

# On the most likely Voronoi diagram and nearest neighbor searching

**Citation for published version (APA):**

Suri, S., & Verbeek, K. A. B. (2016). On the most likely Voronoi diagram and nearest neighbor searching. *International Journal of Computational Geometry and Applications*, 26(3-4), 151-166.  
<https://doi.org/10.1142/S0218195916600025>

**Document license:**

Unspecified

**DOI:**

[10.1142/S0218195916600025](https://doi.org/10.1142/S0218195916600025)

**Document status and date:**

Published: 01/12/2016

**Document Version:**

Accepted manuscript including changes made at the peer-review stage

**Please check the document version of this publication:**

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

**General rights**

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

[www.tue.nl/taverne](http://www.tue.nl/taverne)

**Take down policy**

If you believe that this document breaches copyright please contact us at:

[openaccess@tue.nl](mailto:openaccess@tue.nl)

providing details and we will investigate your claim.

International Journal of Computational Geometry & Applications  
 © World Scientific Publishing Company

## ON THE MOST LIKELY VORONOI DIAGRAM AND NEAREST NEIGHBOR SEARCHING

SUBHASH SURI

*Department of Computer Science, University of California, Santa Barbara  
 Santa Barbara, 93106, United States of America  
 suri@cs.ucsb.edu*

KEVIN VERBEEK

*Department of Mathematics and Computer Science, TU Eindhoven, Den Dolech 2  
 Eindhoven, 5612 AZ, The Netherlands  
 k.a.b.verbeek@tue.nl*

Let  $\mathcal{S} = \{(s_1, \pi_1), (s_2, \pi_2), \dots, (s_n, \pi_n)\}$  be a set of stochastic sites, where each site is a tuple  $(s_i, \pi_i)$  consisting of a point  $s_i$  in  $d$ -dimensional space and a probability  $\pi_i$  of existence. Given a query point  $q$ , we define its *most likely nearest neighbor* (LNN) as the site with the largest probability of being  $q$ 's nearest neighbor. The *Most Likely Voronoi Diagram* (LVD) of  $\mathcal{S}$  is a partition of the space into regions with the same LNN. We investigate the complexity of LVD in one dimension and show that it can have size  $\Omega(n^2)$  in the worst-case. We then show that under non-adversarial conditions, the size of the 1-dimensional LVD is significantly smaller: (1)  $\Theta(kn)$  if the input has only  $k$  distinct probability values, (2)  $O(n \log n)$  on average, and (3)  $O(n\sqrt{n})$  under smoothed analysis. We also describe a framework for LNN search using *Pareto sets*, which gives a linear-space data structure and sub-linear query time in 1D for average and smoothed analysis models as well as the worst-case with a bounded number of distinct probabilities. The Pareto-set framework is also applicable to multi-dimensional LNN search via reduction to a sequence of nearest neighbor and spherical range queries.

*Keywords:* Uncertain data; nearest neighbors; Voronoi diagram; geometric data structures; Pareto sets.

### 1. Introduction

There is a growing interest in algorithms and data structures that deal with data uncertainty, driven in part by the rapid growth of unstructured databases where many attributes are missing or difficult to quantify<sup>5,6,10</sup>. Furthermore, an increasing amount of analytics today happens on data generated by machine learning systems, which is inherently probabilistic unlike the data produced by traditional methods. In computational geometry, the data uncertainty has typically been thought of as imprecision in the *positions* of objects—this viewpoint is quite useful for data produced by noisy sensors (e.g. LiDAR or MRI scanners) or associated with mobile entities, and many classical geometric problems including nearest-neighbors, convex hull, range searching and geometric optimization have been investigated in recent years<sup>2,3,4,14,16,17,18</sup>.

Our focus, in this paper, is on a different form of uncertainty: each object's location is known precisely but its *presence*, or activation, is subject to uncertainty. For instance,

2 Subhash Suri and Kevin Verbeek

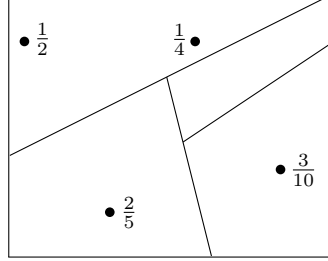


Fig. 1. An example of a two-dimensional LVD of four points, with respective probabilities  $\frac{1}{2}$ ,  $\frac{2}{5}$ ,  $\frac{3}{10}$ ,  $\frac{1}{4}$ . The first three points lie in their own Voronoi regions, but the point with probability  $\frac{1}{4}$  does not.

a company planning to open stores may know all the residents' locations but has only a probabilistic knowledge about their interest in its products. Similarly, many phenomena where *influence* is transmitted through *physical proximity* involve entities whose positions are known but their ability to influence others is best modeled probabilistically: opinions, diseases, political views, etc. With this underlying motivation, we investigate one of the most basic *proximity* search problems for stochastic input.

Let a *stochastic site* be a tuple  $(s_i, \pi_i)$ , where  $s_i$  is a point in  $d$ -dimensional Euclidean space and  $\pi_i$  is the probability of its existence (namely, activation). Let  $\mathcal{S} = \{(s_1, \pi_1), (s_2, \pi_2), \dots, (s_n, \pi_n)\}$  be a set of stochastic sites, where we assume that the points  $s_i$ 's are distinct, and that the individual probabilities  $\pi_i$  are independent. Whenever convenient, we will simply use  $s_i$  to refer to the site  $(s_i, \pi_i)$ . We want to preprocess  $\mathcal{S}$  for answering *most likely nearest neighbor* (LNN) queries: a site  $s_i$  is the LNN of a query point  $q$  if  $s_i$  is present *and* all other sites closer than  $s_i$  to  $q$  are not present. More formally, let  $\bar{\pi}_i = 1 - \pi_i$ , and let  $B(q, s_i)$  be the set of sites  $s_j$  for which  $\|q - s_j\| < \|q - s_i\|$ . Then the probability that  $s_i$  is the LNN of  $q$  is  $\pi_i \times \prod_{s_j \in B(q, s_i)} \bar{\pi}_j$ . For ease of reference, we call this probability the *likeliness* of  $s_i$  with respect to  $q$ , and denote it as

$$\ell(s_i, q) = \pi_i \times \prod_{s_j \in B(q, s_i)} \bar{\pi}_j. \quad (1)$$

The LNN of a query point  $q$  is the site  $s$  for which  $\ell(s, q)$  is maximized.

An important concept related to nearest neighbors is the Voronoi Diagram: it partitions the space into regions with the same nearest neighbor. In our stochastic setting, we seek the *most likely Voronoi Diagram* (LVD) of  $\mathcal{S}$ : a partition of the space into regions so that all query points in a region have the same LNN (see Figure 1). In addition to serving the role of a convenient *data structure* for LNN of query points, the structure of LVD also provides a compact representation of each stochastic site's region of *likely influence*.

**Related Work.** The topic of uncertain data has received a great deal of attention in recent years in the research communities of databases, machine learning, AI, algorithms and computational geometry. Due to limited space, we mention just a small number of papers that are directly relevant to our work. A number of researchers have explored nearest-neighbors and Voronoi diagrams for uncertain data<sup>2,4,14</sup>, however, these papers focus on the

*locational* uncertainty, with the goal of finding a neighbor minimizing the *expected* distance. Kamousi-Chan-Suri<sup>19</sup> consider the stochastic (existence uncertainty) model but they also focus on the expected distance. Unfortunately, nearest neighbors under the expected measure can give non-sensical answers—a very low probability neighbor gets a large weight simply by being near the query point. Instead, the most likely nearest neighbor gives a more intuitive answer.

Over the past decade, smoothed analysis has emerged as a useful approach for analyzing problems in which the complexity of typical cases deviates significantly from the worst-case. A classical example is the Simplex algorithm whose worst-case complexity is exponential and yet it runs remarkably well on most practical instances of linear programming. The smoothed analysis framework<sup>22</sup> offers a more insightful analysis than simple average case. Smoothed analysis is also quite appropriate for many geometric problems<sup>7,8,11,12</sup>, because data is often the result of physical measurements that are inherently noisy.

**Our Results.** We first show that the most likely Voronoi diagram (LVD) has worst-case complexity  $\Omega(n^2)$  even in 1D, which is easily seen to be tight. We then show that under more natural, and less pathological, conditions the LVD has significantly better behavior. Specifically, (1) if the input has only  $k$  distinct probability values, then the LVD has worst case size  $\Theta(nk)$ ; (2) if the probability values are randomly chosen (*average-case analysis*), then the LVD has expected size  $O(n \log n)$ ; (3) if the probability values (or the site positions) are worst-case but can be perturbed by some small random value (*smoothed analysis*), then the LVD has expected size  $O(n\sqrt{n})$ . Of course, the LVD immediately gives an  $O(\log n)$  time data structure for LNN queries. Next, we propose an alternative data structure for LNN queries using Pareto sets. In 1-dimension, this data structure has linear size and answers LNN queries in worst-case  $O(k \log n)$  time when the input has only  $k$  distinct probability values, and in  $O(\log^2 n)$  and  $O(\sqrt{n} \log n)$  (expected) time under the average case and smoothed analysis models, respectively. In the worst case, however, the query time is  $O(n \log n)$ , which is as fast as computing the LNN directly. Finally, the Pareto-set approach can be generalized to higher dimensions by reducing the problem to a sequence of nearest neighbor and spherical range queries. We give a concrete example of this generalization to finding the LNN in two dimensions.

## 2. The LVD can have Quadratic Complexity in 1D

The most likely nearest neighbor problem has non-trivial complexity even in the simplest of all settings: points on a line. Indeed, the LNN even violates a basic property often used in data structure design: *decomposability*. With deterministic data, one can split the input into a number of subsets, compute the nearest neighbor in each subset, and then choose the closest of those neighbors. As the following simple example shows, this basic property does not hold for the LNN.

Let the input have 3 sites  $\{(-2, \frac{1}{4}), (1, \frac{1}{3}), (3, \frac{3}{5})\}$ , and consider the query point  $q = 0$  (see Figure 2). Suppose we decompose the input into two subsets, sites to the left, and sites to the right of the query point. Then,

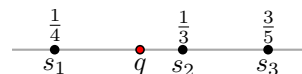
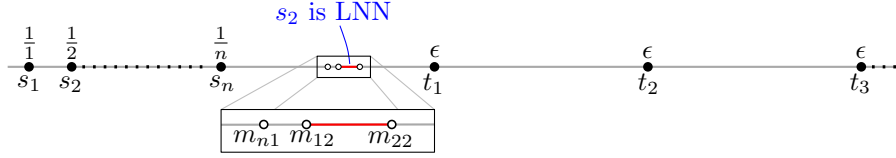


Fig. 2. The LNN of  $q$  is  $s_2$ .

4 Subhash Suri and Kevin Verbeek


 Fig. 3. The lower bound example of Theorem 1 with  $\Omega(n^2)$  complexity.

it is easy to check that  $s_1$  is the LNN on the left, and  $s_3$  is the LNN for the right subset. However, the overall LNN of  $q$  turns out to be  $s_2$ , as is easily verified by the likeliness probabilities:  $\ell(s_1, q) = \frac{2}{3} \cdot \frac{1}{4} = \frac{1}{6}$ ,  $\ell(s_2, q) = \frac{1}{3}$ , and  $\ell(s_3, q) = \frac{2}{3} \cdot \frac{3}{4} \cdot \frac{3}{5} = \frac{3}{10}$ .

The likeliness region for a site is also not necessarily connected: in fact, the following theorem shows that the LVD on a line can have quadratic complexity.

**Theorem 1.** *The most likely Voronoi diagram (LVD) of  $n$  stochastic sites on a line can have complexity  $\Omega(n^2)$ .*

**Proof.** Our input set realizing the lower bound consists of two groups of  $n$  sites each, for a total of  $2n$  sites. In the first group, which we call  $S$ , the  $i$ th site has position  $s_i = i/n$ , and probability  $\pi_i = 1/i$ , for  $i = 1, 2, \dots, n$ . In the second group, which we call  $T$ , the  $i$ th site has position  $t_i = i + 1$ , and probability  $\epsilon$ , for a choice of  $\epsilon$  specified later (see Figure 3). We will focus on the  $n^2$  midpoints  $m_{ij}$ , namely the bisectors, of pairs of sites  $s_i \in S$  and  $t_j \in T$ , and argue that the LNN changes in the neighborhood of each of these midpoints, proving the lower bound.

By construction, the midpoints  $m_{ij}$  are ordered lexicographically on the line, first by  $j$  and then by  $i$ . We will show that the LNN in the interval immediately to the left of the midpoint  $m_{ij}$  is  $s_i$ , which implies that the LVD has size  $\Omega(n^2)$ . For now we assume that if two sites have the same likeliness then the site with the lower index is chosen as the LNN. Let us consider a query point  $q$  that lies immediately to the left of the first midpoint  $m_{11}$ . It is easy to verify that  $\ell(s_i, q) = \frac{1}{n}$ , for all  $1 \leq i \leq n$ , and therefore  $s_1$  is  $q$ 's LNN. As the query point moves past  $m_{11}$ , only the likeliness of  $s_1$  changes to  $\frac{1-\epsilon}{n}$ , making  $s_2$  the LNN. The same argument holds as  $q$  moves past other midpoints towards the right, with the likeliness of corresponding sites changing to  $\frac{1-\epsilon}{n}$  in order, resulting in  $s_i$  becoming the new LNN when  $q$  lies just to the left of  $m_{i1}$ . After  $q$  passes  $m_{n1}$ , all sites of  $S$  have the same likeliness again, and the pattern is repeated for the remaining midpoints.

To avoid the tie-breaking rule mentioned above, we subtract a small value  $\delta > 0$  from the probability  $\pi_i$  belonging to each site  $s_i$  (the probabilities for sites in  $T$  remain  $\epsilon$ ). If we ignore the sites in  $T$ , then we get the relation  $\ell(s_i, q) = \frac{\pi_{i+1}\pi_i}{\pi_{i+1}} \ell(s_{i+1}, q)$ . We require that  $\ell(s_1, q) > \ell(s_2, q) > \dots > \ell(s_n, q)$ , so we need that  $\frac{\pi_{i+1}\pi_i}{\pi_{i+1}} > 1$  for all  $i$ . We have the

following for  $\delta < 1/n$ .

$$\frac{\pi_{i+1}\pi_i}{\pi_{i+1}} = \left(\frac{i}{i+1} + \delta\right) \left(\frac{1}{i} - \delta\right) = \frac{1}{i+1} - \delta + \delta \left(\frac{1}{i} + \frac{1}{i+1}\right) - \delta^2 \quad (2)$$

$$> \frac{1}{i+1} - \delta = \pi_{i+1} \quad (3)$$

Also, if  $\delta < n^{-2}$ , then we have the following.

$$\frac{\pi_{i+1}\pi_i}{\pi_{i+1}} = 1 + \frac{\delta\left(\frac{1}{i} + \frac{1}{i+1}\right) - \delta^2}{\frac{1}{i+1} - \delta} < 1 + \frac{\frac{2\delta}{i}}{\frac{1}{i+1} - \delta} \quad (4)$$

$$< 1 + \frac{\frac{2\delta}{i}}{\frac{1}{i+1} - \frac{1}{(i+1)^2}} = 1 + \frac{2\delta(i+1)^2}{i^2} \leq 1 + 8\delta \quad (5)$$

Hence, if  $\delta < n^{-2}$ , then  $1 < \frac{\pi_{i+1}\pi_i}{\pi_{i+1}} < 1 + 8\delta$  for all  $i$ .

Furthermore we need to ensure that no site in  $T$  can ever be the LNN, which holds if  $\frac{(1-\epsilon)^n}{n} > \epsilon$ . This can be achieved by choosing  $\epsilon = n^{-2}$ . Note that  $(1 - n^{-2})^n = 1 - 1/n + O(n^{-2})$ , and so  $(1 - \epsilon)^n > 1 - 1/n > 1/n = \epsilon n$  for  $n > 2$ .

Finally we require that, if the query point passes some midpoint like  $m_{11}$ , the likeliness of  $s_1$  becomes lower than the likeliness of all other sites, i.e.,  $\ell(s_1, q)(1 - \epsilon) < \ell(s_n, q)$ . If we choose  $\delta = \frac{1}{9n^3}$ , then we get the following.

$$\frac{\ell(s_1, q)}{\ell(s_n, q)} < (1 + 8\delta)^n = 1 + 8n\delta + O(n^2\delta^2) \quad (6)$$

$$= 1 + \frac{8}{9}n^{-2} + O(n^{-4}) < 1 + n^{-2} < \frac{1}{1 - \epsilon} \quad (7)$$

This completes the construction.  $\square$

### 3. Upper Bounds for the LVD in 1D

A matching upper bound of  $O(n^2)$  for the 1-dimensional LVD is easy: only the midpoints of pairs of sites can determine the boundary points of the LVD. In this section, we prove a number of stronger upper bounds, which may be more reflective of practical data sets. In particular, we show that if the number of distinct probability values among the stochastic sites is  $k$ , then the LVD has size  $\Theta(kn)$ , where clearly  $k \leq n$ . Thus, the LVD has size only  $O(n)$  if the input probabilities come from a fixed, constant size universe. This is not an unrealistic assumption in practice: a common technique in many applications is to discretize continuous data to improve computational efficiency. Probability distributions are a natural candidate for such a ‘‘binning’’ approach. Our results show that the complexity of the LVD grows linearly with the number of bins used in discretization.

Second, the lower bound construction of Theorem 1 requires a highly pathological arrangement of sites and their probabilities, unlikely to arise in practice. We therefore analyze the LVD complexity using average-case and smoothed analysis, and prove upper bounds of  $O(n \log n)$  and  $O(n\sqrt{n})$ , respectively.

### 3.1. Structure of the LVD

We first establish some structural properties of the LVD; in particular, which midpoints (bisectors) form the boundaries between adjacent cells of the LVD. For ease of reference, let us call these midpoints *critical*. Given a query point  $q$ , let  $\mathcal{L}(q)$  denote the sorted list of sites in  $\mathcal{S}$  by their (increasing) distance to  $q$ . Clearly, as long as the list  $\mathcal{L}(q)$  does not change by moving  $q$  along the line, its LNN remains unchanged. The order only changes at a midpoint  $m_{ij}$ , in which case  $s_i$  and  $s_j$  swap their positions in the list. The following lemmas provide a simple rule for determining critical midpoints.

**Lemma 1.** *Suppose that the midpoint  $m_{ij}$  of two sites  $s_i$  and  $s_j$  ( $s_i < s_j$ ) is critical, and consider the points  $q'$  immediately to the left of  $m_{ij}$ , and  $q''$  immediately to the right of  $m_{ij}$ . Then,  $s_i$  is the LNN of  $q'$ , or  $s_j$  is the LNN of  $q''$ .*

**Proof.** Suppose, for the sake of contradiction, that the LNN of  $q'$  is not  $s_i$ , but instead some other site  $s_z$ . Consider the list  $\mathcal{L}(q')$  of sites ordered by their distance to the query, and consider the change to this list as the query point shifts from  $q'$  to  $q''$ . The only change is swapping of  $s_i$  and  $s_j$ . Then the likelihood of  $s_i$  and  $s_j$  satisfy  $\ell(s_i, q'') < \ell(s_i, q')$  and  $\ell(s_j, q'') > \ell(s_j, q')$ , while for all other sites  $s$ , we have  $\ell(s, q') = \ell(s, q'')$ . Therefore, the LNN of  $q''$  is either  $s_j$  or  $s_z$ . If  $s_z$  is the LNN of  $q''$ , then  $m_{ij}$  is not critical (a contradiction). So  $s_j$  must be the LNN of  $q''$  satisfying the condition of the lemma.  $\square$

**Lemma 2.** *If the midpoint  $m_{ij}$  of sites  $s_i$  and  $s_j$ , for  $s_i < s_j$ , is critical, then there cannot be a site  $s_z$  with  $s_z \in [s_i, s_j]$  and  $\pi_z \geq \max(\pi_i, \pi_j)$ .*

**Proof.** Suppose, for the sake of contradiction, that such a site  $s_z$  exists. By the position of  $s_z$ , we must have  $\|s_z - m_{ij}\| < \min\{\|s_i - m_{ij}\|, \|s_j - m_{ij}\|\}$ , and the same also holds for any query point  $q$  arbitrary close to  $m_{ij}$ . Because  $\pi_z \geq \max(\pi_i, \pi_j)$ , we have  $\ell(s_z, q) > \ell(s_i, q)$  and  $\ell(s_z, q) > \ell(s_j, q)$ , implying that  $s_z$  is more likely than both  $s_i$  and  $s_j$  to be the nearest neighbor of any  $q$  arbitrary close to  $m_{ij}$ . By Lemma 1, however, if  $m_{ij}$  is critical, then there exists a  $q$  close to  $m_{ij}$  for which the LNN is either  $s_i$  or  $s_j$ . Hence  $s_z$  cannot exist.  $\square$

### 3.2. Refined Upper Bounds

Our first result shows that if the stochastic input has only  $k$  distinct probabilities, then the LVD has size  $O(kn)$ . Let  $\{S_1, \dots, S_k\}$  be the partition of the input so that each group has sites of the same probability, ordered by increasing probability; that is, any site in  $S_j$  has higher probability than a site in  $S_i$ , for  $j > i$ . We write  $n_i = |S_i|$ , where  $\sum_{i=1}^k n_i = n$ .

**Lemma 3.** *The LVD of  $n$  stochastic sites on a line, with at most  $k$  distinct probabilities, has worst-case complexity  $\Theta(kn)$ .*

**Proof.** The lower bound on the size follows from an easy modification of the construction in Theorem 1: we use only  $k - 1$  points for the left side of the construction. We now analyze

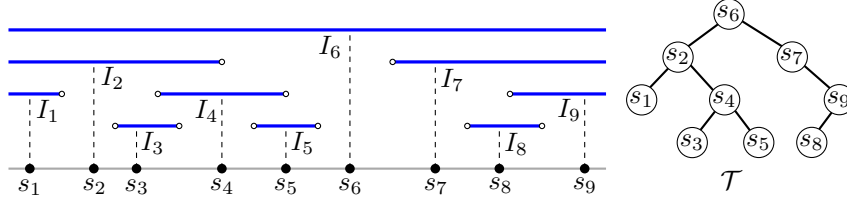


Fig. 4. The influence intervals (left) and the partition tree (right).

the upper bound. Suppose the midpoint  $m_{ij}$  defined by two sites  $s_i \in S_a$  and  $s_j \in S_b$  is critical, where  $1 \leq a < b \leq k$ , and without loss of generality, assume that  $s_i$  lies to the left of  $s_j$ . The sites in  $S_b$  have higher probability than those in  $S_a$ , because of our assumption that  $a < b$ . Hence, by Lemma 2, there cannot be a site  $s \in S_b$  such that  $s \in [s_i, s_j]$ . By the same reasoning, the midpoint of  $s_i$  and a site  $s \in S_b$  with  $s > s_j$  also cannot be critical. Therefore,  $s_i$  can form critical midpoints with at most two sites in  $S_b$ : one on each side. Altogether,  $s_i$  can form critical midpoints with at most  $2k$  other sites  $s_j$  with  $\pi_j \geq \pi_i$ . Thus,  $|LVD| \leq 2k \sum_{i=1}^k n_i = 2kn$ .  $\square$

### 3.3. Average-case and Smoothed Analysis of the LVD

We now show that even with  $n$  distinct probability values among the stochastic sites, the LVD has significantly smaller complexity as long as those probabilities are either assigned randomly to the points, or they can be perturbed slightly to get rid of the highly unstable pathological cases. More formally, for the average-case analysis we assume that we have a fixed set of  $n$  probabilities, and we randomly assign these probabilities to the sites. That is, we consider the average over all possible assignments of probabilities to sites. The smoothed analysis fixes a *noise* parameter  $a > 0$ , and draws a noise value  $\delta_i \in [-a, a]$  uniformly at random for each site  $(s_i, \pi_i)$ . This noise is used to perturb the input, either the location of a site or its probability. The location perturbation changes each site's position to  $s'_i = s_i + \delta_i$ , resulting in the randomly perturbed input  $\mathcal{S}' = \{(s'_1, \pi_1), \dots, (s'_n, \pi_n)\}$ , which is a random variable. The smoothed complexity of the LVD is the expected complexity of the LVD of  $\mathcal{S}'$ , where we take the worst case over all inputs  $\mathcal{S}$ . The smoothed complexity naturally depends on the noise parameter  $a$ , which for the sake of simplicity we assume to be a constant—more detailed bounds involving  $a$  can easily be obtained. Of course, for this model we need to restrict the positions of sites to  $[0, 1]$ . The smoothed model perturbing the probabilities instead of the positions is defined analogously.

Our analysis uses a *partition tree*  $\mathcal{T}$  defined on the sites as follows. The tree is rooted at the site  $s_i$  with the highest probability. The remaining sites are split into a set  $S_1$ , containing the sites on the left of  $s_i$ , and a set  $S_2$  containing the rest (excluding  $s_i$ , see Figure 4 right). We then recursively construct the partition trees for  $S_1$  and  $S_2$ , whose roots become the children of  $s_i$ . (In case of ties, choose  $s_i$  to make the partition as balanced as possible.) The partition tree has the following useful property.



**Lemma 4.** *Let  $s_i$  and  $s_j$  be two sites with  $\pi_i \leq \pi_j$ . If the midpoint  $m_{ij}$  is critical, then  $s_j$  is an ancestor of  $s_i$  in  $\mathcal{T}$ .*

**Proof.** Let  $s_z$  be the lowest common ancestor of  $s_i$  and  $s_j$  in  $\mathcal{T}$ , assuming  $s_z \neq s_j$ . By construction,  $s_z \in [s_i, s_j]$  and  $\pi_z \geq \pi_j$ . Hence, by Lemma 2,  $m_{ij}$  cannot be critical.  $\square$

**Corollary 1.** *If the depth of  $\mathcal{T}$  is  $d$ , then the size of the LVD is  $O(dn)$ .*

Thus, we can bound the average and smoothed complexity of the LVD by analyzing the average and smoothed depth of the partition tree  $\mathcal{T}$ . In the average case,  $\mathcal{T}$  is essentially a *random binary search tree*. The keys are given by the positions of the sites, and the order in which these keys are added to the binary tree is determined by the probabilities of the sites (decreasing order on probabilities). Note that it does not matter if we take the average over all positions or over all probabilities (or both); for a random binary search tree, randomly permuting the values of the keys is equivalent to randomly permuting the insertion order of the keys. It is well known<sup>21</sup> that the expected depth of a random binary search tree, and thus also the average depth of the partition tree, is  $O(\log n)$ . In the smoothed model, if the perturbation is on the *position* of the sites, then a result by Manthey and Tantau<sup>20</sup> shows that the smoothed depth of  $\mathcal{T}$  is  $O(\sqrt{n})$ .<sup>a</sup> We can easily extend that analysis to the perturbation on the probability values, instead of the positions of the sites (perturbed probability values that fall outside  $[0, 1]$  are clamped). In a nutshell, the proof by Manthey and Tantau relies on the fact that the input elements can be partitioned into  $O(\sqrt{n}/\log n)$  groups such that the binary search tree of a single group is essentially random, and in this random tree, we can simply swap the roles of probabilities and positions as explained above. Thus, the smoothed depth of  $\mathcal{T}$  is also  $O(\sqrt{n})$  if the probabilities are perturbed. (The analysis also holds for clamped values due to our tie-breaking rule.)

**Theorem 2.** *Given a set of  $n$  stochastic sites on the line, its most likely Voronoi Diagram (LVD) has average-case complexity  $O(n \log n)$ , and smoothed complexity  $O(n\sqrt{n})$ .*

#### 4. Algorithms for Constructing the LVD

Our main tool for constructing the LVD is the *likeliness curve*  $\ell(s_i): \mathbb{R} \rightarrow \mathbb{R}$  of a site  $s_i$ , which is simply the function  $\ell(s_i, q)$  with  $q$  ranging over the entire real line  $\mathbb{R}$ . A likeliness curve  $\ell(s_i)$  has  $O(n)$  complexity and it is a bimodal step function, achieving its maximum value at  $q = s_i$  (see Figure 5). By presorting all the sites in the left-to-right order, we can easily compute each  $\ell(s_i)$  in  $O(n)$  time, as follows. Start at  $q = s_i$  and walk to the left updating the value  $\ell(s_i, q)$  at every midpoint of the form  $m_{ij}$  with  $1 \leq j < i$ . We do the same for the right portion of  $\ell(s_i)$ , walking to the right instead (and  $i < j \leq n$ ). In the same way we can compute a restriction of  $\ell(s_i)$  to some interval  $I$ : assuming  $s_i \in I$ , it is easy to see that this restriction can be computed in time proportional to its complexity.

<sup>a</sup>Manthey and Tantau<sup>20</sup> construct a binary search tree from a sequence of real numbers. We obtain this sequence from our input by ordering the stochastic sites by decreasing probabilities. Their construction of binary search trees then matches our construction of  $\mathcal{T}$ .

We can now compute the LVD by constructing the upper envelope  $\mathcal{U}$  of all  $\ell(s_i)$ , for  $i = 1, \dots, n$ . A naive construction, however, still takes  $O(n^2)$  time since the total complexity of all likeliness curves is quadratic. Instead, we restrict the likeliness curve of every site to a critical subpart such that the upper envelope of these partial curves gives the correct  $\mathcal{U}$ . In particular, for each site  $s_i$ , define the *influence interval*  $I_i$  as follows. Let  $s_j$  be the first site encountered on the left of  $s_i$  for which  $\pi_j \geq \pi_i$ , and let  $s_z$  be the first site encountered on the right side of  $s_i$ . Then we define  $I_i = [m_{ji}, m_{iz}]$ . (If  $s_j$  and/or  $s_z$  does not exist, we replace  $m_{ji}$  with  $-\infty$  and/or  $m_{iz}$  with  $+\infty$ , respectively.) Observe that, for any  $q \notin I_i$ , either  $\ell(s_i, q) < \ell(s_j, q)$  or  $\ell(s_i, q) < \ell(s_z, q)$ , since either  $s_j$  or  $s_z$  is closer to  $q$  and  $\pi_j, \pi_z \geq \pi_i$ . We define  $\ell'(s_i)$  as the restriction of  $\ell(s_i)$  to the interval  $I_i$  (see Figure 5). Clearly,  $\mathcal{U}$  can be constructed by computing the upper envelope of just these restrictions  $\ell'(s_i)$ , and the complexity of each  $\ell'(s_i)$  is exactly the number of midpoints involving  $s_i$  that lie in  $I_i$ . Thus, given the defining sites  $s_j$  and  $s_z$  of  $I_i$ , the complexity of  $\ell'(s_i)$  is the number of sites in the interval  $[s_j, s_z]$  minus one (excluding  $s_i$ ).

**Lemma 5.** *The complexity of the union of all  $\ell'(s_i)$ , for  $i = 1, 2, \dots, n$ , is  $O(nd)$ , where  $d$  is the depth of the partition tree  $\mathcal{T}$  of the input sites. Furthermore, the union of  $\ell'(s_i)$  can be represented by  $d$  curves of  $O(n)$  complexity each.*

**Proof.** Let  $\sigma_1, \dots, \sigma_r$  be the set of sites at a fixed depth in the partition tree  $\mathcal{T}$  in order, and let  $\tau_i$ , for  $1 \leq i < r$ , be the lowest common ancestor of  $\sigma_i$  and  $\sigma_{i+1}$  in the tree. It is easy to see that the influence interval of a site  $\sigma_i$  is defined by a site in  $[\tau_{i-1}, \sigma_i]$  (possibly  $\tau_{i-1}$ ) and a site in  $[\sigma_i, \tau_i]$  (possibly  $\tau_i$ ), assuming  $1 < i < r$  (otherwise the influence interval may extend to  $-\infty$  or  $+\infty$ , see Figure 4). Hence the complexity of  $\ell'(\sigma_i)$  is bounded by the number of sites in the interval  $[\tau_{i-1}, \tau_i]$ . Furthermore, all influence intervals of the sites  $\sigma_1, \dots, \sigma_r$  are disjoint, and so we can combine all  $\ell'(\sigma_i)$  into a single curve with  $O(n)$  complexity. The result follows by constructing such a curve for each level of the partition tree.  $\square$

We can use Lemma 5 to efficiently compute the upper envelope  $\mathcal{U}$ . First, we compute the  $d$  curves  $f_1, \dots, f_d$  mentioned in Lemma 5, one for each level of  $\mathcal{T}$ , as follows. As we construct  $\mathcal{T}$ , we simultaneously compute  $\ell'(s_i)$  for each site  $s_i$ , in time  $O(|\ell'(s_i)|)$  time. This takes  $O(n)$  time per level of  $\mathcal{T}$ . We can then easily combine the individual parts  $\ell'(s_i)$  to obtain the curves  $f_1, \dots, f_d$ . The total running time of computing the curves  $f_1, \dots, f_d$  is  $O(n \log n + dn)$ , consisting of  $O(n \log n)$  time to compute  $\mathcal{T}$ , and  $O(n)$  time to compute each curve  $f_i$  ( $1 \leq i \leq d$ ).

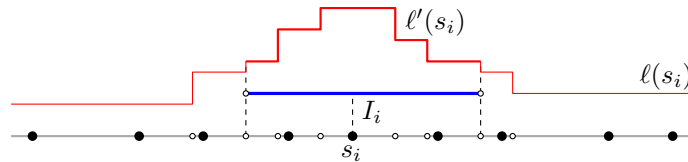


Fig. 5. The likeliness curve  $\ell(s_i)$  of  $s_i$  and its restriction  $\ell'(s_i)$  to  $I_i$ .

Finally we can construct  $\mathcal{U}$  by computing the upper envelope of the curves  $f_1, \dots, f_d$ . We scan through the curves from left to right, maintaining two priority queues: (1) a priority queue for the events at which a curve changes, and (2) a priority queue for maintaining the curve with the highest likeliness. Both priority queues have size  $d$ , which means that each event can be handled in  $O(\log d)$  time.

**Lemma 6.** *If  $d$  is the depth of  $\mathcal{T}$ , then the LVD can be constructed in  $O(n \log n + dn \log d)$  time.*

The algorithm is easily adapted for the case of  $k$  distinct probabilities. Consider the sites  $\sigma_1, \dots, \sigma_r$  (in order) for a single probability value. Since they all have the same probability, they bound each other's influence intervals, and hence all influence intervals are interior disjoint. Now assume that a site  $s_j$  is contained in the interval  $[\sigma_i, \sigma_{i+1}]$ . Then  $s_j$  can add to the complexity of only  $\ell'(\sigma_i)$  and  $\ell'(\sigma_{i+1})$ , and no other  $\ell'(\sigma_z)$  with  $z \neq i, i+1$ . Thus, we can combine the partial likeliness curves  $\ell'(\sigma_i)$  into a single curve of  $O(n)$  complexity. In total we obtain  $k$  curves of  $O(n)$  complexity each, from which we can construct the LVD.

**Theorem 3.** *The LVD of  $n$  stochastic sites in 1D can be computed in worst-case time  $O(n \log n + nk \log k)$  if the sites involve  $k$  distinct probabilities. Without the assumption on distinct probabilities, the construction takes  $O(n \log n \log \log n)$  time in the average case and  $O(n\sqrt{n} \log n)$  time in the smoothed analysis model.*

**Proof.** The result for  $k$  distinct probabilities directly follows from the discussion above. For the smoothed analysis model, note that  $O(n \log n + dn \log d) = O(n \log n + dn \log n)$ . Since  $d = O(\sqrt{n})$  in the smoothed analysis model, then construction time is  $O(n\sqrt{n} \log n)$ . For the average case, the construction time is  $E[O(n \log n + dn \log d)]$ , which can be simplified to  $O(n \log n + nE[d \log d])$ . Note that, in general,  $E[d \log d] \neq E[d] \log E[d]$ . However, we can use the following existing result<sup>13</sup>:

$$P[d \geq x] \leq \frac{1}{n} \left( \frac{2e \log n}{x} \right)^x$$

Therefore we can compute  $E[d \log d]$  as follows:

$$\begin{aligned} E[d \log d] &= \sum_{x=1}^n P[d = x] x \log x \\ &\leq \sum_{x=1}^{2e \log n} P[d = x] x \log x + P[d \geq 2e \log n] n \log n \\ &\leq \sum_{x=1}^{2e \log n} P[d = x] x O(\log \log n) + \log n \\ &\leq E[d] O(\log \log n) + \log n \\ &= O(\log n \log \log n). \end{aligned}$$

It follows that the construction takes  $O(n \log n \log \log n)$  time in the average case.  $\square$

## 5. Time-Space Tradeoffs for LNN Searching

The worst-case complexity of the LVD is  $\Omega(n^2)$  even in 1 dimension and the Voronoi region of a single site can have  $\Omega(n)$  disjoint intervals. This raises a natural question: can the 1-dimensional LNN search be solved by a data structure of *subquadratic* size and *sub-linear* query time? While we cannot answer that question definitively, we offer an argument suggesting its hardness below.

### 5.1. A 3SUM Hard Problem

Consider the following problem, which we call the NEXT MIDPOINT PROBLEM: *given a set of  $n$  sites on a line, preprocess them so that for a query  $q$  we can efficiently compute the midpoint (for some pair of sites) that is immediately to the right of  $q$ .* The problem is inspired by the fact that an LNN query essentially needs to decide the location of the query point among the (potentially  $\Omega(n^2)$  critical) midpoints of the input. The following lemma proves 3SUM-hardness of this problem. (Recall that the 3SUM problem asks, given a set of numbers  $a_1, \dots, a_n$ , does there exist a triple  $(a_i, a_j, a_z)$  satisfying  $a_i + a_j + a_z = 0$ .)

**Lemma 7.** *Building the data structure plus answering  $2n$  queries of the NEXT MIDPOINT PROBLEM is 3SUM-hard.*

**Proof.** Consider an instance of the 3SUM problem consisting of numbers  $a_1, \dots, a_n$ . We use these numbers directly as sites for the NEXT MIDPOINT PROBLEM. If there exists a triple for which  $a_i + a_j + a_z = 0$ , then the midpoint  $m_{ij}$  is at  $-a_z/2$ . Thus, for every input number  $a_z$ , we query the NEXT MIDPOINT data structure just to the left and just to the right of  $-a_z/2$  (all numbers are integers, so this is easy). If the next midpoint is different for the two queries, then there exists a triple for which  $a_i + a_j + a_z = 0$ . Otherwise, such a triple does not exist.  $\square$

**Remark.** Thus, unless 3SUM can be solved in significantly faster than  $O(n^2)$  time, either the preprocessing time for the Next Midpoint problem is close to  $\Omega(n^2)$ , or the query time is close to  $\Omega(n)$ . Specifically, if 3SUM requires  $\Omega(f(n))$  time, then either the preprocessing time is  $\Omega(f(n))$ , or the query time is  $\Omega(f(n)/n)$ . However, our reduction does not imply a hardness for the LNN problem in general: the order of the midpoints in the example of Theorem 1 follows a very simple pattern, which can be encoded efficiently.

### 5.2. LNN Search Using Pareto Sets

We now propose an alternative approach to LNN search using Pareto sets, which trades query time for space. Consider a query point  $q$ , and suppose that its LNN is the site  $s_i$ . Then,  $s_i$  must be Pareto optimal with respect to  $q$ , that is, there cannot be a site  $s_j$  closer to  $q$  with  $\pi_j \geq \pi_i$ . In fact, recalling the influence intervals  $I_i$  from the previous section, it is easy to check that  $s_i$  is Pareto optimal for  $q$  if and only if  $q \in I_i$ . This observation suggests the following algorithm for LNN: (1) compute the set  $S$  of sites  $s_i$  with  $q \in I_i$ , (2) compute  $\ell(s, q)$  for each  $s \in S$ , and (3) return  $s \in S$  with the maximum likeliness.

Step (1) requires computing the influence intervals for all sites, which is easily done as follows. Sort the sites in descending order of probability, and suppose they are numbered in this order. We incrementally add the sites to a balanced binary search tree, using the position of a site as its key. When we add a site  $s_i$  to the tree, all the sites with a higher probability are already in the tree. The interval  $I_i$  is defined by the two consecutive sites  $s_j$  and  $s_z$  in the tree such that  $s_i \in [s_j, s_z]$ . Thus, we can find  $s_j$  and  $s_z$  in  $O(\log n)$  time when adding  $s_i$  to the tree, and compute all the influence intervals in  $O(n \log n)$  total time<sup>b</sup>. To find the intervals containing the query point, we organize the influence intervals in an interval tree, which takes  $O(n \log n)$  time and  $O(n)$  space, and solves the query in  $O(\log n + r)$  time, where  $r$  is the output size. By the results in previous sections, we have  $r \leq \min\{k, d\}$ , where  $k$  is the number of distinct probabilities and  $d$  is the depth of  $\mathcal{T}$ .

Step (2) requires computing the likeliness of each site efficiently, and we do this by rewriting the likeliness function as follows:

$$\ell(s_i, q) = \pi_i \times \prod_{s_j \in (q-a, q+a)} \bar{\pi}_j \quad \text{where } a = |q - s_i| \quad (8)$$

With Equation (8), we can compute the likeliness of a site by a single range search query. For that we use an augmented balanced binary search tree, where the augmentation stores the product of inverse probabilities. This solution requires  $O(n)$  space,  $O(n \log n)$  construction time, and can compute  $\ell(s_i, q)$  in  $O(\log n)$  time.

**Theorem 4.** *There is a data structure for 1D LNN search that needs  $O(n)$  space and  $O(n \log n)$  construction time and answers queries in (1) worst-case  $O(k \log n)$  time if the sites involve  $k$  distinct probabilities, (2) expected time  $O(\log^2 n)$  in the average case, and (3) expected time  $O(\sqrt{n} \log n)$  in the smoothed analysis model.*

**Remark.** The query bounds of Theorem 4 for the average and smoothed analysis model are strong in the sense that they hold for *all* query points simultaneously, and not just for a fixed query point. That is, the bounds are for the expected worst case query time, rather than the expected query time.

## 6. The Pareto-Set Approach in Higher Dimensions

Our Pareto-set approach essentially requires the following two operations: (1) find the Pareto set for a query point  $q$ , and (2) compute the likeliness of a site w.r.t.  $q$ . In higher dimensions, the second operation can be performed with a spherical range query data structure, for which nearly optimal data structures exist<sup>1</sup>. The first operation can be reduced to a sequence of nearest neighbor queries, as follows: (1) find the nearest neighbor of  $q$ , say  $s_i$ , among all sites and add  $s_i$  to the Pareto set, (2) remove all sites with probability at most  $\pi_i$ , and (3) repeat steps (1) and (2) until no sites are left. We, therefore, need a data structure supporting

<sup>b</sup>If there are sites with the same probability, we must first determine their influence intervals among sites with the same probability, before adding them to the tree. This can easily be achieved by first sorting the sites on position.

the following query: given a query point  $q$  and a probability  $\pi$ , find the closest site to  $q$  with probability higher than  $\pi$ . A dynamic nearest neighbor data structure can be adapted to answer this query as follows: incrementally add sites in decreasing order of probability, and make the data structure partially persistent<sup>c</sup>. In this way, the data structure can answer the query we need, and partially persistent data structures often require only little extra space.

The required number of nearest neighbor and spherical range queries is precisely the number of elements in the Pareto set. For a query point  $q$ , consider the sequence of the sites' probabilities ordered by their increasing distance to  $q$ . Observe that the size of the Pareto set is precisely the number of left-to-right maxima in this sequence<sup>20</sup>. Therefore, the size of the Pareto set is (1) at most  $k$  when the input has at most  $k$  distinct probabilities, (2)  $O(\log n)$  in the average case model, and (3)  $O(\sqrt{n})$  in the smoothed analysis model. (Unlike the bound of Section 5.2, however, this result holds for any arbitrary query but not for the worst case among all queries.) A concrete realization of this abstract approach is discussed below for LNN search in 2D.

**2D Euclidean LNN Search.** For the sake of illustration, we consider only the average case of LNN queries. In this case, an incremental construction ordered by decreasing probabilities is simply a randomized incremental construction. We can then use the algorithm by Guibas *et al.*<sup>15</sup> to incrementally construct the Voronoi diagram including a planar point location data structure, which uses  $O(n)$  space on average. Although not explicitly mentioned in the paper, this data structure is partially persistent. Using this data structure we can answer a nearest neighbor query in  $O(\log^2 n)$  time. For the circular range queries, we use the data structure by Chazelle and Welzl<sup>9</sup>, which uses  $O(n \log n)$  space and can answer queries in  $O(\sqrt{n} \log^2 n)$  time. The final result is a data structure that uses, on average,  $O(n \log n)$  space and can answer LNN queries in  $O(\log^2 n \cdot \log n + \sqrt{n} \log^2 n \cdot \log n) = O(\sqrt{n} \log^3 n)$  time.

## 7. Concluding Remarks

The introduction of uncertainty seems to make even simple geometric problems quite hard, at least in the worst case. At the same time, uncertain data problems and algorithms may be particularly well-suited for average-case and smoothed analyses: after all, probabilities associated with uncertain data are inherently fuzzy measures, and problem instances whose answer changes dramatically with minor perturbations of input may suggest fragility of those probabilistic assumptions.

Our research suggests a number of open problems and research questions. In the 1-dimensional setting, we are able to settle the complexity of the LVD in the worst-case, and provide stronger upper bounds for average-case and smoothed analysis. It will be interesting to extend the results to higher dimensions. In particular, we believe the worst-case complexity of the  $d$ -dimensional LVD is  $\Omega(n^{2d})$ , but that is work in progress. Settling that complexity in the average or smoothed analysis case, as well as in the case of  $k$  distinct

<sup>c</sup>A persistent data structure always preserves the previous version of itself when modified; a partially persistent data structure only allows modifications to the newest version of the data structure.

14 Subhash Suri and Kevin Verbeek

probabilities, is entirely open.

**Acknowledgments.** The authors gratefully acknowledge support from the National Science Foundation, under the grants CNS-1035917 and CCF-11611495, and DARPA.

## References

1. P. Agarwal. Range Searching. In *CRC Handbook of Discrete and Computational Geometry* (J. Goodman and J. O'Rourke, eds.), CRC Press, New York, 2004.
2. P. Agarwal, B. Aronov, S. Har-Peled, J. Phillips, K. Yi, and W. Zhang. Nearest neighbor searching under uncertainty II. In *Proc. 32nd PODS*, pp. 115–126, 2013.
3. P. Agarwal, S. Cheng, and K. Yi. Range searching on uncertain data. *ACM Trans. on Alg.*, 8(4):43, 2012.
4. P. Agarwal, A. Efrat, S. Sankararaman, and W. Zhang. Nearest-neighbor searching under uncertainty. In *Proc. 31st PODS*, pp. 225–236, 2012.
5. C. Aggarwal. *Managing and Mining Uncertain Data*. Advances in Database Systems Vol. 35, Springer, 1st edition, 2009.
6. C. Aggarwal, and P. Yu. A survey of uncertain data algorithms and applications. *IEEE Trans. Knowl. Data Eng.*, 21(5):609–623, 2009.
7. M. de Berg, H. Haverkort, and C. Tsirgiannis. Visibility maps of realistic terrains have linear smoothed complexity. *J. of Comp. Geom.*, 1(1):57–71, 2010.
8. S. Chaudhuri, V. Koltun. Smoothed analysis of probabilistic roadmaps. *Comp. Geom. Theor. Appl.*, 42(8):731–747, 2009.
9. B. Chazelle and E. Welzl. Quasi-optimal range searching in spaces of finite VC-dimension. *Discrete Comput. Geom.*, 4:467–489, 1989.
10. N. Dalvi, C. Ré, and D. Suci. Probabilistic databases: diamonds in the dirt. *Communications of the ACM*, 52(7):86–94, 2009.
11. V. Damerow, F. Meyer auf der Heide, H. Räcke, C. Scheideler, and C. Sohler. Smoothed motion complexity. In *Proc. 11th ESA*, pp. 161–171, 2003.
12. V. Damerow and C. Sohler. Extreme points under random noise. In *Proc. 12th ESA*, pp. 264–274, 2004.
13. L. Devroye. A note on the height of binary search trees. *J. ACM*, 33(3):489–498, 1986.
14. W. Evans and J. Sember. Guaranteed Voronoi diagrams of uncertain sites. In *Proc. 20th CCCG*, pp. 207–210, 2008.
15. L. Guibas, D. Knuth, and M. Sharir. Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica*, 7, 381–413, 1992.
16. A. Jørgensen, M. Löffler, and J. Phillips. Geometric computations on indecisive and uncertain points. *CoRR*, abs/1205.0273, 2012.
17. M. Löffler. *Data imprecision in computational geometry*. PhD Thesis, Utrecht University, 2009.
18. M. Löffler and M. van Kreveld. Largest and smallest convex hulls for imprecise points. *Algorithmica*, 56:235–269, 2010.
19. P. Kamousi, T. Chan, and S. Suri. Closest pair and the post office problem for stochastic points. *Comp. Geom. Theor. Appl.*, 47(2):214–223, 2014.
20. B. Manthey and T. Tantau. Smoothed analysis of binary search trees and Quicksort under additive noise. In *Proc. 33rd Int. Symp. Math. Found. Comp. Sci.*, pp. 467–478, 2008.
21. B. Reed. The height of a random binary search tree. *J. ACM*, 50(3):306–332, 2003.
22. D. Spielman and S. Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *J. ACM*, 51:385–463, 2004.
23. S. Suri, K. Verbeek, and H. Yıldız. On the most likely convex hull of uncertain points. In *Proc. 21st ESA*, pp. 791–802, 2013.