Chapter 24

Building a Spanning Tree with a Few Crossings

By Sariel Har-Peled, April 13, 2023[1]

In addition, the sirloin which I threw overboard, instead of drifting off into the void, didn't seem to want to leave the rocket and revolved about it, a second artificial satellite, which produced a brief eclipse of the sun every eleven minutes and four seconds. To calm my nerves I calculated till evening the components of its trajectory, as well as the orbital perturbation caused by the presence of the lost wrench. I figured out that for the next six million years the sirloin, rotating about the ship in circular path, would lead the wrench, then catch up with it from behind and pass it again.

The Star Diaries, Stanislaw Lem

In this chapter, we will introduce a powerful technique for "structure" approximation. The basic idea is to perform a search by assigning elements weights and picking the elements according to their weights. The element's weight indicates its importance. By repeatedly picking elements according to their weights and updating the weights of objects that are being neglected (i.e., they are more important than the current weights indicate), we end up with a structure that has some desired properties.

We will demonstrate this technique for two problems. In the first problem, we will compute a spanning tree of points that has low stabbing number. In the second problem, we will show how the set cover problem can be approximated efficiently in geometric settings, yielding a better bound than the general approximation algorithm for this problem.

24.1. Preliminaries

In this section, we describe how to implement efficiently some low-level operations required by the algorithms described in this chapter. The reader uninterested in such low-level details can safely skip to Section 24.2.

The following algorithms assign and manipulate weights associated with various entities. There are two main technical problems:

(i) elements might have weights that are very large (i.e., require n bits) and
(ii) we need to perform (efficiently) random sampling from a set of elements that are weighted using such numbers.

24.1.1. Handling large weights

In the following, some of the weights we have to handle are of the form 2^i, where $i \geq 0$ is some integer. Such numbers can easily be represented efficiently by storing only the index i. In our algorithm i is polynomial in n.

We will also need to handle numbers that are the sum of m such numbers, where m is usually a polynomial of the input size. Naturally, if we care only about polynomial running time, we could just use exact arithmetic to handle these numbers. Alternatively, we can approximately compute such numbers. So, consider such a number $x = \sum_{i=1}^{m} 2^{b_i}$, where $b_1, \ldots, b_m$ are non-negative integers. If the powers are sufficiently small (say all the b_is are at most 16382), we can just use floating-point numbers directly (here, you would have to use the `long double` floating-point number type). Otherwise, one needs to implement a variant of floating-point numbers that can handle large powers. This can easily be done by explicitly storing the exponent part of the floating-point number. Naturally, we need the implementation of only a few specific operations on these numbers, and as such the implementation does not need to be exhaustive. It is easy to verify that one can implement all the required operations so that they take constant time.

The error introduced by this extended floating-point representation is pretty small in our applications. Indeed, all the operations the algorithms below do are addition, multiplication, and division (but no subtraction). As such, the error caused by the representation is too minuscule for us to worry about, and we will ignore it.

24.1.2. Random sampling from a weighted set.

We need to do r independent draws from a weighted set $\mathcal{C}$ having (say n) elements, where each weight is a large real number (as described above). In particular, for an element $x \in \mathcal{C}$, let $\omega(x)$ denote its weight. Assume that we are given a function that can randomly and uniformly pick an integer number in a range $[1, M]$, where M is a parameter given to the function.

One way to do this sampling is to compute the element x of $\mathcal{C}$ having the maximum weight; that is, $\omega(x) = \max_{y \in \mathcal{C}} \omega(y)$. Observe that all the elements of weight $\leq \omega(x)/n^{10}$ have weight which is so tiny that they can be ignored. Thus, normalize all the weights by dividing them by $2^{\lfloor \lg \omega(x)/n^{10} \rfloor}$, and remove all the elements with weights smaller than 1. For an element $z \in \mathcal{C}$, let $\widehat{\omega}(z)$ denote its normalized weight. Clearly, all the normalized weights are integers in the range $1, \ldots, 2n^{10}$.

Thus, we now have to pick elements from a set with (relatively small) integer weights. Place the elements in an array, and compute the prefix sum array of their weights. That is, $\alpha_k = \sum_{i=1}^{k} \widehat{\omega}(z_i)$, for $k = 0, \ldots, n$ (as such, $\alpha_0 = 0$). Next, pick a random (integer) number γ uniformly in the range $[1, \alpha_n]$, and using a binary search, find the j, such that $\alpha_{j-1} < \gamma \leq \alpha_j$. This picks the element z_j to be in the random sample. This requires $O(n)$ preprocessing, but then a single random sample can be done in $O(\log n)$ time. We need to perform r independent samples. Thus, this takes $O(n + r \log n)$ time overall.

Corollary 24.1.1. *Given a set $\mathcal{C}$ of n weighted elements, one can preprocess it in linear time, such that one can randomly pick an element of $\mathcal{C}$ uniformly at random (according to the weights of the elements of $\mathcal{C}$) in $O(\log n)$ time. In particular, picking r elements (with replacement) from $\mathcal{C}$ can be done in $O(n + r \log n)$ time.*

24.2. Computing a spanning tree with low crossing number

The ***crossing number*** of a set of segments in the plane is the maximum number of segments that can be stabbed by a single line. For a tree T drawn in the plane, it is the maximum number of intersections of a line with T.

Let P be a set of n points in the plane. We would like to compute a tree T that spans the points of P such that every line in the plane crosses the edges of the tree at most $O(\sqrt{n})$ times. If the points are the $\sqrt{n} \times \sqrt{n}$ grid, this easily holds if we pick any spanning tree that connects only points that are adjacent on the grid. It is not hard to show that one cannot do better (see Exercise 24.4.2).

Definition 24.2.1. Given a weighted set of lines L, the ***crossing distance*** $\mathrm{d}_{\times}(p, q)$ between two points p and q is the minimum weight of the lines one has to cross (i.e., cut) to get from p to q. The function $\mathrm{d}_{\times}(p, q)$ complies with the triangle inequality, and it is a ***pseudo-metric*** (since distinct points can have distance zero between them).

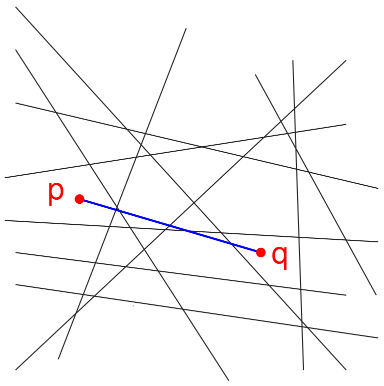
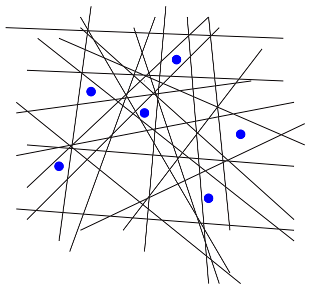


Figure 24.2.1: $\mathrm{d}_{\times}(p, q) = 5$.

Because of the triangle inequality, the crossing distance $\mathrm{d}_{\times}(p, q)$ is equal to the total weight of the lines intersecting the close segment pq; see Figure 24.2.1.

24.2.1. The algorithm



So, consider the set of all separating lines $\widehat{\mathsf{L}}$ of P. Formally, we will consider two lines ℓ and ℓ' to be equivalent if the closed halfplane above ℓ contains the same set of points as the closed halfplane above ℓ' (we assume no two points in P have the same x coordinate, and as such we can ignore vertical lines). For each equivalent class of $\widehat{\mathsf{L}}$ pick one representative line into a set L; see the figure on the right. Clearly, given such a line, we can translate and rotate it till it passes through two points of P or till it is horizontal and passes through one point of P. If it passes through two points, we need to specify for these two points whether they belong to the set defined. As such, there are at most $4\binom{n}{2} + n + 1 = O(n^2)$ different lines in L.

Idea.

The algorithm would build the spanning tree by adding the edges one by one. So, consider a candidate edge (i.e., segment) qt, and consider a line $\ell \in \mathsf{L}$. If ℓ already intersects k edges of E_i and k is large (i.e., $\Omega(\sqrt{n})$), then we would like to discourage the usage of qt in the spanning tree. The problem is that the desirability of an edge is determined by the lines that intersect the edge. Furthermore, a line that is already crossing many edges might be close to the limit of what it is allowed to cross, and as such this line would "prefer" not to cross any more edges. Intuitively, as the load (i.e., number of edges it crosses) of a line gets higher, edges that cross this line becomes less desirable.

Algorithm.

The input is the set P of n points in the plane. Let $\mathsf{E}_0 = \emptyset$, and for $i > 0$ let E_i be the set of edges added by the algorithm by the end of the ith iteration. The weight $\omega(\ell)$ of a line $\ell \in \mathsf{L}$ is initialized to

1. The ***weight*** of the line at the beginning of the ith iteration would be denoted by $\omega_{i-1}(\ell) = 2^{n_{i-1}(\ell)}$, where

$$n_{i-1}(\ell) = \left|\left\{ s \in \mathsf{E}_{i-1} \;\middle|\; s \cap \ell \neq \emptyset \right\}\right|$$

is the number of segments of E_{i-1} that intersect ℓ. The ***weight*** of a segment s, in the beginning of the ith iteration, is

$$\omega_{i-1}(s) = \sum_{\ell \in \mathsf{L}, \ell \cap s \neq \emptyset} \omega_{i-1}(\ell).$$

Specifically, the weight of $\omega_{i-1}(s)$ is the total weight of the lines intersecting s. Clearly, the heavier a segment is the less desirable it is to be used for the spanning tree. Motivated by this, we would always pick an edge qt such that $q, t \in \mathsf{P}$ belong to two different connected components of the forest induced by E_{i-1}, and its weight is minimal among all such edges. We repeat this process till we end up with a spanning tree of P. To simplify the implementation of the algorithm, when adding s to the set of edges in the forest, we also remove one its endpoints from P (i.e., every connected component of this forest has a single representative point). Thus, the algorithm terminates when P has a single point in it.

24.2.2. Analysis

24.2.2.1. Proof of correctness

We claim that the resulting spanning tree has the required properties. The algorithm performs $n - 1$ iterations and as such, the largest weight used is $\leq 2^n$, and such integer numbers can be manipulated in polynomial time. Thus, overall the running time of the algorithm is polynomial.

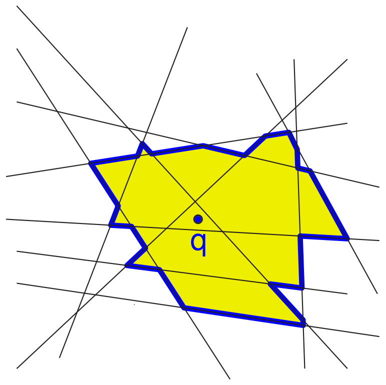


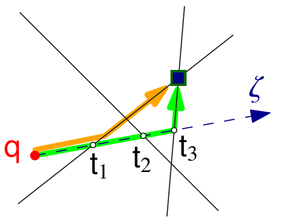
Figure 24.2.2

For a point $q \in \mathbb{R}^2$ and a set of lines L, consider the set of all the vertices of the arrangement of $\mathcal{A} = \mathcal{A}(\mathsf{L})$ that are in crossing distance at most r from q. We will refer to this set of vertices of the arrangement, denoted by $\mathsf{b}_{\times}(q, r)$, as the ***ball*** of radius r under the crossing metric. Such a ball under the crossing distance is depicted Figure 24.2.2. It is star shaped but not necessarily convex.

Lemma 24.2.2. *Let L be a set of n lines in the plane, and let $q \in \mathbb{R}^2$ be a point (not lying on any of the lines of L). Then, for any $r \leq n/2$, we have that $|\mathsf{b}_{\times}(q, r)| \geq r^2/8$.*

Proof: Indeed, one can shoot a ray ζ from q that intersects at least $n/2$ lines of L. Let $\ell_1, \ldots, \ell_{r/2}$ be the first $r/2$ lines hit by the ray ζ, and let $t_1, \ldots, t_{r/2}$ be the respective intersection points between these lines and ζ. Now, mark all the intersection points of the arrangement $\mathcal{A}(\mathsf{L})$ along the line ℓ_i that are in distance at most $r/2$ from t_i, for $i = 1, \ldots, r/2$.

The picture on the right depicts one such vertex being collected in this way. Clearly, we overall marked at least $(r/2)(r/2)/2$ vertices of the arrangement, since we marked (at least) $r/2$ vertices along each of the lines $\ell_1, \ldots, \ell_{r/2}$. Furthermore, each vertex can be counted in this way at most twice. Now, observe that all these vertices are in distance at most $r/2 + r/2$ from q because of the triangle inequality, implying the claim. ■



Lemma 24.2.3. *Let* P *be a set of* n *points in the plane, and let* L *be a set of lines in the plane with total weight* W*. One can always find a pair of points* q *and* t *in* P*, such that the total weight of the segment* $s = qt$ *(i.e., the total weight of the lines of* L *intersecting* s*) is at most* $4W/\sqrt{n} + 3 \leq \mathsf{c}W/\sqrt{n}$*, for some constant* c*.*

Proof: First, since the weights considered are always integers, we can consider all the weights to be 1 by replacing a line ℓ of weight $\omega(\ell)$ by $\omega(\ell)$ copies of it. Perturb slightly the lines, so that there is no pair of them which is parallel.

Next, consider the set of vertices $X(r) = \bigcup_{p \in \mathsf{P}} \mathsf{b}_{\times}(p, r)$. Clearly, as long as the balls of $X(r)$ are disjoint, the number of vertices of the arrangement $\mathcal{A}$ included in $X(r)$ is at least $nr^2/8$, by Lemma 24.2.2. In particular, the overall number of vertices in the arrangement is $\binom{W}{2}$, and as such it must be true that two balls of $X(r)$ are not disjoint when $nr^2/8 > \binom{W}{2} = W(W-1)/2$. Namely, this happens when $r^2 > 4W^2/n$. This happens when $r > 2W/\sqrt{n}$. As such, for $r = \lceil 2W/\sqrt{n} \rceil + 1$ there must be a vertex v in the arrangement $\mathcal{A}$ and two points $q, t \in \mathsf{P}$, such that $\mathrm{d}_{\times}(q, v) \leq r$ and $\mathrm{d}_{\times}(v, t) \leq r$, and by the triangle inequality, we have that

$$\mathrm{d}_{\times}(q, t) \leq \mathrm{d}_{\times}(q, v) + \mathrm{d}_{\times}(v, t) \leq 2r \leq 4W/\sqrt{n} + 3.$$

Namely, q and t are within the required crossing distance. ■

Claim 24.2.4. *Any line in the plane crosses at most* $O(\sqrt{n})$ *edges of the resulting spanning tree* T*.*

Proof: Let W_i denote the total weight of the lines in L in the end of the ith iteration. We have that $W_0 = |\mathsf{L}| \leq 6\binom{n}{2}$, and since there are $n_i = n - i + 1$ points in P in the beginning of the ith iteration, it follows, by Lemma 24.2.3, that the algorithm found a segment s_i of weight at most $\mathsf{c}W_{i-1}/\sqrt{n_i}$. We double the weight of all the lines that intersect s_i. Thus,

$$W_i \leq W_{i-1} + \mathsf{c}W_{i-1}/\sqrt{n_i} \leq \left(1 + \frac{\mathsf{c}}{\sqrt{n_i}}\right) W_{i-1} \leq \prod_{k=1}^{i} \left(1 + \frac{\mathsf{c}}{\sqrt{n_k}}\right) W_0$$
$$\leq W_0 \prod_{k=1}^{i} \exp\left(\frac{\mathsf{c}}{\sqrt{n_k}}\right) = W_0 \exp\left(\sum_{k=1}^{i} \frac{\mathsf{c}}{\sqrt{n-k+1}}\right),$$

since $1 + x \leq e^x$, for all $x \geq 0$. In particular, we have that

$$W_n \leq W_0 \exp\left(\sum_{k=1}^{n} \frac{\mathsf{c}}{\sqrt{n-k+1}}\right) \leq 6\binom{n}{2} \exp\left(\sum_{k=1}^{n} \frac{\mathsf{c}}{\sqrt{k}}\right) \leq 3n^2 \exp\left(4\mathsf{c}\sqrt{n}\right),$$

since $\sum_{k=1}^{n} 1/\sqrt{k} \leq 1 + \int_{x=1}^{n} (1/\sqrt{x}) dx = 1 + [2\sqrt{x}]_{x=1}^{x=n+1} \leq 4\sqrt{n}$. As for the other direction, consider the heaviest line $\hbar$ in L in the end of the execution of the algorithm. If it crosses Δ segments T, then its weight is 2^{Δ}, and as such

$$2^{\Delta} = \omega(\hbar) \leq W_n \leq 3n^2 \exp\left(4\mathsf{c}\sqrt{n}\right).$$

It follows that $\Delta = O(\log n + \sqrt{n})$, as required. Namely, any line in the plane crosses at most $O(\sqrt{n})$ edges of T. ■

Theorem 24.2.5. *Given a set* P *of* n *points in the plane, one can compute a spanning tree* T *of* P *such that any line crosses at most* $O(\sqrt{n})$ *edges of* T*. The running time is polynomial in* n*.*

This result also holds in higher dimensions. The proof is left as an exercise (see Exercise 24.4.1).

Theorem 24.2.6. *Given a set* P *of* n *points in* $\mathbb{R}^d$*, one can compute a spanning tree* T *of* P *such that any hyperplane crosses at most* $O(n^{1-1/d})$ *edges of* T*. The running time is polynomial in* n*.*

24.2.3. An application – better discrepancy

Before extending this result to more abstract settings, let us quickly outline why this spanning tree of low crossing number leads to better discrepancy and a smaller ε-sample for halfplanes.

Indeed, let us turn T into a cycle by drawing a tour walking around the edges of T; formally, we double every edge of T and observe that the resulting graph is Eulerian, and we extract the Eulerian tour from this graph. Clearly, this cycle C has twice the crossing number of the tree.

Next, consider a curve γ in the plane and a segment s with the same endpoints, and observe that a line that intersects s must also intersect γ (but not vice versa!). As such, if we shortcut parts of C by replacing them by straight segments, we are only decreasing the crossing number of C. To this end, start traversing C from some arbitrary point $p_0 \in \mathsf{P}$, and start "walking" along C. Whenever we are at point $p \in \mathsf{P}$, along the cycle C, we go directly from there (i.e., shortcut) to the next point visited by C that was not visited yet. Let C' be the resulting cycle. Clearly, it visits all the points of P and it has a crossing number which is at most twice the crossing number of T. Now, assuming that $n = |\mathsf{P}|$ is even, pick all the even edges of C' so they form a prefect matching M of P. We have:

Lemma 24.2.7. *One can compute a perfect matching* M *of a set of* n *points in the plane, such that every line crosses at most* $O(\sqrt{n})$ *edges of the matching.*

Now, going back to the discrepancy question, we remind the reader that we would like to color the points by $\{-1, 1\}$ such that for any halfplane the 'balance' of the coloring is as close to perfect as possible. To this end, we use the matching of the above lemma and plug it into Theorem **??**. Since any line ℓ crosses at most $\#_\ell = O(\sqrt{n})$ edges of M, we get the following result.

Theorem 24.2.8. *Let* P *be a set of* n *points in the plane. One can compute a coloring* χ *of* P *by* $\{-1, 1\}$*, such that for all halfplanes* h*, it holds that* $|\chi(h)| = O\Big(n^{1/4}\sqrt{\ln n}\Big)$*.*

In words, the discrepancy of n points in relation to halfplanes is $O\Big(n^{1/4}\sqrt{\ln n}\Big)$. This also implies that one can construct a better ε-sample in this case. But before dwelling on this, let us prove a more general version of the spanning tree lemma.

24.2.4. Spanning tree for space with bounded shattering dimension

Let $\mathsf{S} = (\mathsf{X}, \mathcal{R})$ be a range space with shattering dimension δ and dual shattering dimension $\delta^\star$ (see Definition **??**). Let $\mathsf{P} \subseteq \mathsf{X}$ be a set of n points. Consider a spanning tree T of P. The tree T is defined by $n-1$ edges $e_i = \{p_i, q_i\} \subseteq \mathsf{P}$. An edge $\{p, q\}$ ***crosses*** a range $\mathbf{r} \in \mathcal{R}$ if

$|\{p, q\} \cap \mathbf{r}| = 1$. Our purpose is to build a spanning tree such that every range of $\mathcal{R}$ crosses a small number of edges of T. The reader can verify that this abstract setting corresponds to the more concrete problem described above.

We will concentrate on the restricted space $\mathsf{S}_{|\mathsf{P}} = \Big(\mathsf{P}, \mathcal{R}_{|\mathsf{P}}\Big)$. It is easy to verify that $\mathsf{S}_{|\mathsf{P}}$ has shuttering dimension $\leq \delta$ and dual shattering dimension bounded by $\delta^\star$. Observe that $m = \Big|\mathcal{R}_{|\mathsf{P}}\Big| \leq O\Big(n^\delta\Big)$. Let $\mathcal{F}$ be a weighted subset of $\mathcal{R}_{|\mathsf{P}}$.

24.2.4.1. The algorithm

The algorithm to compute a spanning tree with low crossing number would work as before: Initialize the weight of all the ranges of $\mathcal{R}_{|\mathsf{P}}$ to 1 and the spanning tree to be empty.

Now, at each iteration, the algorithm computes a pair $\{p, q\} \subseteq \mathsf{P}$ such that the total weight of the ranges of $\mathcal{F}$ it crosses is minimized. Next, the algorithm doubles the weight of these ranges, adds the edge $\{p, q\}$ to the spanning tree, and deletes, say, q from P. The algorithm repeats this process till there remains only a single point in P. Clearly, we have computed a spanning tree.

24.2.4.2. Analysis

We need to bound the crossing number of the generated tree. As before, the analysis boils down to proving the existence of two "close" points.

Lemma 24.2.9. *Let $\mathsf{S} = (\mathsf{X}, \mathcal{R})$ be a range space with dual shattering dimension $\delta^\star$. Let $\mathsf{P} \subseteq \mathsf{X}$ be a set of n points, and let $\mathcal{F}$ be a weighted set of ranges from $\mathcal{R}_{|\mathsf{P}}$, with total weight W. Then, there is a pair of points $\mathsf{e} = \{p, q\} \subseteq \mathsf{P}$, such that the total weight of the ranges crossed by e is at most $O\Big(W\delta^\star n^{-1/\delta^\star} \log n\Big)$.*

Proof: Let ε be a parameter to be specified shortly. For an edge $\{u, v\} \subseteq \mathsf{P}$, consider the set of ranges that its crosses:

$$C(u, v) = \Big\{\mathbf{r} \;\Big|\; (u \in \mathbf{r} \text{ and } v \notin \mathbf{r}) \quad \text{or} \quad (u \notin \mathbf{r} \text{ and } v \in \mathbf{r}), \quad \text{for} \quad \mathbf{r} \in \mathcal{F}\Big\}.$$

Observe that the range space $\mathsf{U} = (\mathsf{P}, \mathcal{F})$ has dual shattering dimension $\delta^\star$. Next, consider the dual range space $\mathsf{U}^\star = (\mathcal{F}, \mathsf{P}^\star)$; see Definition **??**. This range space has (primal) shattering dimension bounded by $\delta^\star$, by assumption. Consider the new range space

$$\mathsf{T}^\star = (\mathcal{F},\ \{\mathcal{F}_p \oplus \mathcal{F}_q \mid \mathcal{F}_p, \mathcal{F}_q \in \mathsf{P}^\star\}),$$

where $\oplus$ is the symmetric difference of the two sets; namely,

$$\mathbf{r} \oplus \mathbf{r}' = (\mathbf{r} \setminus \mathbf{r}') \cup (\mathbf{r}' \setminus \mathbf{r}), \qquad \text{and} \qquad \mathcal{F}_p = \{\mathbf{r} \mid \mathbf{r} \in \mathcal{F} \text{ and } p \in \mathbf{r}\}.$$

By arguing as in Corollary **??**, we have that $\mathsf{T}^\star$ has a shattering dimension at most $2\delta^\star$. Furthermore, the projected range space $\mathsf{T}^\star$ has $C(u, v)$ as one of its ranges, for any $u, v \in \mathsf{P}$.

So consider a random sample $\mathcal{C}$ of size $O((\delta^\star/\varepsilon)\log(\delta^\star/\varepsilon))$ from $\mathcal{F}$ (note that $\mathcal{F}$ is weighted and the random sampling is done accordingly). By the ε-net theorem (see the ε-net theorem and Remark **??**), we know that with constant probability, this is an ε-net of $\mathsf{T}^\star$. Namely, a set $C(u, v)$ which does not contain any range of $\mathcal{C}$ has weight at most εW (indeed, if the weight of $C(u, v)$ exceeds εW, then it must contain an element of the ε-net $\mathcal{C}$).

On the other hand, the projection of the range space $\mathsf{U}^\star = (\mathcal{F}, \mathsf{P}^\star)$ to $\mathcal{C}$ is the range space $\mathsf{U}^\star_{|\mathcal{C}} = \Big(\mathcal{C}, \mathsf{P}^\star_{|\mathcal{C}}\Big)$. This range space has shattering dimension at most $\delta^\star$, and as such

$$\mu(\varepsilon) = \Big|\mathsf{P}^\star_{|\mathcal{C}}\Big| = O\Big(|\mathcal{C}|^{\delta^\star}\Big) \leq \left(\mathsf{c}\frac{\delta^\star}{\varepsilon} \log \frac{\delta^\star}{\varepsilon}\right)^{\delta^\star}$$

ranges[2], where c is an appropriate constant. In particular, let us pick ε as small as possible, such that $\mu(\varepsilon) < n = |\mathsf{P}|$. We are then guaranteed that there are two points $p, q \in \mathsf{P}$ such that $\mathcal{C}_p = \mathcal{C}_q$.

[2]If the ranges of $\mathcal{R}$ are geometric shapes, it means that the arrangement formed by the shapes of $\mathcal{C}$ has at most $\mu(\varepsilon)$ faces.

Let us regroup. We found two points p and q that do not cross any range of $\mathcal{C}$. But $\mathcal{C}$ is an ε-net for $\mathsf{T}^\star$ (with constant probability, so assume it is a net). This implies that $C(p,q)$ has total weight at most εW. Namely, the total weight of the ranges crossing the edge $\{p,q\}$ is at most εW. As such, we found a "light" edge. Note that we can verify the weight of $\{p,q\}$ by computing explicitly all the ranges of $\mathcal{F}$ that cross it. If it is not light, we can repeat this algorithm till we succeed.

We are left with the task of picking ε. Naturally, we would like ε to be as small as possible. Now, for $\varepsilon = \mathsf{c}_1 \delta^\star n^{-1/\delta^\star} \log(n)$ it holds that $\mu(\varepsilon) < n$ if c_1 is a sufficiently large constant, as can be easily verified. ■

To complete the argument, we need to bound the total weight of the ranges in the end of this process. For some constant c_2, it is bounded by

$$U \leq \left|\mathcal{R}_{|\mathsf{P}}\right| \prod_{i=1}^{n}\left(1 + \mathsf{c}_1 \frac{\log i}{i^{1/\delta^\star}}\right) \leq \mathsf{c}_2 n^\delta \prod_{i=1}^{n}\left(1 + \mathsf{c}_1 \frac{\log i}{i^{1/\delta^\star}}\right) \leq \mathsf{c}_2 n^\delta \exp\left(\sum_i \mathsf{c}_1 \frac{\log i}{i^{1/\delta^\star}}\right)$$
$$\leq \mathsf{c}_2 n^\delta \exp\Bigl(O\Bigl(\mathsf{c}_1 n^{1-1/\delta^\star} \log n\Bigr)\Bigr).$$

Now, the crossing number of the resulting tree T, for any range $\mathbf{r} \in \mathcal{R}$, is bounded by $\lg U = O\Bigl(\delta \log n + n^{1-1/\delta^\star} \log n\Bigr)$. We thus conclude:

Theorem 24.2.10. *Given a range space $\mathsf{S} = (\mathsf{X}, \mathcal{R})$ with shattering dimension δ and dual shattering dimension $\delta^\star$ and a set $\mathsf{P} \subseteq \mathsf{X}$ of n points, one can compute, in polynomial time (assuming that δ and $\delta^\star$ are constants), a spanning tree T of P, such that any range of $\mathcal{R}$ crosses at most $O\Bigl(\delta \log n + n^{1-1/\delta^\star} \log n\Bigr)$ edges of T.*

Plugging this into the discrepancy machinery, we get the following result.

Theorem 24.2.11. *Given a range space $\mathsf{S} = (\mathsf{X}, \mathcal{R})$ with shattering dimension δ and dual shattering dimension $\delta^\star$ and a set $\mathsf{P} \subseteq \mathsf{X}$ of n points, one can compute, in polynomial time (assuming that δ and $\delta^\star$ are constants), a coloring of P by $\{-1, 1\}$, such that for any range $\mathbf{r} \in \mathcal{R}$, we have that $|\chi(\mathbf{r})| = O\Bigl(\delta n^{1/2 - 1/2\delta^\star} \log n\Bigr)$.*

Proof: Indeed, compute a spanning tree with low crossing number using the algorithm of Theorem 24.2.10. Next, extract a matching from it with low crossing number, and use this matching in computing a good discrepancy; see Theorem **??**. ■

Now, we can extract a small ε-sample from such a range space by using the construction of an ε-sample via discrepancy. We get the following result.

Theorem 24.2.12. *Given a range space $\mathsf{S} = (\mathsf{X}, \mathcal{R})$ with shattering dimension δ and dual shattering dimension $\delta^\star$, a set $\mathsf{P} \subseteq \mathsf{X}$ of n points, and $\varepsilon > 0$, one can compute, in polynomial time (assuming that δ and $\delta^\star$ are constants), an ε-sample for the pointset P of size $O\left(\Bigl((\delta/\varepsilon)\log(\delta/\varepsilon)\Bigr)^{2-2/(\delta^\star+1)}\right)$.*

Proof: We plug the improved bound of Theorem 24.2.11 into Theorem **??**. Clearly, $\tau(n)$ (which is how good of a sample the points colored by $+1$ are in one round of coloring) in our case is

$$\tau(n) = \frac{\mathrm{disc}\Bigl(\mathsf{S}_{|\mathsf{P}}\Bigr)}{|n|} = O\Bigl(\delta n^{-(\delta^\star+1)/2\delta^\star} \log n\Bigr).$$

We repeat this halving process, and the required ε-sample is the smallest $m = n/2^i$, such that $\mathsf{c}_3\tau(m) < \varepsilon$, for some constant c_3. Namely, it is sufficient that, for some constant c_4, we have that

$$\mathsf{c}_4 \delta m^{-(\delta^\star+1)/2\delta^\star} \log m < \varepsilon.$$

Now, since $2\delta^\star/(\delta^\star+1) = 2-2/(\delta^\star+1)$, it is now easy to verify that this holds for $m = O\Bigl(\bigl(\frac{\delta}{\varepsilon}\log\frac{\delta}{\varepsilon}\bigr)^{2-2/(\delta^\star+1)}\Bigr)$. ■

Theorem 24.2.12 is a surprising result. It implies that one can construct ε-samples of size with subquadratic dependency of ε. Note that a regular random sample cannot provide such a guarantee even for a single range! Namely, we have broken the glass ceiling of what can be achieved by random sampling.

24.3. Bibliographical notes

The stabbing tree result is due to Welzl [Wel92], which is in turn a simplification of the work of Chazelle and Welzl [CW89]. The running time of computing the good spanning tree can be improved to $O(n^{1+\varepsilon})$ [Wel92].

The extension of the spanning tree result to range spaces with low VC dimension is due to Matoušek *et al.* [MWW93].

24.4. Exercises

Exercise 24.4.1 (Spanning tree with low crossing number in $\mathbb{R}^d$). Prove Theorem 24.2.6.

Exercise 24.4.2 (Tight example for spanning tree with low crossing number). Show that in the worst case, one can pick a point set P of n points in the plane, such that for any spanning tree T of P, there exists a line ℓ, such that ℓ crosses $\Omega(\sqrt{n})$ edges of T.

Exercise 24.4.3 (Spanning tree with relative crossing number). Let P be a set of n points in the plane. For a line ℓ, let w_ℓ^+ (resp., w_ℓ^-) be the number of points of P lying above (resp., below or on) ℓ, and define the *weight* of ℓ, denoted by $\omega\ell$, to be $\min(w_\ell^+, w_\ell^-)$.

Show that one can construct a spanning tree T for P such that any line ℓ crosses $O\bigl(\sqrt{\omega\ell}\log(n/\omega\ell)\bigr)$ edges of T.

References

[CW89] B. Chazelle and E. Welzl. Quasi-optimal range searching in space of finite vc-dimension. *Discrete Comput. Geom.*, 4: 467–489, 1989.

[MWW93] J. Matoušek, E. Welzl, and L. Wernisch. Discrepancy and ε-approximations for bounded VC-dimension. *Combinatorica*, 13: 455–466, 1993.

[Wel92] E. Welzl. On spanning trees with low crossing numbers. *Data Structures and Efficient Algorithms, Final Report on the DFG Special Joint Initiative.* Vol. 594. Lect. Notes in Comp. Sci. Berlin, Heidelberg: Springer-Verlag, 1992, pp. 233–249.