to determine the position of a segment endpoint relative to a(nother) segment using an orientation test, one cannot, for example, compare a segment endpoint with a point of intersection lexicographically. Even computing the coordinates for a point of intersection is not possible. Therefore the output of intersection points is done implicitly, as "intersection of s and t".

Graphically speaking we can deform any segment—keeping its endpoints fixed—as long as it remains monotone and it does not reach or cross any segment endpoint (it

did not touch before). With help of degree two predicates there is no way to tell the difference.

Witnesses. Using such transformations the processing of intersection points is deferred as long as possible (lazy computation). The last possible point $w(s,t)$ where an intersection between two segments s and t has to be processed we call the *witness* of the intersection. The witness $w(s,t)$ is the lexicographically smallest segment endpoint that is located within the closed wedge formed by the two intersecting segments s and t to the right of the point of intersection (Figure A.2). Note that each such wedge contains at least two segment endpoints, namely the right endpoints of the two intersecting segments. Therefore for any pair of intersecting segments its witness is well-defined.

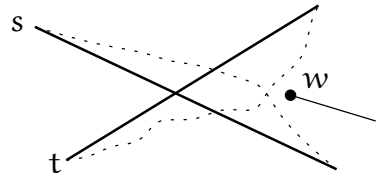


Figure A.2: *The witness* $w(s,t)$ *of an intersection* $s \cap t$.

As a consequence, only the segment endpoints are EPs and the EPS can be determined by lexicographic sorting during initialization. On the other hand, the SLS structure gets more complicated because its order of segments does not necessarily reflect the order in which the segments intersect the sweep line.

Invariants. The invariants of the algorithm have to take the relaxed notion of order into account. Denote the sweep line by ℓ.

1. L is the sequence of segments from $S = R \cup B$ intersecting ℓ; s appears before t in $L \Longrightarrow s$ intersects ℓ above t or s intersects t and the witness of this intersection is to the right of ℓ.

2. All intersections of segments from S whose witness is to the left of ℓ have been reported.

SLS Data Structure. The SLS structure consist of three levels. We use the fact that segments of the same color do not interact, except by possibly sharing endpoints.

1. Collect adjacent segments of the same color in *bundles*, stored as balanced search trees. For each bundle store pointers to the topmost and bottommost segment. (As the segments within one bundle are interior-disjoint, their order remains static and thus correct under possible deformations due to lazy computation.)

2. All bundles are stored in a doubly linked list, sorted by y-coordinate.

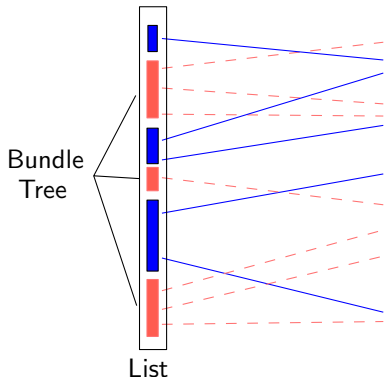


Figure A.3: *Graphical representation of the SLS data structure.*

3. All red bundles are stored in a balanced search tree (*bundle tree*).

The search tree structure should support insert, delete, split and merge in (amortized) logarithmic time each. For instance, splay trees [12] meet these requirements. (If you have never heard about splay trees so far, you do not need to know for our purposes here. Just treat them as a black box. However, you should take a note and read about splay trees still. They are a fascinating data structure.)

EP handling. An EP p is processed as follows.

1. We first want to classify all bundles with respect to their position relative to p: as either lying *above* or *below* p or as *ending* at p. There are $\leqslant 2$ bundles that have no clear such characterization.
   (a) Localize p in bundle tree $\rightarrow$ Find $\leqslant 2$ bundles without clear characterization. For the localization we use the pointers to the topmost and bottommost segment within a bundle.
   (b) Localize p in $\leqslant 2$ red bundles found and split them at p. (If p is above the topmost or below the bottommost segment of the bundle, there is no split.) All red bundles are now either above, ending, or below with respect to p.
   (c) Localize p within the blue bundles by linear search.
   (d) Localize p in the $\leqslant 2$ blue bundles found and split them at p. (If p is above the topmost or below the bottommost segment of the bundle, there is no split.) All bundles are now either above, ending, or below with respect to p.
2. Run through the list of bundles around p. More precisely, start from one of the bundles containing p found in Step 1 and walk up in the doubly linked list of bundles until two successive bundles (hence of opposite color) are both above p. Similarly, walk down in the doubly linked list of bundles, until two successive bundles are both below p. Handle all adjacent pairs of bundles (A, B) that are in

wrong order and report all pairs of segments as intersecting. (Exchange A and B in the bundle list and merge them with their new neighbors.)

3. Report all two-colored pairs from $\text{begin}(p) \times \text{end}(p)$ as intersecting.

4. Remove ending bundles and insert starting segments, sorted by slope and bundled by color and possibly merge with the closest bundle above or below.

Remark: As both the red and the blue segments are interior-disjoint, at every EP there can be at most one segment that passes through the EP. Should this happen, for the purpose of EP processing split this segment into two, one ending and one starting at this EP. But make sure to not report an intersection between the two parts!

Analysis. Sorting the EPS: $O(n \log n)$ time. Every EP generates a constant number of tree searches and splits of $O(\log n)$ each. Every exchange in Step 2 generates at least one intersection. New bundles are created only by inserting a new segment or by one of the constant number of splits at an EP. Therefore $O(n)$ bundles are created in total. The total number of merge operations is $O(n)$, as every merge kills one bundle and $O(n)$ bundles are created overall. The linear search in steps 1 and 2 can be charged either to the ending bundle or—for continuing bundles—to the subsequent merge operation. In summary, we have a runtime of $O(n \log n + k)$ and space usage is linear obviously.

Theorem A.17. *For two sets R and B, each consisting of interior-disjoint line segments in $\mathbb{R}^2$, one can find all intersecting pairs of segments in $O(n \log n + k)$ time and linear space, using predicates of maximum degree two. Here $n = |R| + |B|$ and k is the number of intersecting pairs.*

Remarks. The first optimal algorithm for the red-blue intersection problem was published in 1988 by Harry Mairson and Jorge Stolfi [7]. In 1994 Timothy Chan [3] described a trapezoidal-sweep algorithm that uses predicates of degree three only. The approach discussed above is due to Andrea Mantler and Jack Snoeyink [8] from 2000.

Exercise A.18. *Let S be a set of n segments each of which is either horizontal or vertical. Describe an $O(n \log n)$ time and $O(n)$ space algorithm that counts the number of pairs in $\binom{S}{2}$ that intersect.*

Questions

73. *How can one test whether a polygon on n vertices is simple?* Describe an $O(n \log n)$ time algorithm.

74. *How can one test whether two simple polygons on altogether n vertices intersect?* Describe an $O(n \log n)$ time algorithm.

75. *How does the line sweep algorithm work that finds all* k *intersecting pairs among* n *line segments in* $\mathbb{R}^2$*?* Describe the algorithm, using $O((n+k)\log n)$ time and $O(n)$ space. In particular, explain the data structures used, how event points are handled, and how to cope with degeneracies.

76. *Given two line segments* s *and* t *in* $\mathbb{R}^2$ *whose endpoints have integer coordinates in* $[0, 2^b)$*; suppose* s *and* t *intersect in a single point* q*, what can you say about the coordinates of* q*?* Give a good (tight up to small additive number of bits) upper bound on the size of these coordinates. (No need to prove tightness, that is, give an example which achieves the bound.)

77. *What is the degree of a predicate and why is it an important parameter? What is the degree of the orientation test and the incircle test in* $\mathbb{R}^2$*?* Explain the term and give two reasons for its relevance. Provide tight upper bounds for the two predicates mentioned. (No need to prove tightness.)

78. *What is the map overlay problem and how can it be solved optimally using degree two predicates only?* Give the problem definition and explain the term "interior-disjoint". Explain the bundle-sweep algorithm, in particular, the data structures used, how an event point is processed, and the concept of witnesses.

References

[1] Ivan J. Balaban, An optimal algorithm for finding segment intersections. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pp. 211–219, 1995.

[2] Jon L. Bentley and Thomas A. Ottmann, Algorithms for reporting and counting geometric intersections. *IEEE Trans. Comput.*, C-28/9, (1979), 643–647.

[3] Timothy M. Chan, A simple trapezoid sweep algorithm for reporting red/blue segment intersections. In *Proc. 6th Canad. Conf. Comput. Geom.*, pp. 263–268, 1994.

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

[5] Bernard Chazelle and Herbert Edelsbrunner, An optimal algorithm for intersecting line segments in the plane. *J. ACM*, 39/1, (1992), 1–54.

[6] Kenneth L. Clarkson and Peter W. Shor, Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.*, 4, (1989), 387–421.

[7] Harry G. Mairson and Jorge Stolfi, Reporting and counting intersections between two sets of line segments. In R. A. Earnshaw, ed., *Theoretical Foundations of Computer Graphics and CAD*, vol. 40 of *NATO ASI Series F*, pp. 307–325, Springer, Berlin, Germany, 1988.

[8] Andrea Mantler and Jack Snoeyink, Intersecting red and blue line segments in optimal time and precision. In J. Akiyama, M. Kano, and M. Urabe, eds., *Proc. Japan Conf. Discrete Comput. Geom.*, vol. 2098 of *Lecture Notes Comput. Sci.*, pp. 244–251, Springer, 2001.

[9] Kurt Mehlhorn and Stefan Näher, Implementation of a sweep line algorithm for the straight line segment intersection problem. Report MPI-I-94-160, Max-Planck-Institut Inform., Saarbrücken, Germany, 1994.

[10] Ketan Mulmuley, A fast planar partition algorithm, I. *J. Symbolic Comput.*, 10/3-4, (1990), 253–280.

[11] Ketan Mulmuley, A fast planar partition algorithm, II. *J. ACM*, 38, (1991), 74–103.

[12] Daniel D. Sleator and Robert E. Tarjan, Self-adjusting binary search trees. *J. ACM*, 32/3, (1985), 652–686.

Appendix B

The Configuration Space Framework

In Section 7.1, we have discussed the incremental construction of the Delaunay triangulation of a finite point set. In this lecture, we want to analyze the runtime of this algorithm if the insertion order is chosen *uniformly at random* among all insertion orders. We will do the analysis not directly for the problem of constructing the Delaunay triangulation but in a somewhat more abstract framework, with the goal of reusing the analysis for other problems.

Throughout this lecture, we again assume general position: no three points on a line, no four on a circle.

B.1 The Delaunay triangulation — an abstract view

The incremental construction constructs and destroys triangles. In this section, we want to take a closer look at these triangles, and we want to understand exactly when a triangle is "there".

Lemma B.1. *Given three points* $p, q, r \in R$*, the triangle* $\Delta(p, q, r)$ *with vertices* p, q, r *is a triangle of* $\mathcal{DT}(R)$ *if and only if the circumcircle of* $\Delta(p, q, r)$ *is empty of points from* R*.*

Proof. The "only if" direction follows from the definition of a Delaunay triangulation (Definition 6.8). The "if" direction is a consequence of general position and Lemma 6.17: if the circumcircle C of $\Delta(p, q, r)$ is empty of points from R, then all the three edges $\overline{pq}, \overline{qr}, \overline{pr}$ are easily seen to be in the Delaunay graph of R. C being empty also implies that the triangle $\Delta(p, q, r)$ is empty, and hence it forms a triangle of $\mathcal{DT}(R)$. □

Next we develop a somewhat more abstract view of $\mathcal{DT}(R)$.

Definition B.2.

(i) For all $p, q, r \in P$*, the triangle* $\Delta = \Delta(p, q, r)$ *is called a* configuration*. The points* p, q *and* r *are called the* defining elements *of* Δ*.*

(ii) A configuration Δ is in conflict *with a point $s \in P$ if s is strictly inside the circumcircle of Δ. In this case, the pair (Δ, s) is called a* conflict*.*

(iii) A configuration Δ is called active *w.r.t. $R \subseteq P$ if (a) the defining elements of Δ are in R, and (b) if Δ is not in conflict with any element of R.*

According to this definition and Lemma B.1, $\mathcal{DT}(R)$ consists of exactly the configurations that are active w.r.t. R. Moreover, if we consider $\mathcal{DT}(R)$ and $\mathcal{DT}(R \cup \{s\})$ as sets of configurations, we can exactly say how these two sets differ.

There are the configurations in $\mathcal{DT}(R)$ that are not in conflict with s. These configurations are still in $\mathcal{DT}(R \cup \{s\})$. The configurations of $\mathcal{DT}(R)$ that *are* in conflict with s will be removed when going from R to $R \cup \{s\}$. Finally, $\mathcal{DT}(R \cup \{s\})$ contains some new configurations, all of which must have s in their defining set. According to Lemma B.1, it cannot happen that we get a new configuration without s in its defining set, as such a configuration would have been present in $\mathcal{DT}(R)$ already.

B.2 Configuration Spaces

Here is the abstract framework that generalizes the previous configuration view of the Delaunay triangulation.

Definition B.3. *Let X (the ground set) and Π (the set of configurations) be finite sets. Furthermore, let*

$$D : \Pi \to 2^X$$

be a function that assigns to every configuration Δ a set of defining elements *$D(\Delta)$. We assume that only a constant number of configurations have the same defining elements. Let*

$$K : \Pi \to 2^X$$

be a function that assigns to every configuration Δ a set of elements in conflict *with Δ (the "killer" elements). We stipulate that $D(\Delta) \cap K(\Delta) = \emptyset$ for all $\Delta \in \Pi$.*

Then the quadruple $\mathcal{S} = (X, \Pi, D, K)$ is called a configuration space*. The number*

$$d = d(\mathcal{S}) := \max_{\Delta \in \Pi} |D(\Delta)|$$

is called the dimension *of $\mathcal{S}$.*

Given $R \subseteq X$, a configuration Δ is called active *w.r.t. R if*

$$D(\Delta) \subseteq R \quad \textit{and} \quad K(\Delta) \cap R = \emptyset,$$

i.e. if all defining elements are in R but no element of R is in conflict with Δ. The set of active configurations w.r.t. R is denoted by $\mathcal{T}_{\mathcal{S}}(R)$, where we drop the subscript if the configuration space is clear from the context.

In case of the Delaunay triangulation, we set $X = P$ (the input point set). Π consists of all triangles $\Delta = \Delta(p, q, r)$ spanned by three points $p, q, r \in X \cup \{a, b, c\}$, where a, b, c are the three artificial far-away points. We set $D(\Delta) := \{p, q, r\} \cap X$. The set $K(\Delta)$ consists of all points strictly inside the circumcircle of Δ. The resulting configuration space has dimension 3, and the technical condition that only a constant number of configurations share the defining set is satisfied as well. In fact, every set of three points defines a *unique* configuration (triangle) in this case. A set of two points or one point defines three triangles (we have to add one or two artificial points which can be done in three ways). The empty set defines one triangle, the initial triangle consisting of just the three artificial points.

Furthermore, in the setting of the Delaunay triangulation, a configuration is active w.r.t. R if it is in $\mathcal{DT}(R \cup \{a, b, c\})$, i.e. we have $\mathcal{T}(R) = \mathcal{DT}(R \cup \{a, b, c\})$.

B.3 Expected structural change

Let us fix a configuration space $\mathcal{S} = (X, \Pi, D, K)$ for the remainder of this lecture. We can also interpret the incremental construction in $\mathcal{S}$. Given $R \subseteq X$ and $s \in X \setminus R$, we want to update $\mathcal{T}(R)$ to $\mathcal{T}(R \cup \{s\})$. What is the number of new configurations that arise during this step? For the case of Delaunay triangulations, this is the relevant question when we want to bound the number of Lawson flips during one update step, since this number is exactly the number of new configurations minus three.

Here is the general picture.

Definition B.4. *For $Q \subseteq X$ and $s \in Q$, $\deg(s, Q)$ is defined as the number of configurations of $\mathcal{T}(Q)$ that have s in their defining set.*

With this, we can say that the number of new configurations in going from $\mathcal{T}(R)$ to $\mathcal{T}(R \cup \{s\})$ is precisely $\deg(s, R \cup \{s\})$, since the new configurations are by definition exactly the ones that have s in their defining set.

Now the random insertion order comes in for the first time: what is

$$E(\deg(s, R \cup \{s\})),$$

averaged over all insertion orders? In such a random insertion order, R is a random r-element subset of X (when we are about to insert the $(r+1)$-st element), and s is a random element of $X \setminus R$. Let $\mathcal{T}_r$ be the "random variable" for the set of active configurations after r insertion steps.

It seems hard to average over all R, but there is a trick: we make a movie of the randomized incremental construction, and then we watch the movie backwards. What we see is elements of X being deleted one after another, again in random order. This is due to the fact that the reverse of a random order is also random. At the point where the $(r+1)$-st element is being deleted, it is going to be a *random* element s of the currently

present $(r+1)$-element subset Q. For fixed Q, the expected degree of s is simply the average degree of an element in Q which is

$$\frac{1}{r+1}\sum_{s\in Q}\deg(s,Q) \leqslant \frac{d}{r+1}|\mathcal{T}(Q)|,$$

since the sum counts every configuration of $\mathcal{T}(Q)$ at most d times. Since Q is a random $(r+1)$-element subset, we get

$$E(\deg(s,R\cup\{s\})) \leqslant \frac{d}{r+1}t_{r+1},$$

where t_{r+1} is defined as the expected number of active configurations w.r.t. a random $(r+1)$-element set.

Here is a more formal derivation that does not use the backwards movie view. It exploits the bijection

$$(R,s) \mapsto (\underbrace{R\cup\{s\}}_{Q}, s)$$

between pairs (R,s) with $|R|=r$ and $s\notin R$ and pairs (Q,s) with $|Q|=r+1$ and $s\in Q$. Let $n=|X|$.

$$\begin{aligned}
E(\deg(s,R\cup\{s\})) &= \frac{1}{\binom{n}{r}}\sum_{R\subseteq X,|R|=r}\frac{1}{n-r}\sum_{s\in X\setminus R}\deg(s,R\cup\{s\})\\
&= \frac{1}{\binom{n}{r}}\sum_{Q\subseteq X,|Q|=r+1}\frac{1}{n-r}\sum_{s\in Q}\deg(s,Q)\\
&= \frac{1}{\binom{n}{r+1}}\sum_{Q\subseteq X,|Q|=r+1}\frac{\binom{n}{r+1}}{\binom{n}{r}}\frac{1}{n-r}\sum_{s\in Q}\deg(s,Q)\\
&= \frac{1}{\binom{n}{r+1}}\sum_{Q\subseteq X,|Q|=r+1}\frac{1}{r+1}\sum_{s\in Q}\deg(s,Q)\\
&\leqslant \frac{1}{\binom{n}{r+1}}\sum_{Q\subseteq X,|Q|=r+1}\frac{d}{r+1}|\mathcal{T}(Q)|\\
&= \frac{d}{r+1}t_{r+1}.
\end{aligned}$$

Thus, the expected number of new active configurations in going from $\mathcal{T}_r$ to $\mathcal{T}_{r+1}$ is bounded by

$$\frac{d}{r+1}t_{r+1},$$

where t_{r+1} is the expected size of $\mathcal{T}_{r+1}$.

What do we get for Delaunay triangulations? We have $d = 3$ and $t_{r+1} \leqslant 2(r+4)-4$ (the maximum number of triangles in a triangulation of $r+4$ points). Hence,

$$E(\deg(s, R \cup \{s\})) \leqslant \frac{6r+12}{r+1} \approx 6.$$

This means that on average, ≈ 3 Lawson flips are done to update $\mathcal{DT}_r$ (the Delaunay triangulation after r insertion steps) to $\mathcal{DT}_{r+1}$. Over the whole algorithm, the expected update cost is thus $O(n)$.

B.4 Bounding location costs by conflict counting

Before we can even update $\mathcal{DT}_r$ to $\mathcal{DT}_{r+1}$ during the incremental construction of the Delaunay triangulation, we need to locate the new point s in $\mathcal{DT}_r$, meaning that we need to find the triangle that contains s. We have done this with the history graph: During the insertion of s we "visit" a sequence of triangles from the history graph, each of which contains s and was created at some previous iteration $k < r$.

However, some of these visited triangles are "ephemeral" triangles (recall the discussion at the end of Section 7.3), and they present a problem to the generic analysis we want to perform. Therefore, we will do a charging scheme, so that all triangles charged are valid Delaunay triangles.

The charging scheme is as follows: If the visited triangle Δ is a valid Delaunay triangle (from some previous iteration), then we simply charge the visit of Δ during the insertion of s to the triangle-point pair (Δ, s).

If, on the other hand, Δ is an "ephemeral" triangle, then Δ was destroyed, together with some neighbor Δ', by a Lawson flip into another pair Δ'', Δ'''. Note that this neighbor Δ' was a valid triangle. Thus, in this case we charge the visit of Δ during the insertion of s to the pair (Δ', s). Observe that s is contained in the circumcircle of Δ', so s is in conflict with Δ'.

This way, we have charged each visit to a triangle in the history graph to a triangle-point pair of the form (Δ, s), such that Δ is in conflict with s. Furthermore, it is easy to see that no such pair gets charged more than once.

We define the notion of a *conflict* in general:

Definition B.5. *A* conflict *is a configuration-element pair* (Δ, s) *where* $\Delta \in \mathcal{T}_r$ *for some* r *and* $s \in K(\Delta)$.

Thus, the running time of the Delaunay algorithm is proportional to the number of conflicts. We now proceed to derive a bound on the expected number of conflicts in the generic configuration-space framework.

B.5 Expected number of conflicts

Since every configuration involved in a conflict has been created in some step r (we include step 0), the total number of conflicts is

$$\sum_{r=0}^{n} \sum_{\Delta \in \mathcal{T}_r \setminus \mathcal{T}_{r-1}} |K(\Delta)|,$$

where $\mathcal{T}_{-1} := \emptyset$. $\mathcal{T}_0$ consists of constantly many configurations only (namely those where the set of defining elements is the empty set), each of which is in conflict with at most all elements; moreover, no conflict is created in step n. Hence,

$$\sum_{r=0}^{n} \sum_{\Delta \in \mathcal{T}_r \setminus \mathcal{T}_{r-1}} |K(\Delta)| = O(n) + \sum_{r=1}^{n-1} \sum_{\Delta \in \mathcal{T}_r \setminus \mathcal{T}_{r-1}} |K(\Delta)|,$$

and we will bound the latter quantity. Let

$$K(r) := \sum_{\Delta \in \mathcal{T}_r \setminus \mathcal{T}_{r-1}} |K(\Delta)|, \quad r = 1, \ldots, n-1.$$

and $k(r) := E(K(r))$ the expected number of conflicts created in step r.

Bounding $k(r)$. We know that $\mathcal{T}_r$ arises from a random r-element set R. Fixing R, the backwards movie view tells us that $\mathcal{T}_{r-1}$ arises from $\mathcal{T}_r$ by deleting a random element s of R. Thus,

$$\begin{aligned} k(r) &= \frac{1}{\binom{n}{r}} \sum_{R \subseteq X, |R|=r} \frac{1}{r} \sum_{s \in R} \sum_{\Delta \in \mathcal{T}(R) \setminus \mathcal{T}(R \setminus \{s\})} |K(\Delta)| \\ &= \frac{1}{\binom{n}{r}} \sum_{R \subseteq X, |R|=r} \frac{1}{r} \sum_{s \in R} \sum_{\Delta \in \mathcal{T}(R), s \in D(\Delta)} |K(\Delta)| \\ &\leqslant \frac{1}{\binom{n}{r}} \sum_{R \subseteq X, |R|=r} \frac{d}{r} \sum_{\Delta \in \mathcal{T}(R)} |K(\Delta)|, \end{aligned}$$

since in the sum over $s \in R$, every configuration is counted at most d times. Since we can rewrite

$$\sum_{\Delta \in \mathcal{T}(R)} |K(\Delta)| = \sum_{y \in X \setminus R} |\{\Delta \in \mathcal{T}(R) : y \in K(\Delta)\}|,$$

we thus have

$$k(r) \leqslant \frac{1}{\binom{n}{r}} \sum_{R \subseteq X, |R|=r} \frac{d}{r} \sum_{y \in X \setminus R} |\{\Delta \in \mathcal{T}(R) : y \in K(\Delta)\}|.$$

To estimate this further, here is a simple but crucial

Lemma B.6. *The configurations in $\mathcal{T}(R)$ that are not in conflict with $y \in X \setminus R$ are the configurations in $\mathcal{T}(R \cup \{y\})$ that do not have y in their defining set; in formulas:*

$$|\mathcal{T}(R)| - |\{\Delta \in \mathcal{T}(R) : y \in K(\Delta)\}| = |\mathcal{T}(R \cup \{y\})| - \deg(y, R \cup \{y\}).$$

The proof is a direct consequence of the definitions: every configuration in $\mathcal{T}(R)$ not in conflict with y is by definition still present in $\mathcal{T}(R \cup \{y\})$ and still does not have y in its defining set. And a configuration in $\mathcal{T}(R \cup \{y\})$ with y not in its defining set is by definition already present in $\mathcal{T}(R)$ and already there not in conflict with y.

The lemma implies that

$$k(r) \leqslant k_1(r) - k_2(r) + k_3(r),$$

where

$$k_1(r) = \frac{1}{\binom{n}{r}} \sum_{R \subseteq X, |R|=r} \frac{d}{r} \sum_{y \in X \setminus R} |\mathcal{T}(R)|,$$

$$k_2(r) = \frac{1}{\binom{n}{r}} \sum_{R \subseteq X, |R|=r} \frac{d}{r} \sum_{y \in X \setminus R} |\mathcal{T}(R \cup \{y\})|,$$

$$k_3(r) = \frac{1}{\binom{n}{r}} \sum_{R \subseteq X, |R|=r} \frac{d}{r} \sum_{y \in X \setminus R} \deg(y, R \cup \{y\}).$$

Estimating $k_1(r)$. This is really simple.

$$\begin{aligned} k_1(r) &= \frac{1}{\binom{n}{r}} \sum_{R \subseteq X, |R|=r} \frac{d}{r} \sum_{y \in X \setminus R} |\mathcal{T}(R)| \\ &= \frac{1}{\binom{n}{r}} \sum_{R \subseteq X, |R|=r} \frac{d}{r}(n-r)|\mathcal{T}(R)| \\ &= \frac{d}{r}(n-r)t_r. \end{aligned}$$

Estimating $k_2(r)$. For this, we need to employ our earlier $(R, y) \mapsto (R \cup \{y\}, y)$ bijection again.

$$\begin{aligned}
k_2(r) &= \frac{1}{\binom{n}{r}} \sum_{R \subseteq X, |R|=r} \frac{d}{r} \sum_{y \in X \setminus R} |\mathcal{T}(R \cup \{y\})| \\
&= \frac{1}{\binom{n}{r}} \sum_{Q \subseteq X, |Q|=r+1} \frac{d}{r} \sum_{y \in Q} |\mathcal{T}(Q)| \\
&= \frac{1}{\binom{n}{r+1}} \sum_{Q \subseteq X, |Q|=r+1} \frac{\binom{n}{r+1}}{\binom{n}{r}} \frac{d}{r}(r+1)|\mathcal{T}(Q)| \\
&= \frac{1}{\binom{n}{r+1}} \sum_{Q \subseteq X, |Q|=r+1} \frac{d}{r}(n-r)|\mathcal{T}(Q)| \\
&= \frac{d}{r}(n-r)t_{r+1} \\
&= \frac{d}{r+1}(n-(r+1))t_{r+1} + \frac{dn}{r(r+1)}t_{r+1} \\
&= k_1(r+1) + \frac{dn}{r(r+1)}t_{r+1}.
\end{aligned}$$

Estimating $k_3(r)$. This is similar to $k_2(r)$ and in addition uses a fact that we have employed before: $\sum_{y \in Q} \deg(y, Q) \leqslant d|\mathcal{T}(Q)|$.

$$\begin{aligned}
k_3(r) &= \frac{1}{\binom{n}{r}} \sum_{R \subseteq X, |R|=r} \frac{d}{r} \sum_{y \in X \setminus R} \deg(y, R \cup \{y\}) \\
&= \frac{1}{\binom{n}{r}} \sum_{Q \subseteq X, |Q|=r+1} \frac{d}{r} \sum_{y \in Q} \deg(y, Q) \\
&\leqslant \frac{1}{\binom{n}{r}} \sum_{Q \subseteq X, |Q|=r+1} \frac{d^2}{r}|\mathcal{T}(Q)| \\
&= \frac{1}{\binom{n}{r+1}} \sum_{Q \subseteq X, |Q|=r+1} \frac{\binom{n}{r+1}}{\binom{n}{r}} \frac{d^2}{r}|\mathcal{T}(Q)| \\
&= \frac{1}{\binom{n}{r+1}} \sum_{Q \subseteq X, |Q|=r+1} \frac{n-r}{r+1} \cdot \frac{d^2}{r}|\mathcal{T}(Q)| \\
&= \frac{d^2}{r(r+1)}(n-r)t_{r+1} \\
&= \frac{d^2 n}{r(r+1)}t_{r+1} - \frac{d^2}{r+1}t_{r+1}.
\end{aligned}$$

Summing up. Let us recapitulate: the overall expected number of conflicts is $O(n)$ plus

$$\sum_{r=1}^{n-1} k(r) \leqslant \sum_{r=1}^{n-1} (k_1(r) - k_2(r) + k_3(r)).$$

Using our previous estimates, $k_1(2), \ldots, k_1(n-1)$ are canceled by the first terms of $k_2(1), \ldots, k_2(n-2)$. The second term of $k_2(r)$ can be combined with the first term of $k_3(r)$, so that we get

$$\begin{aligned}
\sum_{r=1}^{n-1} (k_1(r) - k_2(r) + k_3(r)) &\leqslant k_1(1) - \underbrace{k_1(n)}_{=0} + n \sum_{r=1}^{n-1} \frac{d(d-1)}{r(r+1)} t_{r+1} - \sum_{r=1}^{n-1} \frac{d^2}{r+1} t_{r+1} \\
&\leqslant d(n-1)t_1 + d(d-1)n \sum_{r=1}^{n-1} \frac{t_{r+1}}{r(r+1)} \\
&= O\left(d^2 n \sum_{r=1}^{n} \frac{t_r}{r^2} \right).
\end{aligned}$$

The Delaunay case. We have argued that the expected number of conflicts asymptotically bounds the expected total location cost over all insertion steps. The previous equation tells us that this cost is proportional to $O(n)$ plus

$$O\left(9n \sum_{r=1}^{n} \frac{2(r+3)-4}{r^2} \right) = O\left(n \sum_{r=1}^{n} \frac{1}{r} \right) = O(n \log n).$$

Here,

$$\sum_{r=1}^{n} \frac{1}{r} =: H_n$$

is the n-th Harmonic Number which is known to be approximately $\ln n$.

By going through the abstract framework of configuration spaces, we have thus analyzed the randomized incremental construction of the Delaunay triangulation of n points. According to Section B.3, the expected update cost itself is only $O(n)$. The steps dominating the runtime are the location steps via the history graph. According to Section B.5, all history graph searches (whose number is proportional to the number of conflicts) can be performed in expected time $O(n \log n)$, and this then also bounds the space requirements of the algorithm.

Exercise B.7. *Design and analyze a sorting algorithm based on randomized incremental construction in configuration spaces. The input is a set S of numbers, and the output should be the sorted sequence (in increasing order).*

a) Define an appropriate configuration space for the problem! In particular, the set of active configurations w.r.t. S should represent the desired sorted sequence.

b) Provide an efficient implementation of the incremental construction algorithm. "Efficient" means that the runtime of the algorithm is asymptotically dominated by the number of conflicts.

c) What is the expected number of conflicts (and thus the asymptotic runtime of your sorting algorithm) for a set S of n numbers?

Questions

79. *What is a configuration space?* Give a precise definition! What is an active configuration?

80. *How do we get a configuration space from the problem of computing the Delaunay triangulation of a finite point set?*

81. *How many new active configurations do we get on average when inserting the r-th element?* Provide an answer for configuration spaces in general, and for the special case of the Delaunay triangulation.

82. *What is a conflict?* Provide an answer for configuration spaces in general, and for the special case of the Delaunay triangulation.

83. *Explain why counting the expected number of conflicts asymptotically bounds the cost for the history searches during the randomized incremental construction of the Delaunay triangulation!*

Appendix C

Trapezoidal Maps

In this section, we will see another application of randomized incremental construction in the abstract configuration space framework. At the same time, this will give us an efficient algorithm for solving the general problem of point location, as well as a faster algorithm for computing all intersections between a given set of line segments.

C.1 The Trapezoidal Map

To start with, let us introduce the concept of a *trapezoidal map*.

We are given a set $S = \{s_1, \ldots, s_n\}$ of line segments in the plane (not necessarily disjoint). We make several general position assumptions.

We assume that no two segment endpoints and intersection points have the same x-coordinate. As an exception, we do allow several segments to share an endpoint. We also assume that no line segment is vertical, that any two line segments intersect in at most one point (which is a common endpoint, or a proper crossing), and that no three line segments have a common point. Finally, we assume that $s_i \subseteq [0,1]^2$ for all i (which can be achieved by scaling the coordinates of the segments accordingly).

Definition C.1. *The* trapezoidal map *of S is the partition of $[0,1]^2$ into vertices, edges, and faces (called* trapezoids*), obtained as follows. Every segment endpoint and point of intersection between segments gets connected by two vertical extensions with the next feature below and above, where a feature is either another line segment or an edge of the bounding box $[0,1]^2$.*

Figure C.1 gives an example.

(The general-position assumptions are made only for convenience and simplicity of the presentation. The various degeneracies can be handled without too much trouble, though we will not get into the details.)

The trapezoids of the trapezoidal map are "true" trapezoids (quadrangles with two parallel vertical sides), and triangles (which may be considered as degenerate trapezoids).

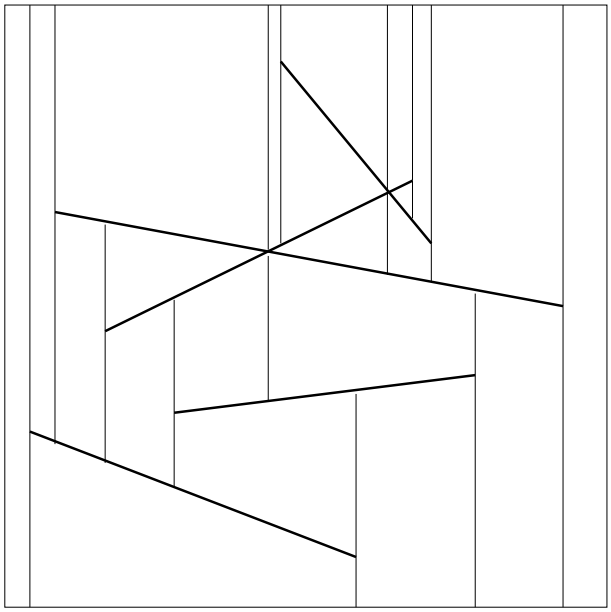


Figure C.1: *The trapezoidal map of five line segments (depicted in bold)*

C.2 Applications of trapezoidal maps

In this chapter we will see two applications of trapezoidal maps (there are others):

1. *Point location:* Given a set of n segments in the plane, we want to preprocess them in order to answer point-location queries: given a point p, return the cell (connected component of the complement of the segments) that contains p (see Figure C.2). This is a more powerful alternative to Kirkpatrick's algorithm that handles only triangulations, and which is treated in Section 8.5. The preprocessing constructs the trapezoidal map of the segments (Figure C.3) in expected time $O(n \log n + K)$, where K is the number of intersections between the input segments; the query time will be $O(\log n)$ in expectation.

2. *Line segment intersection:* Given a set of n segments, we will report all segment intersections in expected time $O(n \log n + K)$. This is a faster alternative to the classical line-sweep algorithm, which takes time $O((n + K) \log n)$.

C.3 Incremental Construction of the Trapezoidal Map

We can construct the trapezoidal map by inserting the segments one by one, in random order, always maintaining the trapezoidal map of the segments inserted so far. In order to perform manipulations efficiently, we can represent the current trapezoidal map as a doubly-connected edge list (see Section 2.2.1), for example.

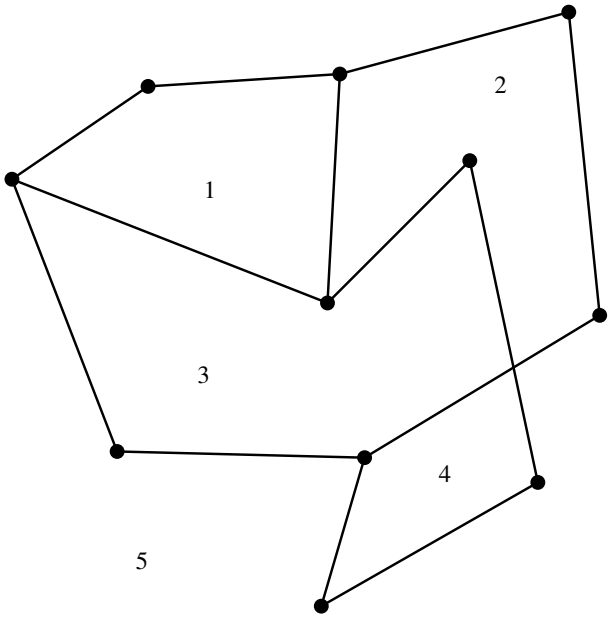


Figure C.2: *The general point location problem defined by a set of (possibly intersecting) line segments. In this example, the segments partition the plane into 5 cells.*

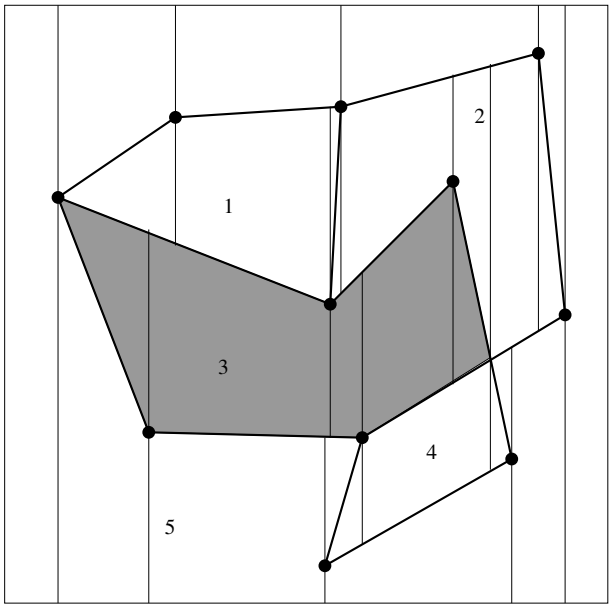


Figure C.3: *The trapezoidal map is a refinement of the partition of the plane into cells. For example, cell 3 is a union of five trapezoids.*

Suppose that we have already inserted segments $s_1, \ldots, s_{r-1}$, and that the resulting trapezoidal map $\mathcal{T}_{r-1}$ looks like in Figure C.1. Now we insert segment s_r (see Figure C.4).

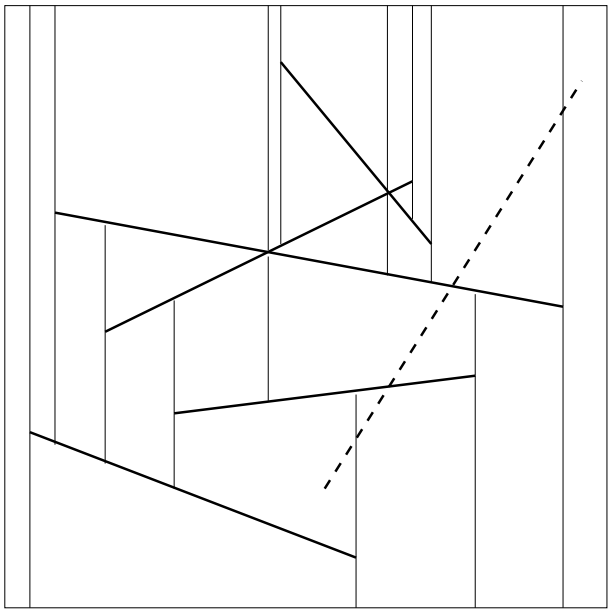


Figure C.4: *A new segment (dashed) is to be inserted*

Here are the four steps that we need to do in order to construct $\mathcal{T}_r$.

1. **Find** the trapezoid $\square_0$ of $\mathcal{T}_{r-1}$ that contains the left endpoint of s_r.
2. **Trace** s_r through $\mathcal{T}_{r-1}$ until the trapezoid containing the right endpoint of s_r is found. To get from the current trapezoid $\square$ to the next one, traverse the boundary of $\square$ until the edge is found through which s_r leaves $\square$.
3. **Split** the trapezoids intersected by s_r. A trapezoid $\square$ may get replaced by
   - two new trapezoids (if s_r intersects two vertical extensions of $\square$);
   - three new trapezoids (if s_r intersects one vertical extension of $\square$);
   - four new trapezoids (if s_r intersects no vertical extension of $\square$).
4. **Merge** trapezoids by removing parts of vertical extensions that do not belong to $\mathcal{T}_r$ anymore.

Figure C.5 illustrates the **Trace** and **Split** steps. s_6 intersects 5 trapezoids, and they are being split into $3, 3, 4, 3$, and 3 trapezoids, respectively.

The **Merge** step is shown in Figure C.6. In the example, there are two vertical edges that need to be removed (indicated with a cross) because they come from vertical extensions that are cut off by the new segment. In both cases, the two trapezoids to the left and right of the removed edge are being merged into one trapezoid (drawn shaded).

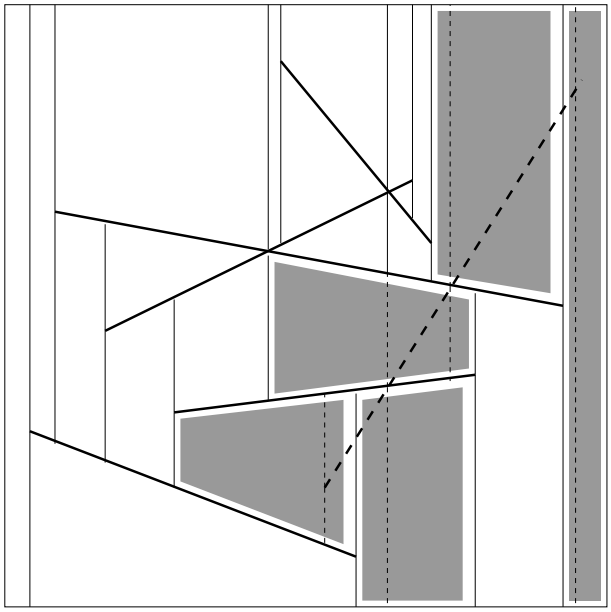


Figure C.5: *The* **Trace** *and* **Split** *steps*

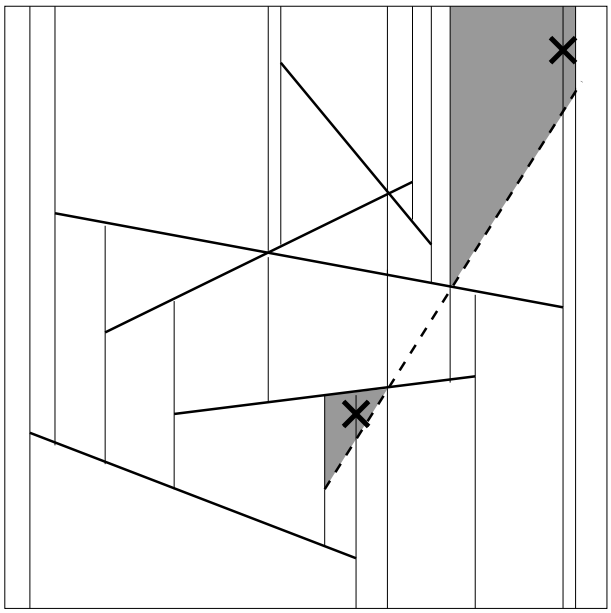


Figure C.6: *The* **Merge** *steps*

C.4 Using trapezoidal maps for point location

Recall that in the point location problem we want to preprocess a given set S of segments in order to answer subsequent point-location queries: S partitions the plane into connected *cells* and we want to know, given a query point q, to which cell q belongs, see Figure C.2.

Note that the trapezoidal map of S is a *refinement* of the partition of the plane into cells, in the sense that a cell might be partitioned into several trapezoids, but every trapezoid belongs to a single cell, see Figure C.3. Thus, once the trapezoidal map of S is constructed, we can easily "glue together" trapezoids that touch along their vertical sides, obtaining the original cells. Then we can answer point-location queries using the same routine that performs the **Find** step (whose implementation will be described below).

C.5 Analysis of the incremental construction

In order to analyze the runtime of the incremental construction, we insert the segments in random order, and we employ the configuration space framework. We also implement the **Find** step in such a way that the analysis boils down to conflict counting, just as for the Delaunay triangulation.

C.5.1 Defining The Right Configurations

Recall that a configuration space is a quadruple $\mathcal{S} = (X, \Pi, D, K)$, where X is the ground set, Π is the set of configurations, D is a mapping that assigns to each configuration its defining elements ("generators"), and K is a mapping that assigns to each configuration its conflict elements ("killers").

It seems natural to choose $X = S$, the set of segments, and to define Π as the set of all possible trapezoids that could appear in the trapezoidal map of some subset of segments. Indeed, this satisfies one important property of configuration spaces: for each configuration, the number of generators is constant.

Lemma C.2. *For every trapezoid $\square$ in the trapezoidal map of $R \subseteq S$, there exists a set $D \subseteq R$ of at most four segments, such that $\square$ is in the trapezoidal map of D.*

Proof. By our general position assumption, each non-vertical side of $\square$ is a subset of a unique segment in R, and each vertical side of $\square$ is induced by a unique (left or right) endpoint, or by the intersection of two unique segments. In the latter case, one of these segments also contributes a non-vertical side, and in the former case, we attribute the endpoint to the "topmost" segment with that (left or right) endpoint. It follows that there is a set of at most four segments whose trapezoidal map already contains $\square$. $\square$

But there is a problem with this definition of configurations. Recall that we can apply the general configuration space analysis only if

(i) the cost of updating $\mathcal{T}_{r-1}$ to $\mathcal{T}_r$ is proportional to the *structural change*, the number of configurations in $\mathcal{T}_r \setminus \mathcal{T}_{r-1}$; and

(ii) the expected cost of all **Find** operations during the randomized incremental construction is proportional to the expected number of conflicts. (This is the "conflict counting" part.)

Here we see that already (i) fails. During the **Trace** step, we traverse the boundary of each trapezoid intersected by s_r in order to find the next trapezoid. Even if s_r intersects only a small number of trapezoids (so that the structural change is small), the traversals may take very long. This is due to the fact that a trapezoid can be incident to a large number of edges. Consider the trapezoid labeled $\square$ in Figure C.7. It has many incident vertical extensions from above. Tracing a segment through such a trapezoid takes time that we cannot charge to the structural change.

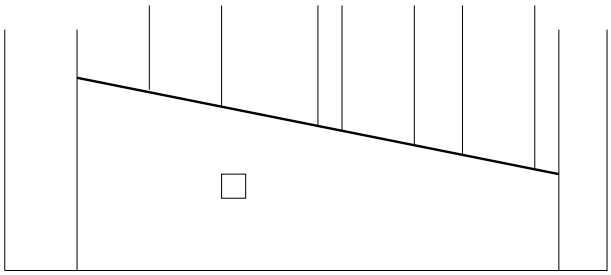


Figure C.7: *Trapezoids may have arbitrarily large complexity*

To deal with this, we slightly adapt our notion of configuration.

Definition C.3. *Let Π be the set of all trapezoids together with at most one incident vertical edge ("trapezoids with tail") that appear in the trapezoidal map of some subset of $X = S$, see Figure C.8. A trapezoid without any incident vertical edge is also considered a trapezoid with tail.*

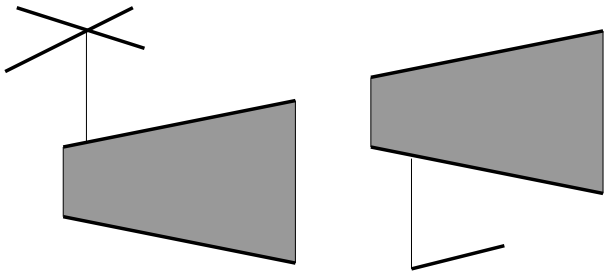


Figure C.8: *A trapezoid with tail is a trapezoid together with at most one vertical edge attached to its upper or its lower segment.*

As it turns out, we still have constantly many generators.

Lemma C.4. *For every trapezoid with tail $\square$ in the trapezoidal map of $R \subseteq S$, there exists a set $D \subseteq R$ of at most six segments, such that $\square$ is in the trapezoidal map of D.*

Proof. We already know from Lemma C.2 that the trapezoid without tail has at most 4 generators. And since the tail is induced by a unique segment endpoint or by the intersection of a unique pair of segments, the claim follows. □

Here is the complete specification of our configuration space $\mathcal{S} = (X, \Pi, D, K)$.

Definition C.5. *Let $X = S$ be the set of segments, and Π the set of all trapezoids with tail. For each trapezoid with tail $\square$, $D(\square)$ is the set of at most 6 generators. $K(\square)$ is the set of all segments that intersect $\square$ in the interior of the trapezoid, or cut off some part of the tail, or replace the topmost generator of the left or right side, see Figure C.9.*

Then $\mathcal{S} = (X, \Pi, D, K)$ is a configuration space of dimension at most 6, by Lemma C.4. The only additional property that we need to check is that $D(\square) \cap K(\square) = \emptyset$ for all trapezoids with tail, but this is clear since no generator of $\square$ properly intersects the trapezoid of $\square$ or cuts off part of its tail.

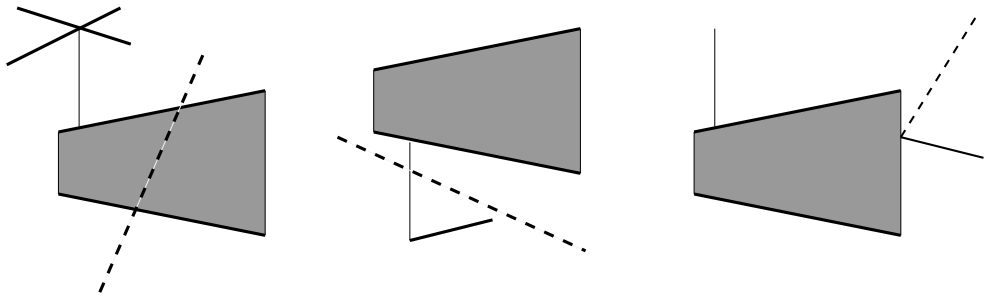


Figure C.9: *A trapezoid with tail $\square$ is in conflict with a segment s (dashed) if s intersects $\square$ in the interior of the trapezoid (left), or cuts off part of the tail (middle), or is a new topmost segment generating a vertical side.*

C.5.2 Update Cost

Now we can argue that the update cost can be bounded by the structural change. We employ the same trick as for Delaunay triangulations. We prove that the update cost is in each step $r-1 \to r$ proportional to the number of configurations that are being *destroyed*. Over the whole algorithm, we cannot destroy more configurations than we create, so the bound that we get is also a bound in terms of the overall structural change.

Lemma C.6. *In updating $\mathcal{T}_{r-1}$ to $\mathcal{T}_r$, the steps* **Trace**, **Split**, *and* **Merge** *can be performed in time proportional to the number of trapezoids with tail in $\mathcal{T}_{r-1} \setminus \mathcal{T}_r$.*

Proof. By definition, the complexity (number of edges) of a trapezoid is proportional to the number of trapezoids with tail that share this trapezoid. This means, the cost of traversing the trapezoid can be charged to the trapezoids with tail containing it, and all of them will be destroyed (this includes the trapezoids with tail that just change their left or right generator; in the configuration space, this is also a destruction). This takes care of the **Trace** step. The **Split** and **Merge** steps can be done within the same asymptotic time bounds since they can be performed by traversing the boundaries of all intersected trapezoids a constant number of times each. For efficiently doing the manipulations on the trapezoidal map, we can for example represent it using a doubly-connected edge list. □

We can now employ the general configuration space analysis to bound the *expected* structural change throughout the randomized incremental construction; as previously shown, this asymptotically bounds the expected cost of the **Trace**, **Split**, and **Merge** steps throughout the algorithm. Let us recall the general bound.

Theorem C.7. *Let $\mathcal{S} = (X, \Pi, D, K)$ be a configuration space of fixed dimension with $|X| = n$. The expected number of configurations that are created throughout the algorithm is bounded by*

$$O\left(\sum_{r=1}^{n} \frac{t_r}{r}\right),$$

where t_r is the expected size of $\mathcal{T}_r$, the expected number of active configurations after r insertion steps.

C.5.3 The History Graph

Here is how we realize the **Find** step (as well as the point-location queries for our point-location application). It is a straightforward history graph approach as for Delaunay triangulations. Every trapezoid that we ever create is a node in the history graph; whenever a trapezoid is destroyed, we add outgoing edges to its (at most four) successor trapezoids. Note that trapezoids are destroyed during the steps **Split** and **Merge**. In the latter step, every destroyed trapezoid has only one successor trapezoid, namely the one it is merged into. It follows that we can prune the nodes of the "ephemeral" trapezoids that exist only between the **Split** and **Merge** steps. What we get is a history graph of degree at most 4, such that every non-leaf node corresponds to a trapezoid in $\mathcal{T}_{r-1} \setminus \mathcal{T}_r$, for some r.

C.5.4 Cost of the Find step

We can use the history graph for point location during the **Find** step. Given a segment endpoint p, we start from the bounding box (the unique trapezoid with no generators)

that is certain to contain p. Since for every trapezoid in the history graph, its area is covered by the at most four successor trapezoids, we can simply traverse the history graph along directed edges until we reach a leaf that contains p. This leaf corresponds to the trapezoid of the current trapezoidal map containing p. By the outdegree-4-property, the cost of the traversal is proportional to the length of the path that we traverse.

Here is the crucial observation that allows us to reduce the analysis of the **Find** step to "conflict counting". Note that this is is precisely what we also did for Delaunay triangulations, except that there, we had to deal explicitly with "ephemeral" triangles.

Recall Definition B.5, according to which a *conflict* is a pair $(\square, s)$ where $\square$ is a trapezoid with tail, contained in some intermediate trapezoidal map, and $s \in K(\square)$.

Lemma C.8. *During a run of the incremental construction algorithm for the trapezoidal map, the total number of history graph nodes traversed during all* **Find** *steps is bounded by the number of conflicts during the run.*

Proof. Whenever we traverse a node (during insertion of segment s_r, say), the node corresponds to a trapezoid $\square$ (which we also consider as a trapezoid with tail) in some set $\mathcal{T}_s, s < r$, such that $p \in \square$, where p is the left endpoint of the segment s_r. We can therefore uniquely identify this edge with the conflict $(\square, s_r)$. The statement follows. $\square$

Now we can use the configuration space analysis that precisely bounds the *expected* number of conflicts, and therefore the expected cost of the **Find** steps over the whole algorithm. Let us recapitulate the bound.

Theorem C.9. *Let $\mathcal{S} = (X, \Pi, D, K)$ be a configuration space of fixed dimension d with $|X| = n$. The expected number of conflicts during randomized incremental construction of $\mathcal{T}_n$ is bounded by*

$$O\left(n \sum_{r=1}^{n} \frac{t_r}{r^2}\right),$$

where t_r is as before the expected size of $\mathcal{T}_r$.

C.5.5 Applying the General Bounds

Let us now apply Theorem C.7 and Theorem C.9 to our concrete situation of trapezoidal maps. What we obviously need to determine for that is the quantity t_r, the expected number of active configurations after r insertion steps.

Recall that the configurations are the trapezoids with tail that exist at this point. The first step is easy.

Observation C.10. *In every trapezoidal map, the number of trapezoids with tail is proportional to the number vertices.*

Proof. Every trapezoid with tail that actually has a tail can be charged to the vertex of the trapezoidal map on the "trapezoid side" of the tail. No vertex can be charged twice in this way. The trapezoids with no tail are exactly the faces of the trapezoidal map, and since the trapezoidal map is a planar graph, their number is also proportional to the number vertices. □

Using this observation, we have therefore reduced the problem of computing t_r to the problem of computing the expected number of vertices in $\mathcal{T}_r$. To count the latter, we note that every segment endpoint and every segment intersection generates 3 vertices: one at the point itself, and two where the vertical extensions hit another feature. Here, we are sweeping the 4 bounding box vertices under the rug.

Observation C.11. *In every trapezoidal map of* r *segments, the number of vertices is*

$$6r + 3k,$$

where k *is the number of pairwise intersections between the* r *segments.*

So far, we have not used the fact that we have a random insertion order, but this comes next.

Lemma C.12. *Let* K *be the total number of pairwise intersections between segments in* S*, and let* k_r *be the random variable for the expected number of pairwise intersections between the first* r *segments inserted during randomized incremental construction. Then*

$$k_r = K\frac{\binom{n-2}{r-2}}{\binom{n}{r}} = K\frac{r(r-1)}{n(n-1)}.$$

Proof. Let us consider the intersection point of two fixed segments s and s'. This intersection point appears in $\mathcal{T}_r$ if and only both s and s' are among the first r segments. There are $\binom{n}{r}$ ways of choosing the set of r segments (and all choices have the same probability); since the number of r-element sets containing s and s' is $\binom{n-2}{r-2}$, the probability for the intersection point to appear is

$$\frac{\binom{n-2}{r-2}}{\binom{n}{r}} = \frac{r(r-1)}{n(n-1)}.$$

Summing this up over all K intersection points, and using linearity of expectation, the statement follows. □

From Observation C.10, Observation C.11 and Lemma C.12, we obtain the following

Corollary C.13. *The expected number* t_r *of active configurations after* r *insertion steps is*

$$t_r = O\left(r + K\frac{r(r-1)}{n^2}\right).$$

Plugging this into Theorem C.7 and Theorem C.9, we obtain the following final

Theorem C.14. *Let S be a set of n segments in the plane, with a total of K pairwise intersections. The randomized incremental construction computes the trapezoidal map of S in time*

$$O(n \log n + K).$$

Proof. We already know that the expected update cost (subsuming steps **Trace**, **Split**, and **Merge**) is proportional to the expected overall structural change, which by Theorem C.7 is

$$O\left(\sum_{r=1}^{n} \frac{t_r}{r}\right) = O(n) + O\left(\frac{K}{n^2} \sum_{r=1}^{n} r\right) = O(n + K).$$

We further know that the expected point location cost (subsuming step **Find**) is proportional to the overall expected number of conflicts which by Theorem C.9 is

$$O\left(n \sum_{r=1}^{n} \frac{t_r}{r^2}\right) = O(n \log n) + O\left(\frac{K}{n} \sum_{r=1}^{n} 1\right) = O(n \log n + K).$$

□

C.6 Analysis of the point location

Finally, we return to the application of trapezoidal maps for point location. We make precise what we mean by saying that "point-location queries are handled in $O(\log n)$ expected time", and we prove our claim.

Lemma C.15. *Let $S = \{s_1, \ldots, s_n\}$ be any set of n segments. Then there exists a constant $c > 0$ such that, with high probability (meaning, with probability tending to 1 as $n \to \infty$), the history graph produced by the random incremental construction answers* every possible *point-location query in time at most $c \log n$.*

Note that our only randomness assumption is over the random permutation of S chosen at the beginning of the incremental construction. We do not make any randomness assumption on the given set of segments.

The proof of Lemma C.15 is by a typical application of Chernoff's bound followed by the union bound.

Recall (or please meet) Chernoff's bound:

Lemma C.16. *Let $X_1, X_2, \ldots, X_n$ be independent 0/1 random variables, and let $X = X_1 + \cdots + X_n$. Let $p_i = \Pr[X_i = 1]$, and let $\mu = E[X] = p_1 + \cdots + p_n$. Then,*

$$\Pr[X < (1-\delta)\mu] < \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^{\mu} \qquad \text{for every } 0 < \delta < 1;$$

$$\Pr[X > (1+\delta)\mu] < \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu} \qquad \text{for every } \delta > 0.$$

The important thing to note is that $e^{-\delta}/(1-\delta)^{1-\delta}$ as well as $e^{\delta}/(1+\delta)^{1+\delta}$ are strictly less than 1 for every fixed $\delta > 0$, and decrease with increasing δ.

Now back to the proof of Lemma C.15:

Proof. First note that, even though there are infinitely many possible query points, there is only a finite number of combinatorially distinct possible *queries*: If two query points lie together in every trapezoid (either both inside or both outside), among all possible trapezoids defined by segments of S, then there is no difference in querying one point versus querying the other, as far as the algorithm is concerned. Since there are $O(n^4)$ possible trapezoids (recall that each trapezoid is defined by at most four segments), there are only $O(n^4)$ queries we have to consider.

Fix a query point q. We will show that there exists a large enough constant $c > 0$, such that only with probability at most $O(n^{-5})$ does the query on q take more than $c \log n$ steps.

Let $s_1, s_2, \ldots, s_n$ be the random order of the segments chosen by the algorithm, and for $1 \leqslant r \leqslant n$ let $\mathcal{T}_r$ be the trapezoidal map generated by the first r segments. Note that for every r, the point q belongs to exactly one trapezoid of $\mathcal{T}_r$. The question is how many times the trapezoid containing q changes during the insertion of the segments, since these are exactly the trapezoids of the history graph that will be visited when we do a point-location query on q.

For $1 \leqslant r \leqslant n$, let A_r be the event that the trapezoid containing q changes from $\mathcal{T}_{r-1}$ to $\mathcal{T}_r$. What is the probability of A_r? As in Section B.3, we "run the movie backwards": To obtain $\mathcal{T}_{r-1}$ from $\mathcal{T}_r$, we delete a random segment from among $s_1, \ldots, s_r$; the probability that the trapezoid containing q in $\mathcal{T}_r$ is destroyed is at most $4/r$, since this trapezoid is defined by at most four segments. Thus, $\Pr[A_r] \leqslant 4/r$, independently of every other A_s, $s \neq r$.

For each r let X_r be a random variable equal to 1 if A_r occurs, and equal to 0 otherwise. We are interested in the quantity $X = X_1 + \cdots + X_r$. Then $\mu = \mathrm{E}[X] = \sum_{r=1}^{n} \Pr[A_r] = 4 \ln n + O(1)$. Applying Chernoff's bound with $\delta = 2$ (it is just a matter of choosing δ large enough), we get

$$\Pr[X > (1+\delta)\mu] < 0.273^{\mu} = O(0.273^{4 \ln n}) = O(n^{-5.19}),$$

so we can take our c to be anything larger than $4(1+\delta) = 12$.

Thus, for every fixed query q, the probability of a "bad event" (a permutation that results in a long query time) is $O(n^{-5})$. Since there are only $O(n^4)$ possible choices for q, by the union bound the probability of *some* q having a bad event is $O(1/n)$, which tends to zero with n. □

C.7 The trapezoidal map of a simple polygon

An important special case of the trapezoidal map is obtained when the input segments form a simple polygon; seee Figure C.10. In this case, we are mostly interested in the

part of the trapezoidal map inside the polygon, since that parts allows us to obtain a triangulation of the polygon in linear time.

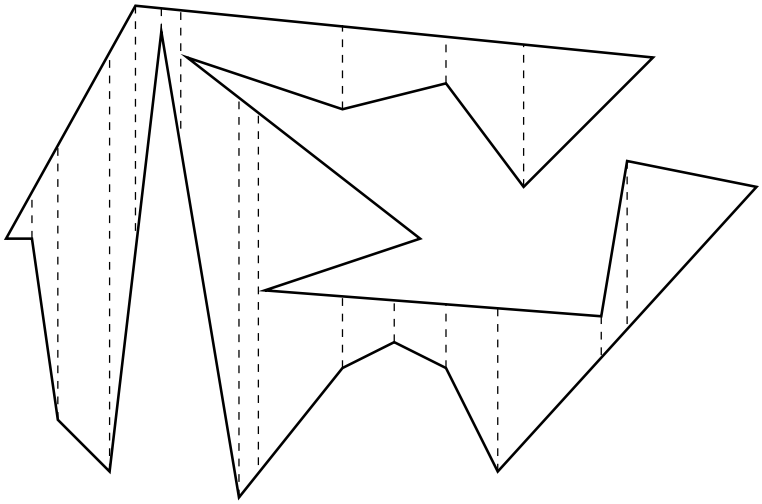


Figure C.10: *The trapezoidal map inside a simple polygon*

To get a triangulation, we first go through all trapezoids; we know that each trapezoid must have one polygon vertex on its left and one on its right boundary (due to general position, there is actually *exactly* one vertex on each of these two boundaries). Whenever the segment connecting these two vertices is not an edge of the polygon, we have a diagonal, and we insert this diagonal. Once we have done this for all trapezoids, it is easily seen that we have obtained a subdivision of the polygon into x-monotone polygons, each of which can be triangualated in linear time; see Exercise C.24. This immediately allows us to improve over the statement of Exercise 4.14.

Corollary C.17. *A simple polygon with n vertices can be triangulated in expected time $O(n \log n)$.*

Proof. By Theorem C.14, the trapezoidal decomposition induced by the segments of a simple polygon can be combuted in expected time $O(n \log n)$, since there are no intersections between the segments. Using the above linear-time triangulation algorithm from the trapezoidal map, the result follows. □

The goal of this section is to further improve this bound and show the following result.

Theorem C.18. *Let $S = \{s_1, s_2, \ldots, s_n\}$ be the set of edges of an n-vertex simple polygon, in counterclockwise order around the polygon. The trapezoidal map induced by S (and thus also a triangulation of S) can be computed in exptected time $O(n \log^* n)$.*

Informally speaking, the function $\log^* n$ is the number of times we have to iterate the operation of taking (binary) logarithms, before we get from n down to 1. Formally, we define

$$\log^{(h)}(n) = \begin{cases} n, & \text{if } h = 0 \\ \log^{(h-1)}(\log n), & \text{otherwise} \end{cases}$$

as the h-times iterated logarithm, and for $n \geqslant 1$ we set

$$\log^* n = \max\{h : \log^{(h)} n \geqslant 1\}.$$

For example, we have

$$\log^*(2^{65536}) = 5,$$

meaning that for all practical purposes, $\log^* n \leqslant 5$; a bound of $O(n \log^* n)$ is therefore very close to a linear bound.

History flattening. Recall that the bottleneck in the randomized incremental construction is the **Find** step. Using the special structure we have (the segments form a simple polygon in this order), we can speed up this step. Suppose that at some point during incremental construction, we have built the trapezoidal map of a subset of r segments, along with the history graph. We now flatten the history by removing all trapezoids that are not in $\mathcal{T}_r$, the current trapezoidal map. To allow for point location also in the future, we need an "entry point" into the flattened history, for every segment not inserted so far (the old entry point for all segments was the bounding unit square $[0,1]^2$).

Lemma C.19. *Let S be the set of edges of an n-vertex simple polygon, in counterclockwise order around the polygon. For $R \subseteq S$, let $\mathcal{T}(R)$ be the trapezoidal map induced by R. In time proportional to n plus the number of conflicts between trapezoids in $\mathcal{T}(R)$ and segments in $S \setminus R$, we can find for all segment $s \in S$ a trapezoid of $\mathcal{T}(R)$ that contains an endpoint of s.*

Proof. In a trivial manner (and in time $O(n)$), we do this for the first segment s_1 and its first endpoint p_1. Knowing the trapezoid $\square_i$ containing p_i, the first endpoint of s_i, we trace the segment s_i through $\mathcal{T}(R)$ until p_{i+1}, its other endpoint and first endpoint of s_{i+1} is found, along with the trapezoid $\square_{i+1}$ containing it. This is precisely what we also did in the **Trace** Step 2 of the incremental construction in Section C.3.

By Lemma C.6 (pretending that we are about to insert s_i), the cost of tracing s_i through $\mathcal{T}(R)$ is proportional to the size of $\mathcal{T}(R) \setminus \mathcal{T}(R \cup \{s_i\})$, equivalently, the number of conflicts between trapezoids in $\mathcal{T}(R)$ and s_i. The statement follows by adding up the costs for all i. □

This is exactly where the special structure of our segments forming a polygon helps. After locating p_i, we can locate the next endpoint p_{i+1} in time proportional to the

structural change that the insertion of s_i would cause. We completely avoid the traversal of old history trapezoids that would be necessary for an efficient location of p_{i+1} from scratch.

Next we show what Lemma C.19 gives us in expectation.

Lemma C.20. *Let S be the set of edges of an n-vertex simple polygon, in counterclockwise order around the polygon, and let $\mathcal{T}_r$ be the trapezoidal map obtained after inserting r segments in random order. In expected time $O(n)$, we can find for each segment s not inserted so far a trapezoid of $\mathcal{T}_r$ containing an endpoint of s.*

Proof. According to Lemma C.19, the expected time is bounded by $O(n + \ell(r))$, where

$$\ell(r) = \frac{1}{\binom{n}{r}} \sum_{R \subseteq S, |R|=r} \sum_{y \in S \setminus R} |\{\square \in \mathcal{T}(R) : y \in K(\square)\}|.$$

This looks very similar to the bound on the quantity $k(r)$ that we have derived in Section B.5 to count the expected number of conflicts in general configuration spaces:

$$k(r) \leqslant \frac{1}{\binom{n}{r}} \sum_{R \subseteq S, |R|=r} \frac{d}{r} \sum_{y \in S \setminus R} |\{\Delta \in \mathcal{T}(R) : y \in K(\Delta)\}|.$$

Indeed, the difference is only an additional factor of $\frac{d}{r}$ in the latter. For $k(r)$, we have then computed the bound

$$k(r) \leqslant k_1(r) - k_2(r) + k_3(r) \leqslant \frac{d}{r}(n-r)t_r - \frac{d}{r}(n-r)t_{r+1} + \frac{d^2}{r(r+1)}(n-r)t_{r+1}, \quad \text{(C.21)}$$

where t_r is the expected size of $\mathcal{T}_r$. Hence, to get a bound for $\ell(r)$, we simply have to cancel $\frac{d}{r}$ from all terms to obtain

$$\ell(r) \leqslant (n-r)t_r - (n-r)t_{r+1} + \frac{d}{r+1}(n-r)t_{r+1} = O(n),$$

since $t_r \leqslant t_{r+1} = O(r)$ in the case of nonintersecting line segments. □

The faster algorithm. We proceed as in the regular randomized incremental construction, except that we frequently and at well-chosen points flatten the history. Let us define

$$N(h) = \left\lceil \frac{n}{\log^{(h)} n} \right\rceil, \quad 0 \leqslant h \leqslant \log^* n.$$

We have $N(0) = 1, N(\log^* n) \leqslant n$ and $N(\log^* n + 1) > n$. We insert the segments in random order, but proceed in $\log^* n + 1$ *rounds*. In round $h = 1, \ldots, \log^* n + 1$, we do the following.

(i) Flatten the history graph by finding for each segment s not inserted so far a trapezoid of the current trapezoidal map containing an endpoint of s.

(ii) Insert the segments $N(h-1)$ up to $N(h)-1$ in the random order, as usual, but starting from the flat history established in (i).

In the last round, we have $N(h)-1 \geqslant n$, so we stop with segment n in the random order.

From Lemma C.20, we know that the expected cost of step (i) over all rounds is bounded by $O(n \log^* n)$ which is our desired overall bound. It remains to prove that the same bound also deals with step (ii). We do not have to worry about the overall expected cost of performing the structural changes in the trapezoidal map: this will be bounded by $O(n)$, using $t_r = O(r)$ and Theorem C.7. It remains to analyze the **Find** step, and this is where the history flattening leads to a speedup. Adapting Lemma C.8 and its proof accordingly, we obtain the following.

Lemma C.22. *During round h of the fast incremental construction algorithm for the trapezoidal map, the total number of history graph nodes traversed during all* **Find** *steps is bounded by $N(h)-N(h-1)$[1], plus the number of conflicts between trapezoids that are created during round h, and segments inserted during round h.*

Proof. The history in round h only contains trapezoids that are active at some point in round h. On the one hand, we have the "root nodes" present immediately after flattening the history, on the other hand the trapezoids that are created during the round. The term $N(h)-N(h-1)$ accounts for the traversals of the root nodes during round h. As in the proof of Lemma C.8, the traversals of other history graph nodes can be charged to the number of conflicts between trapezoids that are created during round h and segments inserted during round h. □

To count the expected number of conflicts involving trapezoids created in round h, we go back to the general configuration space framework once more. With $k_h(r)$ being the expected number of conflicts created in step r, *restricted* to the ones involving segments inserted in round h, we need to bound

$$\kappa(h) = \sum_{r=N(h-1)}^{N(h)-1} k_h(r).$$

As in (C.21), we get

$$k_h(r) \leqslant \frac{d}{r}(N(h)-r)t_r - \frac{d}{r}(N(h)-r)t_{r+1} + \frac{d^2}{r(r+1)}(N(h)-r)t_{r+1},$$

[1] this amounts to $O(n)$ throughout the algorithm and can therefore be neglected

where we have replaced n with $N(h)$, due to the fact that we do not need to consider segments in later rounds. Then $t_r = O(r)$ yields

$$\begin{aligned}
\kappa(h) &\leqslant O\left(N(h) \sum_{r=N(h-1)}^{N(h)-1} \frac{1}{r}\right) \\
&= O\left(N(h) \log \frac{N(h)}{N(h-1)}\right) \\
&= O\left(N(h) \log \frac{\log^{(h-1)} n}{\log^{(h)} n}\right) \\
&= O\left(N(h) \log^{(h)} n\right) = O(n).
\end{aligned}$$

It follows that step (ii) of the fast algorithm also requires expected linear time per round, and Theorem C.18 follows.

Exercise C.23. *a) You are given a set of n pairwise disjoint line segments. Find an algorithm to answer* vertical ray shooting *queries in $O(\log n)$ time. That is, preprocess the data such that given a query point q you can report in $O(\log n)$ time which segment is the first above q (or if there are none). Analyze the running time and the space consumption of the preprocessing.*

b) What happens if we allow intersections of the line segments? Explain in a few words how you have to adapt your solution and how the time and space complexity would change.

Exercise C.24. *Show that an n-vertex x-monotone polygon can be triangulated in time $O(n)$. (As usual a polygon is given as a list of its vertices in counterclockwise order. A polygon P is called x-monotone if for all vertical lines ℓ, the intersection $\ell \cap P$ has at most one component.)*

Questions

84. *What is the definition of a trapezoidal map?*
85. *How does the random incremental construction of a trapezoidal map proceed?* What are the main steps to be executed at each iteration?
86. *How can trapezoidal maps be used for the point location problem?*
87. *What is the configuration space framework?* Recall Section B.3.
88. *What is a naive way of defining a configuration in the case of trapezoids, and why does it fail?*
89. *What is a more successful way of defining a configuration?* Why do things work in this case?

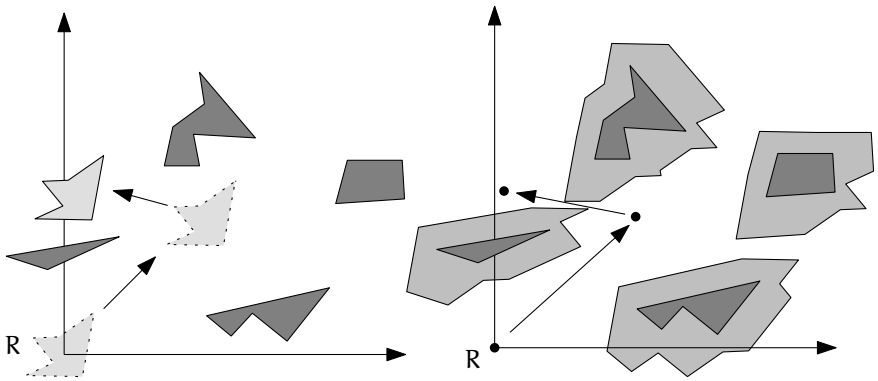
90. *What is the history graph, and how is it used to answer point location queries?*
91. *What is the performance of the random incremental construction of the trapezoidal map when used for the point-location problem?* Be precise!
92. *What probabilistic techniques are used in proving this performance bound?*
93. *How can you speed up the randomized incremental construction in the case where the input segments from a simple polygon? Sketch the necessary changes to the algorithm, and how they affect the analysis.*

Appendix D

Translational Motion Planning

In a basic instance of motion planning, a robot—modeled as a simple polygon R—moves by translation amidst a set $\mathcal{P}$ of polygonal obstacles. Throughout its motion the robot must not penetrate any of the obstacles but touching them is allowed. In other words, the interior of R must be disjoint from the obstacles at any time. Formulated in this way, we are looking at the motion planning problem in *working space* (Figure D.1a).

However, often it is useful to look at the problem from a different angle, in so-called *configuration space*. Starting point is the observation that the placement of R is fully determined by a vector $\vec{v} = (x, y) \in \mathbb{R}^2$ specifying the translation of R with respect to the origin. Hence, by fixing a *reference point* in R and considering it the origin, the robot becomes a point in configuration space (Figure D.1b). The advantage of this point of view is that it is much easier to think of a point moving around as compared to a simple polygon moving around.



(a) Working Space. **(b)** Configuration Space.

Figure D.1: *Working space and configuration space for a robot* R *and a collection of polygonal obstacles.*

The next question is: How do obstacles look like in configuration space? For an obstacle $P \in \mathcal{P}$ the set $\mathcal{C}(P) = \{\vec{v} \in \mathbb{R}^2 \mid R + \vec{v} \cap P \neq \emptyset\}$ in configuration space corresponds

to the obstacle P in the original setting. We write $R+\vec{v}$ for the *Minkowski sum* $\{\vec{r}+\vec{v} \mid \vec{r} \in R\}$. Our interest is focused on the set $\mathcal{F} = \mathbb{R}^2 \setminus \bigcup_{P \in \mathcal{P}} \mathcal{C}(P)$ of *free placements* in which the robot does not intersect any obstacle.

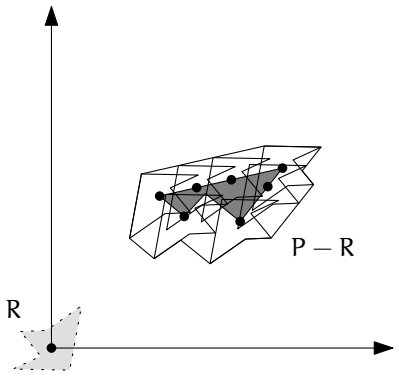


Figure D.2: *The Minkowski sum of an obstacle with an inverted robot.*

Proposition D.1. $\mathcal{C}(P) = P - R$.

Proof. $\vec{v} \in P - R \iff \vec{v} = \vec{p} - \vec{r}$, for some $\vec{p} \in P$ and $\vec{r} \in R$. On the other hand, $R + \vec{v} \cap P \neq \emptyset \iff \vec{r} + \vec{v} = \vec{p}$, for some $\vec{p} \in P$ and $\vec{r} \in R$. □

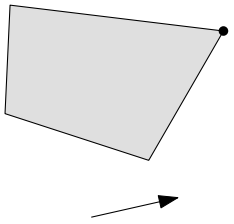
D.1 Complexity of Minkowski sums

Recall that the Minkowski sum of two point sets $P, Q \subseteq \mathbb{R}$ is defined as $P + Q = \{p + q \mid p \in P, q \in Q\}$.

Theorem D.2. *Let* P *be an* m*-vertex polygon and* Q *an* n*-vertex polygon. Then:*

1. *If both* P *and* Q *are convex, then their Minkowski sum* $P+Q$ *has at most* $m+n$ *vertices.*
2. *If* P *or* Q *is convex, then* $P + Q$ *has* $O(mn)$ *vertices.*
3. *In any case,* $P + Q$ *has* $O(m^2n^2)$ *vertices.*

Proof. The first claim can be proven using the notion of *extremal points*. If $\overrightarrow{d} = (d_x, d_y)$ is a direction in the plane and $P \subseteq \mathbb{R}^2$ is a set of points, then an *extremal point of* P *in direction* $\overrightarrow{d}$ is a point $p = (p_x, p_y) \in P$ that maximizes $p_x d_y + p_y d_x$:



It is easy to see that, if P and Q are convex polygons and $r \in P + Q$ is an extremal point of $P + Q$ in some direction $\overrightarrow{d}$, then r is the sum of some extremal points $p \in P$ and $q \in Q$ in direction $\overrightarrow{d}$. If $\overrightarrow{d}$ is chosen in "general position", then p, q, and r will be *vertices* of P, Q, and $P + Q$, respectively.

For every $\overrightarrow{d}$ in general position, let $p_{\overrightarrow{d}}$ and $q_{\overrightarrow{d}}$ be the extremal points in P and Q, respectively, in direction $\overrightarrow{d}$. Then the vertices of $P + Q$ are the precisely sums $p_{\overrightarrow{d}} + q_{\overrightarrow{d}}$ over all $\overrightarrow{d}$. But as $\overrightarrow{d}$ varies continuously and makes a full turn, the pair $(p_{\overrightarrow{d}}, q_{\overrightarrow{d}})$ can change at most $m + n$ times. This proves the first claim.

For the second claim we need the notion of *pseudodiscs*. A set $\mathcal{P} = \{P_1, \ldots, P_n\}$ of objects in the plane is called a *set of pseudodiscs* if, for every distinct i and j, the boundaries of P_i and P_j intersect in at most two points. That is, the objects "behave" the way discs behave, even though they might not be actual discs.

(Note that it makes no sense to ask whether a single object P_i is a pseudodisc; it only makes sense to ask whether a *set* of objects is a set of pseudodiscs.)

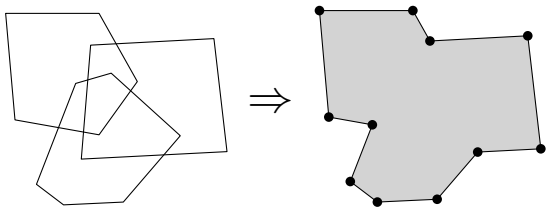
Now we need an auxiliary lemma:

Lemma D.3. *Let $\mathcal{P}$ be a set of convex objects in the plane that are pairwise interior-disjoint, and let Q be a convex object in the plane. Then the set $\mathcal{P} + Q = \{P + Q \mid P \in \mathcal{P}\}$ is a set of pseudodiscs.*

Proof. Suppose for a contradiction that the boundaries of $P_1 + Q$ and $P_2 + Q$ intersect four times, for some $P_1, P_2 \in \mathcal{P}$. This means that there are four directions $\overrightarrow{d}_1$, $\overrightarrow{d}_2$, $\overrightarrow{d}_3$, $\overrightarrow{d}_4$, in this circular order, such that P_1 is more extreme than P_2 in directions $\overrightarrow{d}_1$ and $\overrightarrow{d}_3$, while P_2 is more extreme than P_1 in directions $\overrightarrow{d}_2$ and $\overrightarrow{d}_4$. But such an alternation cannot happen since P_1 and P_2 are interior-disjoint. □

And we need a second auxiliary lemma:

Lemma D.4. *Let $\mathcal{P}$ be a set of polygonal pseudodiscs with n vertices in total. Then their union $U = \bigcup \mathcal{P}$ hast at most $2n$ vertices.*



Proof. We charge every vertex of the union U to a vertex of $\mathcal{P}$ in such a way that every vertex of $\mathcal{P}$ receives at most two charges.

Let v be a vertex of U. If v is a vertex of $\mathcal{P}$ then we simply charge v to itself. The other case is where v is the intersection point of two edges e_1, e_2, belonging to the boundaries of two distinct polygons $P_1, P_2 \in \mathcal{P}$. In such a case there must be an endpoint w of one edge e_1 or e_2 that lies inside the other polygon P_2 or P_1 (since $\mathcal{P}$ is a set of pseudodiscs). We charge v to w. It is clear that w can receive at most two charges (coming from the

two edges adjacent to w). And every vertex of a polygon in $\mathcal{P}$ that is not contained in any other polygon receives at most one charge. □

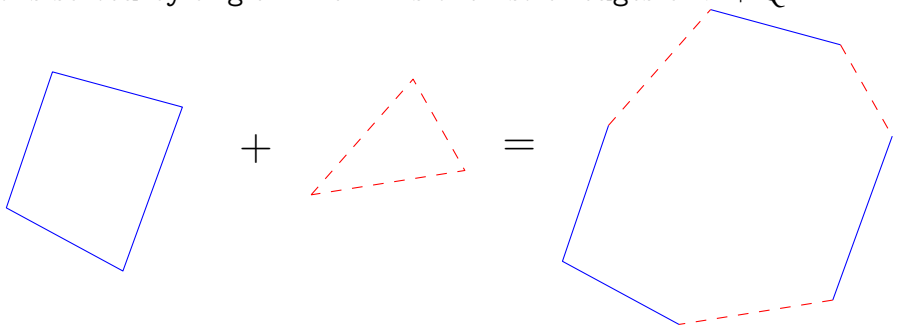
Now we are ready to prove the second claim of Theorem D.2: Suppose P is convex. Triangulate Q into $n-2$ triangles $T_1, T_2, \ldots, T_{n-2}$. Then $P+Q = \bigcup_i (P+T_i)$. By the first claim of our Theorem, each $P+T_i$ is a convex polygon with at most $m+3$ vertices. Therefore, by Lemmas D.3 and D.4, their union has at most $2(m+3)(n-2) = O(mn)$ vertices.

For the third part of our Theorem, let P and Q be arbitrary polygons. Triangulate them into triangles $S_1, \ldots, S_{m-2}$ and $T_1, \ldots, T_{n-2}$, respectively. Then $P+Q = \bigcup_{i,j}(S_i + T_j)$. Arguing as before, for every fixed i, the union $X_i = \bigcup_j (S_i + T_j)$ has at most $12(n-2)$ vertices and as many edges. Each vertex in $P+Q = \bigcup_i X_i$ is either a vertex of some X_i, or the intersection of two edges in two different X_i, X_j. There are at most $\binom{12(m-2)(n-2)}{2} = O(m^2n^2)$ vertices of the latter type. □

We leave as an exercise to show that each case of Theorem D.2 is tight in the worst case.

D.2 Minkowski sum of two convex polygons

Let P and Q be convex polygons, given as circular lists of vertices. Construct the corresponding circular lists of edges E_P and E_Q. Merge E_P and E_Q into a single list of edges E that is sorted by angle. Then E is the list of edges of $P+Q$:



Merging of two sorted lists can be done in linear time.

D.3 Constructing a single face

Theorem D.5. *Given a set S of n line segments and a point $x \in \mathbb{R}^2$, the face of $\mathcal{A}(S)$ that contains x can be constructed in $O(\lambda_3(n) \log n)$ expected time.*

Phrased in terms of translational motion planning this means the following.

Corollary D.6. *Consider a simple polygon R with k edges (robot) and a polygonal environment $\mathcal{P}$ that consists of n edges in total. The free space of all positions of R*

that can be reached by translating it without properly intersecting an obstacle from $\mathcal{P}$ *has complexity* $O(\lambda_3(kn))$ *and it can be constructed in* $O(\lambda_3(kn)\log(kn))$ *expected time.*

Below we sketch[1] a proof of Theorem D.5 using a randomized incremental construction, by constructing the trapezoidal map induced by the given set S of segments. As before, suppose without loss of generality that no two points (intersection points or endpoints) have the same x-coordinate.

In contrast to the algorithm you know, here we want to construct a single cell only, the cell that contains x. Whenever a segment closes a face, splitting it into two, we discard one of the two resulting faces and keep only the one that contains x. To detect whether a face is closed, use a disjoint-set (union-find) data structure on S. Initially, all segments are in separate components. The runtime needed for the disjoint-set data structure is $O(n\alpha(n))$, which is not a dominating factor in the bound we are heading for.

Insert the segments of S in order $s_1, \ldots, s_n$, chosen uniformly at random. Maintain (as a doubly connected edge list) the trapezoidal decomposition of the face f_i, $1 \leqslant i \leqslant n$, of the arrangement $\mathcal{A}_i$ of $\{s_1, \ldots, s_i\}$ that contains x.

As a third data structure, maintain a *history dag* (directed acyclic graph) on all trapezoids that appeared at some point during the construction. For each trapezoid, store the (at most four) segments that define it. The root of this dag corresponds to the entire plane and has no segments associated to it.

Those trapezoids that are part of the current face f_i appear as *active* leaves in the history dag. There are two more categories of vertices: Either the trapezoid was destroyed at some step by a segment crossing it; in this case, it is an interior vertex of the history dag and stores links to the (at most four) new trapezoids that replaced it. Or the trapezoid was cut off at some step by a segment that did not cross it but excluded it from the face containing x; these vertices are called *inactive* leaves and they will remain so for the rest of the construction.

Insertion of a segment s_{r+1} comprises the following steps.

1. Find the cells of the trapezoidal map f_r^* of f_r that s_{r+1} intersects by propagating s_{r+1} down the history dag.

2. Trace s_{r+1} through the active cells found in Step 1. For each split, store the new trapezoids with the old one that is replaced.

   Wherever in a split s_{r+1} connects two segments s_j and s_k, join the components of s_j and s_k in the union find data structure. If they were in the same component already, then f_r is split into two faces. Determine which trapezoids are cut off from f_{r+1} at this point by alternately exploring both components using the DCEL structure. (Start two depth-first searches one each from the two local trapezoids incident to s_{r+1}. Proceed in both searches alternately until one is finished. Mark

[1] For more details refer to the book of Agarwal and Sharir [1].

all trapezoids as discarded that are in the component that does not contain x.) In this way, the time spent for the exploration is proportional to the number of trapezoids discarded and every trapezoid can be discarded at most once.

3. Update the history dag using the information stored during splits. This is done only after all splits have been processed in order to avoid updating trapezoids that are discarded in this step.

The analysis is completely analogous to the case where the whole arrangement is constructed, except for the expected number of trapezoids created during the algorithm. Recall that any potential trapezoid τ is defined by at most four segments from S. Denote by t_r the expected number of trapezoids created by the algorithm after insertion of $s_1, \ldots, s_r$. Then in order for τ to be created at a certain step of the algorithm, one of these defining segments has to be inserted last. Therefore,

$$\Pr[\tau \text{ is created by inserting } s_r] \leqslant \frac{4}{r} \Pr[\tau \text{ appears in } f_r^*]$$

and

$$\begin{aligned}
t_r &= \sum_\tau \Pr[\tau \text{ is created in one of the first } r \text{ steps}] \\
&\leqslant \sum_\tau \sum_{i=1}^{r} \frac{4}{i} \Pr[\tau \text{ appears in } f_i^*] \\
&= \sum_{i=1}^{r} \frac{4}{i} \sum_\tau \Pr[\tau \text{ appears in } f_i^*] \\
&\overset{\text{Theorem 9.37}}{\leqslant} \sum_{i=1}^{r} \frac{4}{i} O(\lambda_3(i)) \\
&\leqslant \sum_{i=1}^{r} \frac{4}{i} ci\alpha(i) = 4c \sum_{i=1}^{r} \alpha(i) \leqslant 4cr\alpha(r) = O(\lambda_3(r)) \; .
\end{aligned}$$

Using the notation of the configuration space framework, the expected number of conflicts is bounded from above by

$$\begin{aligned}
\sum_{r=1}^{n-1} (k_1 - k_2 + k_3) &\leqslant 16(n-1) + 12n \sum_{r=1}^{n-1} \frac{\lambda_3(r+1)}{r(r+1)} \\
&\leqslant 16(n-1) + 12 \sum_{r=1}^{n-1} \frac{n}{r+1} \lambda_3(r+1) \frac{1}{r} \\
&\leqslant 16(n-1) + 12 \sum_{r=1}^{n-1} \frac{\lambda_3(n)}{r} \\
&= 16(n-1) + 12\lambda_3(n) H_{n-1} \\
&= O(\lambda_3(n) \log n) \; .
\end{aligned}$$

Exercise D.7. *Show that the Minkowski sum of two convex polygons* P *and* Q *with* m *and* n *vertices, respectively, is a convex polygon with at most* $m + n$ *edges. Give an* $O(m + n)$ *time algorithm to construct it.*

Exercise D.8. *Given an ordered set* $X = (x_1, ..., x_n)$ *and a weight function* $w : X \to \mathbb{R}^+$*, show how to construct in* $O(n)$ *time a binary search tree on* X *in which* x_k *has depth* $O(1 + \log(W/w(x_k)))$*, for* $1 \leqslant k \leqslant n$*, where* $W = \sum_{i=1}^{n} w(x_i)$*.*

Questions

94. *What is the configuration space model for (translational) motion planning, and what does it have to do with arrangements (of line segments)?* Explain the working space/configuration space duality and how to model obstacles in configuration space.
95. *What is a Minkowski sum?*
96. *What is the maximum complexity of the Minkowski sum of two polygons? What if one of them is convex? If both are convex?*
97. *How can one compute the Minkowski sum of two convex polygons in linear time?*
98. *Can one construct a single face of an arrangement (of line segments) more efficiently compared to constructing the whole arrangement?* Explain the statement of Theorem D.5 and give a rough sketch of the proof.

References

[1] Pankaj K. Agarwal and Micha Sharir, *Davenport-Schinzel sequences and their geometric applications*, Cambridge University Press, New York, NY, 1995.

Appendix E

Linear Programming

This lecture is about a special type of optimization problems, namely *linear programs*. We start with a geometric problem that can directly be formulated as a linear program.

E.1 Linear Separability of Point Sets

Let $P \subseteq \mathbb{R}^d$ and $Q \subseteq \mathbb{R}^d$ be two finite point sets in d-dimensional space. We want to know whether there exists a hyperplane that separates P from Q (we allow non-strict separation, i.e. some points are allowed to be on the hyperplane). Figure E.1 illustrates the 2-dimensional case.

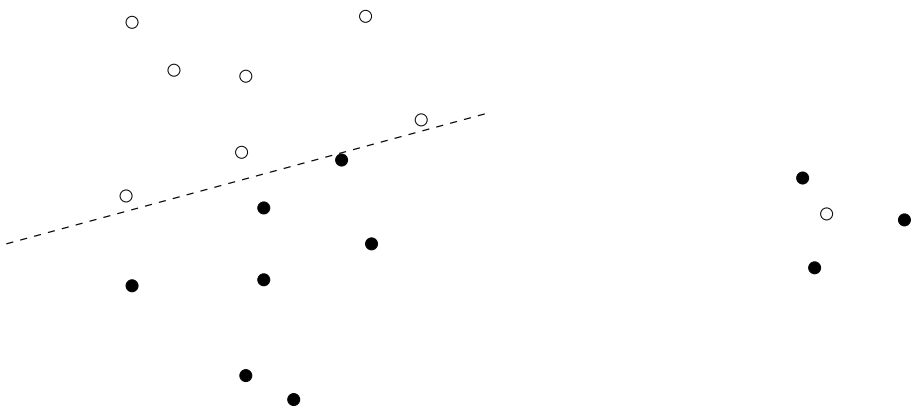


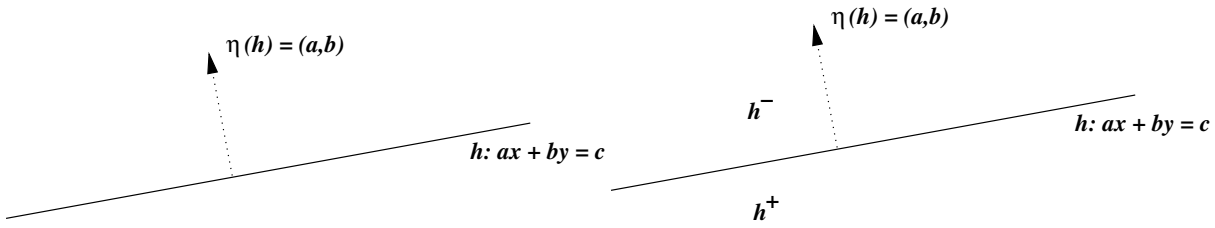
Figure E.1: *Left: there is a separating hyperplane; Right: there is no separating hyperplane*

How can we formalize this problem? A hyperplane is a set of the form

$$h = \{x \in \mathbb{R}^d : h_1x_1 + h_2x_2 + \cdots + h_dx_d = h_0\},$$

where $h_i \in \mathbb{R}, i = 0, \ldots, d$. For example, a line in the plane has an equation of the form $ax + by = c$.

The vector $\eta(h) = (h_1, h_2, \ldots, h_d) \in \mathbb{R}^d$ is called the *normal vector* of h. It is orthogonal to the hyperplane and usually visualized as in Figure E.2(a).



(a) The normal vector of a hyperplane **(b)** The two halfspaces of a hyperplane

Figure E.2: *The concepts of hyperplane, normal vector, and halfspace*

Every hyperplane h defines two closed *halfspaces*

$$\begin{aligned} h^+ &= \{x \in \mathbb{R}^d : h_1x_1 + h_2x_2 + \cdots + h_dx_d \leqslant h_0\}, \\ h^- &= \{x \in \mathbb{R}^d : h_1x_1 + h_2x_2 + \cdots + h_dx_d \geqslant h_0\}. \end{aligned}$$

Each of the two halfpsaces is the region of space "on one side" of h (including h itself). The normal vector $\eta(h)$ points into h^-, see Figure E.2(b). Now we can formally define linear separability.

Definition E.1. *Two point sets* $P \subseteq \mathbb{R}^d$ *and* $Q \subseteq \mathbb{R}^d$ *are called* linearly separable *if there exists a hyperplane* h *such that* $P \subseteq h^+$ *and* $Q \subseteq h^-$. *In formulas, if there exist real numbers* $h_0, h_1, \ldots, h_d$ *such that*

$$\begin{aligned} h_1p_1 + h_2p_2 + \cdots + h_dp_d &\leqslant h_0, \quad p \in P, \\ h_1q_1 + h_2q_2 + \cdots + h_dq_d &\geqslant h_0, \quad q \in Q. \end{aligned}$$

As we see from Figure E.1, such $h_0, h_1, \ldots, h_d$ may or may not exist. How can we find out?

E.2 Linear Programming

The problem of testing for linear separability of point sets is a special case of the general linear programming problem.

Definition E.2. *Given* $n, d \in \mathbb{N}$ *and real numbers*

$$\begin{aligned} b_i &, \quad i = 1, \ldots, n, \\ c_j &, \quad j = 1, \ldots, d, \\ a_{ij} &, \quad i = 1, \ldots, n, \quad j = 1, \ldots, d, \end{aligned}$$

the linear program *defined by these numbers is the problem of finding real numbers* $x_1, x_2, \ldots, x_d$ *such that*

(i) $\sum_{j=1}^{d} a_{ij}x_j \leqslant b_i, \quad i = 1, \ldots, n$, *and*

(ii) $\sum_{j=1}^{d} c_j x_j$ *is as large as possible subject to (i).*

Let us get a geometric intuition: each of the n *constraints* in (i) requires $x = (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d$ to be in the positive halfspace of some hyperplane. The intersection of all these halfspaces is the *feasible region* of the linear program. If it is empty, there is no solution—the linear program is called *infeasible*.

Otherwise—and now (ii) enters the picture—we are looking for a *feasible solution* x (a point inside the feasible region) that maximizes $\sum_{j=1}^{d} c_j x_j$. For every possible value γ of this sum, the feasible solutions for which the sum attains this value are contained in the hyperplane

$$\{x \in \mathbb{R}^d : \sum_{j=1}^{d} c_j x_j = \gamma\}$$

with normal vector $c = (c_1, \ldots, c_d)$. Increasing γ means to shift the hyperplane into direction c. The highest γ is thus obtained from the hyperplane that is most extreme in direction c among all hyperplanes that intersect the feasible region, see Figure E.3.

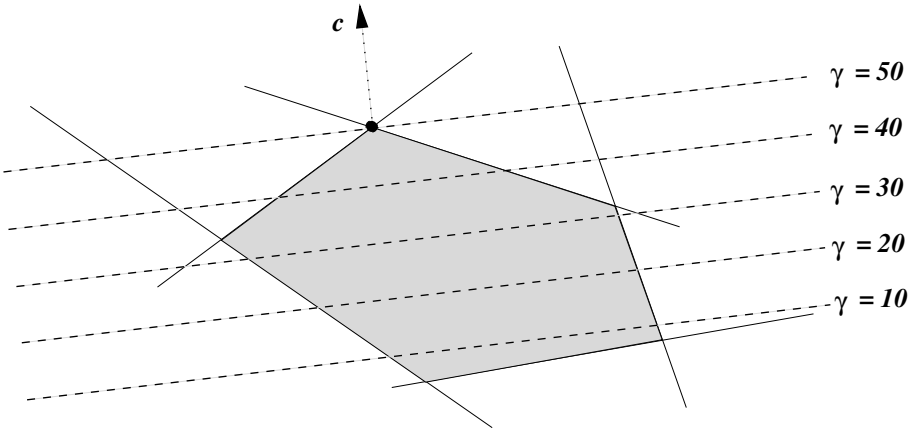


Figure E.3: *A linear program: finding the feasible solution in the intersection of five positive halfspaces that is most extreme in direction* c *(has highest value* $\gamma = \sum_{j=1}^{d} c_j x_j$*)*

In Figure E.3, we do have an *optimal solution* (a feasible solution x of highest value $\sum_{j=1}^{d} c_j x_j$), but in general, there might be feasible solutions of arbitrarily high γ-value. In this case, the linear program is called *unbounded*, see Figure E.4.

It can be shown that infeasibility and unboundedness are the only obstacles for the existence of an optimal solution. If the linear program is feasible and bounded, there exists an optimal solution.

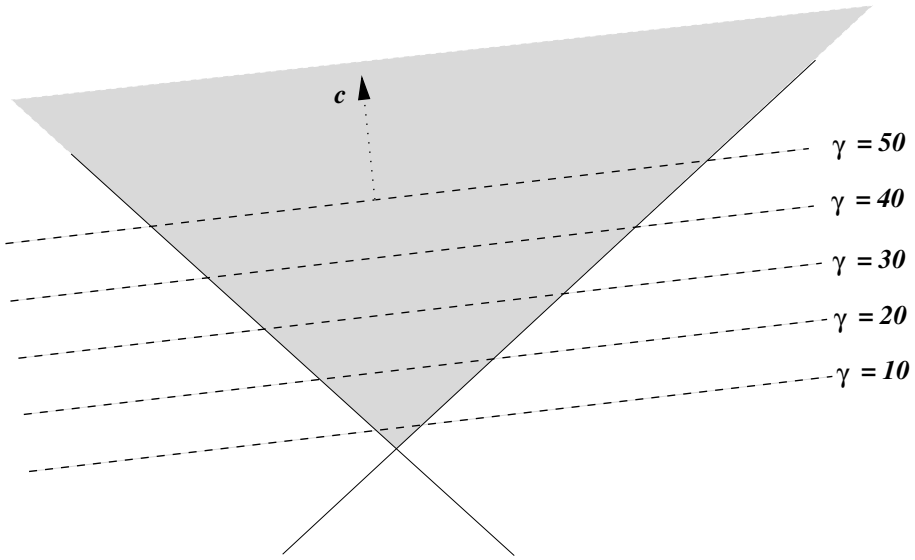


Figure E.4: *An unbounded linear program*

This is not entirely trivial, though. To appreciate the statement, consider the problem of finding a point (x, y) that (i) satisfies $y \geqslant e^x$ and (ii) has smallest value of y among all (x, y) that satisfy (i). This is not a linear program, but in the above sense it is feasible (there are (x, y) that satisfy (i)) and bounded (y is bounded below from 0 over the set of feasible solutions). Still, there is no optimal solution, since values of y arbitrarily close to 0 can be attained but not 0 itself.

Even if a linear program has an optimal solution, it is in general not unique. For example, if you rotate c in Figure E.3 such that it becomes orthogonal to the top-right edge of the feasible region, then every point of this edge is an optimal solution. Why is this called a *linear* program? Because all constraints are linear inequalities, and the objective function is a linear function. There is also a reason why it is called a linear *program*, but we won't get into this here (see [3] for more background).

Using vector and matrix notation, a linear program can succinctly be written as follows.

$$\begin{array}{lll} \text{(LP)} & \text{maximize} & c^\top x \\ & \text{subject to} & Ax \leqslant b \end{array}$$

Here, $c, x \in \mathbb{R}^d$, $b \in \mathbb{R}^n$, $A \in \mathbb{R}^{n \times d}$, and $\cdot^\top$ denotes the transpose operation. The inequality "$\leqslant$" is interpreted componentwise. The vector x represents the *variables*, c is called the *objective function vector*, b the *right-hand side*, and A the *constraint matrix*.

To *solve* a linear programs means to either report that the problem is infeasible or

unbounded, or to compute an optimal solution x^*. If we can solve linear programs, we can also decide linear separability of point sets. For this, we check whether the linear program

$$\begin{array}{ll} \text{maximize} & 0 \\ \text{subject to} & h_1 p_1 + h_2 p_2 + \cdots + h_d p_d - h_0 \leqslant 0, \quad p \in P, \\ & h_1 q_1 + h_2 q_2 + \cdots + h_d q_d - h_0 \geqslant 0, \quad q \in Q. \end{array}$$

in the $d+1$ variables $h_0, h_1, h_2, \ldots, h_d$ and objective function vector $c = 0$ is feasible or not. The fact that some inequalities are of the form "$\geqslant$" is no problem, of course, since we can multiply an inequality by -1 to turn "$\geqslant$" into "$\leqslant$".

E.3 Minimum-area Enclosing Annulus

Here is another geometric problem that we can write as a linear program, although this is less obvious. Given a point set $P \subseteq \mathbb{R}^2$, find a *minimum-area annulus* (region between two concentric circles) that contains P; see Figure E.5 for an illustration.

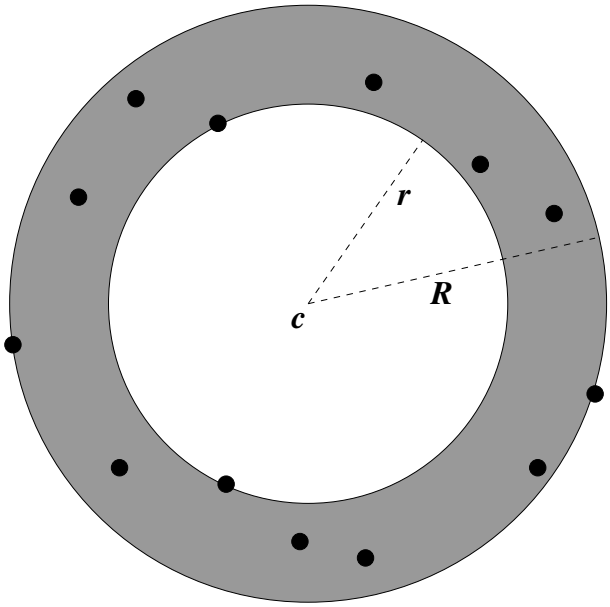


Figure E.5: *A minimum-area annulus containing* P

The optimal annulus can be used to test whether the point set P is (approximately) on a common circle which is the case if the annulus has zero (or small) area.

Let us write this as an optimization problem in the variables $c = (c_1, c_2) \in \mathbb{R}^2$ (the center) and $r, R \in \mathbb{R}$ (the small and the large radius).

$$\begin{array}{ll} \text{minimize} & \pi(R^2 - r^2) \\ \text{subject to} & r^2 \leqslant \|p - c\|^2 \leqslant R^2, \quad p \in P. \end{array}$$

This neither has a linear objective function nor are the constraints linear inequalities. But a variable substitution will take care of this. We define new variables

$$u := r^2 - \|c\|^2, \tag{E.3}$$
$$v := R^2 - \|c\|^2. \tag{E.4}$$

Omitting the factor π in the objective function does not affect the optimal solution (only its value), hence we can equivalently work with the objective function $v - u = R^2 - r^2$. The constraint $r^2 \leqslant \|p - c\|^2$ is equivalent to $r^2 \leqslant \|p\|^2 - 2p^\top c + \|c\|^2$, or

$$u + 2p^\top c \leqslant \|p\|^2.$$

In the same way, $\|p - c\| \leqslant R$ turns out to be equivalent to

$$v + 2p^\top c \geqslant \|p\|^2.$$

This means, we now have a *linear* program in the variables u, v, c_1, c_2:

$$\begin{array}{ll} \text{maximize} & u - v \\ \text{subject to} & u + 2p^\top c \leqslant \|p\|^2, \quad p \in P \\ & v + 2p^\top c \geqslant \|p\|^2, \quad p \in P. \end{array}$$

From optimal values for u, v and c, we can also reconstruct r^2 and R^2 via (E.3) and (E.4). It cannot happen that r^2 obtained in this way is negative: since we have $r^2 \leqslant \|p - c\|^2$ for all p, we could still increase u (and hence r^2 to at least 0), which is a contradicition to $u - v$ being maximal.

Exercise E.5. *a) Prove that the problem of finding a largest disk inside a convex polygon can be formulated as a linear program! What is the number of variables in your linear program?*

b) Prove that the problem of testing whether a simple polygon is starshaped can be formulated as a linear program.

Exercise E.6. *Given a simple polygon P as a list of vertices along its boundary. Describe a linear time algorithm to decide whether P is star-shaped and—if so—to construct the so-called* kernel *of P, that is, the set of all star-points.*

E.4 Solving a Linear Program

Linear programming was first studied in the 1930's -1950's, and some of its original applications were of a military nature. In the 1950's, Dantzig invented the *simplex method* for solving linear programs, a method that is fast in practice but is not known to come with any theoretical guarantees [1].

The computational complexity of solving linear programs was unresolved until 1979 when Leonid Khachiyan discovered a polynomial-time algorithm known as the *ellipsoid method* [2]. This result even made it into the New York times.

From a computational geometry point of view, linear programs with a *fixed* number of variables are of particular interest (see our two applications above, with d and 4 variables, respectively, where d may be 2 or 3 in some relavant cases). As was first shown by Megiddo, such linear programs can be solved in time $O(n)$, where n is the number of constraints [4]. In the next lecture, we will describe a much simpler randomized $O(n)$ algorithm due to Seidel [5].

Questions

99. *What is a linear program?* Give a precise definition! How can you visualize a linear program? What does it mean that the linear program is infeasible / unbounded?

100. *Show an application of linear programming!* Describe a geometric problem that can be formulated as a linear program, and give that formulation!

References

[1] George B. Dantzig, *Linear programming and extensions*, Princeton University Press, Princeton, NJ, 1963.

[2] Leonid G. Khachiyan, Polynomial algorithms in linear programming. *U.S.S.R. Comput. Math. and Math. Phys*, 20, (1980), 53–72.

[3] Jiří Matoušek and Bernd Gärtner, *Understanding and using linear programming*, Universitext, Springer, 2007.

[4] Nimrod Megiddo, Linear programming in linear time when the dimension is fixed. *J. ACM*, 31, (1984), 114–127.

[5] Raimund Seidel, Small-dimensional linear programming and convex hulls made easy. *Discrete Comput. Geom.*, 6, (1991), 423–434.

Appendix F

A randomized Algorithm for Linear Programming

Let us recall the setup from last lecture: we have a linear program of the form

$$\begin{array}{lll} \text{(LP)} & \text{maximize} & c^\top x \\ & \text{subject to} & Ax \leqslant b, \end{array} \tag{F.1}$$

where $c, x \in \mathbb{R}^d$ (there are d variables), $b \in \mathbb{R}^n$ (there are n constraints), and $A \in \mathbb{R}^{n \times d}$. The scenario that we are interested in here is that d is a (small) constant, while n tends to infinity.

The goal of this lecture is to present a randomized algorithm (due to Seidel [2]) for solving a linear program whose expected runtime is $O(n)$. The constant behind the big-O will depend exponentially on d, meaning that this algorithm is practical only for small values of d.

To prepare the ground, let us first get rid of the unboundedness issue. We add to our linear program a set of $2d$ constraints

$$-M \leqslant x_i \leqslant M, \quad i = 1, 2, \ldots d, \tag{F.2}$$

where M is a symbolic constant assumed to be larger than any real number that it is compared with. Formally, this can be done by computing with rational functions in M (quotients of polynomials of degree d in the "variable "M), rather than real numbers. The original problem is bounded if and only if the solution of the new (and bounded) problem does not depend on M. This is called the *big-M method*.

Now let H, $|H| = n$, denote the set of original constraints. For $h \in H$, we write the corresponding constraint as $a_h x \leqslant b_h$.

Definition F.3. *For $Q, R \subseteq H$, $Q \cap R = \emptyset$, let $x^*(Q, R)$ denote the lexicographically largest optimal solution of the linear program*

$$\begin{array}{lll} \textit{LP(Q,R)} & \textit{maximize} & c^\top x \\ & \textit{subject to} & a_h x \leqslant b_h, \qquad h \in Q \\ & & a_h x = b_h, \qquad h \in R \\ & & -M \leqslant x_i \leqslant M, \quad i = 1, 2, \ldots, d. \end{array}$$

If this linear program has no feasible solution, we set $x^*(Q, R) = \infty$.

What does this mean? We delete some of the original constraints (the ones not in $Q \cup R$, and we require some of the constraints to hold with equality (the ones in R). Since every linear equation $a_h x = b_h$ can be simulated by two linear inequalities $a_h x \leqslant b_h$ and $a_h x \geqslant b_h$, this again assumes the form of a linear program. By the big-M method, this linear program is bounded, but it may be infeasible. If it is feasible, there may be several optimal solutions, but choosing the lexicographically largest one leads to a unique solution $x^*(Q, R)$.

Our algorithm will compute $x^*(H, \emptyset)$, the lexicographically largest optimal solution of (F.1) subject to the additional bounding-box constraint (F.2). We also assume that $x^*(H, \emptyset) \neq \infty$, meaning that (F.1) is feasible. At the expense of solving an auxiliary problem with one extra variable, this may be assumed w.l.o.g. (Exercise).

Exercise F.4. *Suppose that you have an algorithm $\mathcal{A}$ for solving* feasible *linear programs of the form*

$$\begin{array}{lll} \textit{(LP)} & \textit{maximize} & c^\top x \\ & \textit{subject to} & Ax \leqslant b, \end{array}$$

where feasible means that there exists $\tilde{x} \in \mathbb{R}^d$ such that $A\tilde{x} \leqslant b$. Extend algorithm $\mathcal{A}$ such that it can deal with arbitrary (not necessarily feasible) linear programs of the above form.

F.1 Helly's Theorem

A crucial ingredient of the algorithm's analysis is that the optimal solution $x^*(H, \emptyset)$ is already determined by a constant number (at most d) of constraints. More generally, the following holds.

Lemma F.5. *Let $Q, R \subseteq H, Q \cap R = \emptyset$, such that the constraints in R are* independent. *This means that the set $\{x \in \mathbb{R}^d : a_h x = b_h, h \in R\}$ has dimension $d - |R|$.*

If $x^*(Q, R) \neq \infty$, then there exists $S \subseteq Q, |S| \leqslant d - |R|$ such that

$$x^*(S, R) = x^*(Q, R).$$

The proof uses *Helly's Theorem*, a classic result in convexity theory.

Theorem F.6 (Helly's Theorem [1]). *Let $C_1, \ldots C_n$ be $n \geqslant d + 1$ convex subsets of $\mathbb{R}^d$. If any $d + 1$ of the sets have a nonempty common intersection, then the common intersection of* all n *sets is nonempty.*

Even in $\mathbb{R}^1$, this is not entirely obvious. There it says that for every set of intervals with pairwise nonempty overlap there is one point contained in all the intervals. We will not prove Helly's Theorem here but just use it to prove Lemma F.5.

Proof. (**Lemma F.5**) The statement is trivial for $|Q| \leqslant d - |R|$, so assume $|Q| > d - |R|$. Let

$$L(R) := \{x \in \mathbb{R}^d : a_h x = b_h, h \in R\}$$

and

$$B := \{x \in \mathbb{R}^d : -M \leqslant x_i \leqslant M, i = 1, \ldots, d\}.$$

For a vector $x = (x_1, \ldots, x_d)$, we define $x_0 = c^\top x$, and we write $x > x'$ if $(x_0, x_1, \ldots, x_d)$ is lexicographically larger than $(x'_0, x'_1, \ldots, x'_d)$.

Let $x^* = x^*(Q, R)$ and consider the $|Q| + 1$ sets

$$C_h = \{x \in \mathbb{R}^d : a_h x \leqslant b_h\} \cap B \cap L(R), \quad h \in Q$$

and

$$C_0 = \{x \in \mathbb{R}^d : x > x^*\} \cap B \cap L(R).$$

The first observation (that may require a little thinking in case of C_0) is that all these sets are convex. The second observation is that their common intersection is empty. Indeed, any point in the common intersection would be a feasible solution $\tilde{x}$ of $LP(Q, R)$ with $\tilde{x} > x^* = x^*(Q, R)$, a contradiction to $x^*(Q, R)$ being the lexicographically largest optimal solution of $LP(Q, R)$. The third observation is that since $L(R)$ has dimension $d - |R|$, we can after an affine transformation assume that all our $|Q| + 1$ convex sets are actually convex subsets of $\mathbb{R}^{d-|R|}$.

Then, applying Helly's Theorem yields a subset of $d - |R| + 1$ constraints with an empty common intersection. Since all the C_h do have $x^*(Q, R)$ in common, this set of constraints must contain C_0. This means, there is $S \subseteq Q, |S| = d - |R|$ such that

$$x \in C_h \ \forall h \in S \quad \Rightarrow \quad x \notin C_0.$$

In particular, $x^*(S, R) \in C_h$ for all $h \in S$, and so it follows that $x^*(S, R) \leqslant x^* = x^*(Q, R)$. But since $S \subseteq Q$, we also have $x^*(S, R) \geqslant x^*(Q, R)$, and $x^*(S, R) = x^*(Q, R)$ follows. □

F.2 Convexity, once more

We need a second property of linear programs on top of Lemma F.5; it is also a consequence of convexity of the constraints, but a much simpler one.

Lemma F.7. *Let* $Q, R \subseteq H, Q \cap R \neq \emptyset$ *and* $x^*(Q, R) \neq \infty$. *Let* $h \in Q$. *If*

$$a_h x^*(Q \setminus \{h\}, R) > b_h,$$

then

$$x^*(Q, R) = x^*(Q \setminus \{h\}, R \cup \{h\}).$$

Before we prove this, let us get an intuition. The vector $x^*(Q \setminus \{h\}, R)$ is the optimal solution of $LP(Q \setminus \{h\}, R)$. And the inequality $a_h x^*(Q \setminus \{h\}, R) > b_h$ means that the constraint h is violated by this solution. The implication of the lemma is that at the optimal solution of $LP(Q, R)$, the constraint h must be satisfied with equality in which case this optimal solution is at the same time the optimal solution of the more restricted problem $LP(Q \setminus \{h\}, R \cup \{h\})$.

Proof. Let us suppose for a contradition that

$$a_h x^*(Q, R) < b_h$$

and consider the line segment s that connects $x^*(Q, R)$ with $x^*(Q \setminus \{h\}, R)$. By the previous strict inequality, we can make a small step (starting from $x^*(Q, R)$) along this line segment without violating the constraint h (Figure F.1). And since both $x^*(Q, R)$ as well as $x^*(Q \setminus \{h\}, R)$ satisfy all other constraints in $(Q \setminus \{h\}, R)$, convexity of the constraints implies that this small step takes us to a feasible solution of $LP(Q, R)$ again. But this solution is lexicographically larger than $x^*(Q, R)$, since we move towards the lexicographically larger vector $x^*(Q \setminus \{h\}, R)$; this is a contradiction. □

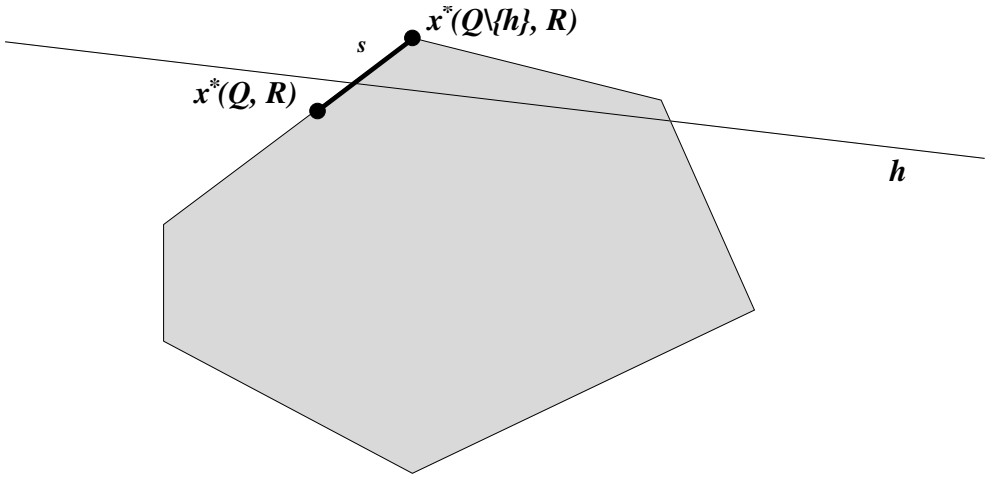


Figure F.1: *Proof of Lemma F.7*

F.3 The Algorithm

The algorithm reduces the computation of $x^*(H, \emptyset)$ to the computation of $x^*(Q, R)$ for various sets Q, R, where R is an *independent* set of constraints. Suppose you want to compute $x^*(Q, R)$ (assuming that $x^*(Q, R) \neq \infty$). If $Q = \emptyset$, this is "easy", since we have a constant-size problem, defined by R with $|R| \leqslant d$ and $2d$ bounding-box constraints $-M \leqslant x_i \leqslant M$.

Otherwise, we choose $h \in Q$ and recursively compute $x^*(Q \setminus \{h\}, R) \neq \infty$. We then check whether constraint h is violated by this solution. If not, we are done, since then $x^*(Q \setminus \{h\}, R) = x^*(Q, R)$ (Think about why!). But if h is violated, we can use Lemma F.7 to conclude that $x^*(Q, R) = x^*(Q \setminus \{h\}, R \cup \{h\})$, and we recursively compute the latter solution. Here is the complete pseudocode.

```
LP(Q, R):
    IF Q = ∅ THEN
        RETURN x*(∅, R)
    ELSE
        choose h ∈ Q uniformly at random
        x* := LP(Q \ {h}, R)
        IF a_h x* ⩽ b_h THEN
            RETURN x*
        ELSE
            RETURN LP(Q \ {h}, R ∪ {h})
        END
    END
```

To solve the original problem, we call this algorithm with $\mathcal{LP}(H, \emptyset)$. It is clear that the algorithm terminates since the first argument Q becomes smaller in every recursive call. It is also true (Exercise) that every set R that comes up during this algorithm is indeed an independent set of constraints and in particular has at most d elements. The correctness of the algorithm then follows from Lemma F.7.

Exercise F.8. *Prove that all sets R of constraints that arise during a call to algorithm $\mathcal{LP}(H, \emptyset)$ are independent, meaning that the set*

$$\{x \in \mathbb{R}^d : a_h x = b_h, h \in R\}$$

of points that satisfy all constraints in R with equality has dimension $d - |R|$.

F.4 Runtime Analysis

Now we get to the analysis of algorithm LP, and this will also reveal why the random choice is important.

We will analyze the algorithm in terms of the expected number of *violation tests* $a_h x^* \leqslant b_h$, and in terms of the expected number of *basis computations* $x^*(\emptyset, R)$ that it performs. This is a good measure, since these are the dominating operations of the algorithm. Moreover, both violation test and basis computation are "cheap" operations in the sense that they can be performed in time $f(d)$ for some f.

More specifically, a violation test can be performed in time $O(d)$; the time needed for a basis computation is less clear, since it amounts to solving a small linear program itself. Let us suppose that it is done by brute-force enumeration of all vertices of the bounded polyhedron defined by the at most $3d$ (in)equalities

$$a_h x = b_h, h \in R$$

and

$$-M \leqslant x_i \leqslant M, i = 1, \ldots, d.$$

F.4.1 Violation Tests

Lemma F.9. *Let $T(n, j)$ be the maximum expected number of violation tests performed in a call to $\mathcal{LP}(Q, R)$ with $|Q| = n$ and $|R| = d - j$. For all $j = 0, \ldots, d$,*

$$\begin{aligned} T(0, j) &= 0 \\ T(n, j) &\leqslant T(n-1, j) + 1 + \frac{j}{n} T(n-1, j-1), \quad n > 0. \end{aligned}$$

Note that in case of $j = 0$, we may get a negative argument on the right-hand side, but due to the factor $0/n$, this does not matter.

Proof. If $|Q| = \emptyset$, there is no violation test. Otherwise, we recursively call $\mathcal{LP}(Q \setminus \{h\}, R)$ for some h which requires at most $T(n-1, j)$ violation tests in expectation. Then there is one violation test within the call to $\mathcal{LP}(Q, R)$ itself, and depending on the outcome, we perform a second recursive call $\mathcal{LP}(Q \setminus \{h\}, R \cup \{h\})$ which requires an expected number of at most $T(n-1, j-1)$ violation tests. The crucial fact is that the probability of performing a second recursive call is at most j/n.

To see this, fix some $S \subseteq Q, |S| \leqslant d - |R| = j$ such that $x^*(Q, R) = x^*(S, R)$. Such a set S exists by Lemma F.5. This means, we can remove from Q all constraints except the ones in S, without changing the solution.

If $h \notin S$, we thus have

$$x^*(Q, R) = x^*(Q \setminus \{h\}, R),$$

meaning that we have already found $x^*(Q, R)$ after the first recursive call; in particular, we will then have $a_h x^* \leqslant b_h$, and there is no second recursive call. Only if $h \in S$ (and this happens with probability $|S|/n \leqslant j/n$), there can be a second recursive call. □

The following can easily be verified by induction.

Theorem F.10.

$$T(n, j) \leqslant \sum_{i=0}^{j} \frac{1}{i!} j! n.$$

Since $\sum_{i=0}^{j} \frac{1}{i!} \leqslant \sum_{i=0}^{\infty} \frac{1}{i!} = e$, we have $T(n, j) = O(j!n)$. If $d \geqslant j$ is constant, this is $O(n)$.

F.4.2 Basis Computations

To count the basis computations, we proceed as in Lemma F.9, except that the "1" now moves to a different place.

Lemma F.11. *Let* $B(n, j)$ *be the maximum expected number of basis computations performed in a call to* $\mathcal{LP}(Q, R)$ *with* $|Q| = n$ *and* $|R| = d - j$*. For all* $j = 0, \ldots, d$*,*

$$\begin{aligned} B(0, j) &= 1 \\ B(n, j) &\leqslant B(n-1, j) + \frac{j}{n} B(n-1, j-1), \quad n > 0. \end{aligned}$$

Interestingly, $B(n, j)$ turns out to be much smaller than $T(n, j)$ which is good since a basic computation is much more expensive than a violation test. Here is the bound that we get.

Theorem F.12.

$$B(n, j) \leqslant (1 + H_n)^j = O(\log^j n),$$

where H_n *is the* n*-th Harmonic number.*

Proof. This is also a simple induction, but let's do this one since it is not entirely obvious. The statement holds for $n = 0$ with the convention that $H_0 = 0$. It also holds for $j = 0$, since Lemma F.11 implies $B(n, 0) = 1$ for all n. For $n, j > 0$, we inductively obtain

$$\begin{aligned} B(n, j) &\leqslant (1 + H_{n-1})^j + \frac{j}{n}(1 + H_{n-1})^{j-1} \\ &\leqslant \sum_{k=0}^{j} \binom{j}{k} (1 + H_{n-1})^{j-k} (\frac{1}{n})^k \\ &= (1 + H_{n-1} + \frac{1}{n})^j = (1 + H_n)^j. \end{aligned}$$

The second inequality uses the fact that the terms $(1 + H_{n-1})^j$ and $\frac{j}{n}(1 + H_{n-1})^{j-1}$ are the first two terms of the sum in the second line. □

F.4.3 The Overall Bound

Putting together Theorems F.10 and F.12 (for $j = d$, corresponding to the case $R = \emptyset$), we obtain the following

Theorem F.13. *A linear program with* n *constraints in* d *variables (*d *a constant) can be solved in time* $O(n)$*.*

Questions

101. *What is Helly's Theorem?* Give a precise statement and outline the application of the theorem for linear programming (Lemma F.5).

102. *Outline an algorithm for solving linear programs*! Sketch the main steps of the algorithm and the correctness proof! Also explain how one may achieve the preconditions of feasibility and boundedness.

103. *Sketch the analysis of the algorithm!* Explain on an intuitive level how randomization helps, and how the recurrence relations for the expected number of violation tests and basis computations are derived. What is the expected runtime of the algorithm?

References

[1] Herbert Edelsbrunner, *Algorithms in combinatorial geometry*, vol. 10 of *EATCS Monographs on Theoretical Computer Science*, Springer, 1987.

[2] Raimund Seidel, Small-dimensional linear programming and convex hulls made easy. *Discrete Comput. Geom.*, 6, (1991), 423–434.

Appendix G

Smallest Enclosing Balls

This problem is related to the linear programming problem, but in a way it is much simpler, since a unique optimal solution always exists.

We let P be a set of n points in $\mathbb{R}^d$. We are interested in finding a closed ball of smallest radius that contains all the points in P, see Figure G.1.

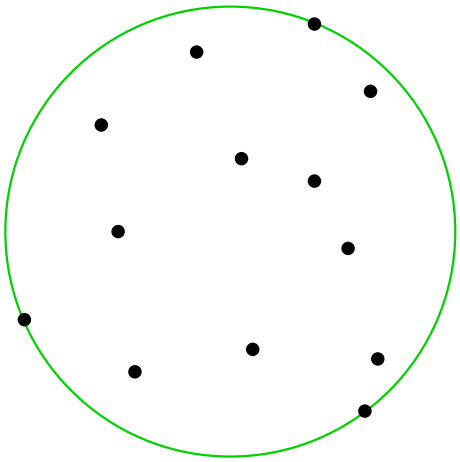


Figure G.1: *The smallest enclosing ball of a set of points in the plane*

As an "application", imagine a village that wants to build a firehouse. The location of the firehouse should be such that the maximum travel time to any house of the village is as small as possible. If we equate travel time with Euclidean distance, the solution is to place the firehouse in the center of the smallest ball that covers all houses.

Existence It is not a priori clear that a smallest ball enclosing P exists, but this follows from standard arguments in calculus. As you usually don't find this worked out in papers and textbooks, let us quickly do the argument here.

Fix P and consider the continuous function $\rho : \mathbb{R}^d \to \mathbb{R}$ defined by

$$\rho(c) = \max_{p \in P} \|p - c\|, c \in \mathbb{R}^d$$

Thus, $\rho(c)$ is the radius of the smallest ball centered at c that encloses all points of P. Let q be any point of P, and consider the closed ball

$$B = B(q, \rho(q)) := \{c \in \mathbb{R}^2 \mid \|c - q\| \leqslant \rho(q)\}.$$

Since B is compact, the function ρ attains its minimum over B at some point c_{opt}, and we claim that c_{opt} is the center of a smallest enclosing ball of P. For this, consider any center $c \in \mathbb{R}^2$. If $c \in B$, we have $\rho(c) \geqslant \rho(c_{opt})$ by optimality of c_{opt} in B, and if $c \notin B$, we get $\rho(c) \geqslant \|c - q\| > \rho(q) \geqslant \rho(c_{opt})$ since $q \in B$. In any case, we get $\rho(c) \geqslant \rho(c_{opt})$, so c_{opt} is indeed a best possible center.

Uniqueness Can it be that there are two distinct smallest enclosing balls of P? No, and to rule this out, we use the concept of *convex combinations* of balls. Let $B = B(c, \rho)$ be a closed ball with center c and radius $\rho > 0$. We define the *characteristic function* of B as the function $f_B : \mathbb{R}^2 \to R$ given by

$$f_B(x) = \frac{\|x - c\|^2}{\rho^2}, \quad x \in \mathbb{R}^2.$$

The name characteristic function comes from the following easy

Observation G.1. *For $x \in \mathbb{R}^2$, we have*

$$x \in B \quad \Leftrightarrow \quad f_B(x) \leqslant 1.$$

Now we are prepared for the convex combination of balls.

Lemma G.2. *Let $B_0 = B(c_0, \rho_0)$ and $B_1 = (c_1, \rho_1)$ be two distinct balls with characteristic functions f_{B_0} and f_{B_1}. For $\lambda \in (0, 1)$, consider the function f_λ defined by*

$$f_\lambda(x) = (1 - \lambda) f_{B_0}(x) + \lambda f_{B_1}(x).$$

Then the following three properties hold.

(i) f_λ *is the characteristic function of a ball* $B_\lambda = (c_\lambda, \rho_\lambda)$. B_λ *is called a (proper)* convex combination *of* B_0 *and* B_1*, and we simply write*

$$B_\lambda = (1 - \lambda) B_0 + \lambda B_1.$$

(ii) $B_\lambda \supseteq B_0 \cap B_1$ *and* $\partial B_\lambda \supseteq \partial B_0 \cap \partial B_1$.

(iii) $\rho_\lambda < \max(\rho_0, \rho_1)$.

A proof of this lemma requires only elementary calculations and can be found for example in the PhD thesis of Kaspar Fischer [3]. Here we will just explain what the lemma means. The family of balls $B_\lambda, \lambda \in (0,1)$ "interpolates" between the balls B_0 and B_1: while we increase λ from 0 to 1, we continuously transform B_0 into B_1. All intermediate balls B_λ "go through" the intersection of the original ball boundaries (a sphere of dimension $d-2$). In addition, each intermediate ball contains the intersection of the original balls. This is property (ii). Property (iii) means that all intermediate balls are smaller than the larger of B_0 and B_1. Figure G.2 illustrates the situation.

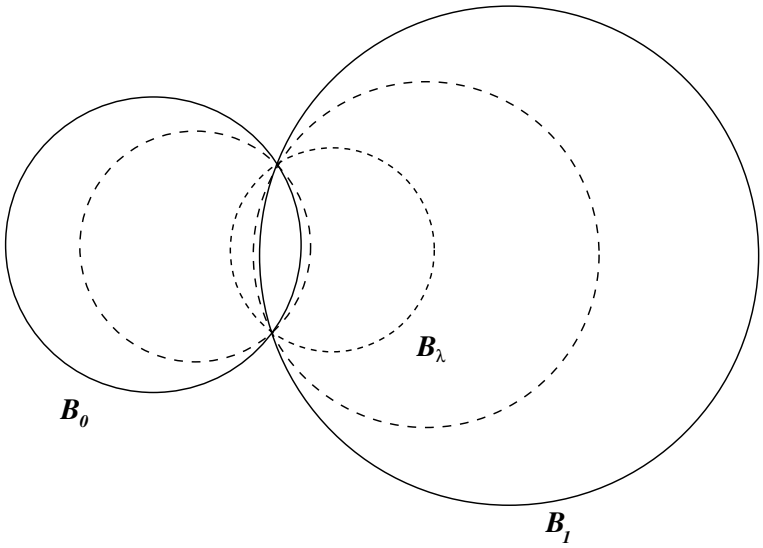


Figure G.2: *Convex combinations* B_λ *of two balls* B_0, B_1

Using this lemma, we can easily prove the following

Theorem G.3. *Given a finite point set* $P \subseteq \mathbb{R}^d$*, there exists a* unique *ball of smallest radius that contains* P*. We will denote this ball by* $B(P)$*.*

Proof. If $P = \{p\}$, the unique smallest enclosing ball is $\{p\}$. Otherwise, any smallest enclosing ball of P has positive radius ρ_{opt}. Assume there are two distinct smallest enclosing balls B_0, B_1. By Lemma G.2, the ball

$$B_{\frac{1}{2}} = \frac{1}{2}B_0 + \frac{1}{2}B_1$$

is also an enclosing ball of P (by property (ii)), but it has smaller radius than ρ_{opt} (by property (iii), a contradiction to B_0, B_1 being smallest enclosing balls. □

Bases When you look at the example of Figure G.1, you notice that only three points are essential for the solution, namely the ones on the boundary of the smallest enclosing ball. Removing all other points from P would not change the smallest enclosing ball. Even in cases where more points are on the boundary, it is always possible to find a subset of at most three points (in the $\mathbb{R}^2$ case) with the same smallest enclosing ball. This is again a consequence of *Helly's Theorem* (Theorem 5.15).

Theorem G.4. *Let $P \subseteq \mathbb{R}^d$ be a finite point set. There is a subset $S \subseteq P, |S| \leqslant d+1$ such that $B(P) = B(S)$.*

Proof. If $|P| < d+1$, we may choose $S = P$. Otherwise, let ρ_{opt} be the radius of the smallest enclosing ball $B(P)$ of $P = \{p_1, \ldots, p_n\}$. Now define

$$C_i = \{x \in \mathbb{R}^d : \|x - p_i\| < \rho_{opt}\}, \quad i = 1, \ldots, n$$

to be the open ball around p_i with radius ρ_{opt}. We know that the common intersection of all the C_i is empty, since any point in the intersection would be a center of an enclosing ball of P with radius smaller than ρ_{opt}. Moreover, the C_i are convex, so Helly's Theorem implies that there is a subset S of $d+1$ points whose C_i's also have an empty common intersection. For this set S, we therefore have no enclosing ball of radius smaller than ρ_{opt} either. Hence, $B(S)$ has radius at leat ρ_{opt}; but since $S \subseteq P$, the radius of $B(S)$ must also be at most ρ_{opt}, and hence it is equal to ρ_{opt}. But then $B(S) = B(P)$ follows, since otherwise, both $B(P)$ and $B(S)$ would be smallest enclosing balls of S, a contradiction. □

The previous theorem motivates the following

Definition G.5. *Let $P \subseteq \mathbb{R}^d$ be a finite point set. A* basis *of P is an inclusion-minimal subset $S \subseteq P$ such that $B(P) = B(S)$.*

It follows that any basis of P has size at most $d+1$. If the points are in general position (no $k+3$ on a common k-dimensional sphere), then P has a unique basis, and this basis is formed by the set of points on the boundary of $B(P)$.

G.1 The trivial algorithm

Theorem G.4 immediately implies the following (rather inefficient) algorithm for computing $B(P)$: for every subset $S \subseteq P, |S| \leqslant d+1$, compute $B(S)$ (in fixed dimension d, this can be done in constant time), and return the one with largest radius.

Indeed, this works: for all $S \subseteq P$, the radius of $B(S)$ is at most that of $B(P)$, and there must be at least one $S, |S| \leqslant d+1$ (a basis of P) with $B(S) = B(P)$. It follows that the ball $B(T)$ being returned has the same radius as $B(P)$ and is therefore equal by $T \subseteq P$.

Assuming that d is fixed, the runtime of this algorithm is

$$O\left(\sum_{i=0}^{d+1} \binom{n}{i}\right) = O(n^{d+1}).$$

If $d = 2$ (the planar case), the trivial algorithm has runtime $O(n^3)$. In the next section, we discuss an algorithm that is substantially better than the trivial one in any dimension.

In order to adapt Seidel's randomized linear programming algorithm to the problem of computing smallest enclosing balls, we need the following statements.

Exercise G.6. *(i) Let $P, R \subseteq \mathbb{R}^d, P \cap R = \emptyset$. If there exists a ball that contains P and has R on the boundary, then there is also a unique smallest such ball which we denote by $B(P, R)$.*

(ii) Let $P, R \subseteq \mathbb{R}^d, P \cap R = \emptyset$. If $B(P, R)$ exists and $p \in P$ satisfies $p \notin B(P \setminus \{p\}, R)$, then p is on the boundary of $B(P, R)$, meaning that $B(P, R) = B(P \setminus \{p\}, R \cup \{p\})$.

Prove these two statements!

G.2 Welzl's Algorithm

The idea of this algorithm is the following. Given $P \subseteq \mathbb{R}^d$, the algorithm first recursively computes $B(P \setminus \{p\})$ where $p \in P$ is chosen uniformly at random. Then there are two cases: if $p \in B(P \setminus \{p\})$ we have $B(P) = B(P \setminus \{p\})$ (it's always good to rethink why this holds) and so we are done already. If $p \notin B(P \setminus \{p\})$, we still need to work, but the key fact (that we prove below) is that in this case, p has to be on the *boundary* of $B(P)$. We can therefore recursively compute the smallest enclosing ball of $P \setminus \{p\}$ that has p on its boundary, and this is a simpler problem because it intuitively has one degree of freedom less.

Let us formalize this idea.

Definition G.7. *Let $P, R \subseteq \mathbb{R}^d$ be disjoint finite point sets. We define $B(P, R)$ as the smallest ball that contains P and has the points of R on its boundary (*if *this ball exists and is unique).*

It is not hard to see that existence cannot always be guaranteed; for example if R is contained in the convex hull of P, there can be no ball that contains P and has even a single point of R on its boundary. But the following Lemma gives a number of useful properties.

Lemma G.8. *Let $P, R \subseteq \mathbb{R}^d$ be disjoint finite point sets, where R is affinely independent.*

(i) If there is any ball that contains P and has R on its boundary, then a unique smallest such ball $B(P, R)$ exists.

(ii) If $B(P, R)$ exists and $p \notin B(P \setminus \{p\}, R)$, then p is on the boundary of $B(P, R)$, meaning that $B(P, R) = B(P \setminus \{p\}, R \cup \{p\})$.

(iii) If $B(P, R)$ exists, there is a subset $S \subseteq P$ of size $|S| \leqslant d + 1 - |R|$ such that $B(P, R) = B(S, R)$.

Before we can prove this, we need a helper lemma.

Lemma G.9. *Let $R \subseteq \mathbb{R}^d, |R| \geqslant 1$ be an affinely independent point set. Then the set*

$$C(R) := \{c \in \mathbb{R}^d \mid c \text{ is the center of a ball that has } R \text{ on its boundary}\} \tag{G.10}$$

is a linear subspace of dimension $d + 1 - |R|$.

Proof. We have $c \in C(R)$ if and only if there exists a number ρ^2 such that

$$\rho^2 = \|c - p\|^2 = c^T c - 2c^T p + p^T p, \quad p \in R. \tag{G.11}$$

Defining $\mu = \rho^2 - c^T c$, this implies

$$\mu = p^T p - 2c^T p, \quad p \in R. \tag{G.12}$$

The set of all (c, μ) satisfying the latter $|R|$ equations is a linear subspace L of R^{d+1}.

Claim. L has dimension $d + 1 - |R|$.

To see this, let us write (G.12) in matrix form as follows.

$$\begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1d} & 1 \\ p_{21} & p_{22} & \cdots & p_{2d} & 1 \\ \vdots & \vdots & & \vdots & \vdots \\ p_{|R|1} & p_{|R|2} & \cdots & p_{|R|d} & 1 \end{pmatrix} \begin{pmatrix} 2c_1 \\ 2c_2 \\ \vdots \\ 2c_d \\ \mu \end{pmatrix} = \begin{pmatrix} p_1^T p_1 \\ p_2^T p_2 \\ \cdots \\ p_{|R|}^T p_{|R|} \end{pmatrix}, \tag{G.13}$$

where $R = p_1, \ldots, p_{|R|}$. We know from linear algebra that the dimension of the solution space L is $d + 1$ minus the rank of the matrix. But this rank is $|R|$ since the rows are linearly independent by affine independence of R (we leave this easy argument to the reader, also as a good exercise to recall the definition of affine independence).

It only remains to show that $C(R)$ is also a linear subspace, and of the same dimension as L. But this holds since the linear function $f : C(R) \to L$ given by

$$f(c) = (c, p_1^T p_1 - 2c^T p_1)$$

is a bijection between $C(R)$ and L. The function is clearly injective, but it is also surjective: if $(c, \mu) \in L$, we satisfy (G.11) with $\rho^2 := \mu + c^T c$, meaning that $c \in C(R)$ and

$$f(c) = (c, p_1^T p_1 - 2c^T p_1) = (c, \|c - p_1\|^2 - c^T c) = (c, \rho^2 - c^T c) = (c, \mu).$$

□

Now we can proceed with the proof of Lemma G.8.

Proof. The proof of (i) works along the same lines as the one for $B(P)$ in Section G, and it differs only for $R \neq \emptyset$. In this case, choose $s \in R$ arbitrarily.

Let $C(P, R)$ be the set of all $c \in \mathbb{R}^d$ that are centers of balls containing P, and with R on the boundary. Note that $C(P, R)$ is closed, since it results from intersecting the linear space $C(R)$ with the closed set

$$\{c \in \mathbb{R}^d \mid \max_{p \in P} \|c - p\| \leqslant \max_{p \in R} \|p - c\|\}.$$

By assumption, the set $C(P, R)$ is nonempty. We define

$$\rho(c) = \|s - c\|, \quad c \in C(P, R).$$

Thus, $\rho(c)$ is the radius of the *unique* ball centered at c that encloses all points of P and has all points of R on the boundary. To prove that there is some smallest ball containing P and with R on the boundary, we need to show that the continuous function ρ attains a minimum over $C(P, R)$. To be able to restrict attention to a closed and bounded (hence compact) subset of $C(P, R)$, we choose some $c_0 \in C(P, R)$; then $\rho(c_0)$ is certainly an upper bound for the minimum value, meaning that any $c \in C(P, R)$ outside the compact set

$$\{c \in C(P, R) \mid \rho(c) \leqslant \rho(c_0)\}$$

canot be a candidate for the center of a smallest ball. Within this compact set, we do get a minimum c_{opt}, and this is the desired center of a smallest enclosing ball of P that has R on the boundary.

To prove uniquenes, we invoke again the convex combination of balls: Assuming that there are two smallest balls B_0, B_1, then the ball

$$\frac{1}{2}B_0 + \frac{1}{2}B_1$$

is a smaller ball that still contains P and still has R on its boundary (Lemma G.2 (ii)), a contradiction.

Now for part (ii). Again, convex combinations of balls come to our help. Consider the two balls $B(P, R)$ and $B(P \setminus \{p\}, R)$ (note that existence of the former implies existence of the latter via part (i)). If p is not on the boundary of $B(P, R)$, we have the situation of Figure G.3.

Then there is some small $\varepsilon > 0$ such that the ball

$$(1 - \varepsilon)B(P, R) + \varepsilon B(P \setminus \{p\}, R)$$

(drawn dashed in Figure G.3) still contains P and has R on the boundary, but has smaller volume than $B(P, R)$ by Lemma G.2 (iii), a contradiction.

Now we turn to (iii).

□

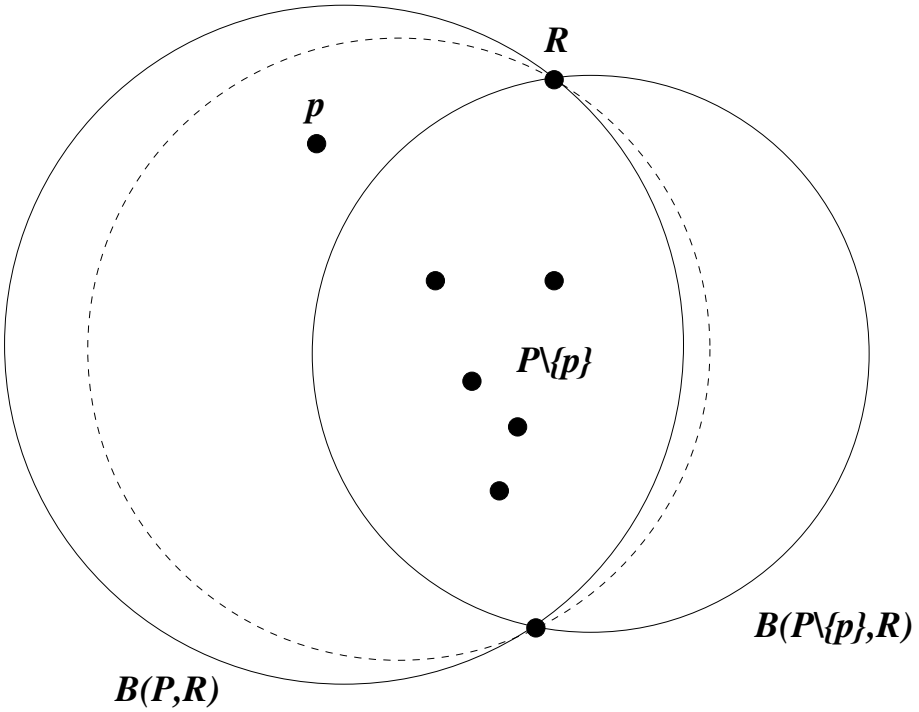


Figure G.3: *Proof of Lemma G.8(ii)*

G.3 The Swiss Algorithm

The name of this algorithm comes from the democratic way in which it works. Let us describe it for the problem of locating the firehouse in a village.

Here is how it is done the Swiss way: a meeting of all n house owners is scheduled, and every house owner is asked to put a ballot of paper with his/her name on it into a voting box. Then a constant number r (to be determined later) of ballots is drawn at random from the voting box, and the selected house owners have the right to negotiate a location for the firehouse among them. They naturally do this in a selfish way, meaning that they agree on the center of the smallest enclosing ball D of just *their* houses as the proposed location.

The house owners that were not in the selected group now fall into two classes: those that are happy with the proposal, and those that are not. Let's say that a house owner p is happy if and only if his/her house is also covered by D. In other words, p is happy if and only if the proposal would have been the same with p as an additional member of the selected group.

Now, the essence of Swiss democracy is to negotiate until everybody is happy, so as long as there are any unhappy house owners at all, the whole process is repeated. But in order to give the unhappy house owners a higher chance of influencing the outcome of the next round, their ballots in the voting box are being doubled before drawing r ballots again. Thus, there are now two ballots for each unhappy house owner, and one

for each happy one.

After round k, a house owner that has been unhappy after exactly i of the k rounds has therefore 2^i ballots in the voting box for the next round.

The obvious question is: how many rounds does it take until all house owners are happy? So far, it is not even clear that the meeting ever ends. But Swiss democracy is efficient, and we will see that the meeting actually ends after an expected number of $O(\log n)$ rounds. We will do the analysis for general dimension d (just imagine the village and its houses to lie in $\mathbb{R}^d$).

G.4 The Forever Swiss Algorithm

In the analysis, we want to argue about a fixed round k, but the above algorithm may never get to this round (for large k, we even strongly hope that it never gets there). But for the purpose of the analysis, we formally let the algorithm continue even if everybody is happy after some round (in such a round, no ballots are being doubled).

We call this extension the Forever Swiss Algorithm. A round is called *controversial* if it renders at least one house owner unhappy.

Definition G.14.

(i) Let m_k be the random variable for the total number of ballots after round k of the Forever Swiss Algorithm. We set $m_0 = n$, the initial number of ballots.

(ii) Let C_k be the event that the first k rounds in the Forever Swiss Algorithm are controversial.

A lower bound for $E(m_k)$ Let $S \subseteq P$ be a basis of P. Recall that this means that S is inclusion-minimal with $B(S) = B(P)$.

Observation G.15. *After every controversial round, there is an unhappy house owner in S.*

Proof. Let Q be the set of selected house owners in the round. Let us write $B \geqslant B'$ for two balls if the radius of B is at least the radius of B'.

If all house owners in S were happy with the outcome of the round, we would have

$$B(Q) = B(Q \cup S) \geqslant B(S) = B(P) \geqslant B(Q),$$

where the inequalities follow from the corresponding superset relations. The whole chain of inequalities would then imply that $B(P)$ and $B(Q)$ have the same radius, meaning that they must be equal (we had this argument before). But then, *nobody* would be unhappy with the round, a contradiction to the current round being controversial. □

Since $|S| \leqslant d + 1$ by Theorem G.4, we know that after k rounds, some element of S must have doubled its ballots at least $k/(d+1)$ times, given that all these rounds were controversial. This implies the following lower bound on the total number m_k of ballots.

Lemma G.16.

$$E(m_k) \geqslant 2^{k/(d+1)} \operatorname{prob}(C_k), \quad k \geqslant 0.$$

Proof. By the partition theorem of conditional expectation, we have

$$E(m_k) = E(m_k \mid C_k) \operatorname{prob}(C_k) + E(m_k \mid \overline{C_k}) \operatorname{prob}(\overline{C_k}) \geqslant 2^{k/(d+1)} \operatorname{prob}(C_k).$$

□

An upper bound for $E(m_k)$ The main step is to show that the expected increase in the number of ballots from one round to the next is bounded.

Lemma G.17. *For all $m \in \mathbb{N}$ and $k > 0$,*

$$E(m_k \mid m_{k-1} = m) \leqslant m\left(1 + \frac{d+1}{r}\right).$$

Proof. Since exactly the "unhappy ballots" are being doubled, the expected increase in the total number of ballots equals the expected number of unhappy ballots, and this number is

$$\frac{1}{\binom{m}{r}} \sum_{|R|=r} \sum_{h \notin R} [h \text{ is unhappy with } R] = \frac{1}{\binom{m}{r}} \sum_{|Q|=r+1} \sum_{h \in Q} [h \text{ is unhappy with } Q \setminus \{h\}]. \tag{G.18}$$

Claim: Every $(r+1)$-element subset Q contains at most $d+1$ ballots such that h is unhappy with $Q \setminus \{h\}$.

To see the claim, choose a basis S, $|S| \leqslant d+1$, of the ball resulting from drawing ballots in Q. Only the removal of a ballot h belonging to some house owner $p \in S$ can have the effect that Q and $Q \setminus \{h\}$ lead to different balls. Moreover, in order for this to happen, the ballot h must be the *only* ballot of the owner p. This means that at most one ballot h per owner $p \in S$ can cause h to be unhappy with $Q \setminus \{h\}$.

We thus get

$$\frac{1}{\binom{m}{r}} \sum_{|R|=r} \sum_{h \notin R} [h \text{ is unhappy with } R] \leqslant (d+1)\frac{\binom{m}{r+1}}{\binom{m}{r}} = (d+1)\frac{m-r}{r+1} \leqslant (d+1)\frac{m}{r}. \tag{G.19}$$

By adding m, we get the new expected total number $E(m_k \mid m_{k-1} = m)$ of ballots. □

From this, we easily get our actual upper bound on $E(m_k)$.

Lemma G.20.

$$E(m_k) \leqslant n\left(1 + \frac{d+1}{r}\right)^k, \quad k \geqslant 0.$$

Proof. We use induction, where the case $k = 0$ follows from $m_0 = n$. For $k > 0$, the partition theorem of conditional expectation gives us

$$\begin{aligned} E(m_k) &= \sum_{m \geqslant 0} E(m_k \mid m_{k-1} = m)\ \mathrm{prob}(m_{k-1} = m) \\ &\leqslant \left(1 + \frac{d+1}{r}\right) \sum_{m \geqslant 0} m\ \mathrm{prob}(m_{k-1} = m) \\ &= \left(1 + \frac{d+1}{r}\right) E(m_{k-1}). \end{aligned}$$

Applying the induction hypothesis to $E(m_{k-1})$, the lemma follows. □

Putting it together Combining Lemmas G.16 and G.20, we know that

$$2^{k/(d+1)}\ \mathrm{prob}(C_k) \leqslant n \left(1 + \frac{d+1}{r}\right)^k,$$

where C_k is the event that there are k or more controversial rounds.

This inequality gives us a useful upper bound on $\mathrm{prob}(C_k)$, because the left-hand side power grows faster than the right-hand side power as a function of k, given that r is chosen large enough.

Let us choose $r = c(d+1)^2$ for some constant $c > \log_2 e \approx 1.44$. We obtain

$$\mathrm{prob}(C_k) \leqslant n \left(1 + \frac{1}{c(d+1)}\right)^k / 2^{k/(d+1)} \leqslant n 2^{k \log_2 e/(c(d+1)) - k/(d+1)},$$

using $1 + x \leqslant e^x = 2^{x \log_2 e}$ for all x. This further gives us

$$\mathrm{prob}(C_k) \leqslant n\alpha^k, \tag{G.21}$$

$$\alpha = \alpha(d, c) = 2^{(\log_2 e - c)/c(d+1)} < 1.$$

This implies the following tail estimate.

Lemma G.22. *For any $\beta > 1$, the probability that the Forever Swiss Algorithm performs at least $\lceil \beta \log_{1/\alpha} n \rceil$ controversial rounds is at most*

$$1/n^{\beta - 1}.$$

Proof. The probability for at least this many controversial rounds is at most

$$\mathrm{prob}(C_{\lceil \beta \log_{1/\alpha} n \rceil}) \leqslant n\alpha^{\lceil \beta \log_{1/\alpha} n \rceil} \leqslant n\alpha^{\beta \log_{1/\alpha} n} = n n^{-\beta} = 1/n^{\beta - 1}.$$

□

In a similar way, we can also bound the expected number of controversial rounds of the Forever Swiss Algorithm. This also bounds the expected number of rounds of the Swiss Algorithm, because the latter terminates upon the first non-controversial round.

Theorem G.23. *For any fixed dimension* d*, and with* $r = \lceil \log_2 e(d+1)^2 \rceil > \log_2 e(d+1)^2$*, the Swiss algorithm terminates after an expected number of* $O(\log n)$ *rounds.*

Proof. By definition of C_k (and using $E(X) = \sum_{m \geqslant 1} \text{prob}(X \geqslant m)$ for a random variable with values in $\mathbb{N}$), the expected number of rounds of the Swiss Algorithm is

$$\sum_{k \geqslant 1} \text{prob}(C_k).$$

For any $\beta > 1$, we can use (G.21) to bound this by

$$\begin{aligned}
\sum_{k=1}^{\lceil \beta \log_{1/\alpha} n \rceil - 1} 1 + n \sum_{k=\lceil \beta \log_{1/\alpha} n \rceil}^{\infty} \alpha^k &= \lceil \beta \log_{1/\alpha} n \rceil - 1 + n \frac{\alpha^{\lceil \beta \log_{1/\alpha} n \rceil}}{1-\alpha} \\
&\leqslant \beta \log_{1/\alpha} n + n \frac{\alpha^{\beta \log_{1/\alpha} n}}{1-\alpha} \\
&= \beta \log_{1/\alpha} n + \frac{n^{-\beta+1}}{1-\alpha} \\
&= \beta \log_{1/\alpha} n + o(1).
\end{aligned}$$

□

What does this mean for $d = 2$? In order to find the location of the firehouse efficiently (meaning in $O(\log n)$ rounds), 13 ballots should be drawn in each round. The resulting constant of proportionality in the $O(\log n)$ bound will be pretty high, though. To reduce the number of rounds, it may be advisable to choose r somewhat larger.

Since a single round can be performed in time $O(n)$ for fixed d, we can summarize our findings as follows.

Theorem G.24. *Using the Swiss Algorithm, the smallest enclosing ball of a set of* n *points in fixed dimension* d *can be computed in expected time* $O(n \log n)$*.*

The Swiss algorithm is a simplification of an algorithm by Clarkson [1, 2].

The bound of the previous Theorem already compares farovably with the bound of $O(n^{d+1})$ for the trivial algorithm, see Section G.1, but it does not stop here. We can even solve the problem in expected linear time $O(n)$, by using an adaptation of Seidel's linear programming algorithm [4].

Exercise G.25. *Let* H *be a set with* n *elements and* $f : 2^H \to \mathbb{R}$ *a function that maps subsets of* H *to real numbers. We say that* $h \in H$ violates $G \subseteq H$ *if* $f(G \cup \{h\}) \neq f(G)$ *(it follows that* $h \notin G$*). We also say that* $h \in H$ *is* extreme *in* G *if* $f(G \setminus \{h\}) \neq f(G)$ *(it follows that* $h \in G$*).*

Now we define two random variables $V_r, X_r : \binom{H}{r} \to \mathbb{R}$ *where* V_r *maps an* r*-element set* R *to the number of elements that violate* R*, and* X_r *maps an* r*-element set* R *to the number of extreme elements in* R*.*

Prove the following equality for $0 \leqslant r < n$*:*

$$\frac{E(V_r)}{n-r} = \frac{E(X_{r+1})}{r+1}.$$

Exercise G.26. *Imagine instead of doubling the ballots of the unhappy house owners in the Swiss Algorithm, we would multiply their number by some integer* $t \in \mathbb{N}$*. Does the analysis of the algorithm improve (i.e., does one get a better bound on the expected number of rounds, following the same approach)?*

Exercise G.27. *We have shown that for* $d = 2$ *and sample size* $r = 13$*, the Swiss algorithm takes an expected number of* $O(\log n)$ *rounds. Compute the constants, i.e., find numbers* c_1, c_2 *such that the expected number of rounds is always bounded by* $c_1 \log_2 n + c_2$*. Try to make* c_1 *as small as possible.*

G.5 Smallest Enclosing Balls in the Manhattan Distance

We can also compute smallest enclosing balls w.r.t. distances other than the Euclidean distance. In general, if $\delta : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a metric, the smallest enclosing ball problem with respect to δ is the following.

Given $P \subseteq \mathbb{R}^d$, find $c \in \mathbb{R}^d$ and $\rho \in \mathbb{R}$ such that

$$d(c, p) \leqslant \rho, \quad p \in P,$$

and ρ is as small as possible.

For example, if $d(x, y) = \|x - y\|_\infty = \max_{i=1}^{d} |x_i - y_i|$, the problem is to find a smallest axis-parallel cube that contains all the points. This can be done in time $O(d^2 n)$ by finding the smallest enclosing box. The largest side-length of the box corresponds to the largest extent of the point set in any of the coordinate directions; to obtain a smallest enclosing cube, we simply extend the box along the other directions until all side lengths are equal.

A more interesting case is $d(x, y) = \|x-y\|_1 = \sum_{i=1}^{d} |x_i - y_i|$. This is the *Manhattan distance*. There, the problem can be written as

$$\begin{array}{ll} \text{minimize} & \rho \\ \text{subject to} & \sum_{i=1}^{d} |p_{ji} - c_i| \leqslant \rho, \quad j = 1, \ldots, n. \end{array}$$

where p_j is the j-th point and p_{ji} it's i-th coordinate Geometrically, the problem is now that of finding a smallest *cross polytope* (generalized octahedron) that contains the points. Algebraically, we can reduce it to a linear program, as follows.

We replace all $|p_{ji} - c_i|$ by new variables y_{ji} and add the additional constraints $y_{ji} \geqslant p_{ji} - c_i$ and $y_{ji} \geqslant c_i - pji$. The problem now is a linear program.

$$\begin{array}{lll} \text{minimize} & \rho & \\ \text{subject to} & \sum_{i=1}^{d} y_{ji} & \leqslant \rho, \quad j = 1, \ldots, n \\ & y_{ji} & \geqslant p_{ji} - c_i, \quad \forall i, j \\ & y_{ji} & \geqslant c_i - p_{ji}, \quad \forall i, j. \end{array}$$

The claim is that the solution to this linear program also solves the original problem. For this, we need to observe two things: first of all, every optimal solution $(\tilde{c}, \tilde{\rho})$ to the original problem induces a feasible solution to the LP with the same value (simply set $y_{ji} := |p_{ji} - \tilde{c}_i|$), so the LP solution has value equal to $\tilde{\rho}$ or better. The second is that every optimal solution $((\tilde{y}_{ji})_{i,j}, \tilde{\rho})$ to the LP induces a feasible solution to the orginal problem with the same value: by $\sum_{i=1}^{d} \tilde{y}_{ji} \leqslant \tilde{\rho}$ and $\tilde{y}_{ji} \geqslant |p_{ji} - c_i|$, we also have $\sum_{i=1}^{d} |p_{ji} - c_i| \leqslant \tilde{\rho}$. This means, the original problem has value $\tilde{\rho}$ or better. From these two observations it follows that both problems have the same optimal value ρ_{opt}, and an LP solution of this value yields a smallest enclosing ball of P w.r.t. the Manhatten distance.

Questions

104. *Formulate the Swiss Algorithm for computing smallest enclosing balls, and discuss its relation with the Forever Swiss algorithm that we employ for the analysis!*

105. *The analysis of the Forever Swiss algorithm depends on a lower and an upper bound for the expected number of ballots after k controversial rounds. Sketch how these lower and upper bounds can be obtained, and how termination of the algorithm (with high probability) can be derived from them.*

106. *What is the expected runtime of the Swiss Algorithm for computing the smallest enclosing ball of a set of n points in fixed dimension d?*

107. *How can you compute smallest enclising balls in the Manhattan metric?*

References

[1] Kenneth L. Clarkson, A Las Vegas algorithm for linear programming when the dimension is small. In *Proc. 29th Annu. IEEE Sympos. Found. Comput. Sci.*, pp. 452–456, 1988.

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

[3] Kaspar Fischer, *Smallest enclosing balls of balls. Combinatorial structure and algorithms*. Ph.D. thesis, ETH Zürich, 2005.

[4] Emo Welzl, Smallest enclosing disks (balls and ellipsoids). In H. Maurer, ed., *New Results and New Trends in Computer Science*, vol. 555 of *Lecture Notes Comput. Sci.*, pp. 359–370, Springer, 1991.

Appendix H

Epsilon Nets

H.1 Motivation

Here is our scenario for this chapter. We are given a set A of points in $\mathbb{R}^d$ and a family $\mathcal{R}$ of *ranges* $r \subseteq \mathbb{R}^d$, for example the set of all balls, halfspaces, or convex sets in $\mathbb{R}^d$. A is huge and probably not even known completely; similarly, $\mathcal{R}$ may not be accessible explicitly (in the examples above, it is an uncountable set). Still, we want to learn something about A and some $r \in \mathcal{R}$.

The situation is familiar, definitely, if we don't insist on the geometric setting. For example, let A be the set of consumers living in Switzerland, and let $\tilde{r}$ be the subset of consumers who frequently eat a certain food product, say Lindt chocolate. We have similar subsets for other food products, and together, they form the family of ranges $\mathcal{R}$.

If we want to learn something about $\tilde{r}$, e.g. the ratio $\frac{|\tilde{r}|}{|A|}$ (the fraction of consumers frequently eating Lindt chocolate), then we typically sample a subset S of A and see what portion of S lies in $\tilde{r}$. We want to believe that

$$\frac{|\tilde{r} \cap S|}{|S|} \quad \text{approximates} \quad \frac{|\tilde{r}|}{|A|},$$

and statistics tells us to what extent this is justified. In fact, consumer surveys are based on this approach: in our example, S is a sample of consumers who are being asked about their chocolate preferences. After this, the quantity $|\tilde{r} \cap S|/|S|$ is known and used to predict the "popularity" $|\tilde{r}|/|A|$ of Lindt chocolate among Swiss consumers.

In this chapter, we consider a different kind of approximation. Suppose that we are interested in the most popular food products in Switzerland, the ones which are frequently eaten by more than an ε-fraction of all consumers, for some fixed $0 \leqslant \varepsilon \leqslant 1$. The goal is to find a small subset N of consumers that "represent" all popular products. Formally, we want to find a set $N \subseteq A$ such that

$$\text{for all } r\text{:} \quad \frac{|r|}{|A|} > \varepsilon \quad \Rightarrow \quad r \cap N \neq \emptyset.$$

Such a subset is called an *epsilon net*. Obviously, $N = A$ is an epsilon net for all ε, but as already mentioned above, the point here is to have a *small* set N.

Epsilon nets are very useful in many contexts that we won't discuss here. But already in the food consumption example above, it is clear that a small representative set of consumers is a good thing to have; for example if you quickly need a statement about a particular popular food product, you know that you will find somebody in your representative set who knows the product.

The material of this chapter is classic and goes back to Haussler and Welzl [1].

H.2 Range spaces and ε-nets.

Here is the formal framework. Let X be a (possibly infinite) set and $\mathcal{R} \subseteq 2^X$. The pair $(X, \mathcal{R})$ is called a *range space*[1], with X its *points* and the elements of $\mathcal{R}$ its *ranges*.

Definition H.1. *Let $(X, \mathcal{R})$ be a range space. Given $A \subseteq X$, finite, and $\varepsilon \in \mathbb{R}$, $0 \leqslant \varepsilon \leqslant 1$, a subset N of A is called an* ε-net of A (w.r.t. $\mathcal{R}$) *if*

$$\text{for all } r \in \mathcal{R}: \quad |r \cap A| > \varepsilon |A| \quad \Rightarrow \quad r \cap N \neq \emptyset \ .$$

This definition is easy to write down, but it is not so easy to grasp, and this is why we will go through a couple of examples below. Note that we have a slightly more general setup here, compared to the motivating Section H.1 where we had $X = A$.

Examples Typical examples of range spaces in our geometric context are

- $(\mathbb{R}, \mathcal{H}_1)$ with $\mathcal{H}_1 := \{(-\infty, a] \mid a \in \mathbb{R}\} \cup \{[a, \infty) \mid a \in \mathbb{R}\}$ (*half-infinite intervals*), and
- $(\mathbb{R}, \mathcal{I})$ with $\mathcal{I} := \{[a, b] \| \, a, b \in \mathbb{R}, a \leqslant b\}$ (*intervals*),

and higher-dimensional counter-parts

- $(\mathbb{R}^d, \mathcal{H}_d)$ with $\mathcal{H}_d$ the closed *halfspaces* in $\mathbb{R}^d$ bounded by hyperplanes,
- $(\mathbb{R}^d, \mathcal{B}_d)$ with $\mathcal{B}_d$ the closed *balls* in $\mathbb{R}^d$,
- $(\mathbb{R}^d, \mathcal{S}_d)$ with $\mathcal{S}_d$ the d-dimensional *simplices* in $\mathbb{R}^d$, and
- $(\mathbb{R}^d, \mathcal{C}_d)$ with $\mathcal{C}_d$ the *convex sets* in $\mathbb{R}^d$.

[1] In order to avoid confusion: A range space is nothing else but a set system, sometimes also called hypergraph. It is the context, where we think of X as points and $\mathcal{R}$ as ranges in some geometric ambient space, that suggests the name at hand.

ε-Nets w.r.t. $(\mathbb{R}, \mathcal{H}_1)$ are particularly simple to obtain. For $A \subseteq \mathbb{R}$, $N := \{\min A, \max A\}$ is an ε-net for every ε—it is even a 0-net. That is, there are ε-nets of size 2, independent from $|A|$ and ε.

The situation gets slightly more interesting for the range space $(\mathbb{R}, \mathcal{I})$ with intervals. Given ε and A with elements

$$a_1 < a_2 < \cdots < a_n \ ,$$

we observe that an ε-net must contain at least one element from any contiguous sequence $\{a_i, a_{i+1}, \ldots, a_{i+k-1}\}$ of $k > \varepsilon n$ (i.e. $k \geqslant \lfloor \varepsilon n \rfloor + 1$) elements in A. In fact, this is a necessary and sufficient condition for ε-nets w.r.t. intervals. Hence,

$$\{a_{\lfloor \varepsilon n \rfloor + 1}, a_{2\lfloor \varepsilon n \rfloor + 2}, \ldots\}$$

is an ε-net of size[2] $\left\lfloor \frac{n}{\lfloor \varepsilon n \rfloor + 1} \right\rfloor \leqslant \lceil \frac{1}{\varepsilon} \rceil - 1$. So while the size of the net depends now on ε, it is still independent of $|A|$.

No point in a large range. Let us start with a simple exercise, showing that large ranges are easy to "catch". Assume that $|r \cap A| > \varepsilon |A|$ for some fixed r and ε, $0 \leqslant \varepsilon \leqslant 1$.

Now consider the set S obtained by drawing s elements uniformly at random from A (with replacement). We write

$$S \sim A^s,$$

indicating that S is chosen uniformly at random from the set A^s of s-element sequences over A.

What is the probability that $S \sim A^s$ fails to intersect with r, i.e. $S \cap r = \emptyset$? For $p := \frac{|r \cap A|}{|A|}$ (note $p > \varepsilon$) we get[3]

$$\text{prob}(S \cap r = \emptyset) = (1-p)^s < (1-\varepsilon)^s \leqslant e^{-\varepsilon s} \ .$$

That is, if $s = \frac{1}{\varepsilon}$ then this probability is at most $e^{-1} \approx 0.368$, and if we choose $s = \lambda \frac{1}{\varepsilon}$, then this probability decreases exponentially with λ: It is at most $e^{-\lambda}$.

For example, if $|A| = 10000$ and $|r \cap A| > 100$ (r contains more than 1% of the points in A), then a sample of 300 points is disjoint from r with probability at most $e^{-3} \approx 0.05$.

Smallest enclosing balls. Here is a potential use of this for a geometric problem. Suppose A is a set of n points in $\mathbb{R}^d$, and we want to compute the smallest enclosing ball of A. In fact, we are willing to accept some mistake, in that, for some given ε, we want a small ball that contains all but at most εn points from A. So let's choose a sample S of $\lambda \frac{1}{\varepsilon}$ points drawn uniformly (with replacement) from A and compute the smallest enclosing

[2]The number L of elements in the set is the largest ℓ such that $\ell(\lfloor \varepsilon n \rfloor + 1) \leqslant n$, hence $L = \left\lfloor \frac{n}{\lfloor \varepsilon n \rfloor + 1} \right\rfloor$. Since $\lfloor \varepsilon n \rfloor + 1 > \varepsilon n$, we have $\frac{n}{\lfloor \varepsilon n \rfloor + 1} < \frac{1}{\varepsilon}$, and so $L < \frac{1}{\varepsilon}$, i.e. $L \leqslant \lceil \frac{1}{\varepsilon} \rceil - 1$.

[3]We make use of the inequality $1 + x \leqslant e^x$ for all $x \in \mathbb{R}$.

ball B of S. Now let $r := \mathbb{R}^d \setminus B$, the complement of B in $\mathbb{R}^d$, play the role of the range in the analysis above. Obviously $r \cap S = \emptyset$, so it is unlikely that $|r \cap A| > \varepsilon|A|$, since—if so—the probability of $S \cap r = \emptyset$ was at most $e^{-\lambda}$.

It is important to understand that this was complete nonsense!

For the probabilistic analysis above we have to first choose r and then draw the sample—and not, as done in the smallest ball example, first draw the sample and then choose r *based on the sample*. That cannot possibly work, since we could always choose r simply as the complement $\mathbb{R}^d \setminus S$—then clearly $r \cap S = \emptyset$ and $|r \cap A| > \varepsilon|A|$, unless $|S| \geqslant (1-\varepsilon)|A|$.

While you hopefully agree on this, you might find the counterargument with $r = \mathbb{R}^d \setminus S$ somewhat artificial, e.g. complements of balls cannot be that selective in 'extracting' points from A. It is exactly the purpose of this chapter to understand to what extent this advocated intuition is justified or not.

H.3 Either almost all is needed or a constant suffices.

Let us reveal the spectrum of possibilities right away, although its proof will have to await some preparatory steps.

Theorem H.2. *Let $(X, \mathcal{R})$ be an infinite range space. Then one of the following two statements holds.*

(1) For every $n \in \mathbb{N}$ there is a set $A_n \subseteq X$ with $|A_n| = n$ such that for every ε, $0 \leqslant \varepsilon \leqslant 1$, an ε-net must have size at least $(1-\varepsilon)n$.

(2) There is a constant δ depending on $(X, \mathcal{R})$, such that for every finite $A \subseteq X$ and every ε, $0 < \varepsilon \leqslant 1$, there is an ε-net of A w.r.t. $\mathcal{R}$ of size at most $\frac{8\delta}{\varepsilon} \log_2 \frac{4\delta}{\varepsilon}$ (independent of the size of A).

That is, either we have always ε-nets of size independent of $|A|$, or we have to do the trivial thing, namely choosing all but εn points for an ε-net. Obviously, the range spaces $(\mathbb{R}, \mathcal{H}_1)$ and $(\mathbb{R}, \mathcal{I})$ fall into category (2) of the theorem.

For an example for (1), consider $(\mathbb{R}^2, \mathcal{C}_2)$. For any $n \in \mathbb{N}$, let A_n be a set of n points in convex position. For every $N \subseteq A_n$ there is a range $r \in \mathcal{C}_2$, namely the convex hull of $A_n \setminus N$, such that $A_n \cap r = A_n \setminus N$ (hence, $r \cap N = \emptyset$); see Figure H.1. Therefore, $N \subseteq A_n$ cannot be an ε-net of A_n w.r.t. $\mathcal{C}_2$ if $|A_n \setminus N| = n - |N| > \varepsilon n$. Consequently, an ε-net must contain at least $n - \varepsilon n = (1-\varepsilon)n$ points.[4]

So what distinguishes $(\mathbb{R}^2, \mathcal{C}_2)$ from $(\mathbb{R}, \mathcal{H}_1)$ and $(\mathbb{R}, \mathcal{I})$? And which of the two cases applies to the many other range spaces we have listed above? Will all of this eventually tell us something about our attempt of computing a small ball covering all but at most εn out of n given points? This and more should be clear by the end of this chapter.

[4] If we were satisfied with any abstract example for category (1), we could have taken $(X, 2^X)$ for any infinite set X.

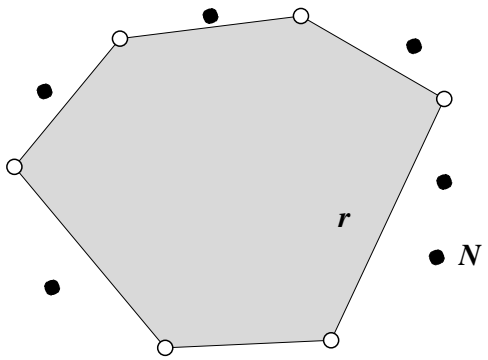


Figure H.1: *If $\mathcal{R}$ consists of all convex sets in the plane, then only trivial epsilon nets exist: for every subset N (black points) of a set A_n in convex position, the range $r = \mathrm{conv}(A_n \setminus N)$ fails to intersect N.*

H.4 What makes the difference: VC-dimension

Given a range space $(X, \mathcal{R})$ and $A \subseteq X$, we let

$$\mathcal{R}|_A := \{r \cap A \mid r \in \mathcal{R}\} ,$$

the *projection of $\mathcal{R}$ to A*. Even if $\mathcal{R}$ is infinite, $\mathcal{R}|_A$ is always of size at most 2^n if A is an n-element set. The significance of projections in our context becomes clear if we rewrite Definition H.1 in terms of projections: $N \subseteq A$ is an ε-net if

$$\text{for all } r \in \mathcal{R}|_A\text{:} \quad |r| > \varepsilon|A| \quad \Rightarrow \quad r \cap N \neq \emptyset.$$

All of a sudden, the conditions for an ε-net have become discrete, and they only depend on the finite range space $(A, \mathcal{R}|_A)$.

Note that, for A a set of n points in convex position in the plane, $\mathcal{C}_2|_A = 2^A$; we get every subset of A by an intersection with a convex set (this is also the message of Figure H.1). That is $|\,\mathcal{C}_2|_A\,| = 2^n$, the highest possible value.

For A a set of n points in $\mathbb{R}$, we can easily see that[5] $|\,\mathcal{I}|_A\,| = \binom{n+1}{2} + 1 = O(n^2)$. A similar argument shows that $|\,\mathcal{H}_1|_A\,| = 2n$. Now comes the crucial definition.

[5]Given A as $a_1 < a_2 \cdots < a_n$ we can choose another $n+1$ points b_i, $0 \leqslant i \leqslant n$, such that

$$b_0 < a_1 < b_1 < a_2 < b_2 < \cdots b_{n-1} < a_n < b_n \ .$$

Each nonempty intersection of A with an interval can be uniquely written as $A \cap [b_i, b_j]$ for $0 \leqslant i < j \leqslant n$. This gives $\binom{n+1}{2}$ plus one for the empty set.

Definition H.3. *Given a range space* $(X, \mathcal{R})$*, a subset* A *of* X *is* shattered by $\mathcal{R}$ *if* $\mathcal{R}|_A = 2^A$*. The* VC-dimension[6] of $(X, \mathcal{R})$, $\mathrm{VCdim}(X, \mathcal{R})$*, is the cardinality (possibly infinite) of the largest subset of* X *that is shattered by* $\mathcal{R}$*. If no set is shattered (i.e. not even the empty set which means that* $\mathcal{R}$ *is empty), we set the VC-dimension to* -1*.*

We had just convinced ourselves that $(\mathbb{R}^2, \mathcal{C}_2)$ has arbitrarily large sets that can be shattered. Therefore, $\mathrm{VCdim}(\mathbb{R}^2, \mathcal{C}_2) = \infty$.

Consider now $(\mathbb{R}, \mathcal{I})$. Two points $A = \{a, b\}$ can be shattered, since for each of the 4 subsets, $\emptyset$, $\{a\}$, $\{b\}$, and $\{a, b\}$, of A, there is an interval that generates that subset by intersection with A. However, for $A = \{a, b, c\}$ with $a < b < c$ there is no interval that contains a and c but not b. Hence, $\mathrm{VCdim}(\mathbb{R}, \mathcal{I}) = 2$.

Exercise H.4. *What is* $\mathrm{VCdim}(\mathbb{R}, \mathcal{H}_1)$*?*

Exercise H.5. *Prove that if* $\mathrm{VCdim}(X, \mathcal{R}) = \infty$*, then we are in case (1) of Theorem H.2, meaning that only trivial epsilon nets always exist.*

The size of projections for finite VC-dimension. Here is the (for our purposes) most important consequence of finite VC dimension: there are only polynomially many ranges in every projection.

Lemma H.6 (Sauer's Lemma). *If* $(X, \mathcal{R})$ *is a range space of finite VC-dimension at most* δ*, then*

$$|\mathcal{R}|_A| \leqslant \Phi_\delta(n) := \sum_{i=0}^{\delta} \binom{n}{i}$$

for all $A \subseteq X$ *with* $|A| = n$*.*

Proof. First let us observe that $\Phi : \mathbb{N}_0 \cup \{-1\} \times \mathbb{N}_0 \to \mathbb{N}_0$ is defined by the recurrence[7]

$$\Phi_\delta(n) = \begin{cases} 0 & \delta = -1, \\ 1 & n = 0 \text{ and } \delta \geqslant 0, \text{ and} \\ \Phi_\delta(n-1) + \Phi_{\delta-1}(n-1) & \text{otherwise.} \end{cases}$$

Second, we note that the VC-dimension cannot increase by passing from $(X, \mathcal{R})$ to a projection (A, R), $R := \mathcal{R}|_A$. Hence, it suffices to consider the finite range space (A, R)—which is of VC-dimension at most δ—and show $|R| \leqslant \Phi_\delta(n)$ (since Φ is monotone in δ).

[6] 'VC' in honor of the Russian statisticians V. N. Vapnik and A. Ya. Chervonenkis, who discovered the crucial role of this parameter in the late sixties.

[7] We recall that the binomial coefficients $\binom{n}{k}$ (with $k, n \in \mathbb{N}_0$) satisfy the recurrence $\binom{n}{k} = 0$ if $n < k$, $\binom{n}{0} = 1$, and $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$.

Now we proceed to a proof by induction of this inequality. If $A = \emptyset$ or $R = \emptyset$ the statement is trivial. Otherwise, we consider the two 'derived' range spaces for some fixed $x \in A$:

$$(A \setminus \{x\}, R - x), \quad \text{with } R - x := \{r \setminus \{x\} \mid r \in R\}$$

(note $R - x = R|_{A\setminus\{x\}}$) and

$$(A \setminus \{x\}, R^{(x)}), \quad \text{with } R^{(x)} := \{r \in R \mid x \notin r, r \cup \{x\} \in R\}.$$

Observe that the ranges in $R^{(x)}$ are exactly those ranges in $R - x$ that have two preimages under the map

$$R \ni \quad r \quad \mapsto \quad r \setminus \{x\} \quad \in R - x\,,$$

all other ranges have a unique preimage. Consequently,

$$|R| = |R - x| + |R^{(x)}|\,.$$

We have $|R - x| \leqslant \Phi_\delta(n-1)$. If $A' \subseteq A\setminus\{x\}$ is shattered by $R^{(x)}$, then $A' \cup \{x\}$ is shattered by R. Hence, $(A \setminus \{x\}, R^{(x)})$ has VC-dimension at most $\delta - 1$ and $|R^{(x)}| \leqslant \Phi_{\delta-1}(n-1)$. Summing up, it follows that

$$|R| \leqslant \Phi_\delta(n-1) + \Phi_{\delta-1}(n-1) = \Phi_\delta(n)$$

which yields the assertion of the lemma. □

In order to see that the bound given in the lemma is tight, consider the range space

$$\left(X, \bigcup_{i=0}^{\delta} \binom{X}{i}\right).$$

Obviously, a set of more than δ elements cannot be shattered (hence, the VC-dimension is at most δ), and for any finite $A \subseteq X$, the projection of the ranges to A is $\bigcup_{i=0}^{\delta} \binom{A}{\delta}$—with cardinality $\Phi_\delta(|A|)$.

We note that a rough, but for our purposes good enough estimate for Φ is given by[8]

$$\Phi_\delta(n) \leqslant n^\delta \quad \text{for } \delta \geqslant 2.$$

We have seen now that the maximum possible size of projections either grows exponentially (2^n in case of infinite VC-dimension) or it is bounded by a polynomial n^δ in case of finite VC-dimension δ). The latter is the key to the existence of small ε-nets. Before shedding light on this, let us better understand when the VC-dimension is finite.

[8] A better estimate, at least for $\delta \geqslant 3$, is given by $\Phi_\delta(n) < \left(\frac{e\,n}{\delta}\right)^d$ for all $n, d \in \mathbb{N}$, $d \leqslant n$.

H.5 VC-dimension of Geometric Range Spaces

Halfspaces. Let us investigate the VC-dimension of $(\mathbb{R}^2, \mathcal{H}_2)$. It is easily seen that three points in the plane can be shattered by halfplanes, as long a they do not lie on a common line. Hence, the VC-dimension is at least 3. Now consider 4 points. If three of them lie on a common line, there is no way to separate the middle point on this line from the other two by a halfplane. So let us assume that no three points lie on a line. Either three of them are vertices of a triangle that contains the fourth point—then we cannot possibly separate the fourth point from the remaining three points by a halfplane. Or the four points are vertices of a convex quadrilateral—then there is no way of separating the endpoints from a diagonal from the other two points. Consequently, four points cannot be shattered, and $\mathrm{VCdim}(\mathbb{R}^2, \mathcal{H}_2) = 3$ is established.

The above argument gets tedious in higher dimensions, if it works in a rigorous way at all. Fortunately, we can employ a classic: By Radon's Lemma (Theorem 5.11), every set A of $\geqslant d+2$ points in $\mathbb{R}^d$ can be partitioned into two disjoint subsets A_1 and A_2 such that $\mathrm{conv}(A_1) \cap \mathrm{conv}(A_2) \neq \emptyset$. We get, as an easy implication, that a set A of at least $d+2$ points in $\mathbb{R}^d$ cannot be shattered by halfspaces. Indeed, let $A_1 \mathbin{\dot\cup} A_2$ be a partition as guaranteed by Radon's Lemma. Now every halfspace containing A_1 must contain at least one point of A_2, hence $h \cap A = A_1$ is impossible for a halfspace h and thus A is not shattered by $\mathcal{H}_d$. Moreover, it is easily seen that the vertex set of a d-dimensional simplex (there are $d+1$ vertices) can be shattered by halfspaces (each subset of the vertices forms a face of the simplex and can thus be separated from the rest by a hyperplane). We summarize that

$$\mathrm{VCdim}(\mathbb{R}^d, \mathcal{H}_d) = d+1 \ .$$

Let us now consider the range space $(\mathbb{R}^d, \check{\mathcal{H}}_d)$, where $\check{\mathcal{H}}_d$ denotes the set of all closed halfspaces below non-vertical hyperplanes[9]–we call these *lower halfspaces*. Since $\check{\mathcal{H}}_d \subseteq \mathcal{H}_d$, the VC-dimension of $(\mathbb{R}^d, \check{\mathcal{H}}_d)$ is at most $d+1$, but, in fact, it not too difficult to show

$$\mathrm{VCdim}(\mathbb{R}^d, \check{\mathcal{H}}_d) = d \ . \tag{H.7}$$

(Check this claim at least for $d = 2$.) This range space is a geometric example where the bound of Sauer's Lemma is attained. Indeed, for any set A of n points in $\mathbb{R}^d$ in general position[10], it can be shown that

$$\left|\check{\mathcal{H}}_d\big|_A\right| = \Phi_d(n).$$

[9] A hyperplane is called non-vertical if it can be specified by a linear equation $\sum_{i=1}^{d} \lambda_i x_i = \lambda_{d+1}$ with $\lambda_d \neq 0$; see also Section 1.2

[10] No $i+2$ on a common i-flat for $i \in \{1, 2, \ldots, d-1\}$; in particular, no $d+1$ points on a common hyperplane.

Balls. It is easy to convince oneself that the VC-dimension of disks in the plane is 3: Three points not on a line can be shattered and four points cannot. Obviously not, if one of the points is in the convex hull of the other, and for four vertices of a convex quadrilateral, it is not possible for both diagonals to be separated from the endpoints of the respective other diagonal by a circle (if you try to draw a picture of this, you see that you get two circles that intersect four times, which we know is not be the case).

A more rigorous argument which works in all dimensions is looming with the help of (H.7), if we employ the following transformation called *lifting map* that we have already encountered for $d = 2$ in Section 6.3:

$$\begin{array}{rcl} \mathbb{R}^d & \longrightarrow & \mathbb{R}^{d+1} \\ (x_1, x_2, \ldots, x_d) = p & \mapsto & \ell(p) = (x_1, x_2, \ldots, x_d, {x_1}^2 + {x_2}^2 + \cdots + {x_d}^2) \end{array}$$

(For a geometric interpretation, this is a vertical projection of $\mathbb{R}^d$ to the unit paraboloid $x_{d+1} = {x_1}^2 + {x_2}^2 + \cdots + {x_d}^2$ in $\mathbb{R}^{d+1}$.) The remarkable property of this transformation is that it maps balls in $\mathbb{R}^d$ to halfspaces in $\mathbb{R}^{d+1}$ in the following sense.

Consider a ball $B_d(c, \rho)$ ($c = (c_1, c_2, \ldots, c_d) \in \mathbb{R}^d$ the center, and $\rho \in \mathbb{R}^+$ the radius). A point $p = (x_1, x_2, \ldots, x_d)$ lies in this ball if and only if

$$\begin{aligned} & \sum_{i=1}^{d} (x_i - c_i)^2 \leqslant \rho^2 \Leftrightarrow \sum_{i=1}^{d} ({x_i}^2 - 2x_i c_i + {c_i}^2) \leqslant \rho^2 \\ \Leftrightarrow \quad & \left(\sum_{i=1}^{d} (-2c_i) x_i \right) + ({x_1}^2 + {x_2}^2 + \cdots + {x_d}^2) \leqslant \rho^2 - \sum_{i=1}^{d} {c_i}^2 \ ; \end{aligned}$$

this equivalently means that $\ell(p)$ lies below the non-vertical hyperplane (in $\mathbb{R}^{d+1}$)

$$\begin{aligned} & h = h(c, \rho) = \left\{ x \in \mathbb{R}^{d+1} \;\middle|\; \sum_{i=1}^{d+1} h_i x_i = h_{d+2} \right\} \quad \text{with} \\ & (h_1, h_2, \ldots, h_d, h_{d+1}, h_{d+2}) = \left((-2c_1), (-2c_2), \ldots, (-2c_d), 1, \rho^2 - \sum_{i=1}^{d} {c_i}^2 \right) \ . \end{aligned}$$

It follows that a set $A \subseteq \mathbb{R}^d$ is shattered by $\mathcal{B}_d$ (the set of closed balls in $\mathbb{R}_d$) if and only if $\ell(A) := \{\ell(p) \,|\, p \in A\}$ is shattered by $\check{\mathcal{H}}_{d+1}$. Assuming (H.7), this readily yields

$$\mathrm{VCdim}(\mathbb{R}^d, \mathcal{B}_d) = d + 1 \ .$$

The lifting map we have employed here is a special case of a more general paradigm called *linearization* which maps non-linear conditions to linear conditions in higher dimensions.

We have clarified the VC-dimension for all examples of range spaces that we have listed in Section H.2, except for the one involving simplices. The following exercise should help to provide an answer.

Exercise H.8. *For a range space $(X, \mathcal{R})$ of VC-dimension d and a number $k \in \mathbb{N}$, consider the range space $(X, \mathcal{I})$ where ranges form intersections of at most k ranges from $\mathcal{R}$, that is, $\mathcal{I} = \{\bigcap_{i=1}^{k} R_i \,|\, \forall i : R_i \in \mathcal{R}\}$. Give an upper bound for the VC-dimension of $(X, \mathcal{I})$.*

H.6 Small ε-Nets, an Easy Warm-up Version

Let us prove a first bound on the size of ε-nets when the VC-dimension is finite.

Theorem H.9. *Let $n \in \mathbb{N}$ and $(X, \mathcal{R})$ be a range space of VC-dimension $d \geqslant 2$. Then for any $A \subseteq X$ with $|A| = n$ and any $\varepsilon \in \mathbb{R}^+$ there exists an ε-net N of A w.r.t. $\mathcal{R}$ with $|N| \leqslant \lceil \frac{d \ln n}{\varepsilon} \rceil$.*

Proof. We restrict our attention to the finite projected range space (A, R), $R := \mathcal{R}|_A$, for which we know $|R| \leqslant \Phi_d(n) \leqslant n^d$. It suffices to show that there is a set $N \subseteq A$ with $|N| \leqslant \frac{d \ln n}{\varepsilon}$ which contains an element from each $r \in R_\varepsilon := \{r \in R : |r| > \varepsilon n\}$.

Suppose, for some $s \in \mathbb{N}$ (to be determined), we let $N \sim A^s$. For each $r \in R_\varepsilon$, we know that $\text{prob}(r \cap N = \emptyset) < (1-\varepsilon)^s \leqslant e^{-\varepsilon s}$. Therefore,

$$\begin{aligned} \text{prob}(N \text{ is not } \varepsilon\text{-net of } A) &= \text{prob}(\exists r \in R_\varepsilon : r \cap N = \emptyset) \\ &= \text{prob}\Big(\bigvee_{r \in R_\varepsilon} (r \cap N = \emptyset)\Big) \\ &\leqslant \sum_{r \in R_\varepsilon} \text{prob}(r \cap N = \emptyset) < |R_\varepsilon| e^{-\varepsilon s} \leqslant n^d e^{-\varepsilon s} \,. \end{aligned}$$

It follows that if s is chosen so that $n^d e^{-\varepsilon s} \leqslant 1$, then $\text{prob}(N \text{ is not } \varepsilon\text{-net of } A) < 1$ and there remains a positive probability for the event that N is an ε-net of A. Now

$$n^d e^{-\varepsilon s} \leqslant 1 \quad \Leftrightarrow \quad n^d \leqslant e^{\varepsilon s} \quad \Leftrightarrow \quad d \ln n \leqslant \varepsilon s.$$

That is, for $s = \lceil \frac{d \ln n}{\varepsilon} \rceil$, the probability of obtaining an ε-net is positive, and therefore an ε-net of that size has to exist.[11] □

If we are willing to invest a little more in the size of the random sample N, then the probability of being an ε-net grows dramatically. More specifically, for $s = \lceil \frac{d \ln n + \lambda}{\varepsilon} \rceil$, we have

$$n^d e^{-\varepsilon s} \leqslant n^d e^{-d \ln n - \lambda} = e^{-\lambda} \,,$$

and, therefore, a sample of that size is an ε-net with probability at least $1 - e^{-\lambda}$.

[11] This line of argument "If an experiment produces a certain object with positive probability, then it has to exist", as trivial as it is, admittedly needs some time to digest. It is called *The Probabilistic Method*, and was used and developed to an amazing extent by the famous Hungarian mathematician Paul Erdős starting in the thirties.

We realize that we need $\frac{d \ln n}{\varepsilon}$ sample size to compensate for the (at most) n^d subsets of A which we have to hit—it suffices to ensure positive success probability. The extra $\frac{\lambda}{\varepsilon}$ allows us to boost the success probability.

Also note that if A were shattered by $\mathcal{R}$, then $R = 2^A$ and $|R| = 2^n$. Using this bound instead of n^d in the proof above would require us to choose s to be roughly $\frac{n \ln 2}{\varepsilon}$, a useless estimate which even exceeds n unless ε is large (at least $\ln 2 \approx 0.69$).

Smallest enclosing balls, again It is time to rehabilitate ourselves a bit concerning the suggested procedure for computing a small ball containing all but at most $\varepsilon|A|$ points from an n-point set $A \subseteq \mathbb{R}^d$.

Let $(\mathbb{R}^d, {\mathcal{B}_d}^{\text{compl}})$ be the range space whose ranges consist of all complements of closed balls in $\mathbb{R}^d$. This has the same VC-dimension $d+1$ as $(\mathbb{R}^d, \mathcal{B}_d)$. Indeed, if $A \cap r = A'$ for a ball r, then $A \cap (\mathbb{R}^d \setminus r) = A \setminus A'$, so A is shattered by $\mathcal{B}_d$ if and only if A is shattered by ${\mathcal{B}_d}^{\text{compl}}$.

Hence, if we choose a sample N of size $s = \lceil \frac{(d+1)\ln n + \lambda}{\varepsilon} \rceil$ then, with probability at least $1 - e^{-\lambda}$ this is an ε-net for A w.r.t. ${\mathcal{B}_d}^{\text{compl}}$. Let us quickly recall what this means: whenever the complement of some ball B has empty intersection with N, then this complement contains at most an $\varepsilon|A|$ points of A. As a consequence, the smallest ball enclosing N has at most $\varepsilon|A|$ points of A outside, with probability at least $1 - e^{-\lambda}$.

H.7 Even Smaller ε-Nets

We still need to prove part 2 of Theorem H.2, the existence of ε-nets whose size is independent of A. For this, we employ the same strategy as in the previous section, i.e. we sample elements from A uniformly at random, with replacement; a refined analysis will show that—compared to the bound of Theorem H.9—much less elements suffice. Here is the main technical lemma.

Lemma H.10. *Let $(X, \mathcal{R})$ be a range space of VC-dimension $\delta \geqslant 2$, and let $A \subseteq X$ be finite. If $x \sim A^m$ for $m \geqslant 8/\varepsilon$, then for the set N_x of elements occurring in x, we have*

$$\text{prob}(N_x \textit{ is not an } \varepsilon\textit{-net for } A \textit{ w.r.t. } \mathcal{R}) \leqslant 2\Phi_\delta(2m)2^{-\frac{\varepsilon m}{2}} .$$

Before we prove this, let us derive the bound of Theorem H.2 from it. Let us recall the statement.

Theorem H.2 (2). Let $(X, \mathcal{R})$ be a range space with VC-dimension $\delta \geqslant 2$. Then for every finite $A \subseteq X$ and every ε, $0 < \varepsilon \leqslant 1$, there is an ε-net of A w.r.t. $\mathcal{R}$ of size at most $\frac{8\delta}{\varepsilon} \log_2 \frac{4\delta}{\varepsilon}$ (independent of the size of A).

We want that

$$2\Phi_\delta(2m)2^{-\frac{\varepsilon m}{2}} < 1,$$

since then we know that an ε-net of size m exists. We have

$$\begin{aligned}
& 2\Phi_\delta(2m)2^{-\frac{\varepsilon m}{2}} < 1 \\
\Leftarrow\ & 2(2m)^\delta < 2^{\frac{\varepsilon m}{2}} \\
\Leftrightarrow\ & 1 + \delta \log_2(2m) < \frac{\varepsilon m}{2} \\
\Leftarrow\ & 2\delta \log_2 m < \frac{\varepsilon m}{2} \\
\Leftrightarrow\ & \frac{4\delta}{\varepsilon} < \frac{m}{\log_2 m}.
\end{aligned}$$

In the second to last implication, we have used $\delta \geqslant 2$ and $m \geqslant 8$. Now we claim that the latter inequality is satisfied for $m = 2\frac{4\delta}{\varepsilon}\log_2\frac{4\delta}{\varepsilon}$. To see this, we need that $\frac{m}{\log_2 m} > \alpha$ for $m = 2\alpha\log_2\alpha$ and $\alpha = \frac{4\delta}{\varepsilon}$. We compute

$$\frac{m}{\log_2 m} = \frac{2\alpha\log_2\alpha}{\log_2(2\alpha\log_2\alpha)} = \frac{2\alpha\log_2\alpha}{1+\log_2\alpha+\log_2\log_2\alpha} = \alpha\frac{2\log_2\alpha}{1+\log_2\alpha+\log_2\log_2\alpha} \geqslant \alpha$$

as long as

$$\log_2\alpha \geqslant 1+\log_2\log_2\alpha \Leftrightarrow \log_2\alpha \geqslant \log_2 2\log_2\alpha \Leftrightarrow \alpha \geqslant 2\log_2\alpha,$$

which holds as long as $\alpha \geqslant 2$, and this is satisfied for $\alpha = \frac{4\delta}{\varepsilon}$.

Proof. (Lemma H.10) By going to the projection $(A, \mathcal{R}|_A)$, we may assume that $X = A$, so we have a finite range space over n elements (see Section H.4). Fix $\varepsilon \in \mathbb{R}^+$. For $t \in \mathbb{N}$, a range $r \in \mathcal{R}$ and $x \in A^t$ (a t-vector of elements from A), we define

$$\mathrm{count}(r, x) = |\{i \in [t] : x_i \in r\}|.$$

Now we consider two events over A^{2m}. The first one is the bad event of not getting an ε-net when we choose m elements at random from A (plus another m elements that we ignore for the moment):

$$Q := \left\{ xy \in A^{2m} \;\middle|\; x \in A^m, y \in A^m, \exists r \in R_\varepsilon : \mathrm{count}(r, x) = 0 \right\}.$$

Recall that $R_\varepsilon = \{r \in \mathcal{R} : |r| > \varepsilon n\}$. Thus $\mathrm{prob}(Q) = \mathrm{prob}(N_x \text{ is not } \varepsilon\text{-net})$ which is exactly what we want to bound.

The second auxiliary event looks somewhat weird at first:

$$J := \left\{ xy \in A^{2m} \;\middle|\; x \in A^m, y \in A^m, \exists r \in R_\varepsilon : \mathrm{count}(r, x) = 0 \text{ and } \mathrm{count}(r, y) \geqslant \frac{\varepsilon m}{2} \right\}.$$

This event satisfies $J \subseteq Q$ and contains pairs of sequences x and y with somewhat contradicting properties for some r. While the first part x fails to contain any element from r, the second part y has many elements from r.

Claim 1. $\text{prob}(J) \leqslant \text{prob}(Q) \leqslant 2\text{prob}(J)$.

The first inequality is a consequence of $J \subseteq Q$. To prove the second inequality, we show that

$$\frac{\text{prob}(J)}{\text{prob}(Q)} = \frac{\text{prob}(J \cap Q)}{\text{prob}(Q)} = \text{prob}(J \mid Q) \geqslant \frac{1}{2}.$$

So suppose that $xy \in Q$, with "witness" r, meaning that $r \in R_\varepsilon$ and $\text{count}(r, x) = 0$. We show that $xy \in J$ with probability at least $1/2$, for every fixed such x and y chosen randomly from A^m. This entails the claim.

The random variable $\text{count}(r, y)$ is a sum of m independent Bernoulli experiments with success probability $p := |r|/n > \varepsilon$ (thus expectation p and variance $p(1-p)$). Using linearity of expectation and variance (the latter requires independence of the experiments), we get

$$\begin{aligned} \text{E}(\text{count}(r, y)) &= pm, \\ \text{Var}(\text{count}(r, y)) &= p(1-p)m. \end{aligned}$$

Now we use Chebyshew's inequality[12] to bound the probability of the "bad" event that $\text{count}(r, y) < \frac{\varepsilon m}{2}$. We have

$$\begin{aligned} & \text{prob}\left(\text{count}(r, y) < \frac{\varepsilon m}{2}\right) \\ \leqslant\ & \text{prob}\left(\text{count}(r, y) < \frac{pm}{2}\right) \\ \leqslant\ & \text{prob}\left(|\text{count}(r, y) - \text{E}(\text{count}(r, y))| > \frac{pm}{2}\right) \\ =\ & \text{prob}\left(|\text{count}(r, y) - \text{E}(\text{count}(r, y))| > \frac{1}{2(1-p)}\text{Var}(\text{count}(r, y))\right) \\ \leqslant\ & \frac{4(1-p)^2}{p(1-p)m} = \frac{4(1-p)}{pm} \leqslant \frac{4}{pm} < \frac{4}{\varepsilon m} \leqslant \frac{1}{2}, \end{aligned}$$

since $m \geqslant \frac{8}{\varepsilon}$. Hence

$$\text{prob}\left(\text{count}(r, y) \geqslant \frac{\varepsilon m}{2}\right) \geqslant \frac{1}{2},$$

and the claim is proved.

Now the weird event J reveals its significance: we can nicely bound its probability. The idea is this: We enumerate A as $A = \{a_1, a_2, \ldots, a_n\}$ and let the *type* of $z \in A^t$ be the n-sequence

$$\text{type}(z) = (\text{count}(a_1, z), \text{count}(a_2, z), \ldots, \text{count}(a_n, z)).$$

For example, the type of $z = (1, 2, 4, 2, 2, 4)$ w.r.t. $A = \{1, 2, 3, 4\}$ is $\text{type}(z) = (1, 3, 0, 2)$.

Now fix an arbitrary type τ. We will bound the conditional probability $\text{prob}(xy \in J \mid xy \text{ has type } \tau)$, and the value that we get is independent of τ. It follows that the same value also bounds $\text{prob}(J)$.

[12] $\text{prob}(|X - \text{E}(X)| \geqslant k\text{Var}(X)) \leqslant \frac{1}{k^2 \text{Var}(X)}$

Claim 2. $\text{prob}(xy \in J \mid xy \text{ has type } \tau) \leqslant \Phi_\delta(2m)2^{-\varepsilon m/2}$.

To analyze the probability in question, we need to sample $z = xy$ uniformly at random from all sequences of type τ. This can be done as follows: take an arbitrary sequence z' of type τ, and then apply a random permutation $\pi \in S_{2m}$ to obtain $z = \pi(z')$, meaning that

$$z_i = z'_{\pi(i)} \text{ for all } i.$$

Why does this work? First of all, π preserves the type, and it is easy to see that all sequences z of type τ can be obtained in this way. Now we simply count the number of permutations that map z' to a fixed z and see that this number only depends on τ, so it is the same for all z. Indeed, for every element $a_i \in A$, there are $\tau_i!$ many ways of mapping the a_i's in z' to the a_i's in z. The number of permutations that map z' to z is therefore given by

$$\prod_{i=1}^{n} \tau_i!.$$

By these considerations,

$$\text{prob}(xy \in J \mid xy \text{ has type } \tau) = \text{prob}(\pi(z') \in J).$$

To estimate this, we let S be the set of distinct elements in z' (this is also a function of the type τ). Since $(X, \mathcal{R})$ has VC-dimension δ, we know from Lemma H.6 that $|\mathcal{R}|_S| \leqslant \Phi_\delta(2m)$, so at most that many different subsets T of S can be obtained by intersections with ranges $r \in \mathcal{R}$, and in particular with ranges $r \in R_\varepsilon$.

Now we look at some fixed such T, consider a permutation π and write $\pi(z') = xy$. We call T a *witness* for π if no element of x is in T, but at least $\varepsilon m/2$ elements of y are in T. According to this definition,

$$\pi(z') \in J \quad \Leftrightarrow \quad \pi \text{ has some witness } T \subseteq S.$$

By the union bound,

$$\text{prob}(\pi(z') \in J) = \text{prob}(\exists T\colon T \text{ is a witness for } \pi) \leqslant \sum_T \text{prob}(T \text{ is a witness for } \pi).$$

Suppose that z' contains $\ell \geqslant \varepsilon m/2$ occurences of elements of T (for smaller ℓ, T cannot be a witness). In order for $\pi(z') = xy$ to be in J, no element of T can appear in the first half x, but all ℓ occurrences of elements from T must fall into the second half y. Therefore, the probability of T being a witness for a random permutation is

$$\frac{\binom{m}{\ell}}{\binom{2m}{\ell}} = \frac{m(m-1)\cdots(m-\ell+1)}{2m(2m-1)\cdots(2m-\ell+1)} \leqslant 2^{-\ell} \leqslant 2^{-\frac{\varepsilon m}{2}},$$

since among all the $\binom{2m}{\ell}$ equally likely ways in which π distributes the ℓ occurences in z, exactly the $\binom{m}{\ell}$ equally likely ways of putting them into y are good.[13] Summing up

[13] This argument can be made more formal by explicitly counting the permutations that map $\{1, 2, \ldots, \ell\}$ to the set $\{1, 2, \ldots, m\}$. We leave this to the reader.

over all at most $\Phi_\delta(2m)$ sets T, we get

$$\text{prob}(xy \in J \mid xy \text{ has type } \tau) \leqslant \Phi_\delta(2m)2^{-\frac{\varepsilon m}{2}},$$

for all τ.

The proof is finished by combining the two claims:

$$\text{prob}(N_x \text{ is not } \varepsilon\text{-net}) = \text{prob}(Q) \leqslant 2\text{prob}(J) \leqslant 2\Phi_\delta(2m)2^{-\frac{\varepsilon m}{2}}.$$

□

Questions

108. *What is a range space?* Give a definition and a few examples.

109. *What is an epsilon net?* Provide a formal definition, and explain in words what it means.

110. *What is the VC dimension of a range space* Give a definition, and compute the VC dimension of a few example range spaces.

111. *Which range spaces always have small epsilon nets (and how small), and which ones don't?* Explain Theorem H.2.

112. *How can you compute small epsilon nets?* Explain the general idea, and give the analysis behind the weaker bound of Theorem H.9.

113. *How can you compute smaller epsilon nets?* Sketch the main steps of the proof of the stronger bound of Theorem H.2.

References

[1] David Haussler and Emo Welzl, Epsilon-nets and simplex range queries. *Discrete Comput. Geom.*, 2, (1987), 127–151.