Advanced Geometric Algorithms

Lecture Notes

Vladlen Koltun[1]

Spring 2006

[1]Computer Science Department, 353 Serra Mall, Gates 464, Stanford University, Stanford, CA 94305, USA; *vladlen@stanford.edu*.

Contents

Chapter 1

Preliminaries from Convex Geometry

1.1 Definitions

Let $\mathbb{R}^d$ denote the d-dimensional Euclidean space. A k-flat passing through the origin is said to be a *linear subspace* of $\mathbb{R}^d$. A general k-flat is called an *affine subspace* of $\mathbb{R}^d$. In both cases, the dimension of the subspace is defined to be k.

An *affine combination* of a set of points $a_1, a_2, \ldots, a_n \in \mathbb{R}^d$ is an expression of the form

$$\sum_{i=1}^{n} c_i a_i, \text{ where } \sum_{i=1}^{n} c_i = 1.$$

A set of points $a_1, a_2, \ldots, a_n \in \mathbb{R}^d$ is said to be *affinely dependent* if one of the points can be expressed as an affine combination of the others. This is the case if and only if there exist $c_1, c_2, \ldots, c_n \in \mathbb{R}$, not all zero, such that

$$\sum_{i=1}^{n} c_i a_i = 0, \text{ where } \sum_{i=1}^{n} c_i = 0.$$

Note the distinction between linear combinations and affine combinations: Given two points in the plane, the set of their linear combinations covers the whole plane, but their set of affine combinations is a single line. Similarly, three points in the plane are linearly dependent, but affinely independent. A useful fact from linear algebra is that any set of $d+1$ or more points in $\mathbb{R}^d$ is linearly dependent and any set of $d+2$ or more points in $\mathbb{R}^d$ is also affinely dependent.

A *convex* combination of a set $A = \{a_1, a_2, \ldots, a_n\} \subseteq \mathbb{R}^d$ is an expression of the form

$$\sum_{i=1}^{n} c_i a_i, \text{ where } \sum_{i=1}^{n} c_i = 1, \text{ and } c_i \geq 0 \text{ for all } 1 \leq i \leq n.$$

The set of all convex combinations of A is called the *convex hull* of A, denoted by $\operatorname{conv}(A)$.

1.2 Radon's theorem

Theorem 1.2.1. *Given $A = \{a_1, a_2, \ldots, a_{d+2}\} \subseteq \mathbb{R}^d$, there exist two disjoint subsets $A_1, A_2 \subset A$ such that*

$$\operatorname{conv}(A_1) \cap \operatorname{conv}(A_2) \neq \emptyset.$$

Proof. We have $d+2$ points in d-space, so they are affinely dependent. Therefore, there exist $c_1, c_2, \ldots, c_{d+2} \in \mathbb{R}$, not all zero, such that

$$\sum_{i=1}^{d+2} c_i a_i = 0 \text{ and } \sum_{i=1}^{d+2} c_i = 0.$$

Let $P = \{i : c_i > 0\}$ and $N = \{i : c_i < 0\}$. Since $\sum_{i=1}^{d+2} c_i = 0$, we have $P, N \neq \emptyset$. We claim that $A_1 = \{a_i : i \in P\}$ and $A_2 = \{a_i : i \in N\}$ are the desired sets.

Indeed, put $S = \sum_{i \in P} c_i = -\sum_{i \in N} c_i$. Consider the point

$$x = \sum_{i \in P} \frac{c_i}{S} a_i.$$

x lies in the convex hull of P since $\sum_{i \in P} \frac{c_i}{S} = 1$ and $\frac{c_i}{S} \geq 0$ for all $i \in P$. Furthermore, recall that

$$\sum_{i=1}^{d+2} c_i a_i = 0 = \sum_{i \in P} c_i a_i + \sum_{i \in N} c_i a_i,$$

and thus

$$\sum_{i \in P} c_i a_i = -\sum_{i \in N} c_i a_i,$$

which implies that

$$x = -\sum_{i \in N} \frac{c_i}{S} a_i.$$

Therefore x also lies in the convex hull of N since $-\sum_{i \in N} \frac{c_i}{S} = 1$ and $\frac{-c_i}{S} \geq 0$ for all $i \in N$. □

1.3 Helly's theorem

We can use Radon's theorem to prove the celebrated result by Helly.

Theorem 1.3.1. *Given a collection of $n \geq d+1$ convex sets in $\mathbb{R}^d$, if the intersection of every $d+1$ of these sets is nonempty, then the intersection of all the sets is nonempty.*

Proof. The proof is by induction on n. The base case of $d+1$ sets is trivial. Suppose the claim holds for collections of $n-1$ sets and consider a collection $C_1, C_2, \ldots, C_n$ of convex sets in $\mathbb{R}^d$. Let $S_i = \{C_j : j \neq i\}$ for $1 \leq i \leq n$. By the induction hypothesis, for any S_i, there exists a point a_i that lies in every $C \in S_i$. Consider the collection

$A = \{a_1, a_2, \ldots, a_n\}$. By Radon's theorem, A can be partitioned into disjoint subsets A_1 and A_2 whose convex hulls intersect. Let $x \in \operatorname{conv}(A_1) \cap \operatorname{conv}(A_2)$. Consider some set C_i and assume without loss of generality that $a_i \in A_1$. Note that $a_j \in C$ for all $C \in S_j$, and that $C_i \in S_j$ for all $j \neq i$. Therefore, for all $a_j \in A_2$, $a_j \in C_i$, which implies $A_2 \subseteq C_i$. Thus, $\operatorname{conv}(A_2) \subseteq C_i$ and $x \in C_i$. Since this holds for all $1 \leq i \leq n$, the proof is complete. □

1.4 Carathéodory's theorem

Theorem 1.4.1. *Any convex combination of a set of points $A = \{a_1, a_2, \ldots, a_n\} \subseteq \mathbb{R}^d$ is a convex combination of at most $d+1$ points in A.*

Proof. By contradiction. Given $x \in \operatorname{conv}(A)$, suppose, without loss of generality, that $\sum_{i=1}^{k} c_i a_i$, where $\sum_{i=1}^{k} c_i = 1$ and $c_i \geq 0$ for all $1 \leq i \leq n$, is a representation of x as a convex combination of a subset of A involving the smallest possible such subset, and that $k \geq d+2$. Thus the points $a_1, a_2, \ldots, a_k$ are affinely dependent and there exist $d_1, d_2, \ldots, d_k \in \mathbb{R}$, not all zero, such that

$$\sum_{i=1}^{k} d_i a_i = 0 \text{ and } \sum_{i=1}^{k} d_i = 0.$$

Assume without loss of generality that $d_k > 0$ and $c_k/d_k \leq c_i/d_i$ for those i $(1 \leq i \leq k-1)$ for which $d_i > 0$. We shall express x as a convex combination of $a_1, a_2, \ldots, a_{k-1}$, obtaining a contradiction. For this, define the coefficients $e_i = c_i - \frac{c_k}{d_k} d_i$ for all $1 \leq i \leq k-1$.

If $d_i \leq 0$ then $e_i \geq c_i \geq 0$, and if $d_i > 0$ then $e_i = c_i - \frac{c_k}{d_k} d_i \geq c_i - \frac{c_i}{d_i} d_i = 0$. Furthermore,

$$\sum_{i=1}^{k-1} e_i = \sum_{i=1}^{k-1} \left(c_i - \frac{c_k}{d_k} d_i \right) = \sum_{i=1}^{k-1} c_i - \frac{c_k}{d_k} \sum_{i=1}^{k-1} d_i = \sum_{i=1}^{k} c_i - \frac{c_k}{d_k} \sum_{i=1}^{k} d_i = \sum_{i=1}^{k} c_i = 1.$$

Thus the coefficients e_i are nonnegative and sum to 1. Finally,

$$x = \sum_{i=1}^{k} c_i a_i = \sum_{i=1}^{k} c_i a_i - \frac{c_k}{d_k} \sum_{i=1}^{k} d_i a_i = \sum_{i=1}^{k-1} \left(c_i - \frac{c_k}{d_k} d_i \right) a_i = \sum_{i=1}^{k-1} e_i a_i.$$

We have reached a contradiction that proves the theorem. □

Chapter 2

ε-nets and VC-dimension

2.1 Defining ε-nets

A remarkably useful tool in modern computational geometry is the notion of ε-nets, which is used to power many randomized geometric constructions, as we shall see below. To define ε-nets, consider a set X with a probability measure μ on X. Most often, μ will be the uniform measure on the set X, which may be finite or infinite. Let F be a collection of subsets of X. Together, the pair (X, F) is called a *set system*. (When X is finite, a set system on X is sometimes referred to as a *hypergraph*.) Given $0 \leq \varepsilon \leq 1$, an ε-net for (X, F) is a subset of X that "hits" all the heavy sets in F, namely all the set in F that have measure at least ε. More formally,

Definition 2.1.1. *Given a set system (X, F) as above, a subset $N \subseteq X$ is called an ε-net for (X, F) if $N \cap S \neq \emptyset$ for all $S \in F$ with $\mu(S) \geq \varepsilon$.*

For example, given a finite set system (X, F) with $|X| = n$, $\mu(S) = \frac{|S|}{n}$ for any $S \in 2^X$, and $r \in \mathbb{N}^+$, a $(1/r)$-net for (X, F) is a subset N that has a nonempty intersection with all sets of F that have at least n/r elements.

We will be interested in finding small ε-nets. Specifically, for constant ε, we will want to find ε-nets of constant size (!), namely, size that depends on ε but not on the size of the set system. This is not always possible, as easy constructions can suggest, but quite often it is. To describe an important class of set systems that admit small ε-nets we need the notion of VC-dimension.

2.2 Defining VC-dimension

The *Vapnik-Chervonenkis dimension*, or the VC-dimension is a numerical parameter of a set system that quantifies how "well behaved", in a certain sense, the system is.

Given a set system (X, F) and a subset $Y \subseteq X$, the *restriction* of F on Y is the set $F|_Y = \{S \cap Y : S \in F\}$.

Definition 2.2.1. *Given a set system (X, F) as above, a subset $A \subseteq X$ is said to be* shattered *by F if each of the subsets of A arises as an intersection $A \cap S$ for some*

$S \in F$, that is, if $F|_Y = 2^A$. The VC-dimension $\dim(F)$ *of F is the size of the largest subset of X that is shattered by F.*

To see that the VC-dimension concept is not vacuous, consider two examples. Let X be the Euclidean plane $\mathbb{R}^2$, F_1 be the set of all convex polygons in the plane, and F_2 be the set of halfplanes. The set system (X, F_1) has VC-dimension ∞. Indeed, consider an arbitrarily large finite set $A \subseteq \mathbb{R}^2$ in convex position; any subset $A' \in A$ can arise as an intersection of A with a convex polygon—just take $\operatorname{conv}(A')$.

On the other hand, the set system (X, F_2) has the bounded VC-dimension 3. In fact, something more general holds.

Proposition 2.2.2. *Let $R[x_1, x_2, \ldots, x_d]_1$ denote the set of all linear functions in d variables, and let*

$$P_{d,1} = \big\{\{x \in \mathbb{R}^d : p(x) \geq 0\} : p \in R[x_1, x_2, \ldots, x_d]_1\big\}.$$

The VC-dimension of the set system $(\mathbb{R}^d, P_{d,1})$ is $d+1$.

Proof. Consider a set A of $d+1$ points in $\mathbb{R}^d$ in general position. It is easy to see that any subset of A can be separated from the rest by a halfspace. On the other hand, given a collection B of at least $d+2$ points in $\mathbb{R}^d$, Radon's theorem asserts that there exist two disjoint subsets $B_1, B_2 \subset B$ such that

$$\operatorname{conv}(B_1) \cap \operatorname{conv}(B_2) \neq \emptyset.$$

By convexity, these two subsets cannot be separated by a hyperplane, so neither B_1 nor B_2 can arise as an intersection of B with a halfspace. $\square$

2.3 The existence of small ε-nets

The next theorem of Haussler and Welzl is the reason we went through the trouble of introducing ε-nets and VC-dimension. It says that for set systems with bounded VC-dimension, small ε-nets exist, and can in fact be found by simple random sampling.

Theorem 2.3.1 (ε-net theorem)**.** *Given a set system (X, F) with $\dim(F) \leq d$, such that $d \geq 2$ and $r \geq 2$ is a parameter, there exists a $(1/r)$-net for (X, F) of size at most $Cdr \ln r$, where C is an absolute constant.*

To prove this theorem we first need to establish two lemmas. The first concerns a so-called *shatter function* of a set system (X, F). This is the function

$$\pi_F(m) = \max_{Y \subseteq X, |Y|=m} \big|F|_Y\big|.$$

To obtain intuition concerning this definition, note that $\pi_F(|X|)$ is simply the size of F. The next lemma then states that the size of a set system with bounded VC-dimension is itself bounded.

Lemma 2.3.2 (Shatter function lemma)**.** *Given a set system* (X, F) *with* $\dim(F) \le d$*, and any* $1 \le m \le |X|$*,*

$$\pi_F(m) \le \sum_{i=0}^{d} \binom{m}{i}.$$

This lemma implies that the size of (X, F) is $O(n^d)$, where $n = |X|$. In fact, a tighter bound can be obtained using estimates for binomial coefficients:

$$\pi_F(m) \le \left(\frac{em}{d}\right)^d.$$

Proof. Note that for any $Y \subseteq X$, the VC-dimension of $(Y, F|_Y)$ is at most d. (Any subset of Y that is shattered by $F|_Y$ is also a subset of X that is shattered by F.) This implies that it suffices to show that the size of (X, F) is at most $\sum_{i=0}^{d} \binom{n}{i}$, where $n = |X|$. To this end, we use induction on d, and for a fixed d we do induction on n.

For the induction step, fix some $x \in X$ and consider the set system $(X \setminus \{x\}, F_1)$, for $F_1 = F|_{X\setminus\{x\}}$. By the induction hypothesis, $|F_1| \le \sum_{i=0}^{d} \binom{n-1}{i}$. Any two distinct sets $A_1, A_2 \in F$ for which $A_1 \cap F_1 \ne A_2 \cap F_1$ are counted as distinct sets of F_1 and are thus considered in $|F_1|$. The only pairs of distinct sets of F that are not thus counted are pairs of sets $A_1, A_2 \in F$, such that $A_1 \subseteq X \setminus \{x\}$ and $A_2 = A_1 \cup \{x\}$.

Consider the set system $(X \setminus \{x\}, F_2)$, defined as $F_2 = \{A \in F_1 : A \in F \text{ and } A \cup \{x\} \in F\}$. The discussion above implies $|F| = |F_1| + |F_2|$. Observe now that $\dim(F_2) \le d - 1$, since if $A \subseteq X \setminus \{x\}$ is shattered by F_2 then $A \cup \{x\}$ is shattered by F. By the induction hypothesis, $|F_2| \le \sum_{i=0}^{d-1} \binom{n-1}{i}$. Therefore,

$$|F| \le \sum_{i=0}^{d-1} \binom{n-1}{i} + \sum_{i=0}^{d} \binom{n-1}{i} = 1 + \sum_{i=1}^{d} \left(\binom{n-1}{i-1} + \binom{n-1}{i} \right) =$$

$$\binom{n}{0} + \sum_{i=1}^{d} \binom{n}{i} = \sum_{i=0}^{d} \binom{n}{i}.$$

□

We still need one more lemma before we commence the proof of the ε-net theorem.

Lemma 2.3.3. *Let* $X = \sum_{i=1}^{n} X_i$*, where the* X_i *are independent random variables, each attaining the value 1 with probability* p *and the value 0 with probability* $1 - p$*. Then*

$$P\left[X \ge \frac{1}{2}np\right] \ge \frac{1}{2}$$

provided that $np \ge 8$*.*

Proof. The estimate in the lemma is very weak and much stronger ones can be obtained. This one is a consequence of Chebyshev's inequality that states that $P\left[|X - E[X]| \ge k\sigma\right] \le \frac{1}{k^2}$. In our case, $E[X] = np$ and $\sigma \le \sqrt{np}$. We have

$$P\left[X \ge \frac{1}{2}np\right] = 1 - P\left[X < \frac{1}{2}np\right] = 1 - P\left[X - E[X] < -\frac{1}{2}np\right] \le$$

$$1 - P\left[|X - E[X]| \ge \frac{1}{2}np\right] \le 1 - P\left[|X - E[X]| \ge \frac{\sqrt{np}}{2}\sigma\right] \ge 1 - \frac{4}{np} \ge \frac{1}{2}.$$

$\square$

Proof of the ε-net theorem. First of all, note that we can assume that all sets of F have measure at least $1/r$, as the small sets do not change anything. Put $s = Cdr \ln r$ and assume for the sake of clarity that it is an integer—the proof can be easily modified to remove this assumption. Let N denote a random sample of s elements from X, drawn with repetitions according to the probability distribution on X. (So N is regarded as a multiset, since it can have repeated elements.) Our goal is to show that N is a $(1/r)$-net with a positive probability—once we show that, it is easy to make this probability arbitrarily close to 1 by increasing the constant C. So let E_0 be the "bad" event that N fails to be a $(1/r)$-net, namely, that there exists $T \in F$ for which $N \cap T = \emptyset$.

We bound $P[E_0]$ away from 1 by a "magic trick" that relates $P[E_0]$ to the probability of another event, which we call E_1. Draw a second random sample of s elements from X and denote the resulting sequence by M. E_1 is the event that there exists a set $T \in F$ that is missed by N but is "heavily hit" by M. Specifically, put $k = s/2r$, again assuming that k is an integer, and let E_1 be the event that there exists $T \in F$ with $N \cap T = \emptyset$ and $|M \cap T| \geq k$.

Now, $P[E_1] \leq P[E_0]$, since E_1 required E_0. We show that $P[E_1] \geq \frac{1}{2}P[E_0]$. For this, we first bound $P[E_1|N]$. If N is a $(1/r)$-net, then $P[E_1] = P[E_0] = 0$. Otherwise, fix a set T_N for which $N \cap T = \emptyset$ and note that $P[E_1|N] \geq P[|M \cap T_N| \geq k]$. Now, the quantity $|M \cap T_N|$ can be viewed as a sum of s Bernoulli trials, each having success probability $1/r$. Thus, by the above lemma, the probability that this sum is at least $k = s/2r = \frac{1}{2}\frac{s}{r}$ is at least $\frac{1}{2}$, namely if N is not a $(1/r)$-net then $P[E_1|N] \geq \frac{1}{2}$. In general then, $P[E_1|N] \geq \frac{1}{2}P[E_0|N]$ for all N, which implies $P[E_1] \geq \frac{1}{2}P[E_0]$.

Now it's time for the second part of the magic trick, which involves showing that $P[E_1]$ is in fact small, strictly smaller than $\frac{1}{2}$. Given the above inequality, this will imply that $P[E_0]$ is less than 1.

Take a random sample A of $2s$ elements from X, and then choose s elements of A uniformly at random (without repetitions this time), denote this collection by N, and denote the rest of A by M. These N and M have the same distribution as above. Let's analyze $P[E_1|A]$.

Let A be fixed. Fix a particular $T \in F$ and consider the probability that "E_1 holds for this particular T", namely, $P_T = P[N \cap T = \emptyset, |M \cap T| \geq k|A]$. If $|A \cap T| < k$ then $P_T = 0$. Otherwise we use the fact that $P_T \leq P[N \cap T = \emptyset|A]$. This is the probability that a random sample of the s elements of N out of the $2s$ elements of A avoids the at least k elements of $A \cap T$. Thus

$$P_T \leq \frac{\binom{2s-k}{s}}{\binom{2s}{s}} = \frac{\frac{(2s-k)!}{(s-k)!}}{\frac{(2s)!}{s!}} = \frac{(2s-k)(2s-k-1)\cdots(s-k+1)}{(2s)(2s-1)\cdots(s+1)} \leq$$
$$\left(\frac{2s-k}{2s}\right)^s = \left(1 - \frac{k}{2s}\right)^s \leq e^{-(k/2s)s} = e^{-k/2} = e^{-(Cd\ln r)/r} = r^{-Cd/4}.$$

This estimates P_T for a fixed $T \in F$. To estimate $P[E_1|A]$ in general we finally use, for the first time in the proof, the VC-dimension of X. The shatter function lemma implies that the number of possible sets $A \cap T$, for $T \in F$ is bounded, and the event "$N \cap T = \emptyset, |M \cap T| \geq k$" depends only on $A \cap T$. We combine this with the above estimate via the union bound:

$$P[E_1|A] \leq \sum_{i=0}^{d} \binom{2s}{i} r^{-Cd/4} \leq \left(\frac{2es}{d}\right)^d r^{-Cd/4} = \left(2Cer^{1-\frac{C}{4}} \ln r\right)^d < \frac{1}{2}$$

when $d, r \geq 2$ and C is chosen to be sufficiently large. Since this holds for all A we have $P[E_1] < \frac{1}{2}$, which implies $P[E_0] < 1$, concluding the proof. $\square$

2.4 ε-samples

I'd like to briefly mention a notion related to ε-nets that will also be useful.

Definition 2.4.1. *Given a set system (X, F), a subset $N \subseteq X$ is called an ε-*sample *for (X, F) if*

$$\left| \frac{|N \cap S|}{|N|} - \mu(S) \right| \leq \varepsilon$$

for all $S \in F$.

The notion of an ε-sample is stronger than that of an ε-net, since an ε-sample not only hits all heavy sets, but also represents them proportionally. ε-samples are in fact what Vapnik and Chervonenkis were after when they wrote their now-famous VC-dimension paper, and they proved that small ε-samples exist and can be found by random sampling for set systems with constant VC-dimension. The proof is similar to the Haussler-Welzl proof of the ε-net theorem. (Actually, the Haussler-Welzl proof is modeled after the original of Vapnik and Chervonenkis.)

Theorem 2.4.2 (ε-sample theorem)**.** *Given a set system (X, F) with $\dim(F) \leq d$, such that $d \geq 2$ and $r \geq 2$ is a parameter, there exists a $(1/r)$-sample for (X, F) of size at most $Cdr^2 \ln r$, where C is an absolute constant.*

2.5 Bounding the VC-dimension

The ε-net theorem provides a great incentive to bound the VC-dimension of various set systems, as this now implies that we can construct small ε-nets. We will soon see algorithmic applications of this, but let's begin with the VC-dimension bounds. We have already seen that the set system defined by halfspaces in $\mathbb{R}^d$ has VC-dimension $d + 1$. We now generalize this result to polynomials.

Theorem 2.5.1. *Let $R[x_1, x_2, \ldots, x_d]_{\leq D}$ denote the set of all real polynomials in d variables of degree at most D, and let*

$$P_{d,D} = \left\{ \{x \in \mathbb{R}^d : p(x) \geq 0\} : p \in R[x_1, x_2, \ldots, x_d]_{\leq D} \right\}.$$

The VC-dimension of the set system $(\mathbb{R}^d, P_{d,D})$ is at most $\binom{d+D}{d}$.

Proof. The proof reduces the case of polynomials to the case of hyperplanes in a higher-dimensional space using *linearization*. Let M be the set of all possible nonconstant monomials of degree at most D in $x_1, x_2, \ldots, x_d$. For example, when $d = D = 2$ we have $M = \{x_1, x_2, x_1x_2, x_1^2, x_2^2\}$. Note that $|M| = \binom{d+D}{d} - 1$, since monomials of M correspond to placements of D identical balls in $d+1$ bins. The bins are the d coordinates, plus an "extra" one, which is used by monomials of degree strictly less than D. There is one configuration that corresponds to a constant monomial, and this is the one configuration that is subtracted from $\binom{d+D}{d}$. Denote $m = |M|$, and let the coordinates in $\mathbb{R}^m$ be indexed by the monomials of M. The linearization we use is the mapping $\varphi : \mathbb{R}^d \to \mathbb{R}^m$, defined by $\varphi(x)_\mu = \mu(x)$. For example, when $d = D = 2$, the map is

$$\varphi(x_1, x_2) = (x_1, x_2, x_1x_2, x_1^2, x_2^2).$$

Now, if $A \in \mathbb{R}^d$ is shattered by $P_{d,D}$, then $\varphi(A)$ is shattered by half-spaces in $\mathbb{R}^m$. Indeed, consider $B \subseteq A$, and let $p \in P_{d,D}$ be a polynomial that is positive over B and negative over $A \backslash B$. Denote $p = a_0 + \sum_{\mu \in M} a_\mu \mu$. Now consider the halfspace h_p in $\mathbb{R}^m$, defined as $\{y \in \mathbb{R}^m : a_0 + \sum_{\mu \in M} a_\mu y_\mu \geq 0\}$. For example, if $p = 2 + 3x_2 - 5x_1x_2 + x_1^2$, then $h_p = \{y \in \mathbb{R}^m : 2 + 3y_2 - 5y_3 + y_4 \geq 0\}$. Then $h_p \cap \varphi(A) = \varphi(B)$, and in general $\varphi(A)$ is shattered by halfspaces in $\mathbb{R}^m$. Thus $\dim(P_{d,D}) \leq m + 1 = \binom{d+D}{d}$. □

Now that we have bounded the VC-dimension of polynomials, we can extend the result even further, to the domain of *semialgebraic sets*. A semialgebraic set is a set definable by a Boolean combination of polynomial inequalities. More formally, a set $A \subseteq \mathbb{R}^d$ is called semialgebraic if there are polynomials $p_1, p_2, \ldots, p_k \in R[x_1, x_2, \ldots, x_d]_{\leq D}$, for some D, and a Boolean formula $F(X_1, X_2, \ldots, X_k)$, such that

$$A = \left\{x \in \mathbb{R}^d : F(p_1 \geq 0, p_2 \geq 0, \ldots, p_k \geq 0)\right\}.$$

We can bound the VC-dimension of a set system defined by semialgebraic sets using the following general result.

Theorem 2.5.2. *Let $F(X_1, X_2, \ldots, X_k)$ be a set-theoretic formula involving the operations of union, intersection, and subtraction. Let (X, S) be a set system with bounded* $\dim(S) = d$. *Let*

$$T = \{F(s_1, s_2, \ldots, s_k) : s_1, s_2, \ldots, s_k \in S\}.$$

Then $\dim(T) = O(dk \log(dk))$.

Proof. Let $A \subseteq X$ be an n-point set. By induction on the structure of F,

$$F(s_1, s_2, \ldots, s_k) \cap A = F(s_1 \cap A, s_2 \cap A, \ldots, s_k \cap A).$$

In particular, $F(s_1, s_2, \ldots, s_k) \cap A$ depends only on the individual intersections $s_i \cap A$. Thus $\pi_T(n) \leq \pi_S(n)^k$. The shatter function lemma implies $\pi_S(n) \leq \sum_{i=0}^{d} \binom{n}{i}$. If A is shattered by T, then $\pi_T(n) = 2^n$. Thus

$$2^n \leq \left(\sum_{i=0}^{d} \binom{n}{i}\right)^k \leq \left(\frac{en}{d}\right)^{dk}.$$

Using elementary calculus, this implies $n = O(dk \log(dk))$. □

Now let H_d be the set of all hyperplanes in $\mathbb{R}^d$, and let

$$P^*_{d,1} = \big\{\{h \in H_d : h \text{ above } x\} : x \in \mathbb{R}^d\big\} \cup \big\{\{h \in H_d : h \text{ below } x\} : x \in \mathbb{R}^d\big\}.$$

Consider the set system $(H_d, P^*_{d,1})$. By duality, the VC-dimension of this set system is $d+1$. Now consider the set system

$$\Gamma_d = \Big(H_d, \big\{\{h \in H_d : h \cap \gamma \neq \emptyset\} : \gamma \text{ is a line segment in } \mathbb{R}^d\big\}\Big).$$

It is easy to see that Γ_d is contained in the set system

$$\Big(H_d, \big\{(s_1 \cap s_2) \cup (s_3 \cap s_4) : s_1, s_2, s_3, s_4 \in P^*_{d,1}\big\}\Big).$$

By above theorem, the VC-dimension of Γ_d is $O(d \log d)$. Now let

$$\Delta_d = \Big(H_d, \big\{\{h \in H_d : h \cap \delta \neq \emptyset\} : \delta \text{ is a } d\text{-simplex in } \mathbb{R}^d\big\}\Big).$$

A hyperplane intersects a simplex if and only if it intersects at least one of its edges. Thus Δ_d is part of a set system obtained from $P^*_{d,1}$ using a set theoretic formula on $O(d)$ variables. Therefore, its VC-dimension is $O(d^3 \log d)$.

Chapter 3

Applications of ε-nets

3.1 Arrangements and point location

A collection Γ of n hyperplanes in $\mathbb{R}^d$ subdivides the space into $\Theta(n^d)$ relatively open cells of dimensions ranging from 0 to d, such that each cell is a maximal connected set of points that lie in the intersection of a fixed (possibly empty) subset of Γ and are disjoint from all other surfaces of Γ. We refer to this subdivision as the *arrangement* of Γ and denote it by $\mathcal{A}(\Gamma)$.

Our first application of ε-nets will be to the problem of point location in arrangements. The problem is as follows. We wish to preprocess a collection Γ as above such that for a query point $p \in \mathbb{R}^d$, we can efficiently locate the cell of $\mathcal{A}(\Gamma)$ that contains p. For concreteness, let's say that we want the data structure to count how many hyperplanes $\gamma \in \Gamma$ satisfy $\gamma(p) > 0$. The data structure we construct illustrates the basic technique we will subsequently use in other applications.

The data structure is a tree T, and every node v of T is associated with a 'canonical' subset $\Gamma(v)$ of hyperplanes. The root is associated with the whole set Γ. Given a node of T, we recursively construct its subtree in T as follows. Let $n = |\Gamma(v)|$. If $n \leq n_0$, for some global constant n_0, this node becomes a leaf and the recursive construction stops. Otherwise, consider the set system

$$\mathcal{S} = \Big(\Gamma(v), \big\{\{h \in \Gamma(v) : h \cap \delta \neq \emptyset\} : \delta \text{ is a } d\text{-simplex in } \mathbb{R}^d\big\}\Big).$$

This is a subsystem of a set system Δ_d whose VC-dimension is at most $O(d^3 \log d)$. Thus the VC-dimension of $\mathcal{S}$ is at most $O(d^3 \log d)$, which is $O(1)$ when d is assumed to be constant. The ε-net theorem then implies that by random sampling in $O(n)$ time we can construct a $(1/r)$-net $R(v) \subseteq \Gamma(v)$ of $\mathcal{S}$ of size $O(r \log r)$. (We do assume $d = O(1)$ throughout this section.) We "decompose" $\mathcal{A}(R(v))$ into simplices using the following construction, called the "bottom-vertex simplicial decomposition".

Bottom-vertex decomposition. Here is what we mean by a decomposition of $\mathcal{A}(R(v))$. Such decomposition is a collection D of simplices that satisfies

(a)

$$\bigcup_{\delta \in D} \delta = \mathbb{R}^d$$

(b)

$$\forall \delta_1, \delta_2 \in D \ . \ \delta_1 \cap \delta_2 = \emptyset$$

(c)

$$\forall \delta \in D, r \in R(v) \ . \ (\delta \cap r = \emptyset) \vee (\delta \subseteq r)$$

The bottom-vertex decomposition is defined inductively on d. When $d = 1$, the decomposition is the arrangement itself. For higher d, given any $h \in R(v)$, we can consider the collection $R(v)|_h = \{s \cap h : s \in R(v)\}$ and its arrangement $\mathcal{A}(R(v)|_h)$ within the hyperplane h. $\mathcal{A}(R(v)|_h)$ is isomorphic to a hyperplane arrangement in $\mathbb{R}^{d-1}$ and can thus be decomposed by the inductive hypothesis. In this way we produce a decomposition of $\mathcal{A}(R(v)|_h)$ for all $h \in R(v)$. This results in a collection of pairwise disjoint simplices that cover all lower-dimensional cells of $\mathcal{A}(R(v))$. To cover the d-dimensional cells, use the following procedure: for a cell τ of $\mathcal{A}(R(v))$, take the lowest vertex v of τ (in the x_d-direction) and a decomposition simplex δ that lies on the boundary of τ but not entirely in the interior of any hyperplane that contains τ. Then add (the relative interior of) $\operatorname{conv}(v, \delta)$ to the decomposition.

It is easy to show that repeating this for all vertices v and simplices τ as above results in a decomposition of $\mathcal{A}(R(v))$. Moreover, this decomposition has $O(|R(v)|^d)$ simplices.

Note that unbounded cell require special handling and unbounded "simplices" are produced to cover those cells. We omit the details.

Upshot: We can cover the complement of the union of the hyperplanes $R(v)$ with $O((r \log r)^d)$ disjoint simplices. Call this collection of simplices D. Crucial observation: $R(v)$ is a $(1/r)$-net for the set system $\mathcal{S}$, which means that a hyperplane of $R(v)$ "hits" every simplex that intersects at least n/r hyperplanes of $\Gamma(v)$. Since no hyperplane of $R(v)$ intersects any simplex of D, a simplex of D intersects at most n/r hyperplanes of $\Gamma(v)$.

We store the decomposition D at v and create a child u_δ of v for every $\delta \in D$, link δ to u_δ and associate with u_δ the set $\Gamma' \subseteq \Gamma(v)$ of hyperplanes that intersect u_δ. Among $\Gamma(v) \setminus \Gamma'$, we count the number of those hyperplanes h that satisfy $h(p) > 0$ for $p \in \delta$, and store this number, denoted by $\mu(u_\delta)$, at u_δ. We then recurse within each u_δ. The resulting tree T has degree $O((r \log r)^d)$ and depth $O(\log n)$.

Given this data structure, a query proceeds as follows. We trace a path in T in a top-down fashion, starting from the root. We maintain a global counter, which, at the end of the query procedure, gives the number of hyperplanes satisfying $\gamma(p) > 0$. The counter is initialized to 0. At each visited node v, if v is a leaf we explicitly count the number of hyperplanes γ associated with v that satisfy $\gamma(p) > 0$, add it to the counter and terminate. The time spent at a leaf is thus $O(n_0)$, which is constant. If v is an internal node, we add $\mu(v)$ to the counter, go over the simplices in the decomposition stored at v, find the unique simplex δ that contains p and recurse within u_δ. The time spent at each internal node is thus $O((r \log r)^d)$, which is again $O(1)$. Since the height of T is $O(\log n)$, the query time is also $O(\log n)$.

Now let $S(n)$ be the maximum storage required by the data structure; then $S(n)$ satisfies the following recurrence:

$$S(n) \leq \begin{cases} c_0 & n \leq n_0, \\ c_1(r \log r)^d S\left(\frac{n}{r}\right) + c_2(r \log r)^d & n > n_0, \end{cases}$$

where c_0, c_1, c_2 are appropriate constants. We claim that the solution of this recurrence is $An^{d+\varepsilon}$ for any $\varepsilon > 0$ and for an appropriate constant A that depends on ε. This claim is true for $n \leq n_0$ if $A \geq c_0$. For $n > n_0$ we use induction:

$$\begin{aligned} S(n) &\leq c_1(r \log r)^d A\left(\frac{n}{r}\right)^{d+\varepsilon} + c_2(r \log r)^d \\ &\leq An^{d+\varepsilon}\left(\frac{c_1(\log r)^d}{r^\varepsilon} + \frac{c_2(r \log r)^d}{An^{d+\varepsilon}}\right) \\ &\leq An^{d+\varepsilon}\left(\frac{c_1(\log r)^d}{r^\varepsilon} + \frac{c_2(r \log r)^d}{A}\right) \\ &\leq An^{d+\varepsilon}. \end{aligned}$$

The last inequality holds if we choose r large enough that $r^\varepsilon \geq 2c_1(\log r)^d$, and choose $A \geq 2c_2(r \log r)^d$.

Notice that there is a lot of slack in this analysis. In particular, it does not take advantage at all of the $n^{d+\varepsilon}$ in the denominator of the second summand. We are going to reclaim this slack very soon. To start with, we still need to analyze the preprocessing time of the data structure. At every internal node with n associated surfaces, the construction algorithm spends $O(n(r \log r)^d)$ time. We claim that the overall preprocessing time is again $An^{d+\varepsilon}$ for any $\varepsilon > 0$ and an appropriate constant A. Similarly to the above, we let $T(n)$ be the maximum preprocessing time; then

$$T(n) \leq \begin{cases} c_0 & n \leq n_0, \\ c_1(r \log r)^d T\left(\frac{n}{r}\right) + c_2 n(r \log r)^d & n > n_0, \end{cases}$$

and, by induction for $n > n_0$,

$$\begin{aligned} T(n) &\leq c_1(r \log r)^d A\left(\frac{n}{r}\right)^{d+\varepsilon} + c_2 n(r \log r)^d \\ &\leq An^{d+\varepsilon}\left(\frac{c_1(\log r)^d}{r^\varepsilon} + \frac{c_2(r \log r)^d}{An^{d-1+\varepsilon}}\right) \\ &\leq An^{d+\varepsilon}\left(\frac{c_1(\log r)^d}{r^\varepsilon} + \frac{c_2(r \log r)^d}{A}\right) \\ &\leq An^{d+\varepsilon}, \end{aligned}$$

again for r and A chosen to be sufficiently large constants. Let us summarize what we have accomplished with a theorem.

Theorem 3.1.1. *Given a collection Γ of n hyperplanes in $\mathbb{R}^d$, one can preprocess it in time $O(n^{d+\varepsilon})$ for any $\varepsilon > 0$, into a data structure of size $O(n^{d+\varepsilon})$, so that a point location query in $\mathcal{A}(\Gamma)$ can be answered in time $O(\log n)$.*

To conclude our discussion of point location, we note that the above data structure is not the most efficient possible. Chazelle and Friedman came up with a data structure with size and preprocessing time $O(n^d/\log^{d-1} n)$ that answers point location queries in time $O(\log n)$.

3.2 Segment intersection searching

We now want to see how far we can push the methodology we have developed for point location. Consider the following problem, called *segment intersection searching.* Given a collection Γ of n hyperplanes in $\mathbb{R}^d$, we want to preprocess Γ into a data structure that, given a query segment s, will efficiently count the number of hyperplanes of Γ that intersect S, that is, $\left|\{h \in \Gamma : h \cap s \neq \emptyset\}\right|$. Surprisingly, even though this problem appears more involved than point location (it is in fact a generalization), we will see that with some cleverness we can construct a data structure with size and preprocessing time $O(n^{d+\varepsilon})$ for any ε, and query time $O(\log n)$.

Denote the two end-points of s by a and b. We begin by observing that the set of hyperplanes of Γ that intersect s is the disjoint union of two sets: The set of hyperplanes of Γ that pass below a and above b, and the set of hyperplanes of Γ that pass below b and above a. This implies that it is sufficient to preprocess Γ into a data structure that, given two query points p_1 and p_2, will count the number of hyperplanes $h \in \Gamma$ that satisfy $h(p_1) > 0$ and $h(p_2) < 0$.

To construct this data structure we build a tree T as in the case of point location. Recall that during the construction of T, when processing an internal node v we created a child u_δ for every decomposition simplex δ, associated with u_δ the set $\Gamma' \subseteq \Gamma(v)$ of hyperplanes that intersect u_δ, and then counted the number of hyperplanes $h \in \Gamma(v) \setminus \Gamma'$ that satisfy $h(p) > 0$ for $p \in \delta$, and stored this number $\mu(u_\delta)$ at u_δ. Now instead of simply storing $\mu(u_\delta)$ at u_δ, we preprocess the set $\Gamma(v) \setminus \Gamma'$ into a point location data structure that counts the number of hyperplanes $h \in \Gamma(v) \setminus \Gamma'$ that satisfy $h(q) < 0$ for a query point q. Denoting $n = \Gamma(v)$, we have seen that this data structure can be constructed in time $O(n^{d+\varepsilon})$, having size $O(n^{d+\varepsilon})$, for any $\varepsilon > 0$, with $O(\log n)$ query time. We construct this data structure and store it at u_δ, repeating the process for all children u_δ of v.

Let's analyze the maximum preprocessing time $T(n)$ of the data structure. (The storage requirement analysis is completely analogous.) For appropriate constants c_0, c_1, c_2 we can write

$$T(n) \leq \begin{cases} c_0 & n \leq n_0, \\ c_1 (r \log r)^d T\left(\frac{n}{r}\right) + c_2 A_1 n^{d+\varepsilon} (r \log r)^d & n > n_0. \end{cases}$$

Now, amazingly enough, even though we are now storing a second-level point location data structure in every node, we are claiming that the overall preprocessing (and storage) requirements increase only by a constant. Namely, we prove that $T(n) \leq A_2 n^{d+\varepsilon}$ for a sufficiently large constant A_2. The proof again proceeds by induction, with the base case $n \leq n_0$ being trivial. For $n > n_0$ we have

$$\begin{aligned} T(n) &\leq c_1(r\log r)^d A_2 \left(\frac{n}{r}\right)^{d+\varepsilon} + c_2 A_1 n^{d+\varepsilon}(r\log r)^d \\ &\leq A_2 n^{d+\varepsilon}\left(\frac{c_1(\log r)^d}{r^{\varepsilon}} + \frac{c_2 A_1 (r\log r)^d}{A_2}\right) \\ &\leq A_2 n^{d+\varepsilon}, \end{aligned}$$

when r is sufficiently large and $A_2 \geq 2c_2A_1(r\log r)^d$.

So the preprocessing time and storage requirements remain $O(n^{d+\varepsilon})$ for an arbitrarily small ε. Let us look at how a query in this data structure is handled. Given two query points p_1 and p_2, we query the main point location data structure with the point p_1 and trace a top-down path in the tree T. Again, we maintain a global counter. However, when we are at an internal node v we do not simply add a stored quantity to the counter. Rather, we query the second-level point location data structure stored at v with the point p_2, and add the result of the query to the counter. When we reach a leaf we go over the stored hyperplanes brute force and output the value of the counter.

To see that the query produces the correct number of hyperplanes, note that, along a search path, every hyperplane $h \in \Gamma$ that satisfies $h(p_1) > 0$ belongs to exactly one second-level data structure that is queried. If h also satisfies $h(p_2) < 0$ then it is counted once during that query. Overall every hyperplane h that satisfies $h(p_1) > 0$ and $h(p_2) < 0$ is counted exactly once, and no other hyperplane is counted. This proves correctness.

Now let us look at the query time. The query algorithm visits $O(\log n)$ nodes in the main search tree. At each node, the algorithm queries a second-level tree, which also takes $O(\log n)$ time. Overall, the query time is now $O(\log^2 n)$, which is not what we want! We want the query time to remain $O(\log n)$, and in fact this can be accomplished with some cleverness.

The trick is to adjust the branching factor r of the main search tree. Instead of taking r to be a large constant we are going to use $r = n^{\varepsilon'}$, for a sufficiently small ε'. Here $n = |\Gamma|$, the overall number of input hyperplanes. This now means that at every internal node v, we need an auxiliary data structure just to choose which child the query should recurse in. This is achieved by a simple point location data structure on the hyperplanes of the $(1/r)$-net. The size and construction time of the data structure are $O((r\log r)^{d+\varepsilon}) = O((n^{\varepsilon'}\log n)^{d+\varepsilon})$. Let us see how this affects the construction time $T(m)$:

$$T(m) \leq \begin{cases} c_0 & m \leq n_0, \\ c_1(n^{\varepsilon'}\log n)^d T\left(\frac{m}{r}\right) + c_2A_1m^{d+\varepsilon}(n^{\varepsilon'}\log n)^d + c_3(n^{\varepsilon'}\log n)^{d+\varepsilon} & m > n_0. \end{cases}$$

We argue by induction below that this solves to $T(m) \leq A_2 m^{d+\varepsilon} n^{\varepsilon}$ for a sufficiently large constant A_2. This means that $T(n) \leq A_2 n^{d+2\varepsilon}$. This holds for an arbitrarily small ε, so we can say that $T(n) = O(n^{d+\varepsilon})$ for any $\varepsilon > 0$. Here is the inductive argument:

$$\begin{aligned}
T(m) &\leq c_1(n^{\varepsilon'} \log n)^d T\left(\frac{m}{r}\right) + c_4 A_1 m^{d+\varepsilon}(n^{\varepsilon'} \log n)^{d+\varepsilon} \\
&\leq c_1 A_2 (n^{\varepsilon'} \log n)^d \left(\frac{m}{n^{\varepsilon'}}\right)^{d+\varepsilon} n^{\varepsilon} + c_4 A_1 m^{d+\varepsilon}(n^{\varepsilon'} \log n)^{d+\varepsilon} \\
&\leq A_2 m^{d+\varepsilon} n^{\varepsilon} \left(\frac{c_1(n^{\varepsilon'} \log n)^d}{n^{\varepsilon+\varepsilon'(d+\varepsilon)}} + \frac{c_4 A_1 (n^{\varepsilon'} \log n)^{d+\varepsilon}}{A_2 n^{\varepsilon}}\right).
\end{aligned}$$

For ε' small enough and n_0 large enough, each of the summands in the parentheses can be made smaller than $1/2$. Indeed, taking $\varepsilon' < \frac{\varepsilon}{2(d+\varepsilon)}$ and assuming n_0 is a sufficiently large constant, it follows that $(n^{\varepsilon'} \log n)^{d+\varepsilon} \leq \frac{1}{c_1} n^{\varepsilon}$. The claim easily follows, implying that $T(m) \leq A_2 m^{d+\varepsilon} n^{\varepsilon}$. This concludes the analysis of the preprocessing time and storage requirements.

The impact of making r depend polynomially on n is felt most in the query time. Notably, the height of the main search tree is now a constant bounded by $1/\varepsilon'$. So a query spends time $O(\log n)$ in each visited node, but the number of nodes visited is now constant, implying that we succeeded in reducing the overall query time to $O(\log n)$. This yields a segment intersection searching data structure with the same asymptotic performance as the point location data structure we developed previously.

3.3 Range searching

The best thing about the methodology by which we developed the segment intersection searching data structure is that it can be extended. Once we have seen how to build a two-level data structure without increasing the asymptotic construction and query times, we can do the same with a three-level data structure, a four-level data structure, or in fact have any constant number of levels while keeping the asymptotic performance bounds. This is precisely the insight we need for an efficient range-searching data structure.

The problem of simplex range searching is as follows. Given a set P of n points in $\mathbb{R}^d$, we want to preprocess P into a data structure that, given a query d-dimensional simplex S can count the number of points of P that lie in S. That is, we want the data structure to output $|P \cap S|$.

To construct the data structure we take advantage of duality. A d-dimensional simplex can be viewed as the intersection of $d+1$ linear halfspaces $H_1, H_2, \ldots, H_{d+1}$, each of the form $< \mathbf{a}_i, \mathbf{x} >> 1$ or $< \mathbf{a}_i, \mathbf{x} >< 1$, where $\mathbf{a}_i \in \mathbb{R}^d$ is a vector that specifies H_i and $\mathbf{x} \in \mathbb{R}^d$ is a variable. This specification of hyperplane equations excludes hyperplanes that pass through the origin, but they can either be assumed away due to general position or handled separately.

Now, point-hyperplane duality is a simple mapping of points to hyperplanes and hyperplanes to points. Thus a hyperplane $H = \big(< \mathbf{a}, \mathbf{x} >= 1 \big)$ is mapped to the point $H^* = \mathbf{a}$, and a point $\mathbf{a}$ is mapped to the hyperplane $\mathbf{a}^* = \big(< \mathbf{a}, \mathbf{x} >= 1 \big)$. Trivially, a point $\mathbf{y}$ is in the positive halfspace defined by a hyperplane $H = \big(< \mathbf{a}, \mathbf{x} >= 1 \big)$ (i.e., $< \mathbf{a}, \mathbf{y} >> 1$) if and only if the point H^* is in the positive halfspace defined by a hyperplane $\mathbf{y}^*$ (i.e., $< \mathbf{y}, \mathbf{a} >> 1$).

Now consider the collection of hyperplanes $P^* = \{p^* : p \in P\}$. The above discussion implies that we can answer a simplex range searching query among P by answering a $(d+1)$-level point location query in $\mathcal{A}(P^*)$. We have seen that we can preprocess $\mathcal{A}(P^*)$ in time $O(n^{d+\varepsilon})$ into a data structure that answers such queries in time $O(\log n)$. This implies a simplex range searching data structure with the same performance!

Now, simplex range searching is considered a rather cumbersome problem, and we just found a rather simple solution. This demonstrates the power of the ε-nets machinery. The amazing thing is that this solution is the best known and is essentially optimal if we want logarithmic query time.

A data structure of size $O(n^{d+\varepsilon})$ is rather large, and sometimes we might be willing to sacrifice query time for storage space. A separate methodology has been developed for constructing *linear-size* data structures. The best data structure in this vein is by Matousek (1993); it has size $O(n)$ and query time $O(n^{1-1/d})$. The above logarithmic-query-time data structure and Matousek's structure can be easily combined to produce the following *space-time trade-off*:

Theorem 3.3.1. *For any $n \le m \le n^{d+\varepsilon}$, a simplex range-counting query can be answered in time*

$$O\left(\frac{n^{1+(\varepsilon/d)}}{m^{1/d}} + \log\left(\frac{m}{n}\right)\right)$$

using $O(m)$ space.

Finally, we remark that the above results are in fact essentially optimal in a certain natural model of computation. The following theorem is due to Chazelle (1989):

Theorem 3.3.2. *Consider positive integers n, m, such that $n \le m \le n^d$, and let S be a random set of points in $[0,1]^d$. If only m units of storage are available, then with high probability, the worst-case query time for a simplex range query in S is $\Omega(n/(m^{1/d}\log n))$ in the* semigroup *model of computation.*

3.4 Nearest-neighbor search

The problem of nearest-neighbor search is as follows. Given a set P of n points in $\mathbb{R}^d$, we want to preprocess P into a data structure that, given a query point q, we wish to find the point of P that is closest to q. That is, we want to find

$$\text{argmin}_{p \in P} \sqrt{\sum_{i=1}^{d} (p_i - q_i)^2}.$$

By monotonicity of the square root function, it is sufficient to find

$$\text{argmin}_{p \in P} \sum_{i=1}^{d} (p_i - q_i)^2 = \text{argmin}_{p \in P} \left(\sum_{i=1}^{d} p_i^2 - 2\sum_{i=1}^{d} p_i q_i + \sum_{i=1}^{d} q_i^2 \right).$$

Now, notice that the term $\sum_{i=1}^{d} q_i^2$ is the same for all points p. Thus, to find the point $p \in P$ that minimizes the above expression, it is sufficient to find

$$\operatorname{argmin}_{p \in P} \left(\sum_{i=1}^{d} p_i^2 - 2 \sum_{i=1}^{d} p_i q_i \right).$$

For each $p \in P$, consider the function

$$f_p(x) = \sum_{i=1}^{d} p_i^2 - 2 \sum_{i=1}^{d} p_i x_i,$$

defined over $x \in \mathbb{R}^d$. This function is a hyperplane. A nearest-neighbor query with a point q thus seeks to find

$$\operatorname{argmin}_{p \in P} \; f_p(q).$$

This is equivalent to a point location query in the minimization diagram of the set $F = \{f_p : p \in P\}$. We now show how to construct a point location data structure for this minimization diagram that has preprocessing time and storage space $O(n^{\lceil \frac{d}{2} \rceil + \varepsilon})$ and query time $O(\log n)$. (This minimization diagram is really the Voronoi diagram of P.)

The data structure is a tree T, similar to the search tree we have built for point location in arrangements. Every node v of the tree has an associated subset $\Gamma(v) \subseteq F$. Let $n = |\Gamma(v)|$. If $n \leq n_0$, where n_0 is a global constant, the construction simply terminates and v becomes a leaf of T. Otherwise, for a global constant r, we construct in time $O(r \log r)$ a $(1/r)$-net $R \subseteq \Gamma(v)$ for the set system

$$\mathcal{S} = \Big(\Gamma(v), \big\{ \{ h \in \Gamma(v) : h \cap \delta \neq \emptyset \} : \delta \text{ is a } d\text{-simplex in } \mathbb{R}^d \big\} \Big).$$

We have now reached a stage of the construction that is crucially different from the setting of point location in arrangements. Instead of decomposing the whole $\mathcal{A}(R)$, we only decompose the region below the lower envelope.

This is accomplished as follows. We compute the minimization diagram of R. It is a convex subdivision of $\mathbb{R}^d$ of complexity $O\left((r \log r)^{\lceil \frac{d}{2} \rceil} \right)$, and therefore can be decomposed into $O\left((r \log r)^{\lceil \frac{d}{2} \rceil} \right)$ simplices in $\mathbb{R}^d$. One simple way to produce these simplices is to decompose the restriction of $\mathcal{A}(R)$ inside every hyperplane of r, and then project onto $\mathbb{R}^d$ the decomposition simplices that lie on the boundary of the lower envelope of $\mathcal{A}(R)$. This can be accomplished in time $O((r \log r)^d)$.

Every simplex δ of this minimization diagram decomposition lies in a cell of the diagram that corresponds to a single hyperplane $h \in R$. The corresponding simplex δ' in the decomposition of the region below the lower envelope of $\mathcal{A}(R)$ is simply the set of points that project onto δ and lie below r. It is easy to argue that the set of such simplices in a decomposition as desired.

Since R is a $(1/r)$-net for the set system $\mathcal{S}$, δ' is intersected by a set Γ' of at most n/r hyperplanes of $\Gamma(v)$. Crucially, for any point $q \in \delta'$, the lowest hyperplane of $\Gamma(v)$

that lies above q belongs to the set $\Gamma' \cup \{h\}$. We thus create a child $u_{\delta'}$ of v for every simplex δ' as above, associate with it the set $\Gamma' \cup \{h\}$ and recurse inside all children of v produced in this way. This completes the description of the construction.

It is easy to verify that the obvious query procedure produces the correct answer in time $O(\log n)$.

Now let us look at the construction time. We claim that the overall preprocessing time is at most $An^{\lceil \frac{d}{2} \rceil + \varepsilon}$ for any $\varepsilon > 0$ and an appropriate constant A. Let $T(n)$ be the maximum preprocessing time; then

$$T(n) \leq \begin{cases} c_0 & n \leq n_0, \\ c_1 (r \log r)^{\lceil \frac{d}{2} \rceil} T\left(\frac{n}{r} + 1\right) + c_2 n (r \log r)^d & n > n_0, \end{cases}$$

and, by induction for $n > n_0$,

$$\begin{aligned} T(n) &\leq Ac_1 (r \log r)^{\lceil \frac{d}{2} \rceil} \left(\frac{n}{r} + 1\right)^{\lceil \frac{d}{2} \rceil + \varepsilon} + c_2 n (r \log r)^d \\ &\leq Ac_1 \frac{\log^{\lceil \frac{d}{2} \rceil} r \ (n+r)^{\lceil \frac{d}{2} \rceil + \varepsilon}}{r^{\varepsilon}} + c_2 n (r \log r)^d \\ &\leq An^{\lceil \frac{d}{2} \rceil + \varepsilon} \left(\frac{c_1 \log^{\lceil \frac{d}{2} \rceil} r \ (n+r)^{\lceil \frac{d}{2} \rceil + \varepsilon}}{r^{\varepsilon} n^{\lceil \frac{d}{2} \rceil + \varepsilon}} + \frac{c_2 n (r \log r)^d}{An^{\lceil \frac{d}{2} \rceil + \varepsilon}} \right) \\ &\leq An^{\lceil \frac{d}{2} \rceil + \varepsilon} \left(\frac{c_1 \log^{\lceil \frac{d}{2} \rceil} r}{r^{\varepsilon}} \left(1 + \frac{r}{n}\right)^{\lceil \frac{d}{2} \rceil + \varepsilon} + \frac{c_2 (r \log r)^d}{A} \right) \end{aligned}$$

The parenthesized expression can be made smaller than 1 by first making r a large enough constant so that

$$\frac{c_1 \log^{\lceil \frac{d}{2} \rceil} r}{r^{\varepsilon}} < \frac{1}{4},$$

then choosing n_0 large enough so that

$$\left(1 + \frac{r}{n}\right)^{\lceil \frac{d}{2} \rceil + \varepsilon} \leq \left(1 + \frac{r}{n_0}\right)^{\lceil \frac{d}{2} \rceil + \varepsilon} < 2,$$

and then choosing

$$A > 2c_2 (r \log r)^d.$$

This implies the promised preprocessing time bound for nearest-neighbor search.

Chapter 4

Centerpoints, Planar Graphs, and Planar Separators

4.1 Rado's centerpoint theorem

Centerpoints are a natural generalization of medians to high dimensions. They are important in robust statistics, as they are a more robust measure of estimating a point set than the mean of the point set. In this brief section we will prove the existence of centerpoints and give a simple efficient algorithm for computing approximate centerpoints. This will become useful in our discussion of planar graph separators.

Theorem 4.1.1 (Rado)**.** *For any n-point set $A \subseteq \mathbb{R}^d$, there exists a point $x \in \mathbb{R}^d$ with the property that any halfspace that contains x covers at least $\frac{1}{d+1}n$ points of P. (Such a point is called a* centerpoint *of A.)*

Proof. The claim is trivial if $|A| \leq d+1$. Otherwise consider the family $\mathcal{H}$ of halfspaces that cover more than $\frac{dn}{d+1}$ points of P. We show below that $\mathcal{H}$ has a nonempty intersection. Consider a point $x \in \bigcap_{h \in \mathcal{H}} h$. By contrapositive, this point is a centerpoint, since if there exists a halfspace h that contains x that covers less than $\frac{1}{d+1}n$ points of P, the complement of h is a halfspace in $\mathcal{H}$ that does not contain x.

We now show that $\bigcap_{h \in \mathcal{H}} h \neq \emptyset$. For any $h \in \mathcal{H}$, consider the set $A|_h = A \cap h$. We prove that the family S of sets $\operatorname{conv}(A|_h)$, for all $h \in \mathcal{H}$, has a nonempty intersection, which is clearly sufficient. Consider some $d+1$ such sets. Each contains more than $\frac{dn}{d+1}$ points of A, and thus their intersection contains at least one such point. By Helly's theorem, since every $(d+1)$-tuple of sets of S intersects, the whole family S has a nonempty intersection. $\square$

It is not known how to efficiently find centerpoints that have precisely the above properties. The computation of a centerpoint can be reduced to a linear programming feasibility problem with $O(n^d)$ constraints in $\mathbb{R}^d$, and when d is constant this can be performed in time $O(n^d)$, which is highly inefficient. On the other hand, we can use the ε-sample theorem to compute an *approximate centerpoint* much more efficiently.

Theorem 4.1.2. *For any n-point set $A \subseteq \mathbb{R}^d$ and any constant $0 < \varepsilon < \frac{1}{d+1}$, there exists an algorithm that computes a point $x \in \mathbb{R}^d$ with the property that any halfspace* *that contains x covers at least $\left(\frac{1}{d+1}-\varepsilon\right)n$ points of A. The algorithm runs in time $O\left(\left(\frac{1}{\varepsilon^2}\log\frac{1}{\varepsilon}\right)^d\right)$.*

Proof. Take a random sample R of $O\left(\frac{1}{\varepsilon^2}\log\frac{1}{\varepsilon}\right)$ points of A and use linear programming to compute its centerpoint x. The ε-sample theorem guarantees that R is an ε-sample for the range space (A, H), where H is the set of subsets of A that can be cut off by hyperplanes. Consider a hyperplane that contains x. It cuts off a set $s \subseteq R$ of measure at least $\frac{1}{d+1}$. By the ε-net theorem, this hyperplane cuts off a set $s' \subseteq A$ that has measure at least $\frac{1}{d+1}-\varepsilon$. Since this holds for all hyperplanes that contain x, we have proved the theorem. □

4.2 Drawing graphs in the plane

As we have seen in class, graphs are often visualized by drawing them in the plane—vertices are drawn as points, and edges as curved segments (called *arcs*) that connect the corresponding points. A graph together with a drawing of it in the plane is called a *topological graph*.

A graph is called *planar* if there exists a drawing of it in which the interior of any arc does not touch or intersect any other arc. That is, two distinct arcs are either disjoint or touch at endpoints that they share. A planar graph together with a planar drawing of it is called a *plane graph*.

It is easy to verify that paths, cycles and trees of any size are planar. Transportation networks often provide examples of planar graphs, and graph planarity became important in computer science due to a connection with VLSI circuit design. Planar drawings are often considered superior when visualizing graphs, as they have no edge crossings that can be mistaken for vertices. In fact, a whole subfield of computer science called *graph drawing* is devoted to the study of various kinds of drawings of graphs.

It might not be obvious at first that there are any nonplanar graphs at all. There are, but we'll have to do some work to prove this, and we'll need two preliminary steps just to approach this issue. The first is to define the *faces* of a plane graph and the second is to mention the (in)famous Jordan curve theorem.

Let us begin with faces. Define an equivalence relation on the plane as follows: Two points $a, b \in \mathbb{R}^2$ are equivalent if they can be connected by an arc that does not intersect the edges of a given plane graph G. Then the set of all points that belong to a particular equivalence class of this relation are said to be a *face* of G. Intuitively, if we draw G on a white sheet of paper with a black pencil, the faces are the white regions; alternatively, if we cut the paper along the edges of the drawing, the faces are the resulting pieces. Note that faces are defined for plane graphs, but not for planar graphs without a drawing: Different drawings of the same graph can produce different sets of faces!

The second piece of mathematical equipment we'll need to study planar graphs is the Jordan curve theorem.[1] It is a classical example of a mathematical statement

[1]Jordan gets all the press even though his proof of the theorem was wrong, and it took almost

that is intuitively obvious, but exceedingly difficult to prove. (Related specimens that arguably fall into this category are Kepler's conjecture and the Kneser-Poulsen conjecture.)

Theorem 4.2.1 (Jordan curve theorem)**.** *Every closed non-self-intersecting curve γ in the plane separates the plane into exactly two regions, one bounded and one unbounded, such that γ is the boundary of both. Alternatively, a plane drawing of any cycle C_i, for $i \geq 3$, has exactly two faces.*

To see why the Jordan curve theorem is not so easy to prove recall that there are some crazy curves out there—just think about fractals like the *Koch snowflake*. How would you go about proving that such monsters have "interior" and "exterior"?

The following corollary follows from the Jordan curve theorem by routine arguments.

Corollary 4.2.2. *Consider a plane graph G and an edge e that is part of a cycle in G. Then e lies on the boundary of exactly two faces of G.*

4.3 Euler's formula

The fundamental tool in the study of planar graphs is Euler's formula, presented by Euler in 1752.[2]

Theorem 4.3.1 (Euler's formula)**.** *Let G be a connected plane graph with n vertices, e edges, and f faces. Then*

$$n - e + f = 2.$$

Note that the theorem need not hold if the graph is not connected—Just think of a collection of isolated vertices. On the other hand, the formula remains true even for (non-simple) graphs with multiple edges and self-loops.

Proof. The proof proceeds by induction on the number of edges. If there are none, the graph consists of a single vertex, the drawing has one face, and the formula holds as $1 - 0 + 1 = 2$. Assume that the formula holds for all plane graphs having k edges. Consider a plane graph $G = (V, E)$ with n vertices, f faces, and $k + 1$ edges. We distinguish between two cases:

G **is a tree.** In this case $n = k + 2$, due to a tree characterization we have seen previously, and $f = 1$ since any planar drawing of a tree has exactly one face. Then the formula holds as $(k + 2) - (k + 1) + 1 = 2$.

G **has a cycle** C**.** Take an edge e that lies on C and consider a plane graph $G' = (V, E \setminus \{e\})$, whose vertices and edges are drawn as in G. By Corollary 4.2.2, the edge e is adjacent to two faces of G, and these faces "merge" into one in G'. Thus G' has n vertices, $f - 1$ faces, and k edges. By the induction hypothesis, $n - k + (f - 1) = 2$, hence $n - (k + 1) + f = 2$.

20 years until Veblen found a correct one in 1905.

[2]Caution: Due to Euler's prodigious output, there are multiple "Euler's formulae", "Euler's theorems", etc.

This completes the proof by induction. □

Euler's formula implies that the number of faces of a plane graph does not depend on the drawing, so even though the faces themselves are only defined for a particular drawing, their number is fixed *a priori* for any planar graph! The formula has a number of other consequences that are frequently used in theoretical computer science. These consequences say that planar graphs have few edges, and always have at least one low-degree vertex. As they make abundantly clear, not only are not all graphs planar, but *most* graphs aren't. (Do you understand the sense in which the theorem below implies this?)

Theorem 4.3.2. *For any simple planar graph G with n vertices and e edges:*

(a) If $n \geq 3$ then $e \leq 3n - 6$. If $e = 3n - 6$ then every face of G is a 3-cycle (a "triangle") and G is called a triangulation.

(b) There is a vertex of G that has degree at most 5.

Proof. The proofs of the two parts are similar in their clever use of Euler's formula:

(a) If G is not connected, we can add edges to connect G while maintaining its planarity. Assume therefore that G is connected. Let f be the number of faces of G. For such a face t, let $\alpha(t)$ be the number of edges adjacent to t and consider the sum $\sum_t \alpha(t)$ that ranges over all faces t of G. As each edge is adjacent to at most two faces, a particular edge is counted at most twice in the above sum. Hence
$$\sum_t \alpha(t) \leq 2e.$$
On the other hand, each face has at least three edges on its boundary, so
$$\sum_t \alpha(t) \geq 3f.$$
We get $3f \leq 2e$, and, using Euler's formula, $3(2 - n + e) \leq 2e$ and
$$e \leq 3n - 6.$$
Finally, if $e = 3n - 6$ then $3f = 2e$ and it must be that every face has exactly three edges on its boundary.

(b) If the graph is disconnected we consider one particular connected component of it, so assume that G is connected. If G has two vertices or less the result is immediate, so assume that $n \geq 3$. Recall that $d_G(x)$ denotes the degree of a vertex x in G. The sum $\sum_x d_G(x)$, ranging over the vertices x of G, counts every edge twice, so
$$\sum_x d_G(x) = 2e.$$

As we have seen, $e \leq 3n - 6$, so

$$\sum_x d_G(x) \leq 6n - 12.$$

If the degree of every vertex is at least 6, we get

$$6n \leq 6n - 12,$$

which is a contradiction. Therefore, there must be a vertex with degree at most 5.

□

An intuitive way to think about Theorem 4.3.2(a) is that once a graph has too many edges, there is no more room for them in the plane and they start intersecting. This gives a way to prove that a particular graph is not planar. Take K_5, for example. It has 5 vertices and 10 edges, and $10 > 3 \cdot 5 - 6$. Thus K_5 is not planar! In fact, no K_n is planar for $n \geq 5$, since they all contain K_5 as a subgraph. On the other hand, K_n is planar for $n \leq 4$, as can be demonstrated by their simple planar drawings. This illustrates a point that might be obvious by now: proving a graph to be planar is often easier than proving the opposite. (Just draw it!™)

How about complete bipartite graphs? It is easy to verify that $K_{i,j}$ is planar when $i \leq 2$ or $j \leq 2$. The smallest remaining suspect is $K_{3,3}$. Playing around with drawings doesn't help: There seems to be no way to draw $K_{3,3}$ without intersections. Let's try the trick that worked for K_5: The number of vertices of $K_{3,3}$ is 6, its number of edges is 9, and $9 \leq 3 \cdot 6 - 6$. No luck. We need a stronger tool, and here it is:

Proposition 4.3.3. *For any simple planar graph G with n vertices and e edges, if G does not contain a cycle of length 3 then $e \leq 2n - 4$.*

Proof. We can assume that G is connected as in Theorem 4.3.2. Let f be the number of faces of G and let $\alpha(t)$ be the number of edges adjacent to a face t. These edges make up a cycle in G, and thus their number is at least 4, implying $\alpha(t) \geq 4$. Consider the sum $\sum_t \alpha(t)$, over all faces t of G. Each edge is adjacent to at most two faces, thus

$$4f \leq \sum_t \alpha(t) \leq 2e.$$

Using Euler's formula, we get $4(2 - n + e) \leq 2e$ and $e \leq 2n - 4$. □

With this result we're finally in business: $K_{3,3}$ does not contain an odd cycle since it is bipartite, thus every cycle in the graph has length at least 4. Since $9 > 2 \cdot 6 - 4$, $K_{3,3}$ is not planar. Let's summarize what we've learned.

Theorem 4.3.4. *K_n is planar if and only if $n \leq 4$ and $K_{i,j}$ is planar if and only if $i \leq 2$ or $j \leq 2$.*

At this point we have laid the groundwork for one of the most striking results concerning planar graphs, known as Kuratowski's theorem. To state it we need the following definition:

Definition 4.3.5. *Given a graph $G = (V, E)$, an* edge subdivision *operation on an edge $\{u, v\}$ of G results in the graph $(V \cup \{x\}, (E \setminus \{\{u, v\}\}) \cup \{\{u, x\}, \{x, v\}\})$, where $x \notin V$ is a new vertex. A graph G' is said to be a* subdivision *of G if it can be obtained from G by successive applications of edge subdivision.*

Kuratowski's theorem says that not only are K_5 and $K_{3,3}$ non-planar, but every non-planar graph contains either a subdivision of K_5 or a subdivision of $K_{3,3}$. That is, the graphs K_5 and $K_{3,3}$ characterize the whole family of non-planar graphs!

Theorem 4.3.6 (Kuratowski's theorem)**.** *A graph is planar if and only if it does not contain a subdivision of K_5 or a subdivision of $K_{3,3}$ as a subgraph.*

4.4 Coloring planar graphs

You might have heard of the four-color problem. It was posed in the mid-19th century and occupied some of the best discrete mathematicians since that time. The original formulation is in terms of political maps. In such maps, neighboring countries are drawn with different colors. The question is how many colors are needed. It is easy to construct simple examples of maps that need at least four colors. The four color problem asks whether four colors always suffice, for any political map. (We require that every country is *connected*, unlike, say, the US.)

This problem is equivalent to whether every planar graph can be colored with four colors. (To see this, construct a graph whose vertices correspond to countries and whose edges connect neighbors through border segments.) It took over a century until Appel and Haken found a proof that four colors always suffice, and even that was possible only by using computers to conduct extensive case enumeration and analysis. To this date no proof of the four color theorem is known that does not rely on computers. On the other hand, in 1890 Heawood discovered a beautiful proof that *five* colors always suffice. To prepare for his proof, let us warm up by showing that every planar graph can be colored with 6 colors. The proof is surprisingly simple.

Theorem 4.4.1. *The chromatic number of a planar graph G is at most six.*

Proof. By induction on the number n of vertices of G. If $n \leq 6$ the claim is trivial. Assume every planar graph with at most k vertices can be colored with 6 colors or less, and consider a graph $G = (V, E)$ with $k + 1$ vertices. By Theorem 4.3.2(b), there is a vertex v of G with degree at most 5. Let G' be the induced subgraph of G on the vertices $V \setminus \{v\}$. By the induction hypothesis, G' can be colored with five colors or less. Color the vertices $V \setminus \{v\}$ of G with the colors that they are assigned in the coloring of G'. Assign to v the color that is not used by its neighbors. Since the degree of v is at most five, such a color exists. This specifies a valid coloring of G. □

We are now ready for Heawood's five color theorem.

Theorem 4.4.2. *The chromatic number of a planar graph $G = (V, E)$ is at most five.*

Proof. The proof proceeds by induction on the number n of vertices of G. The base case is trivial. Assume every planar graph with at most k vertices can be colored with 5 colors or less, and consider a graph $G = (V, E)$ with $k + 1$ vertices. Let v be a vertex of G with degree at most 5. If $d_G(v) < 5$ we can produce a 5-coloring of G as in the proof of Theorem 4.4.1. Assume $d_G(v) = 5$ and let $c : (V \setminus \{v\}) \rightarrow \{1, 2, 3, 4, 5\}$ be a 5-coloring of the induced subgraph G' of G on the vertices $V \setminus \{v\}$. This coloring exists by the induction hypothesis.

We consider a particular drawing of G in the plane and henceforth regard G as a plane graph. Let v_1, v_2, v_3, v_4, v_5 be the neighbors of v in the order they appear around v in G. (That is, according to one of the circular orders in which the corresponding edges emanate from v in G.) Without loss of generality, assume that $c(v_i) = i$ for $1 \leq i \leq 5$. (Note that if some color is unused by v_1, v_2, v_3, v_4, v_5, we can simply assign that color to v.) We distinguish between two cases: Either there does not exist a path between v_1 and v_3 in G that uses only vertices of colors 1 and 3, or there does.

There is no such path. In this case consider the subgraph G'' of G that is the union of all paths that begin at v_1 and use only vertices with colors 1 and 3. Note that neither v_3 nor its neighbors belong to G''. We produce a 5-coloring of G as follows: All the vertices of G'' of color 1 are assigned the color 3, all the vertices of G'' of color 3 are assigned the color 1, the vertex v is assigned the color 1, and all other vertices of G keep the color assigned by c. No monochromatic edges are created by this assignment and the coloring is valid.

There is such a path. Consider a path P from v_1 to v_3 that uses only vertices with colors 1 and 3. Together with the edges $\{v, v_1\}$ and $\{v, v_3\}$ this forms a cycle. The vertices v_2 and v_4 lie on different sides of this cycle. (Here we use the Jordan curve theorem.) Therefore there is no path between v_2 and v_4 that uses only vertices with colors 2 and 4, and we can apply the reasoning of the previous case.

□

4.5 Koebe's Theorem

Two of the most striking results associated with planar graphs are Fáry's theorem and Koebe's theorem. Fáry's theorem states that every planar graph can be drawn in the plane without edge crossings, such that all the arcs are *straight line segments*. Koebe's theorem says that every planar graph is in fact isomorphic to an "incidence graph" of a collection of nonoverlapping discs in the plane. (The vertices of this graph correspond to the discs, and two vertices are adjacent if and only if the corresponding disks are tangent.) Fáry's theorem is an immediate consequence of Koebe's theorem, although they were discovered independently. We will give a complete proof of Koebe's theorem.

Theorem 4.5.1 (Koebe (1936)). *Given any planar graph G, with vertex set $\{v_1, v_2, \ldots, v_n\}$, we can find a packing of a collection of circular disks $\mathcal{C} = \{C_1, C_2, \ldots, C_n\}$ in the plane with the property that C_i and C_j touch iff $\{v_i, v_j\}$ is an edge of G, for $1 \leq i, j \leq n$.*

Proof. First observation: We can assume all faces are (combinatorial) triangles, i.e., the graph is *maximal planar*. Indeed, if this is not the case, put a vertex in every face of G and connect it to all the vertices on the boundary of the face, obtaining a graph G'. Suppose we can represent G' as the theorem states. Then we simply remove the disks corresponding to the artificial vertices in G' from the representation and obtain a valid representation for G.

Let V, E, F be the vertex, edge, and face sets of G. (F includes the unbounded face.) By Euler's formula,

$$|V| - |E| + |F| = 2,$$

and since $3|F| = 2|E|$ we get

$$|F| = 2|V| - 4 = 2n - 4.$$

Now, let r_i be the radius of a hypothetical disk corresponding to v_i and let $r = (r_1, r_2, \ldots, r_n)$ be the vector of these radii. We can assume without loss of generality that $\sum_{i=1}^{n} r_i = 1$. For a face $\{v_i, v_j, v_k\}$ of F, consider a triangle formed by three touching disks of radii r_i, r_j, r_k. Let $\sigma_r(v_i)$ be the sum of the angles incident to v_i in the triangles that correspond as above to the faces of G incident to v_i. Without loss of generality, let the three vertices of the unbounded face be v_1, v_2, v_3. The following claim is intuitively obvious, and we will not prove it formally:

Claim 4.5.2. *For a Koebe representation with radius vector r to exist it is sufficient that $\sigma_r(v_i) = 2\pi$ for all vertices other than v_1, v_2, v_3.*

Our goal is thus to prove that there exists some radius vector r with $\sum_{i=1}^{n} r_i = 1$, such that the *angle vector* $(\sigma_r(v_1), \sigma_r(v_2), \ldots, \sigma_r(v_n))$ equals, say,

$$x^* = \left(\frac{2\pi}{3}, \frac{2\pi}{3}, \frac{2\pi}{3}, 2\pi, 2\pi, \ldots, 2\pi\right).$$

We know from the above discussion on Euler's formula that

$$\sum_{i=1}^{n} \sigma_r(v_i) = |F|\pi = (2n-4)\pi.$$

Let $S \subseteq \mathbb{R}^n$ define the simplex of valid radius vectors:

$$S = \left\{ r = (r_1, r_2, \ldots, r_n) : r_i > 0 \text{ for all } i, \text{ and } \sum_{i=1}^{n} r_i = 1 \right\}.$$

Also let H be the hyperplane of potential angle vectors:

$$H = \left\{ x = (x_1, x_2, \ldots, x_n) : \sum_{i=1}^{n} x_i = (2n-4)\pi \right\}.$$

Consider the continuous mapping $f : S \to H$, where

$$f(r) = \big(\sigma_r(v_1), \sigma_r(v_2), \ldots, \sigma_r(v_n)\big).$$

We have finally reached a point at which we can give the broad outline of the proof. The above discussion implies that to prove Koebe's theorem it is sufficient to show that x^* lies in the image of f. We accomplish this through the following four steps.

Step 1. Show that $f : S \to H$ is one-to-one.

Step 2. Show that the image of f lies in the interior of a certain polytope $P^* \subset H$.

Step 3. Use the property that f is one-to-one to prove that $f : S \to P^*$ is *onto.*

Step 4. Prove that $x^* \in P^*$, which by Step 3 implies that x^* lies in the image of f.

Step 1.

Claim 4.5.3. *$f : S \to H$ is a one-to-one mapping.*

Proof. Pick any $r, r' \in S$ and let I denote the set of indices i for which $r_i < r'_i$. Note that $I \neq \emptyset$ and $I \neq \{1, 2, \ldots, n\}$. Consider a triangle $v_i v_j v_k$ determined by touching disks of radii r_i, r_j, r_k. If we increase r_i but decrease or leave as is the radii r_j, r_k, then the angle at v_i will decrease. Similarly, if we increase r_i and r_j but decrease or leave as is the radius r_k then the sum of the angles at v_i and v_k will decrease. These are simple consequences of the law of cosines. Thus,

$$\sum_{i \in I} \sigma_r(v_i) > \sum_{i \in I} \sigma_{r'}(v_i), \tag{4.1}$$

which yields $f(r) \neq f(r')$. □

Step 2. Let $s = (s_1, s_2, \ldots, s_n)$ be a boundary point of S and let I denote the set of indices for which $s_i = 0$. Note that if r tends to s then in each triangle that has at least one vertex from the set $\{v_i : i \in I\}$, the sum of the angles at these vertices tends to π. Hence

$$\lim_{r \to s} \sum_{i \in I} \sigma_r(v_i) = |F(I)|\pi, \tag{4.2}$$

where $F(I)$ denotes the set of faces of G with at least one vertex in $\{v_i : i \in I\}$.

For any fixed $r \in S$ and for any nonempty proper subset $I \subset \{1, 2, \ldots, n\}$, there exists a point $s \in \partial S$ with $s_i = 0$ for $i \in I$ and $s_i > r_i$ for $i \notin I$. If we move r towards s along a line, then by inequality (4.1), $\sum_{i \in I} \sigma_r(v_i)$ will increase. Then by the limit equation (4.2), we get

$$\sum_{i \in I} \sigma_r(v_i) < |F(I)|\pi. \tag{4.3}$$

Note that this holds for any subset I. This implies that the image of the map f lies in the interior of the convex polytope P^* determined by the relations

$$\sum_{i=1}^{n} x_i = (2n - 4)\pi, \text{ and } \sum_{i \in I} x_i < |F(I)|\pi \text{ for all } \emptyset \subset I \subset \{1, 2, \ldots, n\}.$$

Step 3.

Claim 4.5.4. *Let $g : A \to B$ be a continuous one-to-one map, where A and B are topological balls in $\mathbb{R}^d$. If all limit points of $f(x)$, as x tends to ∂A, lie on ∂B, then $g : A \to B$ is onto.*

The claim is elementary topology and we do not prove it. In our case, f is a continuous one-to-one map from the interior of S to the interior of P^*, and by the limit equation (4.2), all limit points of $f(r)$, as r tends to ∂S, lie on ∂P^*. Thus the claim implies that $f : S \to P^*$ is a surjective mapping.

Step 4. It now remains to show that $x^* = (x_1^*, x_2^*, \ldots, x_n^*)$ belongs to P^*, namely that it satisfies all the inequalities in the definition of P^*. Let I be a subset as above. If $|I| \geq n-2$, then all faces of G have at least one vertex in $\{v_i : i \in I\}$, implying $|F(I)| = 2n-4$. In this case,

$$\sum_{i \in I} x_i \leq 2\pi(n-3) + \frac{4\pi}{3} < (2n-4)\pi = |F(I)|\pi,$$

as required. To handle the remaining subsets I we have the following claim:

Claim 4.5.5. *For any subset $I \subset \{1, 2, \ldots, n\}$ with $1 \leq |I| \leq n-3$, G has more than $2|I|$ faces that have at least one vertex belonging to $U = \{v_i : i \in I\}$. (Call these faces $F(U)$.)*

Proof. Let F' be the set of faces *all* of whose vertices belong to $U' = V \backslash U$. These faces do not form a perfect triangulation of the vertex set U'. (Otherwise they would cover "the whole plane".) Since for perfect triangulations we have seen that $f = 2n-4$, in our case

$$\begin{aligned}
& |F'| < 2|U'| - 4 \\
\Rightarrow\ & |F| - |F(U)| < 2(|V| - |U|) - 4 \\
\Rightarrow\ & 2n - 4 - |F(U)| < 2n - 2|U| - 4 \\
\Rightarrow\ & |F(U)| > 2|U| = 2|I|
\end{aligned}$$

□

The claim implies that for any subset $I \subset \{1, 2, \ldots, n\}$ with $1 \leq |I| \leq n-3$,

$$\sum_{i \in I} x_i^* \leq 2\pi|I| < |F(I)|\pi.$$

This finally implies that x^* belongs to P^* and thus lies in the image of f, which concludes the proof of Koebe's theorem. □

4.6 Existence of Planar Separators

We will now use Koebe's Theorem to prove a theorem in the spirit of Lipton and Tarjan (1979). This theorem, and particularly its extensions, has important applications in the design of divide-and-conquer algorithms.

Theorem 4.6.1. *Let G be a planar graph with n vertices. Then the vertex set of G can be partitioned into three sets A, B, C, such that $|A|, |B| \leq \frac{3}{4}n$, $|C| < 2\sqrt{n}$, and no vertex of A is adjacent to any vertex of B.*

In other words, there exists a small set B whose removal disconnects the graph. This set is called a *separator* of G.

Proof. Consider a Koebe representation of the graph G that assigns to the vertices of G the points $P = \{p_1, p_2, \ldots, p_n\}$ in the plane. A *circle-preserving* (or *conformal*) map is a function that maps circles to circles. For example, translations, rotations, and dilations of the plane are circle-preserving maps from $\mathbb{R}^2$ to $\mathbb{R}^2$. It can be shown that a stereographic projection from $\mathbb{R}^2$ to $\mathbb{S}^2$, or from $\mathbb{S}^2$ to $\mathbb{R}^2$ is a circle-preserving map.

We will apply a circle-preserving map Φ to the Koebe representation of G that will produce a packing of circular caps on $\mathbb{S}^2$ that correspond to vertices of G, such that two caps touch iff the corresponding vertices of G are adjacent. We will take particular care to ensure that the origin is a centerpoint of $\Phi(P)$. We will then show that in this case a random plane through the origin produces a separator.

Step 1: Mapping to a circle packing on a sphere. Let us begin by describing a sphere-preserving map with the required properties. Let Π be a stereographic projection of $\mathbb{R}^2$ onto $\mathbb{S}^2$, and let c be a centerpoint of $\Pi(P)$. Let U_c be a rotation in $\mathbb{R}^3$ for which $c' = U_c(c) = (0, 0, \|c\|)$. Let Q be the point set $U_c \circ \Pi(P)$. Let $D_\alpha = \Pi \circ (\alpha I) \circ \Pi^{-1}$ be the map from $\mathbb{S}^2$ to itself that first projects the sphere onto the plane, then dilates the plane, and then projects the plane back onto the sphere. We show below that if $\alpha = \sqrt{(1 - \|c\|)/(1 + \|c\|)}$ then the origin is a centerpoint of $D_\alpha(Q)$. Thus $\Phi := D_\alpha \circ U_c \circ \Pi$ is a sphere-preserving map with the required properties.

A circle on $\mathbb{S}^2$ is the intersection of $\mathbb{S}^2$ with a plane. Let $\Gamma_{c'}$ be the set of circles on $\mathbb{S}^2$ whose hyperplanes contain c'. We now show that for any circle $h \in \Gamma_{c'}$, $D_\alpha(h)$ is a great circle on $\mathbb{S}^2$. First consider the circle $h_0 \in \Gamma_{c'}$ whose hyperplane is orthogonal to the z-axis. By similarity of triangles, the radius of $\Pi^{-1}(h_0)$ equals

$$\frac{\sqrt{1 - \|c\|^2}}{1 - \|c\|} = \sqrt{\frac{1 + \|c\|}{1 - \|c\|}}.$$

Thus D_α maps h_0 to the equator of $\mathbb{S}^2$. Now let h be any circle on $\mathbb{S}^2$ that contains c'. This circle intersects h_0 at two antipodal points on h_0. As above, we can show that D_α maps these two points to two antipodal points on the equator of S^2. Since $D_\alpha(h)$ contains these two points, it also contains the origin.

This means that any hyperplane that contains the centerpoint c is mapped by D_α to a hyperplane that contains the origin. Therefore any hyperplane that contains the origin has at least $\frac{1}{4}n$ points of $\Phi(P)$ on both sides, which implies that the origin is a centerpoint of $\Phi(P)$.

Step 2: A random plane yields a good separator. Let $\mathcal{C} = \{C_1, C_2, \ldots, C_n\}$ be a set of spherical caps on $\mathbb{S}^2$, such that C_i corresponds to the point p_i of P. For any plane Π through the origin, we can consider the set $C(\Pi) \subseteq \mathcal{C}$ of caps that are hit by Π, and sets $A(\Pi), B(\Pi) \subseteq \mathcal{C} \setminus C(\Pi)$ of caps that lie above (resp., below) Π. Since the origin is a centerpoint of $\Phi(P)$, the size of each of $A(\Pi)$ and $B(\Pi)$ is at most $\frac{3}{4}n$. We show below that there exists a plane that contains the origin and intersects less than $2\sqrt{n}$ caps, which implies the theorem.

By the radius of a spherical cap we mean the radius of the planar disk that supports it. Let r_i denote the radius of C_i. The locus of points antipodal to the great circles that intersect C_i forms a circular band B_i on $\mathbb{S}^2$ of width $2r_i$, where by the width of a band we mean the width of the slab that supports it. Consider a random hyperplane through the origin. It intersects $\mathbb{S}^2$ at a random great circle. The expected number of caps C_i hit by this great circle is the same as the expected number of belts B_i hit by a random point on $\mathbb{S}^2$, which is simply the total area of the belts divided by the area of the sphere.

The area of B_i is $4\pi r_i$. This implies that the above expectation is

$$\frac{\sum_{i=1}^{n} \mathrm{Area}(B_i)}{4\pi} = \sum_{i=1}^{n} r_i.$$

Clearly, the area of C_i is at least πr_i^2. The area of the whole sphere $\mathbb{S}^2$ is 4π. The caps $C_1, C_2, \ldots, C_n$ form a packing, which implies

$$\sum_{i=1}^{n} \pi r_i^2 < \sum_{i=1}^{n} \mathrm{Area}(C_i) \leq 4\pi$$

and

$$\sum_{i=1}^{n} r_i^2 < 4.$$

Recall Jensen's Inequality: If f is convex then "the function of a convex combination of x_i's is at most the convex combination of the $f(x_i)$'s." A special case is

$$f\left(\frac{1}{n}\sum_{i=1}^{n} x_i\right) \leq \frac{1}{n}\sum_{i=1}^{n} f(x_i).$$

In our case, Jensen's inequality implies

$$4 > \sum_{i=1}^{n} r_i^2 = n\left(\frac{1}{n}\sum_{i=1}^{n} r_i^2\right) \geq n\left(\frac{1}{n}\sum_{i=1}^{n} r_i\right)^2 = \frac{1}{n}\left(\sum_{i=1}^{n} r_i\right)^2,$$

which in turn leads to

$$\sum_{i=1}^{n} r_i < 2\sqrt{n}.$$

This concludes the proof of the theorem. $\square$

This proof implies that trying a number of random planes will produce a separator with probability arbitrarily close to 1. Given a Koebe representation of G, the only algorithmically non-trivial step in the above proof is finding a centerpoint c of the set $\Pi(P)$. However, the proof works just as well with an approximate centerpoint, resulting in a separator of size at most $2\sqrt{n}$ that produces two vertex sets A, B, such that $|A|, |B| \leq (\frac{3}{4} + \varepsilon)n$ and no vertex of A is adjacent to any vertex of B.

Thus a separator for a planar graph can be produced in linear time given a Koebe representation of the graph. Actually, having a Koebe representation is not a very realistic requirement for general planar graphs, as no polynomial-time algorithm for constructing such a representation is known. The real power of geometric separators is not in their applications to general planar graphs, but in the construction of separators of systems of sparsely-overlapping spheres.

(Note that the original Lipton-Tarjan paper described separators of size at most $2\sqrt{2n}$ that produce parts of size at most $\frac{2}{3}n$, and gave a linear-time algorithm for computing them. Their proof is limited to planar graphs.)

Definition 4.6.2. *A k-ply system in $\mathbb{R}^d$ is a set of n balls in $\mathbb{R}^d$ such that no point in $\mathbb{R}^d$ lies in the interior of more than k balls.*

So a Koebe representation is a 1-ply system in $\mathbb{R}^2$. The above separator construction can be easily extended to provide the following result.

Theorem 4.6.3. *Suppose Γ is a k-ply system of n balls in $\mathbb{R}^d$. Then there is a sphere S that intersects a set $\Gamma_O(S)$ of balls, with a set $\Gamma_I(S) \subset \Gamma$ lying inside S, a set $\Gamma_E(S) \subset \Gamma$ lying outside S, and*

$$|\Gamma_O(S)| = O(k^{1/d} n^{1-1/d}),$$

and

$$|\Gamma_I(S)|, |\Gamma_E(S)| \leq \frac{d+1}{d+2} n.$$

Furthermore, for any constant $0 < \varepsilon < 1/(d+2)$ we can compute a sphere S such that $|\Gamma_I(S)|, |\Gamma_E(S)| \leq \left(\frac{d+1}{d+2} + \varepsilon\right) n$ and $|\Gamma_O(S)| = O(k^{1/d} n^{1-1/d})$ with probability at least $1/2$ in time $O(nd)$.

k-ply systems are prevalent. For example, the k-nearest neighbor graph in $\mathbb{R}^d$ gives rise to an $O(k)$-ply system. The above separator theorem has been applied to finite-element mesh processing and the solution of sparse linear systems, among others.

Chapter 5

Well-separated Pair Decompositions and Their Applications

5.1 WSPD Definition and Applications

A well-separated pair decomposition is a data structure on point sets in $\mathbb{R}^d$. It has linear size and can be constructed by a simple and fast algorithm. In light of its simplicity, it has a truly amazing array of powerful applications. It is like a "magic bullet" that generates simple and elegant solutions to a whole battery of problems.

A WSPD of a point set P is a collection of pairs of subsets $\big\{\{A_1, B_1\}, \{A_2, B_2\}, \ldots, \{A_m, B_m\}\big\}$ such that $A_i, B_i \subseteq P$, and the following two properties hold:

(a) Every pair of points is represented in *exactly one* pair $\{A_i, B_i\}$ (i.e, for all distinct $p, q \in P$, there is a unique i for which $p \in A_i, q \in B_i$). (Note that the pairs $\{A_i, B_i\}$ are unordered.)

(b) For every $1 \leq i \leq m$, A_i and B_i can be respectively enclosed by two balls of diameter D, such that the distance between the balls is at least sD, where s is a global strictly positive parameter.

The second condition is called the separation condition and s the separation parameter. A well-separated decomposition with separation parameter s can be concisely referred to as an s-WSPD.

Every point set P has an obvious s-WSPD for any $s > 0$: Just take the collection of all pairs $\{\{p\}, \{q\}\}$ for $p, q \in P$. The amazing thing, however, first observed by Callahan and Kosaraju, is that every point set has an s-WSPD with a *linear* number of pairs for any constant s.

Theorem 5.1.1. *Given a set P of n points in $\mathbb{R}^d$, where $d = O(1)$, an s-WSPD with $O(s^d n)$ pairs can be constructed in time $O(n \log n + s^d n)$.*

Before we show how to construct WSPDs with the bounds given in the theorem, let us see some of the applications of this data structure.

5.1.1 Closest pair

Let us see how to find the closest pair of points in P in time $O(n \log n)$ using a WSPD. This is of course not the fastest known algorithm, but is quite instructive.

Construct an s-WSPD for any $s > 1$. Consider the closest pair of points $p, q \in P$, and let $\{A, B\}$ be the pair in the WSPD such that $p \in A$ and $q \in B$. We claim that $A = \{p\}$ and $B = \{q\}$, i.e., that A and B are in fact singletons. Indeed, suppose A also contains some point $p' \neq p$. By the separation criterion, the points of A and of B can be respectively enclosed in two balls of diameter D, such that the distance between the balls is at least sD. Thus $\|p-p'\| \leq D$ and $\|p-q\| \geq sD > D \geq \|p-p'\|$, which is a contradiction to p, q being the closest pair.

This means that the closest pair is a singleton pair in the WSPD. We can thus go over the WSPD and among all such pairs take the one with the smallest distance. This is guaranteed to be the closest pair.

5.1.2 All nearest neighbors

Instead of just finding the closest pair in P, let us use the WSPD to output all nearest neighbor pairs, i.e. the pairs $p, q \in P$ such that q is the nearest neighbor of p in P. Note that this relationship is asymmetric, so the number of nearest neighbor pairs is between $n/2$ and n. We can find them all in time $O(n \log n)$ using a WSPD.

Construct an s-WSPD for $s > 1$. By a similar argument to the one we used for the closest pair, if q is the nearest neighbor of p in P, the s-WSPD contains a pair of the form $\{\{p\}, Q\}$, where $Q \subseteq P$ and $q \in Q$. We can thus go over the data structure, and for each pair one of the members of which is a singleton $\{p\}$, iterate over the other member Q, pick the point $q \in Q$ closest to p, and associate it with p if the point currently associated with p is farther away than q. This produces all the nearest neighbor pairs.

5.1.3 Spanners

A t-spanner for a weighted graph G is a graph S on the same vertex set such that the graph distance between any two vertices in S is at least their distance in G and at most t times that distance. More precisely,

$$d_G(x, y) \leq d_S(x, y) \leq t \cdot d_G(x, y).$$

A spanner provides an approximation for a shortest path metric. This is particularly interesting when the spanner has much fewer edges than the original graph. Suppose the graph G represents the Euclidean metric on a set P of n points in $\mathbb{R}^d$. We shall see that G can be approximated arbitrarily well by a spanner with a *linear* number of edges. Intuitively, a linear number of distances suffice to approximately encode all the distances between n points in $\mathbb{R}^d$.

Construct an s-WSPD for P with a parameter s that will be specified below. The spanner S for the Euclidean metric on P is constructed as follows: For every pair $\{A, B\}$ of the WSPD, take an arbitrary $p \in A$, $q \in B$, and add the edge $\{p, q\}$ to S

with weight $\|p-q\|$. This produces a graph S with a linear number of edges in time $O(n \log n)$. Let us show that

$$\|x-y\| \leq d_S(x,y) \leq t\|x-y\|$$

for all $x, y \in P$.

The inequality $\|x-y\| \leq d_S(x,y)$ is obvious by the triangle inequality: if there is a path from x to y in S, its weight is the sum of Euclidean distances along a continuous path from x to y in $\mathbb{R}^d$, and this can never be less than $\|x-y\|$. We now need to demonstrate a path from x to y in S the weight of which is at most $t\|x-y\|$. Let us construct such a path recursively. Consider the pair $\{A, B\}$ of the WSPD such that $x \in A$ and $y \in B$. There is an edge of $\{p, q\}$ in S such that $p \in A$ and $q \in B$. Recursively construct a path between x and p in S and a path between q and y in S, and connect them by the edge $\{p, q\}$.

This can be turned into a formal argument using induction when $s > 1$. The induction is on the set of the $O(n^2)$ interpoint distances in P. The base case is distance between the closest pair x, y, and we have seen that the pair $\{\{x\}, \{y\}\}$ is included in the WSPD, which implies that $d_S(x,y) = \|x-y\|$. For any other pair x, y, the inductive hypothesis implies that $d_S(x,p) \leq t\|x-p\|$ and $d_S(q,y) \leq t\|q-y\|$. Thus

$$d_S(x,y) \leq d_S(x,p) + d_S(p,q) + d_S(q,y) \leq t\|x-p\| + \|p-q\| + t\|q-y\|.$$

By the separation criterion, there exists a value D such that $\|x-p\|, \|q-y\| \leq D$ and $\|p-q\| \geq sD$. This implies

$$\|x-p\|, \|q-y\| \leq \frac{\|p-q\|}{s},$$

leading to

$$d_S(x,y) \leq \frac{2t\|p-q\|}{s} + \|p-q\| \leq \frac{2t+s}{s}\|p-q\|.$$

Note that

$$\|x-y\| \geq \|p-q\| - \|x-p\| - \|y-q\| \geq \frac{s-2}{s}\|p-q\|.$$

When

$$s \geq \frac{4t}{t-1}$$

it holds that $\frac{2t+s}{s} \leq t\frac{s-2}{s}$ and $d_S(x,y) \leq t\|p-q\|$, as required. Thus we can produce a t-spanner of linear size for P, for t arbitrarily close to 1, by constructing a $\frac{4t}{t-1}$-WSPD.

5.1.4 Euclidean minimum spanning trees

Given a set P of n points in $\mathbb{R}^d$, we want to connect them into a tree by a collection of line segments with minimal total weight. This is the Euclidean MST problem. The best algorithms for constructing the Euclidean MST exactly require close to quadratic time when d is large. (In the plane the Euclidean MST is a subgraph of the Delaunay triangulation and thus can be constructed in time $O(n \log n)$.) We will now see how to approximate the MST arbitrarily well in time $O(n \log n)$ by constructing a WSPD.

Actually, our construction is a direct application of the spanner construction we have seen above. Suppose we want to construct a t-approximate MST of the set P. That is, we want to connect P into a tree whose weight is at most t times the weight of the MST of P. To this end we simply construct a t-spanner S of P and construct its minimum spanning tree T in time $O(n \log n)$ using any of the standard graph MST algorithms. Let us verify that T is a t-approximate MST of P.

Let M be the exact MST of P. For every edge $\{u, v\}$ of M, let $\pi_{u,v}$ be the shortest path connecting u and v in S. The weight of $\pi_{u,v}$ is at most $t\|u - v\|$, and the weight of the subgraph S' of S obtained as the union of all such paths $\pi_{u,v}$ is at most t times the weight of M. Clearly, S' is connected and thus the MST of S has weight at most that of S' and thus at most t times the weight of M.

5.1.5 Diameter

We can also approximate arbitrarily well the diameter of a point set using WSPD. Suppose we want to find two points in P whose distance is at least $(1-\varepsilon)\text{diam}(P)$. Construct an s-WSPD for P with $s \geq \frac{2}{\varepsilon}$. (This bound can be tightened.) Consider an arbitrary pair of points from each pair in the WSPD. Let $\{p, q\}$ be the farthest pair among those considered. We claim that $\|p - q\| \geq (1-\varepsilon)\text{diam}(P)$. Indeed, let $x, y \in P$ be the pair that defines the diameter. There is a pair $\{A, B\}$ in the WSPD such that $x \in A$ and $y \in B$. Let $p \in A$ and $q \in B$ be the representative pair that we considered for $\{A, B\}$. By the separation criterion, there is a value D for which $\|p - q\| \geq sD$ and $\text{diam}(P) = \|x - y\| \leq (s+2)D$. Thus $\|p - q\| \geq \frac{s}{s+2}\text{diam}(P)$. It is easy to check that $\frac{s}{s+2} \geq 1 - \varepsilon$ for $s \geq \frac{2}{\varepsilon}$. This concludes the proof.

5.2 Constructing WSPDs

In this section we will prove Theorem 5.1.1 by describing an algorithm that constructs an s-WSPD of a set P of n points in $\mathbb{R}^d$ with $O(s^d n)$ pairs in time $O(n \log n + s^d n)$, such that every point of P is present in $O(s^d)$ pairs. In order to construct the WSPD, we will first preprocess P into a *compressed quadtree* of linear size in time $O(n \log n)$. Given the compressed quadtree, a very simple algorithm will construct the WSPD, although some cleverness will be needed to prove the bound on its size.

5.2.1 Introducing quadtrees

Recall the notion of a quadtree $\mathcal{Q}$ for P in $\mathbb{R}^d$. It is a space decomposition, constructed recursively as follows. The construction starts from a bounding cube of P. At every stage, the recursion terminates if there is at most one point in the cube. Otherwise the current cube is subdivided into 2^d equal cubes with half the side-length and the recursion continues inside each of them. The quadtree is stored as a tree, corresponding to the computation tree of the above recursive process. Each node of the tree stores the associated cube. The leaves also store the point of P contained in the associated cube, if any. To save space, we do not construct leaves for empty cubes, so the number of children of an internal node may vary between 1 and 2^d.

In general the size of $\mathcal{Q}$ is not bounded in terms of n. For the sake of general education, let's show that the size of $\mathcal{Q}$ can be bounded in terms of n and the *spread* of P, where the spread is defined as

$$\Phi(P) = \frac{\max_{p,q \in P} \|p - q\|}{\min_{p,q \in P, p \neq q} \|p - q\|}.$$

Namely, $\Phi(P)$ is the ratio between the diameter and the closest point distance.

Lemma 5.2.1. *The depth of $\mathcal{Q}$ is $O(\log \Phi(P))$ and the size and construction time of $\mathcal{Q}$ are $O(n \log \Phi(P))$.*

Proof. Assume without loss of generality that the bounding cube of P that the construction of $\mathcal{Q}$ starts with is the unit hypercube $[0,1]^d$, and that $\operatorname{diam}(P) = 1$. Let lg denote the binary logarithm. Let the level of a node of $\mathcal{Q}$ be its distance from the root. Note that the side length of a node at level i is $1/2^i$ and the diameter of the associated cube is $\sqrt{d}/2^i$. An internal node of $\mathcal{Q}$ contains at least two points of P whose distance is at least $1/\Phi(P)$. Thus, if m is the maximal level of an internal node of $\mathcal{Q}$,

$$\begin{aligned} \frac{1}{\Phi(P)} &\leq \frac{\sqrt{d}}{2^m} \\ 2^m &\leq \sqrt{d}\Phi(P) \\ m &= O(\log \Phi(P)) \end{aligned}$$

This implies the lemma. □

Well, that was somewhat of a diversion. The matter remains that the size of a quadtree cannot be bounded in terms of n alone. Thinking about it for a moment, we can notice that this can only be due to a profusion of internal nodes with a single child. Indeed, the number of leaves is n and the number of nodes of degree at least 2 is therefore at most $n - 1$. (Otherwise the sum of the degrees in the tree $\mathcal{Q}$ would be more than twice the number of edges.) The trouble must be that there are long degree-2 paths in $\mathcal{Q}$. To address this, we define the *compressed quadtree $\mathcal{T}$*, which is simply $\mathcal{Q}$ with each such path shrunk to a single edge. By the above discussion, the size of $\mathcal{T}$ is $O(n)$ and it can be constructed in time $O(n \log \Phi(P))$. In the next subsection we shall see how the construction time can be reduced to $O(n \log n)$, which is what we want in order to use compressed quadtrees to build WSPDs.

5.2.2 Constructing compressed quadtrees

In order to construct a compressed quadtree in linear time we need an efficient algorithm that approximates the smallest ball that contains k points of P.

Lemma 5.2.2. *Let $r_{opt}(k)$ be the radius of the smallest ball $B_{opt}(k)$ that contains k points of P. A ball B that contains k points of P and has radius at most $2r_{opt}(k)$ can be constructed in time $O\left(n\,(n/k)^d\right)$.*

Proof. For every coordinate x_i, compute $m = O(n/k)$ x_i-orthogonal hyperplanes, such that every slab between two consecutive hyperplanes contains at most $k/(d+1)$ points. This can be done recursively in time $O(n \log(n/k))$ using linear-time median selection.

Consider the non-uniform grid G formed by these hyperplanes. We claim that $B_{opt}(k)$ contains at least one point of G. Indeed, assume that $B_{opt}(k)$ does not contain a grid point and consider its center c. Clearly, $B_{opt}(k)$ is contained in the union of the d slabs of G that contain c. Therefore, the number of points in $B_{opt}(k)$ is at most $dk/(d+1)$, a contradiction.

This implies that there exists a ball centered at a point of G of radius at most $2r_{opt}(k)$ that contains at least k points of P. We can find such a ball by spending linear time at each point of G, resulting in overall running time $O\left(n\,(n/k)^d\right)$. □

To construct a compressed quadtree $\mathcal{T}$ of P, find a ball B as that contains $n/(2A)$ points of P and has radius r, such that $r \leq 2r_{opt}(n/(2A))$, where A is a constant specified below. Lemma 5.2.2 implies that this can be accomplished in time $O(n)$.

As above, assume without loss of generality that P is contained in the unit hypercube $[0,1]^d$ and has diameter at least $1/2$. (But do not assume anything about the spread.) Let $l = 2^{\lfloor \lg r \rfloor}$. Since $r < 1$, $\lg r$ is a negative number and $\lfloor \lg r \rfloor$ is the greatest negative integer not greater than $\lg r$. Consider the uniform grid G_l with side length l on the unit hypercube. Every cell of G_l is a valid potential cell of the compressed quadtree.

Note that $l > r/2$ and the diameter of B is less than $4l$. Therefore B is covered by 5^d cells of G_l. This implies that one of the cells of G_l contains at least $n/(5^d 2A)$ points of P. Consider such a cell c. Let A be the minimal number of balls of radius $l/3$ needed to cover c. Clearly, A is a constant. If the cell c contains $n/2$ or more points of P then at least one of these A balls covering c contains at least $n/(2A)$ points. Since $l/3 \leq r/3 \leq \frac{2}{3} r_{opt}(n/(2A))$, we have found a ball of radius strictly less than $r_{opt}(n/(2A))$ that contains at least $n/(2A)$ points of P. This contradiction implies that the cell c contains between $n/(5^d 2A)$ and $n/2$ points of P.

Our algorithm for constructing the compressed quadtree $\mathcal{T}$ now proceeds as follows. We implicitly construct the grid G_l by determining for every point $p = (p_1, p_2, \ldots, p_d)$ of P the values $\lfloor \frac{p_i}{l} \rfloor$. The point $(\lfloor \frac{p_1}{l} \rfloor, \lfloor \frac{p_2}{l} \rfloor, \ldots, \lfloor \frac{p_d}{l} \rfloor)$ is the "lowermost" corner of the grid cell that contains p. Using hashing, we can find the heaviest cell c of G_l and the subset of P that lies inside c in linear time. Let P_{in} and P_{out} be the subsets of P lying inside and outside c, respectively. Construct compressed quadtrees $\mathcal{T}_{in}$ and $\mathcal{T}_{out}$ for these point sets. As mentioned above, c is a valid potential quadtree cell of $\mathcal{T}$. We thus simply hang $\mathcal{T}_{in}$ off the proper node of $\mathcal{T}_{out}$. This can be done in linear time. The resulting tree is the compressed quadtree $\mathcal{T}$. By a standard divide-and-conquer argument, the size of $\mathcal{T}$ is $O(n)$ and the construction time is $O(n \log n)$.

5.2.3 Constructing WSPDs

Given a compressed quadtree $\mathcal{T}$ for P, we construct an s-WSPD using the following simple recursive procedure. The procedure $\mathsf{ConstWSPD}(\{u, v\})$ takes as input a pair of nodes u and v of $\mathcal{T}$ and outputs a set of pairs of nodes $\{s, t\}$ of $\mathcal{T}$, such that the corresponding pairs of subsets of P satisfy the separation condition. (We can assume that a node of T links to the list of points that lie inside the corresponding quadtree cube, so the subset of P that corresponds to a node of T can be easily recovered.) To construct an s-WSPD of P we execute $\mathsf{ConstWSPD}(\{r, r\})$ for the root r of $\mathcal{T}$.

Here is the procedure $\mathsf{ConstWSPD}(\{u, v\})$. Assume without loss of generality that the cube associated with u is at least as large as the cube associated with v. (Otherwise swap u and v.) If these cubes are s-separated, add the pair $\{u, v\}$ to the WSPD and terminate. Otherwise, if the cubes are not s-separated, call $\mathsf{ConstWSPD}(\{w, v\})$ for all children w of u. Two cubes of side lengths l_1 and l_2 are said to be well-separated if the distance between their centers is at least $(s + 1)\sqrt{d} \max(l_1, l_2)$. (This ensures that the point sets inside the cubes are s-separated.) Let us prove that this algorithm constructs an s-WSPD as described in Theorem 5.1.1, namely that the s-WSPD has $O(s^d n)$ pairs and is constructed in time $O(n \log n + s^d n)$.

Consider a pair $\{u, v\}$ of the WSPD, and assume without loss of generality that $\mathsf{ConstWSPD}(\{u, v\})$ was called within $\mathsf{ConstWSPD}(\{u, p(v)\})$. This implies $u \le p(v)$. (We define the $\le$ and $\ge$ relations between cubes in terms of their side lengths.) The fact that $\mathsf{ConstWSPD}(\{u, p(v)\})$ was invoked implies that a pair $\mathsf{ConstWSPD}(\{p(u), a(v)\})$ has been considered and the parent $p(u)$ of u was split when compared to some ancestor $a(v)$ of v. This implies that $p(u) \ge a(v) \ge p(v)$. To summarize, $p(u) \ge p(v) \ge u$. We charge the pair $\{u, v\}$ to $p(v)$.

Let us prove that each node v' is charged $O(s^d)$ times in this way. This will imply that the overall number of pairs generated by the algorithm is $O(s^d)$. Consider the set of nodes u that participate in pairs $\{u, v\}$ as above. Since $p(u) \ge v' \ge u$, either $p(u)$ is at the same level as v', or u is at the same level as v', or there is a cell at the level of v' that is part of a compressed path from $p(u)$ to u. In any of these cases, there is a set of pairwise disjoint cells u' at the level of v', such that a single such node u' corresponds to at most 4^d pairs $\{u, v\}$ that charge v': at most 2^d possibilities for u as a child of u' (or u' itself) and at most 2^d possibilities for v as a child of v'.

Since the pairs $\{u, v'\}$ are not well-separated, neither are the pairs $\{u', v'\}$. Thus the cubes u' all lie within a cube with side length $(4s+9)\sqrt{d}l(v')$ centered at v'. Their number is at most $\left((4s + 9)\sqrt{d}\right)^d = O(s^d)$. This proves the claim.

Chapter 6

Euclidean Traveling Salesperson

6.1 Background

Given a weighted graph G, the TSP problem is to find a Hamiltonian cycle of minimal cost. (A Hamiltonian cycle is one that visits each vertex exactly once.) Clearly, the TSP problem is NP-hard simply because the Hamiltonian cycle problem is. More remarkable, though, is that the problem is hard to approximate within an arbitrarily large factor. Specifically, let $f(n)$ be any polynomial-time computable function of n. We claim that TSP cannot be approximated in polynomial time within a factor of $f(n)$ unless $P = NP$. Indeed, given a graph G' that we want to decide Hamiltonian Cycle for, construct a complete weighted graph G on the same vertex set, such that edges that are present in G' have weight 1 in G, and all the others have weight $n \cdot f(n) + 1$. If G' has a Hamiltonian cycle, there is a TSP tour in G of cost n. Otherwise the cheapest TSP tour in G has cost at least $n \cdot f(n) + 1$. An $f(n)$-approximation algorithm for TSP has to distinguish between these two cases, which implies the inapproximability result.

In general the above reduction produces a graph G that violates the triangle inequality and is thus not a metric. Can TSP be approximated on metrics? The answer is yes. Given a metric G, a simple 2-approximation is as follows.

(a) Construct an MST T for G. (Kruskal.)

(b) Double every edge of T to obtain a Euclidean graph.

(c) Find an Eulerian tour R on this graph. (Simple greedy algorithm.)

(d) Output the *shortcut tour* R' of R. That is, R' visits the vertices of G in the order of their first appearance in R.

The tour output by the algorithm is a Hamiltonian tour and has weight at most $2 \cdot OPT$, where OPT is the weight of the minimal TSP tour of G. Indeed, the MST produced at step (a) has weight at most OPT, since a TSP tour is a spanning tree of G, whereat an MST is a minimal such tree. The graph produced at step (b) has overall weight at most $2 \cdot OPT$, and so does the tour R from step (c). The triangle inequality then ensures that the weight of R' is not greater than the weight of R.

In step (b) we doubled the edges of T to produce an Eulerian graph. We can reduce the approximation factor to $\frac{3}{2}$ by noting that instead of this doubling we can compute a minimum weight matching on the odd-degree vertices of T (within G) and add this matching to T. This does produce an Eulerian graph. What can we say about its weight? Consider the set V' of odd-degree vertices of T. Given an optimal TSP tour on G, we can produce a TSP tour on V' of weight at most OPT by simple shortcutting. Observe that there is a perfect matching on V' of weight at most *half* of the weight of this TSP tour, and thus at most $\frac{1}{2}OPT$. This implies that the weight of the TSP tour produced by this modified algorithm is at most $\frac{3}{2}OPT$. This algorithm is due to Christofides (1976).

Can we do better for Metric TSP? This is an open problem. However, it is known that the problem is MAX-SNP complete (Arora et al., 1992), so no PTAS for it exists. How about interesting special classes of metrics, like the Euclidean metric? Even in this case, Trevisan (1997) proved that if the points are in a Euclidean space of dimension $\Omega(\log n)$ the problem becomes MAX-SNP complete. We are thus left to consider Euclidean TSP in low-dimensional. When the dimension is constant, Arora (1998) showed that a PTAS does indeed exist. Specifically, given a set of points P in $\mathbb{R}^d$, for $d = O(1)$ and any constant ε, an algorithm exists that produces a TSP tour of cost at most $(1+\varepsilon)OPT$ in time $n(\log n)^{1/\varepsilon}$. The rest of this section is devoted to deriving this algorithm.

6.2 The algorithm

We describe the algorithm for $d = 2$. The construction and analysis we describe work for any constant d, with some simple modifications.

Snapping onto grid. The first ingredient of the algorithm is to rescale the points and snap them onto a grid. Clearly, the problem is scale- and translation-invariant. We rescale P so that its smallest bounding square has side length n^2, and translate P so that the bottom left bounding square corner coincides with the origin. We then snap each point of P to its closest point on the unit grid.

We claim that the weight of the optimal TSP tour on the snapped point set is at most $(1+\varepsilon)OPT$, and therefore it is sufficient to approximate TSP on the snapped point set. Indeed, assume $n > 1/\varepsilon$. (If this does not hold to begin with, we can replace n by n/ε; since $1/\varepsilon = O(1)$ this does not disrupt the asymptotic bounds.) Note that $OPT \geq 2n^2$. On the other hand, each point $v \in P$ is moved by at most $\sqrt{2}/2$ by the snapping, and thus the length of each edge of the optimal tour increases by at most $\sqrt{2}$. Since the tour has n edges, the length increase is at most $\sqrt{2}n$, which is at most $OPT/(\sqrt{2}n) \leq \varepsilon OPT$.

Thus a $(1+\varepsilon)$-approximate tour on the snapped points has cost at most $(1+\varepsilon)^2 OPT$. We still need to show that we can reconstruct a tour on the original point set given a tour on the snapped point set, without increasing the cost by much.

Notice that a number of points of P could have been snapped onto the same grid point. Given a tour T on the grid points, we produce a tour on P by simply adding a *detour* (back and forth) between every point of P and the grid point onto which it was snapped. The total length of the added detours is $\sqrt{2}n$. We then produce a TSP tour of P by shortcutting the above network onto P. Clearly, the length of the produced tour is at most $\text{cost}(T) + \sqrt{2}n \leq ((1+\varepsilon)^2 + \varepsilon)OPT$. Since we can adjust ε appropriately, we assume that the point set P we are dealing with is a set of grid points as above.

Portal placement on a randomly-shifted quadtree. Let $l = n^2$ and let $L = 2l$ be twice the side length of the bounding square considered in the previous section. We enclose P in a randomly-shifted square B of side length L as follows. Choose independently uniformly at random two integers $a, b \in [-l, 0]$. Let the point (a, b) be the bottom left corner of B. Assume without loss of generality that n is a power of 2, which implies that so is L. We subdivide B into a quadtree Q. The *level* of a square of Q is the distance of the corresponding tree node from the root. (So the whole B has level 0, while the leaves have level $\log L$.) The 2^i horizontal and 2^i vertical lines that divide i-level squares into $(i+1)$-level squares are also said to have level i. The following is a simple but important estimate. Consider a line l that intersects the bounding square of P. The probability that l has level i is precisely

$$\frac{2^i}{l} = \frac{2^{i+1}}{L}.$$

We place special points called *portals* on each grid line. An i-level line has $2^{i+1}m$ equally spaced portals, where $m = O((\log n)/\varepsilon)$. (Specifically, $m = (8 \log n)/\varepsilon$ will suffice.) We also use the corners of each square as portals. Each i-level line has 2^{i+1} $(i+1)$-level square incident to it. Therefore each side of an i-level square has $m+2$ portals on it—m in its interior and 2 at the endpoints. By the same reasoning, each square has $4m+4$ portals on its boundary.

A *portal-respecting* tour is one that crosses the grid lines only at portals. A portal-respecting tour is *k-light* if it crosses each side of each quadtree square at most k times. The optimal portal-respecting tour does not have more than two incoming edges at any portal. Indeed, the existence of more than 2 incoming edges implies that the tour has to enter and exit the portal to the same side of the line. We can then shortcut the tour on that side. This implies that the optimum portal-respecting tour is $(m+2)$-light.

Dynamic programming. We find the optimal k-light portal-respecting tour using dynamic programming. For every square s of Q, we consider the possible *interfaces* for s. Each interface is an *ordered* sequence of at most $4k$ portals on the boundary of s. Clearly, an interface completely characterizes the interaction between the inside and the outside of s along a k-light portal-respecting tour. The number of possible interfaces is at most

$$\sum_{i=0}^{4k} \binom{4m+4}{i} i! \le (4k)! \left(\frac{e(4m+4)}{4k} \right)^{4k} = m^{O(k)}.$$

The dynamic programming fills out a table that has an entry for each

quadtree square × interface

combination. The entry for an interface I at a quadtree square s corresponds to the cost of the minimal k-light portal-respecting tour of the points of P that lie within s, such that the tour uses the interface I. The size of the table is $O(n^4)m^{O(k)}$ and the amount of time spent for filling out every entry is $m^{O(k)}$. The table is filled out in a bottom-top order. For an interface I at a square s we enumerate all possible combinations of interfaces for the children of s. For each combination, we check whether its parts are consistent among themselves and also with I. If so, we compute its cost. The minimal cost thus computed is stored at the table entry for $s \times I$.

Since $m = O((\log n)/\varepsilon)$ and $k \le m + 2$,

$$m^{O(k)} = \left(\frac{\log n}{\varepsilon} \right)^{O\left(\frac{\log n}{\varepsilon}\right)},$$

which is only a *quasi-polynomial*, but not a truly polynomial bound. This problem is resolved below by showing that considering $O(1/\varepsilon)$-light tours is in fact sufficient for getting an ε-approximation of the optimal tour. The factor $m^{O(k)}$ is then bounded by

$$\left(\frac{\log n}{\varepsilon} \right)^{O\left(\frac{1}{\varepsilon}\right)}$$

and the overall running time of the dynamic programming is $O(n^4 (\log n)^{O(1/\varepsilon)})$. In fact, it is easy to see that we do not need to subdivide B into all $O(n^4)$ unit squares, but that it is sufficient to only construct the quadtree every leaf of which contains a single point. Due to the polynomial spread of P, the size of the quadtree is $O(n \log n)$ and the overall running time is then bounded by $O(n(\log n)^{O(1/\varepsilon)})$.

Bounding the approximation factor. Let us first see that when $m = O(\log n/\varepsilon)$, the optimal portal-respecting tour of P approximates the optimal tour of P by a factor of $(1 + \varepsilon)$. Note that in this preliminary analysis we do not place any restrictions on the k-lightness of the tour, so it may enter and leave each square $8m + 8$ times. Denote the cost of the optimal tour for specific values of a, b by $OPT_{a,b}$.

The following lemma bounds the increase in the cost of a single edge when it is made portal-respecting. The global bound will then follow by linearity of expectation. For two nodes $u, v \in \mathbb{R}^2$ let the *portal-respecting distance* between u and v, denoted $d_{a,b}(u, v)$ be the shortest distance between them when all intermediate grid lines have to be crossed at portals.

Lemma 6.2.1.
$$\mathbb{E}[d_{a,b}(u,v) - \|u-v\|] \leq \frac{2\log L}{m}\|u-v\|.$$

Proof. The straight line path from u to v crosses the grid at most $2\|u-v\|$ times. To get a portal-respecting path, we move each crossing to the nearest portal on the grid line, which involves a detour whose length is at most the interportal distance on that line. If the line has level i, the interportal distance is $L/(2^{i+1}m)$. The probability that the line has level i is $2^{i+1}/L$. The expected length of the detour is thus

$$\sum_{i=0}^{\log L-1} \frac{L}{2^{i+1}m} \cdot \frac{2^{i+1}}{L} = \frac{\log L}{M}.$$

Summing over all the $2\|u-v\|$ lines crossed by the path from u to v, we get

$$E[d_{a,b}(u,v) - \|u-v\|] \leq \frac{2\log L}{m}\|u-v\|.$$

□

By linearity of expectation, the lemma implies

$$E[OPT_{a,b} - OPT] \leq \frac{2\log L}{m} OPT.$$

Therefore, with probability at least 1/2,

$$OPT_{a,b} \leq \left(1 + \frac{4\log L}{m}\right) OPT = \left(1 + \frac{8\log n}{m}\right) OPT = (1+\varepsilon)OPT.$$

Bounding the approximation factor of $(1/\varepsilon)$-light tours. As remarked above, the preceding analysis suffices to get a quasi-polynomial running time. To get a polynomial running time we prove that we can $(1+\varepsilon)$-approximate OPT not just by a portal-respecting tour, but by a $(1/\varepsilon)$-light portal-respecting tour. To this end, we show that we can make any portal-respecting tour k-light while increasing its length by a factor of at most $24/(k-5)$. Choosing $k = 24/\varepsilon + 5$ then yields the result. The proof makes extensive use of the following *Patching Lemma*.

Lemma 6.2.2. *Let S be any line segment of length s and let π be a closed path that crosses S at least three times. Then we can break the path in all but two of these places, and add to it line segments lying on S of total length at most $6s$ such that π changes into a closed path π' that crosses S at most twice.*

Proof. Let $M_1, \ldots, M_t$ be the points on which π crosses S. Break π at those points, thus causing it to fall apart into t paths $P_1, \ldots, P_t$. In what follows, we will need two copies of each M_i, one on each side of S. Let M_i' and M_i'' be these copies.

Let $2j$ be the largest even number less than t. Let J be the multiset of line segments consisting of the following: (i) A minimum cost salesman tour through $M_1, \ldots, M_t$, and (ii) A minimum cost perfect matching among $M_1, \ldots, M_{2j}$. Note that the line segments of J lie on S and their total length is at most $3s$. We take two copies J' and J'' of J and add them to π. We think of J' as lying on the left of S and of J'' as lying on the right.

If $t = 2j + 1$ we add an edge between M'_{2j+1} and M''_{2j+1}. If $t = 2j + 2$ we add an edge between M'_{2j+1} and M''_{2j+1} and an edge between M'_{2j+2} and M''_{2j+2}. Together with the paths $P_1, \dots, P_t$, these added segments define a connected Eulerian graph, in which the degree of the vertices M'_i and M''_i is 4. An Eulerian tour of this graph is a closed tour with the desired properties. $\square$

We will also use the rather obvious property that if $t(\pi, l)$ is the number of times a tour π crosses a grid line l, then

$$\sum_{l \text{ vertical}} t(\pi, l) + \sum_{l \text{ horizontal}} t(\pi, l) \leq 2\text{cost}(\pi).$$

We transform a tour into a k-light one as follows. For every vertical grid line l we consider the sides of quadtree cells incident to l in a bottom-top fashion, from sides of level $\log L - 1$ to sides of level $i + 1$, where i is the level of l. For each considered side, if it is crossed by more than $s = k - 4$ then we apply the Patching Lemma to reduce the number of crossings to at most 2. Note that l is touched by 2^j sides of j-level squares, each of length $L/2^j$, for $j \geq i + 1$.

Let $X_{l,j}(b)$ be a random variable denoting the number of overloaded j-level segments encountered by this procedure applied to a vertical line l that has level i. We claim that for every b,

$$\sum_{j \geq i+1} X_{l,j}(b) \leq \frac{t(\pi, l)}{s - 1}.$$

The reason is that the tour π crosses l only $t(\pi, l)$ times, and each application of the Patching Lemma replaces at least $s + 1$ crossings with at most 2, thus eliminating at least $s - 1$ crossings every time. The same argument implies that

$$\sum_{j \geq 0} X_{l,j}(b) \leq \frac{t(\pi, l)}{s - 1}.$$

Using the Patching Lemma and the fact that a j-level side has length $L/2^j$, the cost of the above transformation applied to l is at most

$$\sum_{j \geq i+1} X_{l,j}(b) \cdot \frac{6L}{2^j}.$$

Let Y_l be the random variable that equals the cost of the transformation applied to l. (The randomness is over the choices of a, which affect the level of l.) Then for any shift b we have

$$\begin{aligned} E[Y_l] &= \sum_{i\geq 0} \frac{2^{i+1}}{L} \cdot \sum_{j\geq i+1} X_{l,j}(b) \cdot \frac{6L}{2^j} \\ &= 6\sum_{i\geq 0}\sum_{j\geq i+1} \frac{X_{l,j}(b)}{2^{j-(i+1)}} \\ &\leq 6\sum_{j\geq 1} 2X_{l,j}(b) \\ &\leq \frac{12t(\pi,l)}{s-1}. \end{aligned}$$

In this way we fix all the vertical lines and then apply the same process to fix all the horizontal lines. Linearity of expectations implies

$$E\left[\sum_l Y_l\right] = \sum_l \frac{12t(\pi,l)}{s-1} \leq \frac{24\text{cost}(\pi)}{s-1} = \frac{24}{k-1}\text{cost}(\pi).$$

There is only one problem left. After we apply the patching procedures to the horizontal lines we create new intersections of the tour with vertical lines, seemingly ruining the work we have previously done fixing the vertical lines. However, this is easily taken care of as follows. Notice that the created crossings of vertical lines are all at vertices of the grid. At every such vertex we can reduce the number of created crossings to 4 by applying the patching lemma on each side of the horizontal line. Since the crossings are all contained in a line segment of length 0, this does not increase the length of the tour. It also does not introduce new horizontal crossings. Thus each side of a quadtree square is crossed at most $s + 4 = k$ times and the tour is k-light. We have thus proved that it suffices to consider k-light portal-respecting tours. This immediately yields an algorithm with running time $O(n(\log n)^{O(1/\varepsilon)})$ for ε-approximating Euclidean TSP in the plane.

Chapter 7

Approximating the Extent of Point Sets

In this section we will mainly see algorithms that estimate the "extent" of a point set in $\mathbb{R}^d$. Specifically, we will look at the problems of computing the diameter, width, k-flat fitting, smallest bounding box, and smallest enclosing ball. The most important thing to get from this section, though, is not necessarily the specific algorithms we consider, but the general techniques we use to derive them. Extent estimation problems are a particularly instructive class of problems to demonstrate these techniques on, which is our reason for concentrating on them.

Our general methodology for designing a geometric approximation algorithm is to use one or both of the following approaches. The first is to construct a sketch of the input, such that finding a solution for the sketch in sufficient. In many cases, the sketch size is very small, much smaller than the size of the original input, so that we can use a much less efficient algorithm, such as a naive exact one, on the sketch. For other problems, the sketch possesses certain desirable geometric properties, like small spread, say. To construct the sketch we will often use a less precise approximation algorithm, such as one with a constant approximation factor, as a bootstrap. And, as alluded to above, we will still need a (possibly inefficient) algorithm for solving the problem on the sketch.

The second approach is to construct a sketch of the solution space, such that an approximate solution lies in the sketch. For example, Arora's approximation algorithm for Euclidean TSP can be seen as a combination of both approaches. First we snap the input to a grid, and this snapped set functions as a well-structured sketch of the input. Then we show that light portal-respecting tours form an appropriate sketch of all possible TSP tours. Finally, we use simple dynamic programming to find the shortest light portal-respecting tour of the sketch.

7.1 Diameter

To warm up and demonstrate the various ideas used throughout this section consider the problem of estimating the *diameter* of a point set P in $\mathbb{R}^d$, i.e., the distance between the farthest pair of points in P.

Constant-factor approximations. Clearly, the problem can be solved exactly in time $O(n^2)$ by enumerating all pairs of points. There are also two simple constant-factor approximation algorithms that run in linear time.

Proposition 7.1.1. *The diameter of a set P of n points in $\mathbb{R}^d$ can be approximated within a factor of 2 in time $O(n)$.*

Proof. Take an arbitrary point $p \in P$ and compute in time $O(n)$ the farthest point $q \in P$ from p. We have

$$\operatorname{diam}(P) = \max_{s,t \in P} \|s - t\| \geq \|p - q\|.$$

On the other hand, P is contained in a ball B of radius $\|p - q\|$ centered at p . Any two points in B have distance at most $2\|p - q\|$, therefore

$$\operatorname{diam}(P) \leq 2\|p - q\|.$$

Hence, $2\|p - q\|$ approximates the diameter of P as desired. □

Proposition 7.1.2. *The diameter of a set P of n points in $\mathbb{R}^d$ can be approximated within a factor of $\sqrt{d}$ in time $O(n)$.*[1]

Proof. Compute in $O(n)$ time the smallest axis-parallel bounding box B of P. Assume without loss of generality that the longest edge of B is x_1-parallel, let l be the length of this edge, and consider the two points $p, q \in P$ whose x_1-coordinates are, respectively, maximal and minimal. We have

$$\operatorname{diam}(P) = \max_{s,t \in P} \|s - t\| \geq \|p - q\|.$$

On the other hand, the maximal distance of any two points in B is at most the length of a diagonal of B, which is at most $\sqrt{d}l$. Therefore

$$\operatorname{diam}(P) \leq \sqrt{d}l = \sqrt{d}\big(p_1 - q_1\big) \leq \sqrt{d}\|p - q\|.$$

Hence, $\sqrt{d}\|p - q\|$ approximates the diameter of P as desired. □

Before we proceed to describe a PTAS for the diameter, let us observe that for $0 < \varepsilon < 1$,

$$1 - \varepsilon < \frac{1}{1 + \varepsilon} < 1 - \frac{\varepsilon}{2}.$$

We will thus use the term ε-approximation to refer interchangeably to an approximation from below with factor $1 - \varepsilon$ and an approximation from below with factor $\frac{1}{1+\varepsilon}$.

[1]We provide this algorithm because it gives a better approximation in the plane and in 3-space, and because the underlying idea is quite instructive.

First PTAS. The above constant-factor approximations can be used to bootstrap a PTAS for the diameter. It constructs a concise sketch of the input and uses an exact brute force algorithm on this sketch. Let D' be a 2-approximation for the diameter found using Proposition 7.1.1. Consider a uniform grid G in $\mathbb{R}^d$ with step $\varepsilon D'$, and round every point of P to the nearest vertex in G. Let P' be the set of grid points obtained in this fashion. P' can be computed in time $O(n)$ using the floor function, and hashing to eliminate duplicates.

Each point of P is moved by at most $\frac{\sqrt{d}}{2}\varepsilon D'$. Thus

$$\operatorname{diam}(P') \geq \operatorname{diam}(P) - \sqrt{d}\varepsilon D'.$$

Let p', q' be the two points that define $\operatorname{diam}(P')$ and let p, q be corresponding points of P. We have

$$\|p-q\| \geq \|p'-q'\| - \sqrt{d}\varepsilon D' \geq \operatorname{diam}(P) - 2\sqrt{d}\varepsilon D' \geq (1-4\sqrt{d}\varepsilon)\operatorname{diam}(P).$$

Thus finding the points that define the diameter of P' and computing the distance between the corresponding points of P gives us an $O(\varepsilon)$-approximation for the diameter of P. Since we can choose ε appropriately, this yields an ε-approximation algorithm for the problem.

How quickly can we compute $\operatorname{diam}(P')$? The crucial observation is that the size of P' is actually constant. Indeed, the points of P lie in a ball of radius $2D'$, which is contained in a box of side length $4D'$. The side length of each grid cell of G is $\varepsilon D'$, therefore $O(\varepsilon^{-d})$ grid cells contain the whole point set P. There are thus $O(\varepsilon^{-d})$ points of P'. In fact, we can reduce the size to $O(\varepsilon^{-(d-1)})$ by keeping only the topmost and bottommost point in every column of G, along a particular orientation. The diameter of P' can then be computed using the naive exact algorithm in time $O(\varepsilon^{-2(d-1)})$. This yields a PTAS for the diameter with running time $O(n+\varepsilon^{-2(d-1)})$.

Second PTAS. Our second PTAS for the diameter constructs a small sketch of the space of solutions and tries each solution in the sketch. Specifically, the sketch is on the space of possible directions of the diameter vector.

Proposition 7.1.3. *There exists a set V of $O(1/\varepsilon^{(d-1)/2})$ unit vectors in $\mathbb{R}^d$, such that for any $x \in \mathbb{R}^d$,*

$$\max_{v\in V}\langle v, x\rangle \geq (1-\varepsilon)\|x\|.$$

Proof. Let C be the cube $[-1,1]^d$ and let F be one of the $2d$ facets of C. Cover F with a uniform grid with step $\sqrt{\varepsilon}$. Iterated over all facets F, this produces $O(1/\varepsilon^{(d-1)/2})$ vectors. Project them onto the unit sphere to obtain the set V. We prove below that V covers the space of orientations as specified in the proposition.

Let $x \in \mathbb{R}^d$ be any vector. We need to prove that there exists $v \in V$ for which $\langle v, x\rangle \geq (1-\varepsilon)\|x\|$. Let us show that there exists $v' \in V$ such that $\cos\angle(v,x) = 1-O(\varepsilon)$. It is sufficient to scale x to obtain a vector x' that lies on a facet F of the cube C, and prove the existence of a vector v' on the grid constructed on F, such that $\cos\angle(v', x') = 1-O(\varepsilon)$. Indeed, there is a grid point within distance $\sqrt{d-1}\sqrt{\varepsilon}/2$ of

x'. Let v' be such a grid point. Consider the triangle $Ox'v'$, where O is the origin. We wish to bound α, the angle in this triangle at O, and can use the law of cosines:

$$\begin{aligned}\cos\alpha &= \frac{\|O - x'\|^2 + \|O, v'\|^2 - \|x' - v'\|^2}{2\|O - x'\|\|O - v'\|} \\ &= \frac{\|O - x'\|^2 + \|O - v'\|^2}{2\|O - x'\|\|O - v'\|} - \frac{\|x' - v'\|^2}{2\|O - x'\|\|O - v'\|} \\ &\geq 1 - \frac{(d-1)\varepsilon/4}{2} = 1 - O(\varepsilon).\end{aligned}$$

By adjusting the step of the grids appropriately we can assume $\cos\alpha \geq 1 - \varepsilon$. Thus

$$\langle v, x\rangle = \cos(\alpha)\|v\|\|x\| \geq (1-\varepsilon)\|x\|.$$

□

Let $p, q \in P$ be the pair that maximizes $\|p - q\|$. By proposition 7.1.3,

$$\begin{aligned}(1-\varepsilon)\mathrm{diam}(P) &= (1-\varepsilon)\|p-q\| \leq \max_{v\in V}\langle v, p-q\rangle \\ &\leq \max_{v\in V}\Big(\langle v, p\rangle - \langle v, q\rangle\Big) \leq \max_{v\in V}\Big(\max_{s\in P}\langle v, s\rangle - \min_{s\in P}\langle v, s\rangle\Big).\end{aligned}$$

It is thus sufficient to find, for every direction $v \in V$, the two extreme points in the projection of P onto v. This can be accomplished in time $O(n/\varepsilon^{(d-1)/2})$.

Third PTAS. After using the first algorithm above to create a set P' of $O(\varepsilon^{-(d-1)})$ grid points, instead of naively computing the diameter of P' in quadratic time, we can use the second PTAS to approximate it. This yields an algorithm with running time $O(n + \varepsilon^{-3(d-1)/2})$.

7.2 Width

The width of a set P of n points in $\mathbb{R}^d$ is the width of the smallest hyperplane slab that encloses P. In this section we devise a similar series of algorithms for the width problem as we did for the diameter. First let us reformulate the problem. It is equivalent to finding the unit vector $x \in \mathbb{R}^d$ such that the extent of the projection of P onto x is minimized. This extent is

$$\max_{s\in P}\langle x, s\rangle - \min_{s\in P}\langle x, s\rangle.$$

If x is not necessarily a unit vector, the extent becomes

$$\Big(\max_{s\in P}\langle x, s\rangle - \min_{s\in P}\langle x, s\rangle\Big)/\|x\|$$

and the width problem reduces to finding

$$\operatorname{argmin}_{x\in\mathbb{R}^d}\left(\max_{s\in P}\langle x,s\rangle-\min_{s\in P}\langle x,s\rangle\right)/\|x\|.$$

Alternatively, the problem is

$$\begin{array}{rl}\text{minimize} & (z-y)/\|x\| \\ \text{subject to} & \forall s\in P.\ \ y\le\langle x,s\rangle\le z \\ & x\in\mathbb{R}^d, y,z\in\mathbb{R}.\end{array}$$

Since the width is invariant to the scaling of x along a particular direction, we can look only at vectors x for which

$$\max_{s\in P}\langle x,s\rangle-\min_{s\in P}\langle x,s\rangle=1.$$

This changes the above problem into

$$\begin{array}{rl}\text{maximize} & \|x\| \\ \text{subject to} & \forall s\in P.\ \ y\le\langle x,s\rangle\le y+1 \\ & x\in\mathbb{R}^d, y\in\mathbb{R}.\end{array}$$

Clearly, the constraints are linear and define a convex polytope in $\mathbb{R}^{d+1}$. The objective function is optimized at a vertex of the polytope. The optimum can therefore be found in time $O(n^{\lceil d/2\rceil})$. This gives an exact algorithm for the problem.

First PTAS. To obtain a PTAS for the width we again snap P onto a grid and run the exact algorithm on the snapped point set. To construct an appropriate grid we need to do some work.

Proposition 7.2.1. *There exists an algorithm that in time $O(n)$ computes a box B that contains P, such that a translated copy of cB is contained in* conv(P)*, for a global constant $c=1/d^{O(d)}$.*

Proof. We construct B by induction. For $d=1$ the optimal bounding interval can be computed in time $O(n)$. For general d we compute $B=B_d$ as follows. Use Proposition 7.1.1 to find two points $p,q\in P$, such that $\|p-q\|\ge\operatorname{diam}(P)/2$. Project P onto the hyperplane orthogonal to $p-q$ to obtain a point set P' in $\mathbb{R}^{d-1}$. Compute a bounding B_{d-1} of P' by induction. B_d is the smallest enclosing box of P with base B_{d-1}, translated appropriately. We prove by induction that $\operatorname{Vol}(B_d)\le 2^d d!\operatorname{Vol}(\operatorname{conv}(P))$.

By the induction hypothesis, $\operatorname{Vol}(B_{d-1})\ge 2^{d-1}(d-1)!\operatorname{Vol}(\operatorname{conv}(P'))$. Assume without loss of generality that the vector $p-q$ is parallel to the x_d-axis, let $\mathcal{U}$ and $\mathcal{L}$ denote the upper and lower hulls of conv(P), and consider a function $f:\mathbb{R}^{d-1}\to\mathbb{R}$, defined as $f(x)=\mathcal{U}(x)-\mathcal{L}(x)$. Clearly, the volume of conv(P) is equal to the volume of the (convex) body enclosed between the hyperplane $x_d=0$ and the graph of f. This body, in turn, encloses a pyramid of height $\|p-q\|$ over the base conv(P'). Thus

$$\operatorname{Vol}(\operatorname{conv}(P))\ge\frac{\operatorname{Vol}(\operatorname{conv}(P'))\|p-q\|}{d}\ge\frac{\operatorname{Vol}(\operatorname{conv}(P'))\cdot\operatorname{diam}(P)}{2d}.$$

Therefore,

$$\mathrm{Vol}(B_d) \le \mathrm{Vol}(B_{d-1})\mathrm{diam}(P) \le 2^{d-1}(d-1)!\mathrm{Vol}(\mathrm{conv}(P'))\mathrm{diam}(P) \le 2^d d!\mathrm{Vol}(\mathrm{conv}(P)).$$

In particular, since the volume of the smallest bounding box of P is at least Vol(conv(P)), the box $B = B_d$ ($2^d d!$)-approximates the volume of the smallest bounding box.

We are left to show that a translate of cB fits inside conv(P). Clearly, our setting is invariant to a translation, rotation and scaling of the ambient space. We thus apply a transformation that maps B onto the unit cube C.

Note that the volume of any hyperplane slice of conv(P) is at most the volume of such a slice of C, which is at most the volume of a $(d-1)$-sphere of radius $\sqrt{d}/2$, which is less than $d^{d/2}$. Clearly, the volume of conv(P) is bounded from above by the product of the width of P and the volume of the largest hyperplane slice of conv(P). This implies

$$\mathrm{width}(P) \ge \frac{\mathrm{Vol}(\mathrm{conv}(P))}{d^{d/2}}.$$

Consider the relationship between the width of P and the radius $r(P)$ of the largest ball inscribed in conv(P). It can be shown that this ratio is maximized by regular simplices, for which it is $2\sqrt{d}$ for odd d and $2(d+1)/\sqrt{d+2}$ for even d. This implies that

$$\frac{\mathrm{width}(P)}{r(P)} \le 2\sqrt{d+1}.$$

Thus

$$r(P) \ge \frac{\mathrm{width}(P)}{\sqrt{d+1}} \ge \frac{\mathrm{Vol}(\mathrm{conv}(P))}{\sqrt{d+1}\cdot d^{d/2}} \ge \frac{1}{\sqrt{d+1}\cdot 2^d d! d^{d/2}}.$$

The largest ball inscribed in conv(P) contains a cube of side length $2r(P)/\sqrt{d} = 1/d^{O(d)}$. Going back to our original setting before the mapping of B to C, this implies that a translate of cB is contained in conv(P), for $c = 1/d^{O(d)}$. □

After all this work the PTAS for width is rather easy. Compute a box B as in Proposition 7.2.1 and consider the grid G whose cells are translates of $c\varepsilon B$, where c is the constant from Proposition 7.2.1. Replace a point p of P by *all* the vertices of the cell of G that contains p. This results in a set P' of $O(\varepsilon^{-d})$ grid points. Again, the size of P' can be reduced to $O(\varepsilon^{-(d-1)})$. P' can be translated to fit inside $P \oplus c\varepsilon B$, which in turn can be translated to fit inside $P \oplus \varepsilon\mathrm{conv}(P)$. Thus if P can be enclosed in a slab S, P' can be enclosed in a translate of the slab $S \oplus \varepsilon S$. Hence

$$\mathrm{width}(P) \le \mathrm{width}(P') \le (1+2\varepsilon)\mathrm{width}(P).$$

It therefore suffices to compute the width of P', which we can do in time $O(\varepsilon^{-(d-1)\lceil d/2\rceil})$. The overall running time of our PTAS is $O(n + \varepsilon^{-(d-1)\lceil d/2\rceil})$.

Remark. The point set P' is a set of $O(\varepsilon^{-(d-1)})$ points that provides an outer ε-approximation to the convex hull of P. Namely,

$$\mathrm{conv}(P) \subseteq \mathrm{conv}(P') \subseteq \mathrm{conv}(P) \oplus \varepsilon\mathrm{conv}(P).$$

The surprising existence of such a concise approximation of the convex hull can be used in many applications. Essentially, we can approximate any "smallest enclosing shape" of $\text{conv}(P)$ by computing such a shape for P'. This includes the smallest enclosing slab (i.e., the width problem), ball, cylinder, bounding box, etc. We still need some algorithm, however inefficient, to run on P'. For example, the smallest bounding box of a set of n points in $\mathbb{R}^3$ can be computed exactly in time $O(n^3)$; the smallest enclosing ball of a set of n points can be computed in time $O(n^{\lceil d/2 \rceil})$ using the farthest-point Voronoi diagram; and so on.

Second PTAS. Our second PTAS produces a sketch of the possible orientations of the normal to the slab that defines the width. In fact, the sketch is the same as in the second PTAS for the diameter, namely the set V from Proposition 7.1.3. That proposition implies that the optimum of the program

$$\begin{array}{ll} \text{maximize} & \langle v, x\rangle \\ \text{subject to} & \forall s \in P.\ \ y \leq \langle x, s\rangle \leq y+1 \\ & x \in \mathbb{R}^d, y \in \mathbb{R}, v \in V \end{array}$$

is at least $(1-\varepsilon)\text{width}(P)$. This program can be solved by $|V|$ linear programming computations, which can be performed in overall time $O(n/\varepsilon^{(d-1)/2})$.

Third PTAS. As in the case of the diameter, we can combine the two approximation schemes, first producing a sketch of the input and then using the second algorithm on the sketch. This gives running time $O(n + \varepsilon^{-3(d-1)/2})$.

7.3 Coresets

The input sketches that were produced for the above approximations of the diameter and width were all collections of grid points. A somewhat more elegant approach is to simply pick a small subset of the input and use it as the desired sketch. In this case the sketch is called a *coreset*. We will now see that it is possible to compute a small coreset by modifying the above construction.

First coreset construction. The modification is as follows: Construct a grid as in the first PTAS for the width. Instead of snapping P to the grid or picking all the vertices of grid cells that enclose points of P, simply pick one point of P from each nonempty grid cell. This yields a set P' of $O(\varepsilon^{-d})$ points. Given a unit vector x, let

$$\text{width}_x(P) = \max_{s \in P}\langle x, s\rangle - \min_{s \in P}\langle x, s\rangle$$

be called the *directional width* of P with respect to x. We claim that P' ε-approximates the directional width of P in all directions, namely, that

$$\forall x \in \mathbb{S}^{d-1}. \quad (1-\varepsilon)\text{width}_x(P) \leq \text{width}_x(P') \leq \text{width}_x(P).$$

Indeed, for some direction $x \in \mathbb{S}^{d-1}$, consider the points u and v that define $\mathrm{width}_x(P)$, i.e., $u = \mathrm{argmax}_{s\in P}\langle x, s\rangle$ and $v = \mathrm{argmin}_{s\in P}\langle x, s\rangle$. Then there exist points $s, t \in P'$, such that s and u (resp., t and v) lie in the same grid cell. Recall that each grid cell is a translate of $c\varepsilon B$. Since $\mathrm{width}_x(cB) \le \mathrm{width}_x(P)$, we obtain

$$\begin{aligned} \mathrm{width}_x(P') &\ge \langle x, s\rangle - \langle x, t\rangle \\ &= \Big(\langle x, u\rangle - \langle x, v\rangle\Big) - \Big(\langle x, u\rangle - \langle x, s\rangle\Big) - \Big(\langle x, t\rangle - \langle x, v\rangle\Big) \\ &\ge (1-2\varepsilon)\mathrm{width}_x(P). \end{aligned}$$

This proves the claim. (As usual, a 2ε-approximation is sufficient since we can adjust ε appropriately.) The set $P' \subseteq P$ is a coreset of size $O(\varepsilon^{-d})$. We can reduce the size to $O(\varepsilon^{-(d-1)})$ by keeping only the topmost and bottommost point in every column of G, along a particular orientation. The resulting coreset is easily seen to still approximate the directional width as above.

Second coreset construction. We can improve the size of the coreset in the following way. As above, we can find in time $O(n)$ a box B such that $cB \subseteq \mathrm{conv}(P) \subseteq B$. We can apply a projective transformation that ensures that $c[-1,1]^d \subseteq \mathrm{conv}(P) \subseteq [-1,1]^d$.

Let S be a sphere or radius $\sqrt{d}+1$ around the origin. Set $\delta = \sqrt{\varepsilon c/2}$. Construct a set V of $O(1/\varepsilon^{(d-1)/2})$ points on S, such that for every $x \in S$, there exists $y \in V$ for which $\|x - y\| \le \delta$. This can be achieved for example by placing an appropriate grid on every facet of the hypercube $[-(\sqrt{d}+1), \sqrt{d}+1]^d$ and projecting the grid points onto S. For $v \in V$, let $\phi(v)$ be the nearest neighbor of v in P. Let P' be this set of nearest neighbors, for all $v \in V$. P' can be computed in time $O(n/\varepsilon^{(d-1)/2})$ and we claim that it is a coreset with the desired properties. To prove this we need the following technical lemma.

Lemma 7.3.1. *Let B be a ball of radius r centered at $(L, 0, \dots, 0) \in \mathbb{R}^d$, where $L \ge 2r$. Let p be a point in B, and let B' the the ball centered at p and touching the origin. Then*

$$\forall x \in B'. \quad x_1 \ge -\frac{r^2}{L}.$$

Proof. It is sufficient to lower bound x_1 for $x = (L - r, r, 0, \dots, 0)$. The radius of the ball B' in this case is $\sqrt{(L-r)^2 + r^2}$, and the x_1-minimal point of B' is

$$(L-r) - \sqrt{(L-r)^2 + r^2} = \frac{(L-r)^2 - ((L-r)^2 + r^2)}{(L-r) + \sqrt{(L-r)^2 + r^2}} \ge \frac{-r^2}{2(L-r)} \ge \frac{-r^2}{L}.$$

□

Fix a direction $u \in \mathbb{R}^{d-1}$. Let $\sigma \in P'$ be the point that maximizes $\langle u, p\rangle$ over $p \in P'$. Let $x \in S$ be the point on S that is hit by the ray emanating from σ in direction u. We know that there exists $y \in V$ such that $\|x - y\| \le \delta$. For the sake of exposition rotate and translate the coordinate system so that σ is the origin and u is the x_1-direction. Let $L = \|x\|$ and $r = \delta$. The point y lies in an r-ball around x, and $L \geq 1 \geq 2\delta \geq 2r$. The point $\phi(y)$ lies in the closed ball centered at y and touching the origin σ. By Lemma 7.3.1,

$$\max_{p \in P'} \langle u, p \rangle \geq \langle u, \phi(y) \rangle \geq \langle u, \sigma \rangle - \frac{r^2}{L} \geq \max_{p \in P} \langle u, p \rangle - \delta^2 = \max_{p \in P} \langle u, p \rangle - \frac{\varepsilon c}{2}.$$

Applying the same analysis for the direction $-u$ implies that

$$\mathrm{width}_u(P') = \max_{s \in P'} \langle u, s \rangle - \min_{s \in P'} \langle u, s \rangle \geq \mathrm{width}_u(P) - 2\frac{\varepsilon c}{2} \geq (1-\varepsilon)\mathrm{width}_u(P),$$

since $\mathrm{width}_u(P) \geq c$ for any u. This concludes the proof that P' is a coreset as desired.

We now combine the two coreset constructions, using the first to produce in time $O(n)$ a coreset P' of size $O(\varepsilon^{-(d-1)})$, and then running the second on P' to produce a final coreset of size $O(\varepsilon^{-(d-1)/2})$ in time $O(\varepsilon^{-3(d-1)/2})$. The overall running time is $O(n + \varepsilon^{-3(d-1)/2})$.

Applications of small coresets. A small coreset P' of P allows us to approximate extent measures of P like width, diameter, volume of minimal bounding box, etc. For example, let d be a unit vector parallel to $s - t$, where $s, t \in P$ are the diametral pair of P. It holds that

$$\mathrm{diam}(P') \geq \mathrm{width}_d(P') \geq (1-\varepsilon)\mathrm{width}_d(P) = (1-\varepsilon)\mathrm{diam}(P).$$

Thus we can approximate the diameter of P by computing the diameter of P'.

As another example, let B' be the smallest bounding box of the coreset P', defined by directions s, t, w. Since the width of P along each of s, t, w is at most $1/(1-\varepsilon) \leq 1 + 2\varepsilon$ times the corresponding width of P', where we assume $\varepsilon < 1/2$, the box $B = (1+4\varepsilon)B'$, appropriately translated, is a bounding box of P of volume $(1 + O(\varepsilon))\mathrm{Vol}(B')$. This gives an algorithm for finding an ε-approximate minimal bounding box.