Revisiting Random Points: Combinatorial Complexity and Algorithms

Sariel Har-Peled* Elfarouk Harb†

November 2, 2023

Abstract

Consider a set P of n points picked uniformly and independently from $[0,1]^d$, where d is a constant. Such a point set is well behaved in many aspects and has several structural properties. For example, for a fixed $r \in [0,1]$, we prove that the number of pairs of $\binom{P}{2}$ at a distance at most r is concentrated within an interval of length $O(n \log n)$ around the expected number of such pairs for the torus distance. We also provide a new proof that the expected complexity of the Delaunay triangulation of P is linear – the new proof is simpler and more direct than previous proofs.

In addition, we present simple linear time algorithms to construct the Delaunay triangulation, Euclidean MST, and the convex hull of the points of P. The MST algorithm uses an interesting divide-and-conquer approach. Finally, we present a simple $\tilde{O}(n^{4/3})$ time algorithm for the distance selection problem, for $d = 2$, providing a new natural justification for the mysterious appearance of $n^{4/3}$ in algorithms for this problem.

1. Introduction

Input model. Fix a constant dimension $d \geq 2$. For $i \in [\![n]\!] = \{1, \ldots, n\}$, uniformly and independently sample a point p_i from $[0,1]^d$. Let $P = \{p_i \mid i \in [\![n]\!]\}$. The *euclidean graph* on P is $G(P) = \left(P, \binom{P}{2}\right)$, with the edge $p_i p_j$ having weight $\omega(p_i p_j) = \|p_i p_j\|$, for $p_i, p_j \in P$, where $\binom{P}{2} = \{pq \mid p, q \in P\}$. This graph has quadratic number of edges, but is defined by only $O(n)$ input numbers. Natural questions to ask about P and $G(P)$ include:

(A) What is the combinatorial complexity of the convex-hull/Delaunay triangulation of P?

(B) How quickly can one compute the convex-hull/Delaunay triangulation/MST/etc of P?

(C) What is the length of the median edge in $G(P)$, and how concentrated is this value?

All these questions have surprisingly good answers – linear complexity, linear running time algorithms, and strong concentration, respectively. Here, we revisit these questions, presenting new simpler proofs and algorithms for them.

1.1. Background

There is a lot of work in stochastic and integral geometry on understanding the behavior of random point sets, and the structures they induce [San53, WW93, Cal10, SW10]. As the name suggests, for many of the questions one states, an integral is set up whose solution is the desired quantity, and one

*Department of Computer Science; University of Illinois; 201 N. Goodwin Avenue; Urbana, IL, 61801, USA; sariel@illinois.edu; http://sarielhp.org/. Work on this paper was partially supported by NSF AF award CCF-2317241.

†Department of Computer Science; University of Illinois; 201 N. Goodwin Avenue; Urbana, IL, 61801, USA; eyharb2@illinois.edu; https://farouky.github.io/.

remains with the (usually painful) task of solving the integral[1]. In this paper, we focus mainly on direct combinatorial arguments of said results.

Closest pair and spread. The ***spread*** of a point set $P \subset \mathbb{R}^d$ is the ratio between the diameter and the closest pair distance of P. Formally, it is the quantity $\Phi = \Phi(P) = \mathrm{diam}(P)/\mathrm{cp}(P)$, where $\mathrm{diam}(P) = \max_{p,q \in P} \|pq\|$ and $\mathrm{cp}(P) = \min_{p,q \in P : p \neq q} \|pq\|$. For a set P of n points sampled uniformly at random from $[0,1]^d$, It is not hard to verify [HJ20] that $\mathbb{E}[\mathrm{cp}(P)] = \Omega(1/n^{2/d})$. This intuitively suggests that $\mathbb{E}[\Phi(P)] = O(n^{2/d})$ - (a formal proof of this requires a bit more effort).

Convex-hull. The Convex-hull of n points in $\mathbb{R}^d$ has combinatorial complexity $\Theta(n^{\lfloor d/2 \rfloor})$ in the worst case (here, combinatorial complexity refers to the number of vertices and faces). It can be computed in $O(n \log n + n^{\lfloor d/2 \rfloor})$ time [Cha93]. Surprisingly, the expected complexity of the convex-hull of random points picked from $[0,1]^d$ is $O(\log^{d-1} n)$ [BKST78]. The exact bound depends on the underlying domain from which the points are sampled. For example, if the sample is taken from a ball in $\mathbb{R}^d$, the expected complexity is $O(n^{(d-1)/(d+1)})$ [Ray70]. See [Har11b] and references therein for more details. Dwyer [Dwy88] provides an expected linear time algorithm for computing the convex hull of a set of points picked from $[0,1]^d$. As hinted to earlier, the analysis is not elementary and uses heavy tools to show the result.

Delaunay triangulation. The Delaunay triangulation $\mathcal{D}$ of n points in $\mathbb{R}^d$ has combinatorial complexity $\Theta(n^{\lceil d/2 \rceil})$ in the worst case. It can be computed in $O(n \log n + n^{\lceil d/2 \rceil})$ time [Cha93]. Dwyer [Dwy91] show that when the points are uniformly sampled from a d-dimensional unit ball (instead of a d-cube), the complexity of the Voronoi diagram (and consequently its dual, $\mathcal{D}$) is also linear, and gave an $O(n)$ time expected time algorithm for constructing it. However, Dwyer's algorithm is involved and its analysis is nontrivial with reliance on algebraic and integral tools.

Minimum spanning trees. There is a lot of work on MST and EMST (Euclidean minimum spanning tree). Since EMST is a subgraph of the *Gabriel graph* of P – that is, the graph where two points $p, q \in P$ are connected by an edge, if their diametrical ball does not contain any point of P in its interior. The Gabriel graph is a subgraph (of the 1-skeleton) of $\mathcal{DT}(P)$, the Delaunay Triangulation of P. Thus, one can calculate $\mathcal{DT}(P)$ (in linear time), and then run Karger *et al.* expected linear time MST algorithm [KKT95] on $\mathcal{DT}(P)$. The algorithm of Karger *et al.* uses as a black box a procedure to identify all the edges in the graph that are too heavy to belong to a minimum spanning tree, given a candidate spanning tree. Such spanning tree "verifiers" are relatively complicated to implement in linear time [Hag09]. Developing deterministic linear time MST algorithm is still an open problem, although Chazelle presented [Cha00] a $O(n + m\alpha(n,m))$ time algorithm where n, m are the number of vertices and edges respectively (as $\alpha(n,m)$ is at most 4 for all practical purposes, this is essentially a linear time algorithm). More bizarrely, a deterministic *optimal* algorithm is known [PR02], but its running time complexity is not known. None of these algorithms can be described as simple.

For minor-closed graphs, Mareš [Mar04] gave two linear time algorithms to construct the MST in $O(n+m)$ time. In the plane, the Delaunay Triangulation is a planar graph, and thus given the Delaunay triangulation the MST can be computed in linear time (this is no longer applicable, already in 3d).

[1]Historically, the field was not named integral geometry *because* it involved integrals in the calculus sense. The origin of the word, which derives from the German "Integralgeometrie", was coined and popularized by Blaschke in their book. We thank an anonymous reviewer for mentioning this.

Distance selection. Given a set P of n points in the plane, and a number k, the distance selection problem asks for the kth small distance in the $\binom{n}{2}$ pairwise distances induced by the points of P. In the plane, this can be computed in $O(n \log n + k)$ time [Cha01], or alternatively in $O(n^{4/3})$ time [CZ21] for general sets of points. An $(1 \pm \varepsilon)$-approximation can be computed in linear time [HR15].

1.2. Our results

We provide simple and elementary proofs for several of the results mentioned above, and we also provide (conceptually) simple algorithms for several of the problems mentioned above:

(A) kTH DISTANCE CONCENTRATION. Fix a value of $r \in [0,1]$. Let $f_r = f_r(P)$ denote the number of pairs p_ip_j with $\|p_ip_j\|_T \leq r$, where $\|p_ip_j\|_T$ is the *torus topology distance* between p_i and p_j (defined in Eq. (3.1)). Note that $f_r(P) \in \{0, ..., \binom{n}{2}\}$. It is not hard to show that $\mathbb{P}\big[|f_r - \mathbb{E}[f_r]| > \widetilde{\Omega}(n^{3/2})\big] \leq 1/n^{O(1)}$ using Chernoff's inequality and the union bound, where $\widetilde{\Omega}$ and $\widetilde{O}$ hide polylogarithmic terms in n. However, in Section 3, we show a significantly stronger concentration, namely that the interval has length $\widetilde{O}(n)$ with high probability[2]: $\mathbb{P}\big[|f_r - \mathbb{E}[f_r]| > \widetilde{\Omega}(n)\big] \leq 1/n^{O(1)}$. The new concentration proof uses martingales together with bounded differences concentration inequality that can handle low probability failure. To the best of our knowledge this result is new, and is an interesting property of random points. (We conjectured this claim after observing this behavior, of strong concentration, in computer simulations we performed.). The proof is an interesting application of a McDiarmid's inequality variant that allows a (small) probability of large variation, when applying the standard McDiarmid's inequality would otherwise fail.

(B) CONVEX HULL. In Section 4, as a warm-up exercise, we provide an $O(n)$ expected time algorithm to construct $\mathcal{C}(P)$, the convex hull of P. Dwyer [Dwy88] presented a divide and conquer algorithm. Our algorithm is somewhat different as it uses a quadtree for the partition scheme, and is the building block for the later algorithms.

(C) LINEAR COMPLEXITY OF DELAUNAY TRIANGULATION. We provide a new proof that the expected complexity of the Delaunay triangulation of P is linear, where P is a set of n points picked uniformly and independently from $[0,1]^d$. The new proof, presented in Section 5, is simpler and more direct than existing proofs. The linear bound is quite easy to derive for points in the inner part of the cube (we refer to this part of the cube as the *fortress*), but the outer part (i.e., the *moat*) requires more work because of boundary issues.

(D) LINEAR TIME ALGORITHM FOR DELAUNAY TRIANGULATION. In Section 6, we present an expected linear time algorithm for computing the Delaunay triangulation. The algorithm computes, for each point, the points it might interact with, and the local Delaunay triangulation of these points. The algorithm then stitch these local structures together to get the global triangulation.

(E) EUCLIDEAN MST. Since the MST of P is a subgraph of (the 1-skeleton) of $\mathcal{DT}(P)$, the (general but more complicated) expected linear time MST algorithm from [KKT95] could be applied to $\mathcal{DT}(P)$ to calculate the EMST of P in linear time. For $d = 2$, it is known that Borůvka's algorithm implemented efficiently[3] takes linear time, since planarity is preserved between rounds. In particular, we conjecture that Borůvka's algorithm takes linear time when run on $\mathcal{DT}(P)$, in higher dimensions, but we were unable to prove it.

[2]Here, an event A_n happens with *high probability* if $\mathbb{P}[A_n] \geq 1 - 1/n^{O(1)}$.
[3]Some textbook implementations would run in $O(n \log n)$ time, even if the graph is planar.

Instead, in Section 7, we present an algorithm for constructing the EMST of P, in expected linear time, using a simple algorithm that is the adaption of Borůvka's algorithm to use divide and conquer over a quadtree storing the points. The correct propagation of subtrees of the MST that can be computed when restricted to a subproblem, together with a "minimal" set of edges that might participate in the MST, is the main new idea of our new algorithm. We believe the new algorithm should be of interest when trying to compute MSTs, or similar structures, for huge graphs where one has to distribute the computation across several computers/nodes.

(F) Distance selection. We show a simple algorithm for distance selection for P that works in expected $O(n^{4/3}\log^{2/3} n)$ time. The new algorithm achieves this running time by partitioning the problem into (roughly) $O(n^{2/3})$ special instances involving (roughly) $O(n^{1/3})$ points concentrated in "tiny" disks, and a set of points that lies in a ring, of radius r, containing (roughly) $O(n^{2/3})$ points. Each of these instances can be solved by a direct point-location algorithm in (roughly) $O(n^{2/3})$ time. In the general case, one has to rely on a more complicated divide and conquer strategy (implemented using cuttings3), together with duality, to reach such unbalanced instances that can be solved using brute force (see [CZ21] and references therein). Thus, the new algorithm provides a new elegant and intuitive explanation where the mysterious $n^{4/3}$ term rises from, in addition for providing a simple algorithm that might work better in practice than previous algorithms.

A comment on the paper organization. Since this paper has many results, and is long, we ordered our results in such a way, that (hopefully) the first ten pages convey our basic approach and ideas. We did move some (more minor) proofs to an appendix.

2. Preliminaries

Notations. The O notation hides constants that depend (usually exponentially) on d.

2.1. VC dimension and the ε-net and ε-sample theorems

The main ingredient in almost all our results is the ε-net/sample theorems. In this subsection, we give a quick introduction, see [AS00] or [Har11a] for more details. We do not assume prior knowledge of this topic.

Definition 2.1. A ***range space*** $S = (\mathcal{C}, \mathcal{F})$ is a pair, where $\mathcal{C}$ is a set, and $\mathcal{F}$ is a family of subsets of $\mathcal{C}$. The elements of $\mathcal{C}$ are *points* and the elements of $\mathcal{F}$ are *ranges*.

A subset $B \subseteq \mathcal{C}$, is ***shattered*** by $\mathcal{F}$ if the $|\{r \cap B \mid r \in \mathcal{F}\}| = 2^{|\mathcal{C}|}$. The ***Vapnik-Chervonenkis*** dimension (or VC-dimension) of the range space $S = (\mathcal{C}, \mathcal{F})$ is the maximum cardinality of a shattered subset of $\mathcal{C}$.

Example 2.2. Suppose $\mathcal{C} = \mathbb{R}^2$ and $\mathcal{F}$ is the set of disks in $\mathbb{R}^2$. For any set of three (not colinear) points $T = \{p_1, p_2, p_3\} \subseteq \mathcal{C}$, and any subset $T' \subseteq T$, one can find a disk containing T', and avoiding the points of $T \setminus T'$. Thus, the VC dimension of disks in the plane is 3. It is easy to verify that no four points can be shattered, and thus the VC dimension of this range space is 3.

Example 2.3. In general, for points in $\mathbb{R}^d$ and balls or halfspace ranges, the VC dimension is $d + 1$. Another noteworthy range is axis-parallel rectangles which have VC dimension $2d$.

For simplicity of exposition, assume $\mathcal{C}$ to be a finite set of points. An ε-***net*** captures all "heavy" ranges. That is, if we sample a "sufficiently" large subset $N \subseteq \mathcal{C}$, then any range $r \in \mathcal{F}$ containing "enough points" from $\mathcal{C}$ must also contain a point from N with high probability. The ε-***sample*** is similar, asserting that for any range $r \in \mathcal{F}$, the fractions $\frac{|N \cap r|}{|N|}$ and $\frac{|\mathcal{C} \cap r|}{|\mathcal{C}|}$ are ε-close, with high probability. The formal definition is stated below.

Definition 2.4. Let $(\mathcal{C}, \mathcal{F})$ be a range space, and let $C \subset \mathcal{C}$ be a finite subset. For $0 < \varepsilon < 1$, a subset $N \subseteq C$, is an ε-***net*** for C if for any range $r \in \mathcal{F}$, we have $|r \cap C| \geq \varepsilon |C| \implies r \cap N \neq \emptyset$.

Definition 2.5. Let $(\mathcal{C}, \mathcal{F})$ be a range space, and let C be a finite subset of $\mathcal{C}$. For $\varepsilon \in (0,1)$, a subset $N \subseteq C$, is an ε-***sample*** for C if for any range $r \in \mathcal{F}$, we have

$$\left| \frac{|N \cap r|}{|N|} - \frac{|C \cap r|}{|C|} \right| \leq \varepsilon.$$

Finally, the ε-net and ε-sample theorems characterizes quantitatively the size of the sample needed to have the desired property.

Theorem 2.6 (ε-net theorem, [HW87]). *Let $(\mathcal{C}, \mathcal{F})$ be a range space of VC-dimension d, let $C \subseteq \mathcal{C}$ be a finite subset, and suppose $\varepsilon > 0, \delta < 1$. Let N be a random sample from C with m independent draws, where $m \geq \max\Big(\frac{4}{\varepsilon}\log\frac{2}{\delta}, \frac{8d}{\varepsilon}\log\frac{8d}{\varepsilon}\Big)$. Then N is an ε-net for C with probability at least $1-\delta$.*

Theorem 2.7 (ε-sample theorem, [VC71, VC13]). *Let $(\mathcal{C}, \mathcal{F})$ be a range space, where its VC-dimension is d. Let $C \subseteq \mathcal{C}$ be a finite subset, and suppose $\varepsilon > 0, \delta < 1$ are parameters. Let N be a random sample of size m from C, where $m \geq \min\Big(|C|, \frac{32}{\varepsilon^2}\big(d\log\frac{d}{\varepsilon} + \log\frac{1}{\delta}\big)\Big)$. Then, N is an ε-sample for C, with probability at least $1-\delta$.*

2.2. Bounding the moments

In the following, n is fixed, and let P be a set of n points picked randomly, uniformly and independently from $[0,1]^d$. Throughout, we use the following fixed quantities:

$$\varphi = \frac{c_d \ln n}{n} \qquad \text{and} \qquad \delta = \varphi^{1/d}, \tag{2.1}$$

where $c_d > 0$ a sufficiently large constant that depend only on d.

Throughout the paper, we often need to bound the moments of the number of points of P that lie in some measurable set Ξ. The following technical lemma bounds the expected number of such points.

Lemma 2.8. *Let $\Xi \subseteq [0,1]^d$ be a measurable set. If $\alpha = vol(\Xi) \geq 1/n$, then for $t > 2e$, we have*

$$\mathbb{P}[|P \cap \Xi| > t \cdot \alpha n] \leq 1/2^{t\alpha n}.$$

Furthermore, for any constant $\kappa \geq 1$, we have that $\mathbb{E}\big[|P \cap \Xi|^\kappa\big] = O\big((\alpha n)^\kappa\big)$ (the O hides here a constant that depend on κ).

Proof: The number of points of P falling into Ξ, is a binomial distribution, and we have

$$\mathbb{P}\big[|P \cap \Xi| > t \cdot \alpha n\big] = \sum_{i=t\alpha n+1}^{n} \binom{n}{i} \alpha^i (1-\alpha)^{n-i} \leq \sum_{i=t\alpha n+1}^{n} \Big(\frac{n\alpha e}{i}\Big)^i \leq \sum_{i=t\alpha n+1}^{n} \Big(\frac{n\alpha e}{2e\alpha n}\Big)^i \leq \frac{1}{2^{t\alpha n}},$$

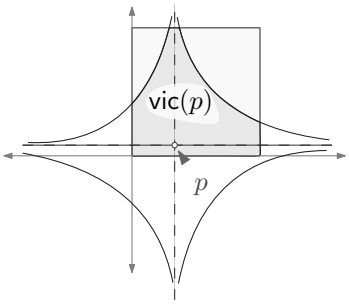

Figure 2.1: The green region is $[0,1]^2 \cap \mathsf{vic}(p)$. The light blue region is $[0,1]^2 \setminus \mathsf{vic}(p)$.

since $\binom{n}{i} \leq \left(\frac{ne}{i}\right)^i$. Thus, we have

$$\mathbb{E}\big[|P \cap \Xi|^\kappa\big] \leq \sum_{t=0}^{\infty} \big((t+1)\alpha n\big)^\kappa \, \mathbb{P}\big[|P \cap \Xi| > t \cdot \alpha n\big] \leq (2\alpha n)^\kappa \sum_{t=0}^{\infty} t^\kappa / 2^{t\alpha n} = O\big((\alpha n)^\kappa\big),$$

since $\sum_{t=0}^{\infty} t^\kappa / 2^{t\alpha n} \leq \sum_{t=0}^{\infty} t^\kappa / 2^t = O(1)$. ∎

2.3. Vicinities

For two points $p, q \in \mathbb{R}^d$, let $\mathsf{R}(p,q)$ denote the axis parallel bounding box of p and q. The ***vicinity*** of a point $p \in [0,1]^d$ is $\mathsf{vic}(p) = \left\{ q \in [0,1]^d \;\middle|\; vol(\mathsf{R}(p,q)) \leq \varphi \right\}$, where φ is specified in Eq. (2.1) (see Figure 2.1). For a number $x > 0$, let $\lceil x \rceil_2 = 2^{\lceil \log_2 x \rceil}$, and observe that $x \leq \lceil x \rceil_2 \leq 2x$, and $\lceil x \rceil_2$ is a power of two. This definition is used in the proof of the following claim.

The following claim can be proved using integration – we provide an alternative combinatorial proof for the sake of completeness.

Lemma 2.9 ([CHR16]). *For any $p \in [0,1]^d$, we have $vol(\mathsf{vic}(p)) = O(\frac{\log^d n}{n})$.*

Proof: Let o denote the origin. The area of a single quadrant of the vicinity is maximized when $p = \mathsf{o}$. There are 2^d quadrants so we have that $vol(\mathsf{vic}(p)) \leq 2^d vol(\mathsf{vic}(\mathsf{o}))$. To bound the later quantity, let $\tau > 0$ be an integer such that $2^d\varphi \leq 2^{-\tau} \leq 2^{d+1}\varphi$. As such, we have $\tau \leq \lg(1/\varphi) - d = O(\log n)$.

A ***canonical box***, is a box of the form $B = \prod_{i=1}^d [0, \alpha_i]$ such that $vol(B) = \prod_i \alpha_i = 2^{-\tau}$, where α_i is a power of two, for all i. For any point $q = (q_1, \ldots, q_d) \in [0,1]^d$, let $\mathsf{pv}(q) = \prod_{i=1}^d q_i$ be the ***point volume***. Consider all the points of $q = (q_1, \ldots, q_d) \in \mathsf{vic}(\mathsf{o})$ (i.e., these are points with $\mathsf{pv}(q) \leq \varphi$), and let $\lceil q \rceil_2 = (\lceil q_1 \rceil_2, \ldots, \lceil q_d \rceil_2)$. Observe that $\mathsf{pv}(\lceil q \rceil_2) \leq 2^d\varphi$. In particular, there exists a canonical box that contains q.

Consider a side $[0, \alpha_i]$ of a canonical box. The number α_i is a power of 2, and $1 \geq \alpha_i \geq 2^{-\tau}$. That is, $\alpha_i \in \{1, 1/2, 1/4, \ldots, 2^{-\tau}\}$. Namely, there are at most $1 + \tau$ choices for the value of each coordinate of a canonical box. As such, the number of canonical boxes is $(1+\tau)^{d-1}$, as fixing $d-1$ coordinates forces the value of the last coordinate. The volume of a canonical box is $\leq 2^{d+1}\varphi$. We conclude $\mathsf{vic}(\mathsf{o})$ is covered by the union of these boxes, and as such, $vol(\mathsf{vic}(\mathsf{o})) \leq \tau^{d-1}\varphi$, which implies the claim. ∎

The intuition for vicinities is that for a lot of the problems discussed in the introduction, any point $p \in P$ only needs to locally consider other points in its vicinity when making decisions of building the desired structures (i.e. points outside the vicinity of p are not relevant for p)

3. Sharp concentration of the kth pairwise distance

Let $P = \{p_1, \ldots, p_n\}$ be a random sequence of points, where p_i is picked uniformly and independently from $[0,1]^d$. For two numbers $x, y \in [0,1]$ their *toroidal distance* is $|x-y|_T = \min\bigl(|x-y|, 1-|x-y|\bigr)$. Let r be a fixed value in $[0,1]$. Let $f_r(P)$ denote the number of pairs p_ip_j in P with $\|p_ip_j\|_T \leq r$. Formally, we have

$$\|p_ip_j\|_T = \sqrt{\sum_{\ell=1}^{d} \bigl|p_i[\ell] - p_j[\ell]\bigr|_T^2} \qquad \text{and} \qquad f_r(P) = \Bigl|\bigl\{p_ip_j \;\bigm|\; i<j \text{ and } \|p_ip_j\|_T \leq r\bigr\}\Bigr|. \tag{3.1}$$

is the ***toroidal distance*** between p_i and p_j, and $p[\ell]$ denotes the ℓth coordinate of a point $p \in \mathbb{R}^d$. We denote the space $[0,1]^d$ under this toroidal topology by $[0,1]^d_T$. Intuitively, this is the space where we allow "wrap-around" in $[0,1]^d$, and the shortest distance between two points can be the wrap-around distance. Using this distance allows us to ignore artifacts that are generated by the boundary of the hypercube $[0,1]^d$.

The claim is that the value of f_r, which is a number in $\{0, \ldots, \binom{n}{2}\}$, is strongly concentrated. Namely, the interval of integers containing f_r, with high probability, is "short". Showing a bound of $\widetilde{O}(n^{3/2})$ on the number of values in this interval is doable via Chernoff's inequality and using the union bound. The resulting guarantee is of the form $\mathbb{P}\bigl[|f_r - \mathbb{E}[f_r]| > \widetilde{\Omega}(n^{3/2})\bigr] \leq 1/n^{O(1)}$. Here, we show a significantly stronger concentration with the interval containing $\widetilde{O}(n)$.

We conjecture this result is true for the Euclidean distance, but handling the boundary cases proved to be quite challenging. Hence the simplifying Toroidal topology assumption. We observed this strong concentration, for both the Toroidal and Euclidean case, in computer simulations.

Consider the closed ball $\mathbb{b}_T(p,r) = \bigl\{x \in [0,1]^d \;\bigm|\; \|px\|_T \leq r\bigr\}$ in $[0,1]^d_T$. We next bound the VC dimension of such balls (as a side, Gillibert *et al.* [GLM22] bounded the VC-dimension of axis-parallel boxes in this space by $O(d \log d)$).

Lemma 3.1. *For $\mathcal{A} = \bigl\{\mathbb{b}_T(p,r) \;\bigm|\; p \in \mathbb{R}^d\bigr\}$, the VC dimension of the range space $([0,1]^d, \mathcal{A})$ is $O(1)$.*

Proof: A toroidal ball consists of at most $O(2^{2d})$ regions R_i, each region being the intersection of a ball and at most $2d$ half spaces (corresponding to the boundaries of $[0,1]^d$). The VC dimension of balls and halfspaces is $d+1$, so the VC dimension of their intersection (and hence each region) is $O(1)$. Taking the union of the at most $O(2^{2d})$ regions, implies the VC dimension is at most $O(1)$, via standard argumentation [Har11a]. ∎

Next, we would like to apply Chernoff-like style inequalities to bound the probability of deviation from the expectation. The most relevant inequality here is McDiarmid's inequality for bounded differences of martingales. Unfortunately, one cannot use McDiarmid's inequality directly because by "sliding" a ball $\mathbb{b}$ in $[0,1]^d$, the number of points inside $\mathbb{b}$ might change by $O(n)$. Of course for random point set P, this is highly unlikely (the change is more likely to be $O(\sqrt{n})$) and so we will have to use a variation of McDiarmid's inequality that allows a "bad" event, where the difference might be large but happening with a small probability, and a "good" typical event where the difference is bounded.

Consider the following extension of McDiarmid's inequality (for bounded differences of martingales) where the differences are only bounded with high probability [Kut02]. It will be useful to view P from two different views[4], one as a set of individual points, and the second as a product $\Omega = \prod_{1 \leq i \leq dn} \Omega_i$ of dn probability spaces for each coordinate.

[4] As with most things in life.

Definition 3.2 ([Kut02]). Let $\Omega_1, ..., \Omega_m$ be probability spaces. Let $\Omega = \prod_i \Omega_i$, and let X be a random variable on Ω. The variable X is *strongly difference-bounded* by (b, c, ς) if the following holds. There is a "bad" subset $B \subseteq \Omega$, where $\varsigma = \mathbb{P}[w \in B]$. In addition, we require that

(i) If $\omega, \omega' \in \Omega$ differ only in the kth coordinate, and $\omega \notin B$ then $|X(\omega) - X(\omega')| \leq c$.

(ii) Furthermore, for any $\omega, \omega' \in \Omega$ differing only in the kth coordinate, $|X(\omega) - X(\omega')| \leq b$.

To decipher this definition consider the case that $c < b$: the difference between "bad" pairs can be large, but the difference between "good" (or mixed) pairs is small. The quantity b behaves like the "worst" case difference, c is the "typical" difference, and ς is the probability of the bad event happening.

Lemma 3.3 ([Kut02], Corollary 3.4). *Let $\Omega_1, ..., \Omega_m$ be probability spaces. Let $\Omega = \prod_{1 \leq i \leq m} \Omega_i$ and let X be a random variable on Ω which is strongly difference-bounded by (b, c, ς). Let $\mu = \mathbb{E}[X]$. Then, for any $\tau > 0$, and any $\alpha > 0$, we have $\mathbb{P}\big[|X - \mu| \geq \tau\big] \leq 2\Big[\exp\big(-\frac{\tau^2}{2m(c+b\alpha)^2}\big) + \frac{m}{\alpha}\varsigma\Big]$.*

In the following, let $P + p = P \cup \{p\}$ and $P - p = P \setminus \{p\}$.

Lemma 3.4. *The random variable f_r is strongly difference-bounded by*

$$(b, c, \varsigma) := (n - 1, O(\sqrt{n \log n}), 1/n^{O(1)}).$$

Proof: If one moves only one point of P, at most $n - 1$ pairwise distances involved with this point can change, implying that $b \leq n - 1$.

By the ε-sample theorem, and Lemma 3.1, a sample of size $O\big(\varepsilon^{-2} \log n\big)$ is an ε-sample for Toroidal balls, with high probability. Interpreting P as an ε-sample for $[0, 1]^d$, implies that this holds for P with $\varepsilon = \sqrt{\varphi} = \sqrt{\frac{c_d \ln n}{n}}$ for sufficiently small constant $c_d > 0$. Let $\mathfrak{v}_d = vol\big(\mathfrak{b}_T(q, r)\big)$, for any point $q \in [0, 1]^d$. The number of points in distance $\leq r$ from a point $p \in [0, 1]^d$, is $X_p = |P \cap \mathfrak{b}_T(p, r)|$. Hence, for *any* Toroidal ball we have

$$|X_p - \mathbb{E}[X_p]| = \big||P \cap \mathfrak{b}_T(p, r)| - \mathfrak{v}_d n\big| \leq \varepsilon n = \sqrt{c_d n \log n} = \widetilde{O}(\sqrt{n}) \tag{3.2}$$

assuming P is indeed an ε-sample. Note that the bound above crucially uses the Toroidal distance properties. Furthermore, the set $P - p$, formed by removing any point $p \in P$, is an ε-sample, and this holds with high probability for all such subsets.

This readily implies that for any two points $p, p' \in [0, 1]^d$, we have

$$\big|X_p - X_{p'}\big| \leq |X_p - \mathfrak{v}_d n| + \big|\mathfrak{v}_d n - X_{p'}\big| = O(\sqrt{n \log n}).$$

Picking a point $p \in P$, and a point $p' \in [0, 1]^d$, and setting $P' = P - p + p'$, we are interested in bounding the "typical" difference between $f_r(P)$ and $f_r(P')$ (this would be the value of c). We have

$$\big|f_r(P) - f_r(P')\big| \leq \big|X_p - X_{p'}\big| + O(1) = O(\sqrt{n \log n}).$$

This implies that $c = O(\sqrt{n \log n})$. This calculation fails, only if P fails to be an ε-sample, which happens with probability $\varsigma \leq 1/n^{O(1)}$. ∎

Theorem 3.5. *For constant c' sufficiently large, we have $\mathbb{P}[|f_r - \mathbb{E}[f_r]| > c' n \log n] \leq 1/n^{O(1)}$.*

Proof: This follows readily by plugging the parameters of Lemma 3.4 into Lemma 3.3. Note that in our case, $m = n, b = n - 1$, and $c = \sqrt{c_d n \log n}$. Choosing $\alpha = 1/n$ and $\varsigma \leq 1/dn^3$ (which can be ensured by making c_d sufficiently large), and $\tau = \Omega(n \log n)$, the result follows by straightforward calculations from Lemma 3.3. For example, plugging $\tau = 100\sqrt{c_d} n \log n$ into Lemma 3.3 yields (for sufficiently large c_d): $\mathbb{P}\Big[\big|f_r(P) - \mathbb{E}[f_r(P)]\big| > 100\sqrt{c_d} dn \log n\Big] \leq 2\big[\exp\big(-\frac{10000 c_d dn^2 \log(n)^2}{2dn(2\sqrt{c_d n \log n}+1)^2}\big) + \frac{dn^2}{dn^3}\big] \leq \frac{4}{n}$. ∎

4. Warm-up: Computing the convex hull in linear time

We present here an algorithm for computing the convex hull of P in $O(n)$ time. This will serve as a warm-up as the tools here will be used later on.

Algorithm. Given P, we build T, a quadtree of height $h = \lceil (\log_2 n)/d \rceil$ and insert the points P in $O(n)$ time to its leaves – this can readily be done by storing the points in the grid formed by the leafs using hashing (or just direct array indexing).

The algorithm computes the convex hull via a bottom-up traversal of the tree. It starts by computing the convex hull (potentially empty) for each leaf of the quadtree using any brute force algorithm. For a node v at level k, the algorithm takes the computed convex-hulls of its children, extracts all their vertices and stores it in a set S, and computes the combined convex-hull of S, using off-the-shelf algorithm [Cha93] in $O(|S| \log |S| + |S|^{\lfloor d/2 \rfloor})$ time.

Analysis. The algorithm correctness is immediate. We next prove an inferior upper bound on the expected complexity of the random convex-hull that holds with higher moments.

Lemma 4.1. *Let $|\mathcal{C}(P)|$ denote the number of vertices in the convex hull of P. For any integer $\kappa > 0$, we have $\mathbb{E}[|\mathcal{C}(P)|^\kappa] = O(\log^{O(\kappa d)} n)$ (the constant hidden by the O depends on both κ and d).*

Proof: Let p be a vertex of the convex-hull $\mathcal{C}(P)$, and consider a tangent (hyper)plane h to $\mathcal{C}(P)$ that passes through p. The plane h separates $\mathcal{C}(P)$ from one of the vertices of the $[0,1]^d$, say q. Let R be the axis parallel box with p and q as antipodal vertices.

The VC dimension of axis aligned boxes is $2d$; see [Har11a]. By the ε-net theorem, a sample of size $O(d\varepsilon^{-1} \log n)$ is an ε-net for axis aligned boxes, with probability $\geq 1 - 1/n^{O(d)}$. Setting $\varepsilon = \varphi = c_d(\log n)/n$, it follows that R contains a point of P with high probability. It follows that all the vertices of $\mathcal{C}(P)$ are in the vicinity of some vertex of $[0,1]^d$. Let Ξ be the union of the vicinities of the vertices of $[0,1]^d$. By Lemma 2.9, we have that $\alpha = vol(\Xi) = O((\log n)^d/n)$. Applying Lemma 2.8 to $|\Xi \cap P|^\kappa$ now implies the claim. ∎

Lemma 4.2. *The above algorithm computes $\mathcal{C}(P)$ is $O(n)$ expected time.*

Proof: Consider the root of the quadtree – it has 2^d children, and let P_i be the set of points of P stored in the ith child. Let $n_i = |P_i|$ and $m_i = |V(\mathcal{C}(P_i))|$. We have that $\sum_i n_i = n$, and $\mathbb{E}[n_i] = n/2^d$. In particular, using Chernoff's inequality we have that $n_i \leq (7/8)n$ with high probability. Similarly, we have that $\mathbb{E}[m_i^\kappa] = O(\log^{O(\kappa d)} n)$ by Lemma 4.1. Let $m = \sum_i m_i$, and observe that computing the convex-hull at the top most level takes $O(m \log m + m^{\lfloor d/2 \rfloor}) = O(m^d) = O(2^d \sum_i m_i^d)$. Thus, ignoring the construction time of the quadtree itself, we have the recurrence

$$T(n) = O\big(\mathbb{E}\big[\textstyle\sum_i m_i^d\big]\big) + \sum_i T(n_i) = \log^{O(d^2)} n + \sum_i T(n_i),$$

and the solution to this recurrence is $O(n)$. ∎

5. Complexity of the Delaunay triangulation of random points

Here we show that the Delaunay triangulation of P has linear complexity in expectation.

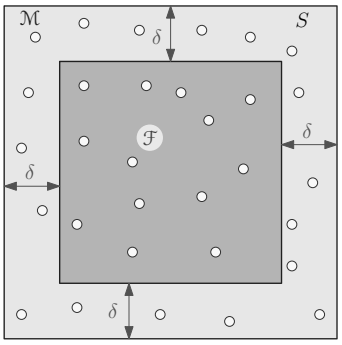

Figure 5.1: The ***inner fortress*** $\mathcal{F}$ in red. The ***moat*** $\mathcal{M}$ in light blue. Here $\delta = \sqrt[d]{\frac{c_d \ln n}{n}}$.

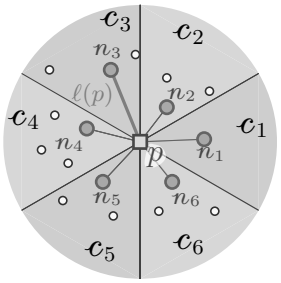

Figure 5.2: Definitions of $\boldsymbol{n}_i = \boldsymbol{n}_{\boldsymbol{c}_i}(p)$ and $\ell(p)$.

Background on Delaunay triangulations. A simplicial complex $\mathcal{D}$ over a set P is a set system with the (hyper) edges being subsets of P, such that for any $\sigma, \sigma' \in \mathcal{D}$, we have that $\sigma \cap \sigma' \in \mathcal{D}$. An edge of $\mathcal{D}$ is a ***simplex***. A simplex is k ***dimensional*** if the affine space its points span is k dimensional. Simplices of dimension $0, 1$ and 2 are ***vertices***, ***edges*** and (two dimensional) ***faces***, respectively.

For a point $p \in \mathbb{R}^d$, and a radius $r > 0$, let $\mathbb{b}(p, r)$ denote the open ***ball*** of radius r centered at p. For any points $p_1, \ldots, p_k \in \mathbb{R}^d$, let $\mathsf{pen}(p_1, \ldots, p_k)$ denote the ***pencil*** of $p_1, \ldots, p_k$: the set of all *open* balls $\mathbb{b}$ in $\mathbb{R}^d$, such that their boundary sphere passes through $p_1, \ldots, p_k$. If $k = d + 1$, and the points are in general position, the pencil is a single ball $\mathsf{circum}(p_1, \ldots, p_k)$ bounded by the ***circumscribed sphere*** of these points.

The ***Delaunay triangulation*** $\mathcal{D} = \mathcal{D}(P)$ of P is a simplicial complex, where $\nabla \in \mathcal{D} \iff$ there is a ball $\mathbb{b} \in \mathsf{pen}(\nabla)$ such that $\mathbb{b} \cap P = \emptyset$. The Delaunay triangulation has the property that if the set of points P is a random set then the points are in general position with probability 1 (i.e., ***almost surely***), and it is then uniquely defined.

5.1. A linear bound in the interior of the hypercube

Let $\mathcal{F} = [\delta, 1 - \delta]^d$ be the ***fortress*** of $[0, 1]^d$, and $\mathcal{M} = [0, 1]^d \setminus \mathcal{F}$ be its ***moat***, where $\delta = \sqrt[d]{\frac{c_d \ln n}{n}}$. See Eq. (2.1) and Figure 5.1.

Definition 5.1. Consider a ray emanating from a point q in a direction v in $\mathbb{R}^d$. A ***cone*** of angle α is the set of all points $p \in \mathbb{R}^d$, such that the angle between $p - q$ and v is at most α. The point q is the ***apex*** of the cone.

One can cover space around a point with $O_d(1)$ cones, with angle $\pi/12$, to cover all of $\mathbb{R}^d$

Lemma 5.2 ([DGL96]). *For any point $p \in \mathbb{R}^d$, one can construct a set $\mathscr{C}_p$ of $2^{O(d)}$ cones with apex p and angle $\pi/12$, such that $\cup \mathscr{C}_p = \mathbb{R}^d$.*

The following shows that all these cones are not empty, if the apex p is in the fortress.

Lemma 5.3. *For any $p \in \mathcal{F}$, and any cone $\boldsymbol{c} \in \mathscr{C}_p$, we have $\mathbb{P}[\boldsymbol{c} \cap (P - p) = \emptyset] < 1/n^{O(1)}$.*

Proof: Since $p \in \mathcal{F}$, it is at a distance of at least δ from the boundary of $[0,1]^d$. Thus, $\alpha = vol\big(\boldsymbol{c} \cap [0,1]^d\big) = \Omega(\delta^d) = \Omega(\varphi) = \Omega\big((\log n)/n\big)$, by Eq. (2.1). This implies that the probability that $\boldsymbol{c}$ does not contain any of the points of $P - p$ is at most

$$(1-\alpha)^{n-1} \leq \exp\big(-\alpha(n-1)\big) \leq \exp(-c \ln n) = \frac{1}{n^{O(1)}},$$

where c is a constant that can be made to be arbitrarily large by increasing the value of c_d. The result now follows by applying the union bound to all the cones in $\mathscr{C}_p$. ∎

Definition 5.4. For a point $p \in P \cap \mathcal{F}$ and a cone $\boldsymbol{c} \in \mathscr{C}_p$, let $\boldsymbol{n}_{\boldsymbol{c}}(p)$ denote the nearest neighbor to p in $\boldsymbol{c} \cap (P - p)$. The ***reach*** of p is $\ell(p) = \max_{\boldsymbol{c} \in \mathscr{C}_p} \|p\boldsymbol{n}_{\boldsymbol{c}}(p)\|$, see Figure 5.2.

The following bounds (in expectation) $\ell(p)$ and the distance between p and $\boldsymbol{n}_{\boldsymbol{c}}(p)$ for $\boldsymbol{c} \in \mathscr{C}_p$.

Lemma 5.5. *For any point $p \in P \cap \mathcal{F}$, a cone $\boldsymbol{c} \in \mathscr{C}(p)$ and $t \leq \delta$, we have:*
(I) $\mathbb{P}[\|p\boldsymbol{n}_{\boldsymbol{c}}(p)\| > t] \leq \exp(-c\, t^d n)$, *where c is constant that depends only on the dimension.*
(II) $\mathbb{P}[\ell(p) > t] \leq f(t,n) = \exp\big(O(d) - c\, t^d n\big)$,
(III) $\mathbb{E}[\ell(p)] = \Theta(1/\sqrt[d]{n})$, *and*
(IV) the reach of all the points of $P \cap \mathcal{F}$ is bounded by δ, with probability $1/n^{O(d)}$.

Proof: (I) If $\|p\boldsymbol{n}_{\boldsymbol{c}}(p)\| > t$ then the set $R = \boldsymbol{c} \cap \mathscr{b}(p,t)$ contains no points of P. The volume of $\boldsymbol{c} \cap \mathscr{b}(p,t)$, for $t \leq \delta$, is $\alpha = \Omega(t^d)$. In this case, all the points of $P - p$ must avoid R. This happens with probability at most $(1-\alpha)^{n-1} \leq \exp\big(-\alpha(n-1)\big) \leq \exp\big(-ct^d n\big)$.

(II) By the union bound, and Lemma 5.2, we have $\mathbb{P}[\ell(p) > t] \leq \sum_{\boldsymbol{c} \in \mathscr{C}(p)} \mathbb{P}[\boldsymbol{n}_{\boldsymbol{c}}(p) > t] \leq f(t,n)$.

(III) Observe that $\mathscr{b}(p, 1/2\sqrt[d]{n})$ contains no other points of $P - p$ with constant probability. This implies that $\mathbb{E}[\ell(p)] = \Omega(1/\sqrt[d]{n})$. The upper bound $\mathbb{E}[\ell(p)] = O(1/\sqrt[d]{n})$ follows by the above exponential decay, as a straightforward calculation shows.

(IV) Setting $t = \delta$, and using the union bound implies this part. ∎

Definition 5.6. For $p \in P \cap \mathcal{F}$, the ***influence*** of p is $\circledcirc_p = \{q \in P \mid \|pq\| \leq 2\ell(p)\}$

Figure 5.3: Sketch of proof of Lemma 5.7

Importantly, all the Delaunay edges adjacent to a point $p \in \mathcal{F}$ are contained in p's region of influence. Hence, locally, it is sufficient to only consider points in the influence when computing the Delaunay triangulation.

Lemma 5.7. *For any point $p \in P \cap \mathcal{F}$, If $pq \in \mathcal{D}$, then $q \in \circledcirc_p$.*

Proof: Consider the largest (open) ball $\mathbb{b}$ with p on its boundary, that does not contain any point of P in its interior, and let r be its radius and z be its center, see Figure 5.3. We claim that $r \leq \ell(p)$, which would imply that $\|pq\| \leq 2r \leq 2\ell(p)$. Assume that $r > \ell(P)$, and consider any cone $\mathbf{c} \in \mathcal{C}_p$, such that $z \in \mathbf{c}$. Let p' be the diametrical point on $\partial\mathbb{b}$ to p. Consider any point $u \in \mathbf{c} \setminus \mathbb{b}$. The distance pu is minimized if $u \in \partial\mathbb{b}$, but then $\angle pup'$ forms the right angle of a right triangle. Observe that $\angle upp' < 30^\circ$ since the cone angle is at most 30°. But then

$$\|pu\| = \|pp'\| \cos \angle p'pu = 2r \cos \angle p'pu > 2\ell(p) \cos 30^\circ = (2\sqrt{3}/2)\ell(p) > \ell(p).$$

However, $\mathbb{b} \cap (P - p) = \emptyset$ implies that the closest point in $(P-p) \cap \mathbf{c}$ to p has distance larger than $\ell(p)$, which contradicts the definition of $\ell(p)$. ∎

We next bound the moments of the size of the set of points inside the influence of a point.

Lemma 5.8. *For $p \in P \cap \mathcal{F}$, and any constant $\kappa \geq 1$, we have $\mathbb{E}\big[|\circledcirc_p|^\kappa\big] = O_\kappa(1)$, see Definition 5.6.*

Proof: Let $L = |\circledcirc_p|$. We break P into two roughly equal sets P_1 and P_2 (this is done before sampling the locations of the points). Let $\ell_i = \ell(p, P_i)$ be the reach of p in P_i for $i \in \{1, 2\}$. Arguing as above, as $p \in \mathcal{F}$, this quantity is well defined. Let U_i be the number of points of P_{3-i} in the ball $\mathbb{b}_i = \mathbb{b}(p, \ell_i)$. Clearly, $\mathbb{E}[L] \leq \mathbb{E}[U_1] + \mathbb{E}[U_2]$, as $\mathbb{b}_i$ contains more points of P_{3-i} than the ball defined by the reach of the whole set.

So, let ψ be the minimum value such that $f(\psi, n/2) \leq 1/2$, where f is the function defined in Lemma 5.5. It is easy to verify that $\psi = O(1/n^{1/d})$. Let $\mathbb{b}_0 = \mathbb{b}(p, \psi)$, and, for $i > 0$, let $\mathbf{r}_i = \mathbb{b}(p, i\psi) \setminus \mathbb{b}(p, (i-1)\psi)$. Observe that $vol(\mathbb{b}_0) = O(1/n)$, and $vol(\mathbf{r}_i) = O(i^d/n)$. By Lemma 2.8, we have that $N_0 = \mathbb{E}\big[|P_2 \cap \mathbb{b}_0|^\kappa\big] = O(1)$, and $N_i = \mathbb{E}[|P_2 \cap \mathbf{r}_i|^\kappa] = O(i^{d\kappa})$. We have that $\mathbb{E}\big[L^\kappa\big] = O(T)$, where

$$T = \sum_{i=0}^{\infty} N_i \, \mathbb{P}[\ell > i\psi] \leq \sum_{i=0} O_\kappa(i^{\kappa d} f(i\psi, n/2)) = \sum_{i=0} O_\kappa(i^{\kappa d}/2^i) = O_\kappa(1),$$

since $f(i\psi, n/2) = \exp\big(O(d) - c\, i^d \psi^d n\big) \leq \big(\exp\big[O(d) - c\,\psi^d n\big]\big)^{i^d} \leq 1/2^{i^d}$. ∎

Combining everything, we get the main result for points in the fortress.

Lemma 5.9. *Let $\mathcal{D}$ be the Delaunay triangulation of P. The expected number of simplices that include any point of $P \cap \mathcal{F}$ is $O(n)$.*

Proof: Let $\tau = |\circledcirc_p|$. All the vertices of a simplex of the Delaunay triangulation containing p must have all its vertices in $\circledcirc_p$ by Lemma 5.7. Thus, the number of such simplices, of all dimensions, is bounded by $\sum_{i=0}^{d+1} \binom{\tau}{i} = O(\tau^d)$. By Lemma 5.8, we have $\mathbb{E}\big[O(\tau^d)\big] = O(1)$. ∎

5.2. The complexity of the Delaunay triangulation near the boundary

We are now left with the tedious technicality of handling points that are too "close" to the boundary[5]. The idea is to use a similar argumentation to the above, but to replace the influence ball induced by the reach by a different region. This inflated region contains significantly more points, but since the number of points in the moat is small, this would still be linear overall. We remind the reader that the *moat* is the area $\mathcal{M} = [0,1]^d \setminus [\delta, 1-\delta]^d$, see Figure 5.1 and Eq. (2.1).

The following is an immediate consequence of Lemma 2.8 and Lemma 2.9.

[5]Ha, the boundary! A source of unmitigated delight to the authors, and hopefully also to the readers.

Lemma 5.10. *For any point $p \in P$, we have $X = |\mathsf{vic}(p) \cap P| = O(\log^d n)$ with probability $\geq 1 - 1/n^{O(1)}$.*

Lemma 5.11. *Consider an axis parallel box $B = \prod_{i=1}^{d} [p_i, q_i]$, and assume that there is a ball $\mathbb{b}$ that contains the points $p = (p_1, \ldots, p_d)$ and $q = (q_1, \ldots, q_d)$. Then $\mathbb{b}$ contains a d-dimensional simplex ∇ defined by $d+1$ vertices of B, such that the volume of this simplex is $\geq vol(B)/d!$. More generally, this holds for any ball that contains two diametrical vertices of B.*

Proof: The proof is by induction on d. The claim is immediate if $d = 1$.

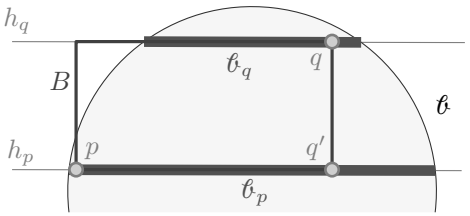

Figure 5.4

For $d > 1$, the idea it to provide a path along the edges of the box B between the two vertices that is contained in $\mathbb{b}$ – the convex-hull of this path would provide the desired simplex. So, consider the two hyperplanes $h_p \equiv x_d = p_d$ and $h_q \equiv x_d = q_d$, see Figure 5.4. Consider the two balls $\mathbb{b}_p = \mathbb{b} \cap h_p$ and $\mathbb{b}_q = \mathbb{b} \cap h_q$. Both balls have the same center if we ignore the dth coordinate, and one of them must have a bigger (or equal) radius to the other. Assume that $\mathbb{b}_p$ has the bigger radius, and observe that it as such must contain the point $q' = (q_1, \ldots, q_{d-1}, p_d)$. This implies that the segment $qq' \subseteq \mathbb{b}$. By induction there is a path on the edges of $B \cap h_p$ between p and q', which implies that there is a path on the edges of B between p and q that lies inside $\mathbb{b}$. ∎

For points $p \in \mathcal{M}$, as the following testifies, one needs to consider only simplices and points that are in $\mathsf{vic}(p)$. This is indeed a larger region than the influence region used before, but is small enough for our purposes.

Lemma 5.12. *Let p be any point in P. With high probability, there are at most $O(\log^{d^2} n)$ simplices in $\mathcal{D} = \mathcal{DT}(P)$ that contains p as a vertex. Furthermore, all the points neighboring p in $\mathcal{D}$ must be in $\mathsf{vic}(p)$, with probability $\geq 1 - 1/n^{O(d)}$.*

Proof: The VC dimension of simplices in $\mathbb{R}^d$ is $O(d^2 \log d)$ as it is the intersection of $d+1$ halfspaces, each of VC dimension $d+1$, see [Har11a]. By the ε-net theorem, a sample of size $O\big((d^2 \log d)\varepsilon^{-1} \log n\big)$ is an ε-net for simplices, with high probability. Interpreting P as an ε-sample for $[0,1]^d$, implies that this holds for P with $\varepsilon = \varphi/d!$, where $\varphi = (c_d \ln n)/n$, see Eq. (2.1) (by making c_d sufficiently large).

Consider a point $q \in P$, such that $q \notin \mathsf{vic}(p)$, and assume that $pq \in \mathcal{DT}(P)$. This implies that there is a close ball $\mathbb{b}$ that has p and q on its boundary, and no other points of P in its interior. By Lemma 5.11, there is an (open) simplex ∇ of volume $\geq vol(\mathsf{R}(p,q))/d! \geq \varepsilon$ that contains p and q on its boundary, and it is contained inside $[0,1]^d \cap \mathbb{b}$. But since P is an ε-net for simplices, it follows that there is a point of P in ∇, which is a contradiction.

We conclude that all the edges adjacent to p in $\mathcal{D}$ must be to points in $\mathsf{vic}(p)$. But there are at most $t = O(\log^d n)$ such points, by Lemma 5.10. Since any simplex involving p in $\mathcal{D}$ must use only points that are in the vicinity, it follows that the number of simplices (of all dimensions) adjacent to p in $\mathcal{D}$ is bounded by $\sum_{i=0}^{d} \binom{t}{i} = O(t^d)$. ∎

Finally, we show that the complexity of the Delaunay triangulation in the moat is sublinear.

Lemma 5.13. *Let $\mathcal{D}$ be the Delaunay triangulation of P. The expected number of simplices in $\mathcal{D}$ that include any point of $P \cap \mathcal{M}$ is $o(n)$.*

Proof: We have $\alpha = vol(\mathcal{M}) \leq 2d\delta = O(\sqrt[d]{(\log n)/n})$, see Eq. (2.1). Thus, the expected number of points of P in $\mathcal{M}$ is $\alpha n = O(n^{1-1/d} \log n)$. (As usual, this bound holds with high probability.) By Lemma 5.12, the total number of simplices in the Delaunay triangulation of P involving points in the moat is bounded by $O(\alpha n \log^{O(d^2)} n) = o(n)$, with high probability. ∎

The result. Combining Lemma 5.9 and Lemma 5.13 implies the following.

Theorem 5.14. *For fixed d, the complexity of the Delaunay triangulation of a set of n random points picked uniformly and independently in $[0,1]^d$ is $O(n)$ in expectation.*

6. Constructing the Delaunay triangulation in linear time

6.1. Algorithm

We established above that the (expected) complexity of the Delaunay triangulation is linear by giving a (linear sized) superset of vertices/simplices that are a superset of the features of $\mathcal{D}$. We are now left with the task of extracting the features that do appear in $\mathcal{D}$. Recall, that the input is a set P of n random points from $[0,1]^d$.

I: Computing the Delaunay simplices attached to points in $P \cap \mathcal{F}$. Let $N = \lceil n^{1/d} \rceil$. The algorithm throws the points of P into a $N \times \cdots \times N$ uniform grid covering $[0,1]^d$. This can be done in linear time using hashing, where one can retrieve a list of all the points stored in a grid cell in constant time. Here a grid cell is uniquely represented by an integer tuple from $\{0, 1 \ldots, N-1\}^d$. Formally, we map a point $p = (p_1, \ldots, p_d) \in [0,1)^d$, to the grid cell with ***id*** $\mathsf{id}(p) = (\lfloor p_1 N \rfloor, \ldots, \lfloor p_d N \rfloor)$; see [Har11a].

For a point $p \in P \cap \mathcal{F}$, the algorithm computes the reach $\ell(p)$ by performing a marching cubes algorithm computing the intersection of the grid with the ball $\mathscr{b}(p, r_i)$, where initially $r_i = 2^i/N$, for $i = 0, 1, \ldots$. The algorithm uses scanning to compute the point set $Q_i = \mathscr{b}(p, r_i) \cap P$ by extracting all the points stored in the intersecting grid cells. The algorithm stops in the ith iteration, if all cones in $\mathscr{C}_p$ contains at least one point of P. At this point one can compute the reach of p by computing for each cone $\mathscr{c} \in \mathscr{C}_p$ the closest point in $\mathscr{c} \cap P_i$ to p. The algorithm then computes the point set $P_p = P \cap \mathscr{b}(p, 2r_i)$, and computes the Delaunay triangulation of P_p using any standard algorithm for computing Delaunay triangulation. Finally, the algorithm extract the star of p from the computed triangulation, and store it. As a reminder, the ***star*** of p, denoted by $\star_p$, is the set of all the simplices in the triangulation that contains p. The algorithm repeats this process for all the points of $P \cap \mathcal{F}$, and returns the union of all the stars computed.

II: Computing the Delaunay simplices attached to points in $P \cap \mathcal{M}$. The algorithm builds an orthogonal range searching data structure on the points $P \cap \mathcal{M}$ (and not on all the whole point set P). Next, for each $p \in P \cap \mathcal{M}$, the algorithm constructs the set of $O(\log^{d-1} n)$ canonical boxes $\mathcal{B}_p$ (as defined in the proof of Lemma 2.9) that their union covers $\mathsf{vic}(p)$. Then for each $\mathsf{r} \in \mathcal{B}_p$, it queries the data structure for points set $P_{\mathsf{r}} = \mathsf{r} \cap \mathcal{M} \cap P$. Next, it loops over $q \in P_{\mathsf{r}}$ and adds points in $P_{\mathsf{r}} \cap \mathsf{vic}(p)$ to the computed set $N_{\mathcal{M}}(p) = P \cap \mathcal{M} \cap \mathsf{vic}(p)$. Next, using the above grid, it computes the set $N_{\mathcal{F}}(p) = P \cap \mathscr{b}(p, 2\delta)$. Finally, the algorithm computes the Delaunay triangulation of $P_p = N_{\mathcal{M}}(p) \cup N_{\mathcal{F}}(p)$ using a standard algorithm and extracts the star $\star_p$ of p, from the computed

triangulation, and stores it. The algorithm repeats this for each $p \in P \cap \mathcal{M}$ and returns the union of all stars computed for all p.

6.2. Analysis

In the following, we prove that the output of the algorithm is correct with probability $\geq 1 - 1/n^{O(d)}$ and the expected running time is $O(n)$.

Part I: The fortress. The correctness of the algorithm is implied by the following claim.

Lemma 6.1. *For $p \in P \cap \mathcal{F}$, we have that $\nabla \in \curlywedge_p$ if and only if $\nabla \in \mathcal{DT}(P)$.*

Proof: If $\nabla \in \mathcal{DT}(P)$, then by Lemma 5.7, $\nabla \subseteq \circledcirc_p$. This implies that $\nabla \in \mathcal{DT}(\circledcirc_p)$, which implies that ∇ is in the computed set $\curlywedge_p$.

If $\nabla \in \curlywedge_p$, then the circumball of ∇ does not contain any point of $\circledcirc_p$ in its interior. If this ball contained any point of P in its interior, then it must be further than $2\ell(p)$, but this is not possible by the argument used in the proof of Lemma 5.7. ∎

Lemma 6.2. *The above algorithm runs in expected $O(n)$ time.*

Proof: The Delaunay triangulation of n points in $\mathbb{R}^d$, can be computed in $O(n^{\lceil d/2 \rceil} + n \log n) = O(n^d)$ [Mul94]. As such, we have that the expected running time is $\mathbb{E}\Big[\sum_{p \in P \cap \mathcal{F}} O(|\circledcirc_p|^d)\Big] = O(n)$, by Lemma 5.8. ∎

II: The moat. Let $\mathcal{DT}(P)_{\mathcal{M}}$ denote the set of simplices in $\mathcal{DT}(P)$ with some vertices in $\mathcal{M}$. The correctness of the algorithm is implied by the following.

Lemma 6.3. *For all $p \in P \cap \mathcal{M}$, we have $\nabla \in \curlywedge_p$ if and only if $\nabla \in \mathcal{DT}(P)_{\mathcal{M}}$ with probability $\geq 1 - 1/n^{O(d)}$.*

Proof: Consider a simple $\nabla \in \mathcal{DT}(P)_{\mathcal{M}}$ with $p \in V(\nabla)$. If ∇ contains a point $q \in V(\nabla)$ that is in the fortress $\mathcal{F}$, and is outside $\mathbb{b}(p, 2\delta)$, then $\ell(q) > \delta$, and Lemma 5.5 implies that this happens with probability $< 1/n^{O(d)}$. Thus, we have that $V(\nabla) \subseteq P \cap \big((\mathcal{M} \cap \mathsf{vic}(p)) \cup \mathbb{b}(p, 2\delta)\big) \subseteq P_p$. Thus, the empty ball $\mathbb{b}$ in $\mathcal{DT}(P)_{\mathcal{M}}$ that circumscribes ∇ is still empty in P_p, $V(\nabla) \subseteq P_p$, and thus $\nabla \in \curlywedge_p$.

If $\nabla \in \curlywedge_p$, then there is an empty ball $\mathbb{b}$ that circumscribes ∇ and is a witness to this. Assume for the sake of contradiction that $\mathbb{b}$ is not empty, and let q be the closest point to p in $\mathbb{b} \cap (P \setminus P_p)$. If $q \in \mathcal{M}$, then $q \notin \mathsf{vic}(p)$ (as $P \cap \mathcal{M} \cap \mathsf{vic}(p) \subseteq P_p$). The probability for that this happens is $< 1/n^{O(d)}$ by Lemma 5.12. If $q \in \mathcal{F}$, then the cone $\mathbb{c} \in \mathcal{C}_q$ that contains p, can not contain any closer point to q (than p) from P. Namely, the reach of q is bigger than 2δ, and probability for that is $\leq 1/n^{O(d)}$, by Lemma 5.5. ∎

Lemma 6.4. *The above algorithm runs in expected $O(n)$ time.*

Proof: We have $n_{\mathcal{M}} = \mathbb{E}[|P \cap \mathcal{M}|] \leq n \cdot d \cdot \delta = O(n^{1-1/d} \log n)$. Building the orthogonal range searching data-structure of $P \cap \mathcal{M}$ takes $O(n + n_{\mathcal{M}} \log^d n) = O(n)$.

For any $p \in P \cap \mathcal{M}$, computing $\mathbb{b}(p, 2\delta) \cap P$ (using the grid) takes $O(\log n)$ time (and this bound holds with high probability). Computing the points in the vicinity of p in the moat takes $O(\log^{O(d)} n)$ time – indeed each orthogonal range query takes $O(\log^d n)$ time, and there are $O(\log^d n)$ such queries. Finally, the time to compute the Delaunay triangulations P_p, is $O(\log^{O(d^2)} n)$.

Putting everything together, we have that the expected running time of the second part of the algorithm is $O(n + n_{\mathcal{M}} \log^{O(d^2)} n) = O(n)$. ∎

Theorem 6.5. *For fixed d, and a uniformly and independently sampled point set $P \subseteq [0,1]^d$ of size n, the above algorithm computes the Delaunay triangulation $\mathcal{DT}(P)$ of P in expected $O(n)$ time. The algorithm succeeds with high probability.*

7. Constructing the MST in linear time

7.1. Preliminaries

Lemma 7.1. *Let $\mathcal{T}$ be the MST of P. The longest edge in $\mathcal{T}$ has length $\leq \delta = \sqrt[d]{c_d(\log n)/n}$ (see Eq. (2.1)), with probability $\geq 1 - 1/n^{O(d)}$, where c_d is a sufficiently large constant.*

Proof: Let pq be the longest edge in $\mathcal{T}$. Observe that diametrical ball $\mathbb{b}$ defined by p and q can not contain any points of P in its interior, as such a point z, would induce a cycle pzq with pq being the longest edge, which implies that it is not the MST. The volume of $\mathbb{b}$ is minimized if its center lies in one the corners of $[0,1]^d$. We conclude that the region $R = \mathbb{b} \cap [0,1]^d$ has $vol(R) = \Omega_d(\|pq\|^d/2^d) = \Omega_d(\|pq\|^d)$. Furthermore, R is formed by the intersection of a hyperbox with ball, and the VC dimension of such ranges is $O(d)$ [Har11a]. The point set P can be interpreted as an ε-net for such ranges, with $\varepsilon = \delta^d/2^d = \Omega_d\big((\log n)/n\big)$, with high probability. We conclude that if $\|pq\| \geq \delta$, then P fails as an ε-net, which implies the claim. ∎

Definition 7.2. The ***Yao graph*** $G_\angle = G_\angle(P)$ [Yao82] of P formed by connecting two points $p, q \in P$ by an edge if q is the nearest point to p in one of the cones of $\mathcal{C}(p)$ (see Lemma 5.2). Let $G_{\angle,\delta}(P)$ be the graph $G_\angle$ after removing from it all the edges with length $\geq \delta$.

It is well known that this graph contains the MST of P [Yao82].

Lemma 7.3. *Let P be a set of n points picked uniformly at random from $[0,1]^d$. One can compute the graph $G_{\angle,\delta}(P)$ in $O(n)$ expected time.*

Proof: We store the points of P in a uniform grid with roughly $\Theta(n)$ cells in $[0,1]^d$. For every point $p \in P$, and every cone $\mathbb{c} \in \mathcal{C}_p$, we perform a marching cube algorithm to compute the closest point to p in $\mathbb{c} \cap P$. If the search distance exceeds δ, we abort the search.

For a point p in the fortress $\mathcal{F}$, computing the edges around p takes $O(1)$ time in expectation, by Lemma 5.8. For points in the moat, their number is $O(n^{1-1/d}\log n)$, with high probability, and the search for each point is truncated after the distance exceeds δ. Per point, such a search takes $O(\log n)$ time. It follows that the overall expected running time is $O(n + n^{1-1/d}\log^2 n) = O(n)$. ∎

A refresher on Borůvka's algorithm. Let $G = (V, E)$ be an undirected graph with n vertices and $m \geq n$ edges, and weights on the edges. Borůvka's algorithm creates an empty forest F_0 over the vertices. Let C_{i-1} be the set of connected components of F_{i-1}. For $p \in P$, let $\sigma_{i-1}(v) \in \mathsf{C}_{i-1}$ denote the connected component of v in F_{i-1}. While $|\mathsf{C}_{i-1}| \geq 2$, for each connected component $C \in \mathsf{C}_{i-1}$, the algorithm adds the cheapest edge leaving $V(C)$ to some other connected component of F_{i-1}. Let F_i be the resulting forest from F_i after adding these edges. The final forest is the desired MST.

Each rounds takes time $O(m)$, and for any i we have $|\mathsf{C}_i| \leq |\mathsf{C}_{i-1}|/2$. Thus, Borůvka's algorithm takes $O(m \log n)$ time.

7.2. An $O(n \log n)$ time algorithm

The underling graph in our case is $G(P) = \big(P, \{uv \mid u, v, \in P\}\big)$ where the weight of each edge is the distance between its endpoints. A naive implementation of Borůvka on $G(P)$ would require roughly quadratic time.

Lemma 7.4. *For a set P of n random points in $[0,1]^d$, one can compute, in $O(n \log n)$ time, the euclidean minimum spanning tree of $G(P)$.*

Proof: One can compute the graph $G_{\angle,\delta}(P)$, see Definition 7.2, in $O(n)$ expected time, using Lemma 7.3. By Lemma 7.1, this graph contains the MST, which can be computed in $O(n \log n)$ time using Borůvka's algorithm. ∎

We did some experiments on Borůvka's algorithm, depicted in Figure 7.1.

7.3. Adapting Borůvka to divide and conquer

The algorithm precomputes the graph $H = G_{\angle,\delta}(P)$. Next, we turn Borůvka into a geometric divide and conquer algorithm. To this end, let $\mathsf{C} \subseteq [0,1]^d$ be some axis-parallel cube, and consider computing the MST of $P \cap \mathsf{C}$. Without any outside information, the output can only be a forest that is part of the final MST, and a set of candidate edges that might participate in the final MST. To this end, the algorithm splits C into $\nu = 2^d$ identical subcells $\mathsf{C}_1, \ldots, \mathsf{C}_\nu$.

The algorithm recursively computes the MST of $P_i = P \cap \mathsf{C}_i$, for all i. Specifically, the edges of the MST are edges of H, and as such, all the edges of the MST with exactly one endpoint in P_i are in the cut $\Gamma(P_i) = \{uv \in E(H) \mid u \in P_i, v \in P \setminus P_i\}$. Intuitively, the size of this cut is quite small (roughly) $O(n^{1-1/d})$, and we can identify the vertices in P_i adjacent to such edges. These vertices are ***portals***, the set of all portals in P_i is denoted by $\partial(P_i)$.

Borůvka's algorithm with portals. Imagine running Borůvka only on the points of P_i. In every round, each connected component (in the current spanning forest) chooses the shortest edge in the cut it defines, and add it to the constructed forest. The catch is that if a connected component contains a portal point, then it might be part of a larger tree (in the larger forest) that is outside P_i. As such, this cut is no longer well defined (as it involves vertices and edges outside P_i). Thus, a connected component that contains a portal is ***frozen*** – it can no longer choose edges to add to the spanning tree. During a Borůvka round, all the components that are active (i.e., not frozen), each chooses the shortest edge in the cut they induce – note, that an active component might choose an edge connected to a frozen component. Thus, a frozen component might grow by active components attaching themselves to it. The algorithm continue doing rounds till all components are frozen.

A natural implementation of Borůvka is via collapsing each tree in the forest being constructed into a single node, and among parallel edges with the same endpoints, preserving the cheapest edge of the bunch. Thus, the execution on the modified Borůvka on P_i results in an induced graph G_i over $\partial(P_i)$ – where the surviving edges are potential edges for use by the MST later on.

Pruning. The number of edges of G_i is potentially too large. The algorithm computes the MST of G_i (treating it as its own graph, ignoring portals) running the standard Borůvka algorithm on G_i. The algorithm deletes from G_i all the edges that do not appear in the computed MST.

To recap – every vertex of G_i is a collapsed tree forming part of the final MST. All the edges of G_i are candidate edges that might appear in the final MST– all these edges form a spanning tree of G_i. See Figure 7.2 and Figure 7.3 for a toy dry run on the Borůvka step and the pruning step.

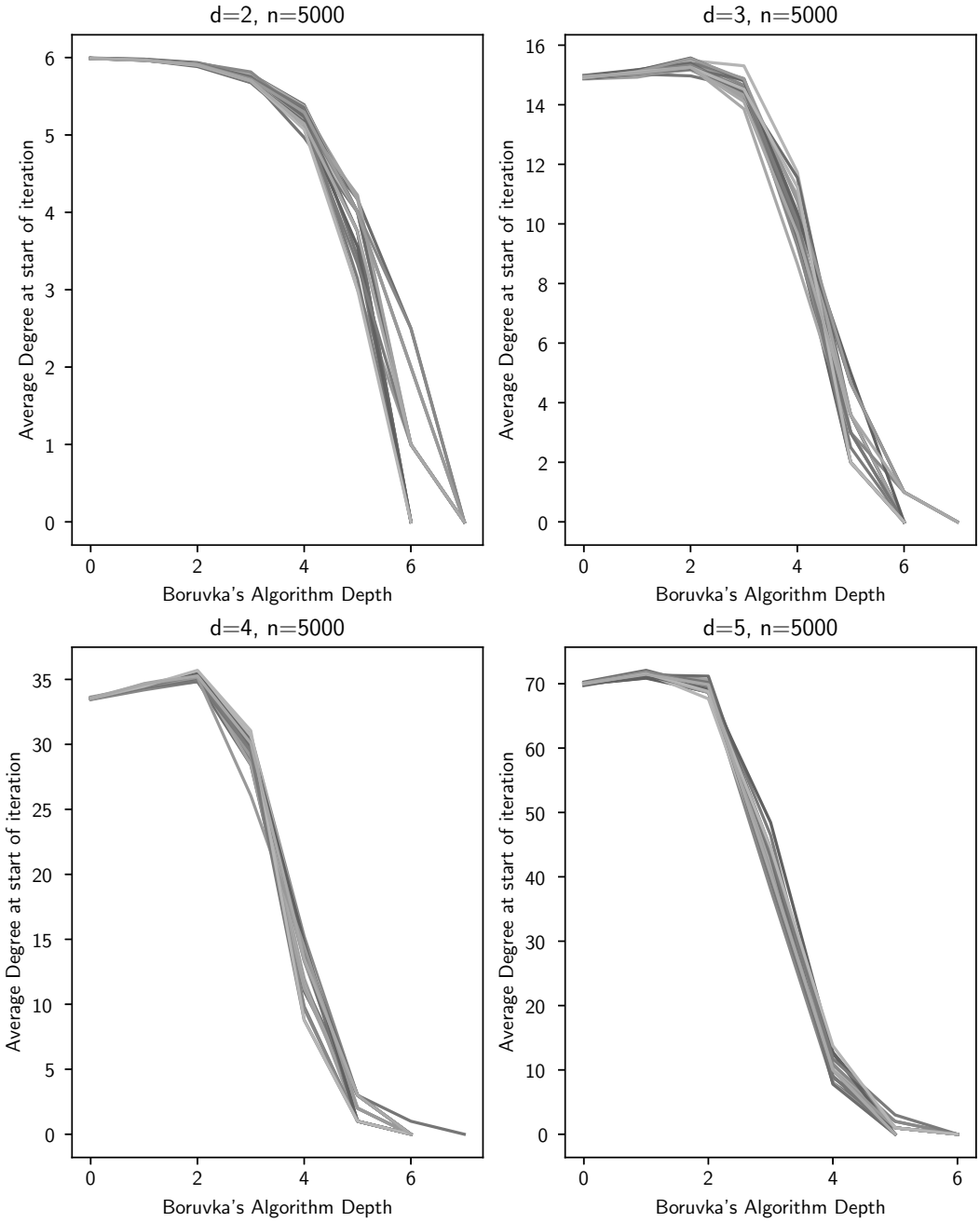

Figure 7.1: We randomly sample 20 samples $P_i, 1 \leq i \leq 20$ where $P_i \subseteq [0,1]^d$ and $|P_i| = 5000$. We run Borůvka algorithm with $\mathcal{DT}(P_i)$ as input for $d \in \{2,3,4,5\}$. In each iteration of Borůvka algorithm, we record the current average degree of the components, and plot the average degree progression for the 20 different samples.

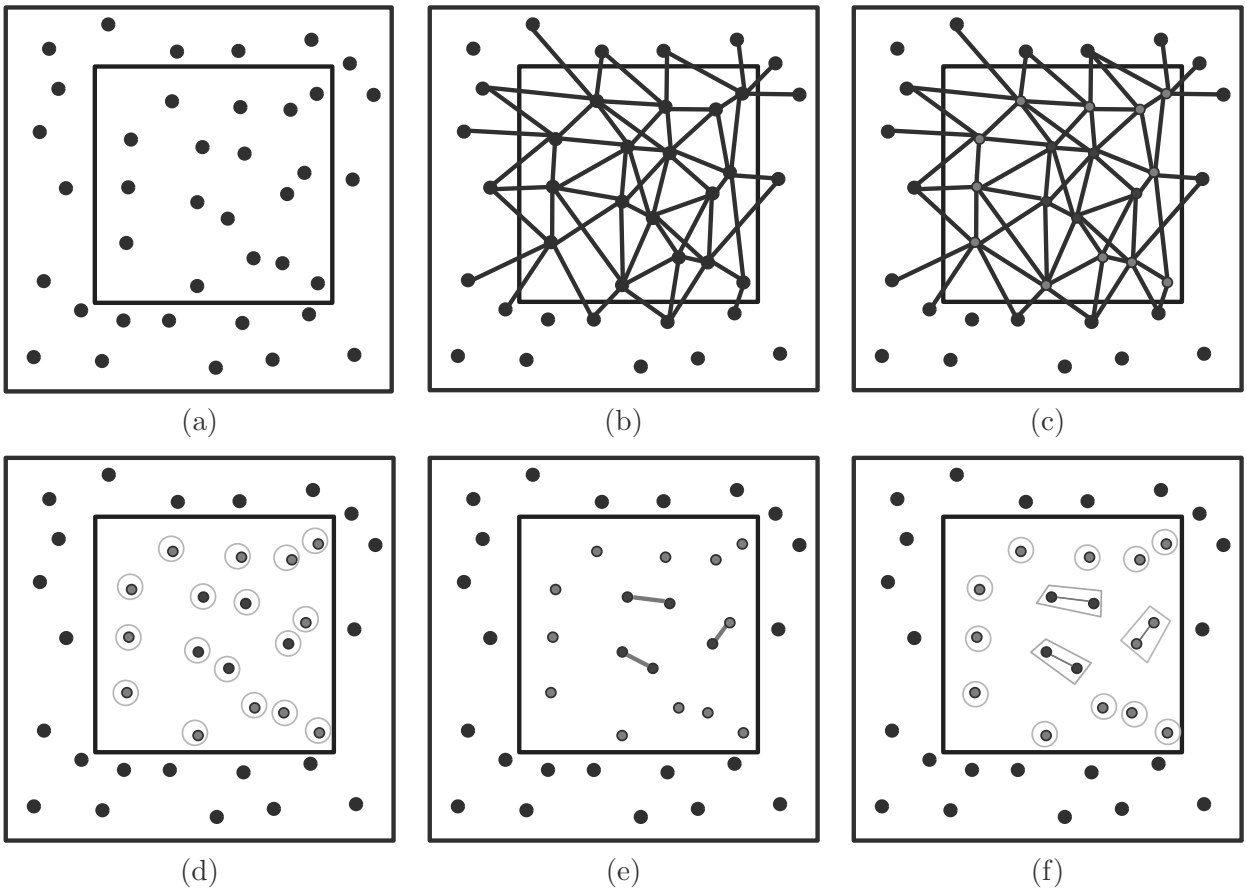

Figure 7.2: (a) shows the axis parallel cube (in blue) and the points inside that we restrict ourselves to. (*b*) shows some of the edges of $G_{\angle,\delta}(P)$ inside the cube. (*c*) shows the ***portal*** vertices in red, and all other points in blue. (*d*) shows the connected components initially for Borůvka algorithm. (*e*) shows the edges in cyan that were added by restricted Borůvka to EMST of P in the first round. (f) Shows the new components after round one of Borůvka algorithm (note the previous blue vertex is now red because it joined a component with a portal). See Figure 7.3 for the rest.

The conquer stage. The algorithm recursively computes the (collapsed) graphs $G_1, \ldots, G_\nu$, for $i = 1, \ldots, \nu$. Next, the algorithm computes the set of portals $\partial = \partial(\mathsf{C} \cap P)$, which is contained in $\cup_i \partial(P_i)$. Let $E_1 = \bigcup_{i<j}(P_i, P_j)$ be the set of all possible edges between the subproblems. Let $E_2 = E_1 \cap E(H)$. Next, the algorithm computes the graph $G_{\mathsf{C}} = \cup_i G_i \cup E_2$. The algorithm runs the modified Borůvka with portals, described above, on the graph G_{C}, with ∂ being the set of portals (thus, all the vertices comping from the children are portals in their own subproblem, but some of them lose their portal status as they migrate to the parent subproblem).

The overall algorithm. We apply the above algorithm to $[0,1]^d$ and P. Note that the root has no portals, so the output is a single tree which is the MST.

Some low level implementation details. We throw the points into a uniform grid over $[0,1]^d$, with each cell having volume $\Theta(1/n)$. We construct the quadtree over this grid in the natural way. We register each edge of H with the lowest node of the quadtree that contains both endpoints. This can

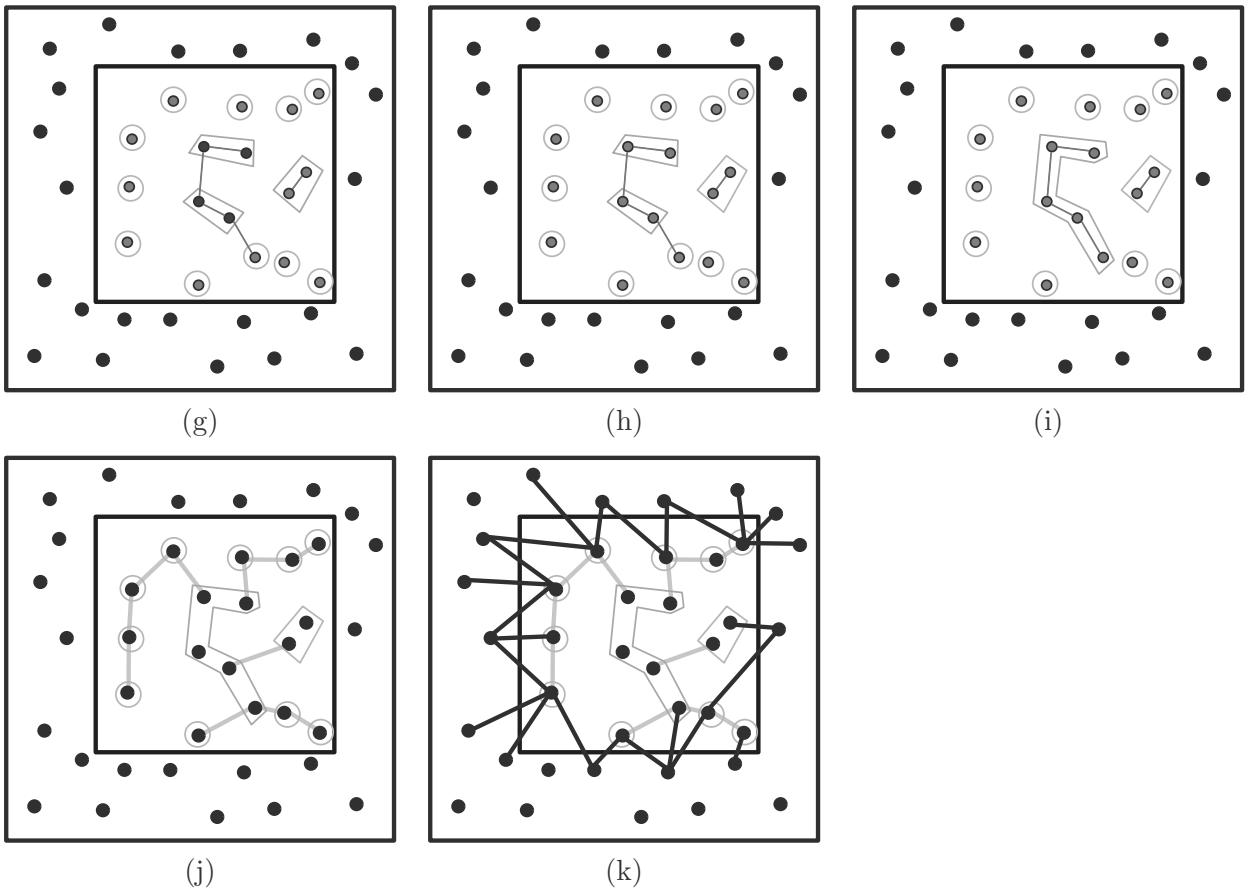

Figure 7.3: (g) again shows the new edges added to EMST in the second round of Borůvka. (i) shows the final connected components since all components have a portal. (j) shows the edges of the minimum spanning tree of the components which might be in the EMST of P. (k) shows the final graph returned by the restricted Borůvka algorithm.

be done in $O(1)$ time per edge using a data-structure for LCA queries in $O(1)$ time. Now, scanning the edges, each vertex can compute the level in the quadtree where it stops being a portal. The LCA operation can be replaced by computing the level of the grid that contains a segment – using the floor operation and bit operations, this can be done in $O(1)$ time, see [Har11a]. The rest of the algorithm implementation is as described above.

7.4. Analysis of the new MST algorithm

Clearly, edges that are added to an active component are edges that are minimal in their respective cuts, and thus must appear in the final MST. The more mysterious step in the pruning stage – let pq be an edge that was deleted by the pruning stage from G_i. Observe that there is a path π between p and q in the graph of G_i using edges that are shorter than pq. Namely, pq is the longest edge in a cycle, and can not appear in the final MST.

7.4.1. Running time analysis

Lemma 7.5. *Let C be a quadtree cell of depth i. Then, the number of portals in $\mathsf{C} \cap P$ is bounded by $O((n/2^{id})^{1-1/d} \log^2 n)$, with high probability. This also bounds the total number of edges in H adjacent to these vertices.*

Proof: A point of $p \in \mathsf{C}$ that is in distance larger than δ from the boundary of C can not be a portal, since H does not contain such long edges. The volume of the moat $\mathcal{M}_{\mathsf{C}}$ containing such points is bounded by the surface area of C multiplied by δ. That is $\alpha = (2d \cdot /2^{id})\delta$. Each such moat point has with high probability $O(\log n)$ edges in H. It follows that the expected number of portal edges is $O(\alpha n) = O((n/2^{id})^{1-1/d} \log^2 n)$, as long as $\alpha > (\log n)/n$, by Lemma 2.8, ∎

Lemma 7.6. *The above algorithm runs in $O(n)$ expected time.*

Proof: Let $\nu = 2^d$, and $P_1, \ldots, P_\nu$ be the points sent to the children of the root of the quadtree. Let $n'_i = |\partial(P_i)|$, for all i. By Lemma 7.5, $n_i = O(n^{1-1/d} \log^2 n)$ with high probability, and this also bounds the number of edges these portals have. Note, that each G_i has exactly $n'_i - 1$ edges. Thus, the graph created in the root has $\sum_i n_i$ vertices, and $O(2^d n^{1-1/d} \log^2 n)$ edges. Running Borůvka algorithm on this graph takes $O(n^{1-1/d} \log^3 n)$ time. We thus get the recurrence

$$T(n) = O(n^{1-1/d} \log^3 n) + \sum_i T(n_i).$$

It is easy to verify that the solution to this recurrence is $O(n)$, as $\sum_i n_i = n$ and $n_i < n/2$ with high probability. (To convince yourself of this, consider the over-simplified recurrence $S(n) = O(n^{1-1/d}) + 2^d S(n/2^d)$.) ∎

Remark 7.7. Note that the linear time MST algorithm can also be extended to a linear time MST algorithm for graphs with small separators. In that case, the portals are the separator vertices in the separator hierarchy, and we run the restricted Borůvka bottom up on the separator decomposition tree.

7.5. The result

The details of the following results are described in Section 7.

Theorem 7.8. *For fixed constant d, the MST of n uniformly and independently sampled points from $[0,1]^d$ can be computed, by the above algorithm, in $O(n)$ expected time.*

8. Simple distance selection in $O(n^{4/3} \log^{2/3} n)$ time in $d = 2$

The task. The input is a set P of n points picked randomly in $[0,1]^2$. For two sets X, Y, let

$$X * Y = \big\{ \{x, y\} \;\big|\; x \in X, y \in Y, x \neq y \big\}$$

be the set of all unordered pairs in $X \times Y$. Let $\Pi = P * P$, and for a fixed radius λ, let $\Pi_{\leq r} = \{\{p, q\} \in \Pi \mid \|pq\| \leq r\}$ be the number of all pairs in P that are in distance at most r from each other. The task at hand is to compute $|\Pi_{\leq r}|$.

Basic idea and some tools. Let K be a uniform $\mathsf{N} \times \mathsf{N}$ grid K, where $\mathsf{N} = \lceil (n/\log n)^{1/3} \rceil$. Let $P_{i,j}$ denote the points of P that fall in the grid cell $\mathsf{C}_{i,j} = [i/\mathsf{N}, (i+1)/\mathsf{N}] \times [j/\mathsf{N}, (j+1)/\mathsf{N}]$. Let $\Delta = \operatorname{diam}(\mathsf{C}_{i,j}) = \sqrt{2}/\mathsf{N}$ be the diameter of a grid cells. We assume here that $r > 8\Delta$. The case for $r \le 8\Delta$ can be handled simply by bruteforce search of a fine grid.

Let $\xi_{i,j} = |\{pq \in \Pi_{\le r} \mid p \in P_{i,j}\}|$. Observe that $|\Pi_{\le r}| = \sum_{i,j} |\xi_{i,j}|/2$. Thus, we restrict our attention to computing the values of $\xi_{i,j}$, for all i, j. For a grid cell $\mathsf{C} \in \mathsf{K}$, consider the sets

$$\mathscr{b}_{\mathsf{K}}(\mathsf{C}) = \left\{ \mathsf{D} \in \mathsf{K} \;\middle|\; \mathsf{D} \subseteq \mathscr{b}(\overline{\mathsf{c}}(\mathsf{C}), r - 2\Delta) \right\} \quad \text{and} \quad \mathcal{B}_{\mathsf{K}}(\mathsf{C}) = \left\{ \mathsf{D} \in \mathsf{K} \;\middle|\; \mathsf{D} \cap \mathscr{b}(\overline{\mathsf{c}}(\mathsf{C}), r + 2\Delta) \neq \emptyset \right\},$$

where $\overline{\mathsf{c}}(\mathsf{C})$ is the *center* of C. All the grid cells of $\mathscr{b}_{\mathsf{K}}(\mathsf{C})$ are contained in any disk of radius r centered at a point of C. Similarly, $\mathcal{B}_{\mathsf{K}}(\mathsf{C})$ is a super set of all the grid cells that cover any disk of radius r centered at any point of C.

Let $\alpha_{i,j} = |(\mathscr{b}_{\mathsf{K}}(\mathsf{C}_{i,j}) \cap P) * P_{i,j}|$ and $\beta_{i,j} = |(\mathcal{B}_{\mathsf{K}}(\mathsf{C}_{i,j}) \cap P) * P_{i,j}|$. Observe that $\alpha_{i,j} \le \xi_{i,j} \le \beta_{i,j}$. The set $\mathsf{O}_{i,j} = \mathcal{B}_{\mathsf{K}}(\mathsf{C}_{i,j}) \setminus \mathscr{b}_{\mathsf{K}}(\mathsf{C}_{i,j})$ is formed by all the grid cells intersecting a ring with outer radius $r + 2\Delta$ and inner radius $r - 2\Delta$. Let $Q_{i,j} = (\cup \mathsf{O}_{i,j}) \cap P_{i,j}$. Observe that $P_{i,j}$ and $Q_{i,j}$ are disjoint. Consider the set of pairs they induce $P_{i,j} * Q_{i,j}$, and let $\tau_{i,j}$ be the number of pairs in $P_{i,j} * Q_{i,j}$ of length at most r. We have that $\xi_{i,j} = \alpha_{i,j} + \tau_{i,j}$. Thus, the algorithm would compute the quantities $\alpha_{i,j}$ and $\tau_{i,j}$ for all i, j. The algorithm would then compute $\sum_{i,j} \xi_{i,j}/2$, which is the desired quantity.

Low level procedures. In the following, we assume that $n_{i,j} = |P_{i,j}| = O(n/\mathsf{N}^2)$.

Lemma 8.1. *After $O(n + \mathsf{N}^2)$ preprocessing, given a query of numbers i, j, one can compute $\alpha_{i,j}$ in $O(\mathsf{N})$ time.*

Proof: The algorithm computes the grid K, the subset of points in P in each grid cell, and their number. The algorithm then preprocess the grid so that given an a contiguous range of cells in a row (of the grid), the algorithm can report the number of points in this range in $O(1)$ time. This can be done using prefix sums for each row of the grid.

The desired quantity is $\alpha_{i,j} = |(\mathscr{b}_{\mathsf{K}}(\mathsf{C}_{i,j}) \cap P) * P_{i,j}| = n_{i,j} \sum_{\mathsf{C}_{u,v} \in \mathscr{b}_{\mathsf{K}}(\mathsf{C}_{i,j})} n_{u,v} - n_{i,j}^2 + \binom{n_{i,j}}{2}$. The set $\mathscr{b}_{\mathsf{K}}(\mathsf{C})$ in a row (of the grid) is just an contiguous box, and one can compute the number of points of P inside this box in $O(1)$ time. Thus, computing $\sum_{\mathsf{C}_{u,v} \in \mathscr{b}_{\mathsf{K}}(\mathsf{C}_{i,j})} n_{u,v}$ can be done in $O(\mathsf{N})$ time. ∎

Lemma 8.2. *After $O(n + \mathsf{N}^2)$ preprocessing, given a query numbers i, j, one can compute the set $Q_{i,j} = (\cup \mathsf{O}_{i,j}) \cap P_{i,j}$ in $O(n/\mathsf{N})$ time (this also bounds its size).*

Proof: The set O is a "ring" of the grid of with 4, and thus $|\mathsf{O}_{i,j}| = O(\mathsf{N})$. In particular, the set $\mathsf{O}_{i,j}$ can be computed in $O(\mathsf{N})$ time. The set $Q_{i,j}$ is formed by collecting all the point sets $P_{i,j}$ for cells $\mathsf{C}_{i,j} \in \mathsf{O}_{i,j}$. By assumption, $|P_{i,j}| = O(n/\mathsf{N}^2)$, which readily implies that $|Q_{i,j}| = O(\mathsf{N} \cdot n/\mathsf{N}^2) = O(n/\mathsf{N})$ ∎

Lemma 8.3. *Let Q and U be two disjoint point sets in the plane, with $|Q| < |U|$. Then one can compute the number of pairs of points in $Q * U$ that are in distance at most r from each other in $O(|Q|^2 + |U| \log |Q|)$ time.*

Proof: Let $\mathcal{D}$ be the set of disks of radius r centered at the points of Q. Compute the arrangement $\mathcal{A} = \mathcal{A}(Q)$, and compute for every face of $\mathcal{A}$ how many disks of $\mathcal{D}$ contain it. Furthermore, preprocess this arrangement for point-location queries in logarithmic time. This is all standard, and can be done in $O(|Q|^2)$ time [BCKO08]. Now compute for each point of U how many points of Q are in distance at most r from it, by performing a point-location query in $\mathcal{A}$, and returning the depth of the query point. ∎

Algorithm restated. The algorithm computes $\alpha_{i,j}, P_{i,j}, Q_{i,j}$ for all i, j using the above procedures. It then computes for all i, j, the quantity $\tau_{i,j}$ by using Lemma 8.3. The algorithm now computes directly $\sum_{i,j}(\alpha_{i,j} + \tau_{i,j})/2$ and return it as the desired quantity.

Analysis.

Lemma 8.4. *Assuming* $\mathsf{N} = O(\sqrt{n}/\log n)$*, with probability* $\geq 1 - 1/n^{O(1)}$*, each grid cell contains* $O(n/\mathsf{N}^2)$ *points of the random point set* P*.*

Proof: Each grid cell in the grid, in expectation, has $n/\mathsf{N}^2 = \Omega(\log^2 n)$ points of P in it. Now using Chernoff's inequality it follows that this quantity is concentrated (say up to $1 \pm 1/2$ around its expectation) with probability $\geq 1 - 1/n^{O(1)}$. Using the union bound on the N^2 grid cells, imply the claim. ∎

Running time analysis. Computing the sets $Q_{i,j}$, for all $i, j \in [\![\mathsf{N}]\!]$, takes $O(n\mathsf{N})$ time, using Lemma 8.2. Computing $\tau_{i,j}$, using Lemma 8.3, takes

$$O(|P_{i,j}|^2 + |Q_{i,j}| \log |P_{i,j}|) = O\big((n/\mathsf{N}^2)^2 + (n/\mathsf{N}) \log n\big)$$

time. doing this for all $i, j \in [\![\mathsf{N}]\!]$ takes $O\big(n^2/\mathsf{N}^2 + n\mathsf{N} \log n\big)$ time. Clearly, this dominates the running time. Solving for $n^2/\mathsf{N}^2 = n\mathsf{N} \log n$, we get $\mathsf{N} = (n/\log n)^{1/3}$. Clearly, the last step dominates the overall running time, which is $o(n\mathsf{N} \log n) = O(n^{4/3} \log^{2/3} n)$.

Theorem 8.5. *Let* P *be a set of* n *points picked uniformly and independently from* $[0, 1]^2$*, and let* r *be a parameter. One can compute, using the algorithm described above, the number of pairs of points in* P *in distance* $\leq r$ *from each other, in* $O(n^{4/3} \log^{2/3} n)$ *time. The result returned by the algorithm is always correct, and the bound on the running time holds with probability* $\geq 1 - 1/n^{O(1)}$*.*

9. Conclusions

To get Borůvka's algorithm to run in $O(n)$ time for MST, we had to restrict its growth phase in each recursive call. This feels unnatural in many ways since it is intentionally slowing down the algorithm's progress, but is necessary for a complete analysis. It remains open whether there is a method of showing Borůvka algorithm takes linear time in three or higher dimensions on random points. One possible direction would be to show that the average degree of the connected components in $G_{\angle,\delta}(P)$, see Definition 7.2, increases (for $d \geq 3$) extremely slowly compared to the halving of connected components. This is an observation the authors noted in numerical simulations, yet were unable to prove. See Figure 7.1. If the average degree increase in every round of Borůvka's algorithm can be bounded to a multiplicative constant $\xi < 2$ in each round then that would imply that Borůvka's algorithm runs in linear time.

References

[AS00] N. Alon and J. H. Spencer. *The Probabilistic Method, Second Edition*. John Wiley, 2000.

[BCKO08] M. de Berg, O. Cheong, M. J. van Kreveld, and M. H. Overmars. *Computational Geometry: Algorithms and Applications*. 3rd. Santa Clara, CA, USA: Springer, 2008.

[BKST78] J. L. Bentley, H. T. Kung, M. Schkolnick, and C. D. Thompson. On the average number of maxima in a set of vectors and applications. *J. Assoc. Comput. Mach.*, 25: 536–543, 1978.

[Cal10] P. Calka. Tessellations. *New Perspectives in Stochastic Geometry*. Ed. by W. S. Kendall and I. Molchanov. London, England: Oxford University Press, 2010, p. 606.

[Cha00] B. Chazelle. A minimum spanning tree algorithm with inverse-Ackermann type complexity. *J. ACM*, 47(6): 1028–1047, 2000.

[Cha01] T. M. Chan. On enumerating and selecting distances. *Int. J. Comput. Geom. Appl.*, 11(3): 291–304, 2001.

[Cha93] B. Chazelle. An optimal convex hull algorithm in any fixed dimension. *Discrete & Computational Geometry*, 10(4): 377–409, 1993.

[CHR16] H. Chang, S. Har-Peled, and B. Raichel. From proximity to utility: A Voronoi partition of pareto optima. *Discrete Comput. Geom.*, 56(3): 631–656, 2016.

[CZ21] T. M. Chan and D. W. Zheng. Hopcroft's problem, log-star shaving, 2d fractional cascading, and decision trees. *CoRR*, abs/2111.03744, 2021. arXiv: 2111.03744.

[DGL96] L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. eng. New York: Springer, 1996, p. 67.

[Dwy88] R. A. Dwyer. On the convex hull of random points in a polytope. *Journal of Applied Probability*, 25(4): 688–699, 1988.

[Dwy91] R. A. Dwyer. Higher-dimensional Voronoi diagrams in linear expected time. *Discrete Comput. Geom.*, 6(3): 343–367, 1991.

[GLM22] P. Gillibert, T. Lachmann, and C. Müllner. The VC-dimension of axis-parallel boxes on the torus. *J. Complexity*, 68: 101600, 2022.

[Hag09] T. Hagerup. An even simpler linear-time algorithm for verifying minimum spanning trees. *35th Int. Work. Graph-Theo. Concepts Comp. Sci.,* vol. 5911. 178–189, 2009.

[Har11a] S. Har-Peled. *Geometric Approximation Algorithms*. Vol. 173. Math. Surveys & Monographs. Boston, MA, USA: Amer. Math. Soc., 2011.

[Har11b] S. Har-Peled. On the expected complexity of random convex hulls. *CoRR*, abs/1111.5340, 2011.

[HJ20] S. Har-Peled and M. Jones. On separating points by lines. *Discret. Comput. Geom.*, 63(3): 705–730, 2020.

[HR15] S. Har-Peled and B. Raichel. Net and prune: A linear time algorithm for Euclidean distance problems. *J. Assoc. Comput. Mach.*, 62(6): 44:1–44:35, 2015.

[HW87] D. Haussler and E. Welzl. ε-nets and simplex range queries. *Discrete Comput. Geom.*, 2: 127–151, 1987.

[KKT95] D. R. Karger, P. N. Klein, and R. E. Tarjan. A randomized linear-time algorithm to find minimum spanning trees. *J. Assoc. Comput. Mach.*, 42(2): 321–328, 1995.

[Kut02] S. Kutin. *Extensions to McDiarmid's inequality when differences are bounded with high probability*. Tech. rep. TR-2002-04. U. Chicago, Apr. 2002.

[Mar04] M. Mareš. Two linear time algorithms for MST on minor closed graph classes. eng. *Archivum Mathematicum*, 040(3): 315–320, 2004.

[Mul94] K. Mulmuley. *Computational Geometry: An Introduction Through Randomized Algorithms*. Englewood Cliffs, NJ: Prentice Hall, 1994.

[PR02] S. Pettie and V. Ramachandran. An optimal minimum spanning tree algorithm. *J. ACM*, 49(1): 16–34, 2002.

[Ray70] H. Raynaud. Sur l'enveloppe convex des nuages de points aleatoires dans R^n. *J. Appl. Probab.*, 7: 35–48, 1970.

[San53] L. Santalo. *Introduction to Integral Geometry*. Paris, Hermann, 1953.

[SW10] R. Schneider and W. Weil. Classical stochastic geometry. *New Perspectives in Stochastic Geometry*. Ed. by W. S. Kendall and I. Molchanov. London, England: Oxford University Press, 2010, p. 606.

[VC13] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of the frequencies of occurrence of events to their probabilities. *Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik*. Ed. by B. Schölkopf, Z. Luo, and V. Vovk. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 7–12.

[VC71] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory Probab. Appl.*, 16: 264–280, 1971.

[WW93] W. Weil and J. A. Wieacker. Stochastic geometry. *Handbook of Convex Geometry*. Ed. by P. M. Gruber and J. M. Wills. Vol. B. North-Holland, 1993. Chap. 5.2, pp. 1393–1438.

[Yao82] A. C. Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM J. Comput.*, 11(4): 721–736, 1982.