Stochastic Closest-pair Problem and Most-likely Nearest-neighbor Search in Tree Spaces

Jie Xue and Yuan Li (✉)

University of Minnesota — Twin Cities, Minneapolis MN 55455, USA
{xuexx193,lixx2100}@umn.edu

Abstract. Let $\mathcal{T}$ be a tree space represented by a weighted tree with t vertices, and S be a set of n stochastic points in $\mathcal{T}$, each of which has a fixed location with an independent existence probability. We investigate two fundamental problems under such a stochastic setting, the closest-pair problem and the nearest-neighbor search. For the former, we propose the first algorithm of computing the ℓ-threshold probability and the expectation of the closest-pair distance of a realization of S. For the latter, we study the k most-likely nearest-neighbor search (k-LNN) via a notion called the k most-likely Voronoi Diagram (k-LVD), where we show the combinatorial complexity of k-LVD is $O(nk)$ under two reasonable assumptions, leading to a logarithmic query time for k-LNN.

1 Introduction

In many real-world applications, due to the existence of noise or limitations of devices, the data obtained may be imprecise or not totally reliable. In this situation, the dataset may fail to capture well the features of the data. Motivated by this, the topic of uncertain data has received significant attention in the last few decades. Many classical problems have been investigated under uncertainty, including convex hull, minimum spanning tree, range search, linear separability, etc. [1,2,3,4,5,7,8,10,13,15,16]. Among these, there are two common models of uncertainty: existential uncertainty and locational uncertainty. In the former, each (stochastic) data point has a fixed location with an uncertain existence depicted by an independent existence probability, while in the latter the location of each point is uncertain and described as a distribution.

The closest-pair problem and nearest-neighbor search are two interrelated fundamental problems, which have numerous applications in various areas. The uncertain versions of both the problems have also been studied recently in [1,9,11,12,14]. Let S be a set of n stochastic points in some metric space $\mathcal{X}$. Concerning the closest pair problem, a basic question one may ask is how to compute elementary statistics about the stochastic closest-pair of S, e.g., the probability that the closest-pair distance of a realization of S is at least ℓ, the expected closest-pair distance, etc. Unfortunately, most problems of this kind have been shown to be NP-hard or #P-hard for general metrics, and some of them remain #P-hard even when $\mathcal{X} = \mathbb{R}^d$ for $d \geq 2$ [9,11]. Concerning the nearest-neighbor search, an important problem is the most-likely nearest-neighbor (LNN) search

F. Ellen et al. (Eds.): WADS 2017, LNCS 10389, pp. 569–580, 2017.
DOI: 10.1007/978-3-319-62127-2_48

[14], which looks for the data point in S with the greatest probability of being the nearest-neighbor of a query point q. The LNN search introduces the concept of most-likely Voronoi diagram (LVD), which decomposes $\mathcal{X}$ into connected cells such that the query points in the same cell have the same LNN. However, as in [12,14], the bound of LVD in $\mathbb{R}^d$ is still high even on average. Due to the difficulties of both problems in general and Euclidean space, it is then natural to ask whether these problems are relatively easier in other metric spaces such as a *tree space*. Indeed, further exploring these problems in tree spaces will be helpful and interesting since any finite metrics (say a road network in practice) can be embedded on a tree space under some reasonable distortions [6].

With the above motivations, in this paper, we study the stochastic closest-pair (SCP) problem and k most-likely nearest-neighbor (k-LNN) search in tree spaces. A *tree space* $\mathcal{T}$ is represented by a positively-weighted tree T where the weight of each edge depicts its "length". Formally, $\mathcal{T}$ is the geometric realization of T, in which each edge weighted by w is isometric to the interval $[0, w]$. There is a natural metric over $\mathcal{T}$ which defines the distance $dist(x, y)$ as the length of the (unique) simple path between x and y in $\mathcal{T}$. See Fig. 1 for an example of tree space. Following [9,11,14], we study the problems under existential uncertainty: each stochastic point has a fixed location (in $\mathcal{T}$) associated with an (independent) existence probability. Due to limited space, the proofs of all lemmas and some theorems are omitted and can be found in the full version [17].

Our results. Let $\mathcal{T}$ be a tree space represented by a t-vertex weighted tree T, and S be the given set of n stochastic points in $\mathcal{T}$ each of which is associated with an existence probability. A *realization* of S refers to a random sample of S in which each point is sampled with its existence probability.

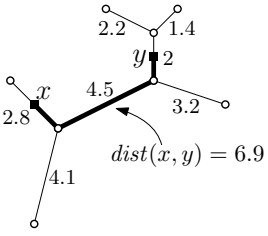

Fig. 1. A tree space and the unique simple path (in bold) between x and y

For the SCP problem, define $\kappa(S)$ as a random variable indicating the closest-pair distance of a realization of S. We first show that the ℓ-threshold probability of $\kappa(S)$ (i.e., the probability that $\kappa(S)$ is at least ℓ) can be computed in $O(t + n \log n + \min\{tn, n^2\})$ time for any given positive threshold ℓ. Based on this, we immediately obtain an $O(t+\min\{tn^3, n^4\})$-time algorithm for computing the expected closest-pair distance, i.e., the expectation of $\kappa(S)$. We then further show that one can approximate the expected closest-pair distance within a factor of $(1+\varepsilon)$ in $O(t+\varepsilon^{-1} \min\{tn^2, n^3\})$ time, by arguing that the expected closest-pair distance can be approximated via $O(\varepsilon^{-1}n)$ threshold probability queries.

For the LNN search, we first study the size of the the k-LVD $\Psi_{\mathcal{T}}^S$ of S on $\mathcal{T}$. A matching $O(n^2)$ upper bound for the worst-case size of $\Psi_{\mathcal{T}}^S$ is given. More interestingly, we show that (1) the worst-case size of $\Psi_{\mathcal{T}}^S$ is $O(kn)$, if the existence probabilities of the points in S are constant-far from 0; (2) the average-case size of $\Psi_{\mathcal{T}}^S$ is $O(kn)$, if the existence probabilities are i.i.d. random variables drawn from a fixed distribution. These results further imply the existence of an LVD data

structure which answers k-LNN queries in $O(\log n + k)$ time using average-case $O(t+k^2n)$ space, and worst-case $O(t+k^2n)$ space if the existence probabilities of the points are constant-far from 0. Finally, we give an $O(t+n^2 \log n+n^2k)$-time algorithm to construct such a data structure.

2 The stochastic closest-pair problem

Let $\mathcal{T}$ be a tree space represented by a t-vertex weighted tree T and $S = \{a_1, \ldots, a_n\} \subset \mathcal{T}$ be a set of stochastic points where a_i has an existence probability π_{a_i}. We use $\kappa(S)$ to denote the random variable indicating the closest-pair distance of a realization of S (if the realization is of size less than 2, we simply set its closest-pair distance to be 0).

2.1 Computing the threshold probability

We study the problem of computing the probability that $\kappa(S)$ is at least ℓ for a given threshold ℓ. We call this quantity the *ℓ-threshold probability* or simply *threshold probability* of $\kappa(S)$, and denote it by $C_{\geq \ell}(S)$. We show that $C_{\geq \ell}(S)$ can be computed in $O(t+n \log n+\min\{tn, n^2\})$ time. This result gives us an $O(t+n^2)$ upper bound for $t = \Omega(n)$ and an $O(n \log n + tn)$ bound for $t = O(n)$. In the rest of this section, we first present an $O(t + n^3)$-time algorithm for computing $C_{\geq \ell}(S)$, and then show how to improve it to achieve the desired bound. For simplicity of exposition, we assume $a_1, \ldots, a_n$ have distinct locations in $\mathcal{T}$.

An $O(t + n^3)$-time algorithm. In order to conveniently and efficiently handle the stochastic points in a tree space, we begin with a preprocessing step, which reduces the problem to a more regular setting.

Theorem 1. *Given $\mathcal{T}$ and S, one can compute in $O(t + n \log n)$ time a new tree space $\mathcal{T}' \subseteq \mathcal{T}$ represented by an $O(n)$-vertex weighted tree T' s.t. $S \subset \mathcal{T}'$ and every point in S is located at some vertex of T'.* (See [17] for a proof.)

By the above theorem, we use $O(t + n \log n)$ time to compute such a new tree space. Using this tree space as well as the $O(n)$-vertex tree representing it, the problem becomes more regular: every stochastic point in S is located at a vertex. We can further put the stochastic points in one-to-one correspondence with the vertices by adding dummy points with existence probability 0 to the "empty" vertices. In such a regular setting, we then consider how to compute the ℓ-threshold probability. For convenience, we still use T to denote the representation of the (new) tree space and $S = \{a_1, \ldots, a_n\}$ the stochastic dataset (though the actual size of S may be larger than n due to the additional dummy points, it is still bounded by $O(n)$). Since the vertices of T are now in one-to-one correspondence with the points in S, we also use a_i to denote the corresponding vertex of T.

As we are working on a tree space, a natural idea for solving the problem is to exploit the recursive structure of the tree and to compute $C_{\geq \ell}(S)$ in a recursive fashion. To this end, we need to define an important concept called

witness. We make T rooted by setting a_1 as its root. The subtree rooted at a vertex x is denoted by T_x. Also, we use $V(T_x)$ to denote the set of the stochastic points lying in T_x, or equivalently, the set of the vertices of T_x. The notations $\bar{p}(x)$ and $ch(x)$ are used to denote the parent of x and the set of the children of x, respectively (for convenience we set $\bar{p}(a_1) = a_1$).

Definition 1. *Let $dep(a_i)$ be the depth of a_i in T, i.e., $dep(a_i) = dist(a_1, a_i)$. For any a_i and a_j, we define $a_i \prec a_j$ if $dep(a_i) < dep(a_j)$, or $dep(a_i) = dep(a_j)$ and $i < j$. Clearly, the relation $\prec$ is a strict total order over S (also, over the vertices of T). For any subset $S' \subseteq S$ and any vertex a_i of T, we define the* ***witness*** *of a_i with respect to S', denoted by $\omega(a_i, S')$, as the smallest vertex in $V(T_{a_i}) \cap S'$ under the $\prec$-order. If $V(T_{a_i}) \cap S' = \emptyset$, we say $\omega(a_i, S')$ is not defined.* See Fig. 2 for an illustration of witness. We say a subset $S' \subseteq S$ is *legal* if the closest-pair distance of S' is at least ℓ.

The following lemma allows us to verify the legality of a subset by using the witnesses, which will be used later.

Lemma 1. *For any $S' \subseteq S$, we have S' is legal if and only if every point $a_i \in S \backslash \{a_1\}$ satisfies one of the following three conditions:*
(1) *$\omega(a_i, S')$ is not defined;*
(2) *$\omega(a_i, S') = \omega(\bar{p}(a_i), S')$;*
(3) *$dist(\omega(a_i, S'), \omega(\bar{p}(a_i), S')) \geq \ell$.*
We say that S' is ***locally legal*** *at a_i whenever a_i satisfies one of the above conditions.*

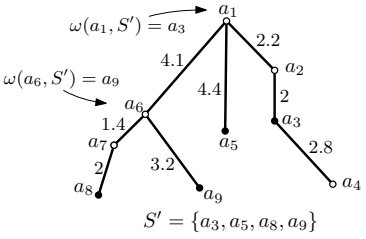

Fig. 2. An illustration of witness

In order to compute $C_{\geq \ell}(S)$, we define, for all $x \in S$ and $y \in V(T_{\bar{p}(x)})$,

$$P_y(x) = \begin{cases} \Pr\limits_{S' \subseteq_{\mathrm{R}} V(T_x)} [S' \text{ is legal and } \omega(x, S') = y] & \text{if } y \in V(T_x), \\ \Pr\limits_{S' \subseteq_{\mathrm{R}} V(T_x)} [S' \cup \{y\} \text{ is legal and } \omega(\bar{p}(x), S' \cup \{y\}) = y] & \text{if } y \in V(T_{\bar{p}(x)}) \backslash V(T_x). \end{cases}$$

Here the notation $\subseteq_{\mathrm{R}}$ means that the former is a realization of the latter, i.e., a random sample obtained by sampling each point with its existence probability. With the above, we immediately have that $C_{\geq \ell}(S) = \sum_{i=1}^{n} P_{a_i}(a_1) - P_0$, where P_0 is the probability that a realization of S contains exactly one point. We then show how $P_y(x)$ can be computed in a recursive way.

Lemma 2. *For $x \in S$ and $y \in V(T_x)$, we have that*

$$P_y(x) = Q \cdot \prod_{c \in ch(x)} P_y(c),$$

where $Q = \pi_x$ if $x = y$ and $Q = 1 - \pi_x$ if $x \neq y$.

Lemma 3. *For $x \in S$ and $y \in V(T_{\bar{p}(x)}) \backslash V(T_x)$, we have that*

$$P_y(x) = \prod_{a_i \in V(T_x)} (1 - \pi_{a_i}) + \sum_{z \in \Gamma} P_z(x),$$

where $\Gamma = \{z \in V(T_x) : y \prec z \text{ and } dist(z, y) \geq \ell\}$.

By the above two lemmas, the values of all $P_y(x)$ can be computed as follows. We enumerate $x \in S$ from the greatest to the smallest under $\prec$-order. For each x, we first compute all $P_y(x)$ for $y \in V(T_x)$ by applying Lemma 2. After this, we are able to compute all $P_y(x)$ for $y \in V(T_{\bar{p}(x)}) \backslash V(T_x)$ by applying Lemma 3. The entire process takes $O(n^3)$ time. Once we have the values of all $P_y(x)$, $C_{\geq \ell}(S)$ can be computed straightforwardly. Including the time for preprocessing, this gives us an $O(t + n^3)$-time algorithm for computing $C_{\geq \ell}(S)$.

In fact, we can further improve the runtime above to $O(t+n^2)$ by speeding up the computation of $P_y(x)$ for $y \in V(T_{\bar{p}(x)}) \backslash V(T_x)$ as they are the bottlenecks. In addition, if $t = O(n)$, we can even further reduce the runtime to $O(t + n \log n + \min(tn, n^2))$. Both optimizations are nontrivial and need new insights. However, due to the limited space, we leave these to [17] and conclude the following.

Theorem 2. *Given a weighted tree T with t vertices and a set S of n stochastic points in its tree space $\mathcal{T}$, one can compute the ℓ-threshold probability of the closest-pair distance of S, $C_{\geq \ell}(S)$, in $O(t + n \log n + \min\{tn, n^2\})$ time.*

2.2 Computing the expected closest-pair distance

Based on our algorithm for computing the threshold probability, we further study the problem of computing the expected closest-pair distance of S, i.e., the expectation of $\kappa(S)$. It is easy to see that our algorithm in Section 2.1 immediately gives us an $O(t+\min\{tn^3, n^4\})$-time algorithm to compute $\mathbf{E}[\kappa(S)]$. This is because the random variable $\kappa(S)$ has at most $\binom{n}{2}$ distinct possible values and hence we can compute $\mathbf{E}[\kappa(S)]$ via $O(n^2)$ threshold probability "queries" with various thresholds ℓ (note that after preprocessing our algorithm answers each threshold probability query in $O(\min\{tn, n^2\})$ time).

If we want to compute the exact value of $\mathbf{E}[\kappa(S)]$ (via threshold probability queries), $\Theta(n^2)$ queries are necessary in worst case. So it is natural to ask whether we can use less queries to approximate $\mathbf{E}[\kappa(S)]$. In the rest of this section, we show that one can use $O(\varepsilon^{-1}n)$ threshold probability queries to achieve a $(1+\varepsilon)$-approximation for $\mathbf{E}[\kappa(S)]$, which in turn gives us an $O(t + \varepsilon^{-1} \min\{tn^2, n^3\})$-time approximation algorithm for computing $\mathbf{E}[\kappa(S)]$.

For simplicity of exposition, we assume that the stochastic points in S are now one-to-one corresponding to the vertices of T (this is what we have after preprocessing). We begin with a simple case, in which the *spread* of T, i.e., the ratio of the length of the longest edge to the length of the shortest edge is bounded by some polynomial of n. In this case, to approximate $\mathbf{E}[\kappa(S)]$ is fairly easy, and we only need $O(\varepsilon^{-1} \log n)$ threshold probability queries.

Definition 2. *For $\beta > \alpha > 0$ and $\tau > 1$, the (α, β, τ)-**jump** is defined as*

$$J = \{\alpha, \tau\alpha, \tau^2\alpha, \ldots, \tau^k\alpha, \beta\},$$

where $\tau^k\alpha < \beta$ and $\tau^{k+1}\alpha \geq \beta$.

Let $d_{\min}$ be the length of the shortest edge of T and $d_{\max}$ be the sum of the lengths of all edges of T. Also, let J be the $(d_{\min}, d_{\max}, 1+\varepsilon)$-jump. Suppose $J =$

$\{\ell_1, \ldots, \ell_{|J|}\}$. Then we do $|J|$ threshold probability queries using the thresholds $\ell_1, \ldots, \ell_{|J|}$, and compute

$$E = \sum_{i=1}^{|J|} C_{\geq \ell_i}(S) \cdot (\ell_i - \ell_{i-1})$$

as an approximation of $\mathbf{E}[\kappa(S)]$ (where $\ell_0 = 0$). Note that $|J| = O(\log_{1+\varepsilon} \frac{d_{\max}}{d_{\min}}) = O(\log_{1+\varepsilon} n) = O(\varepsilon^{-1} \log n)$. It is easy to verify that $E \leq \mathbf{E}[\kappa(S)] \leq (1+\varepsilon)E$.

The problem becomes interesting when the spread of T is unbounded. In this case, although the above method still correctly approximates $\mathbf{E}[\kappa(S)]$, the number of the threshold probability queries is no longer well bounded. Imagine that the $O(n^2)$ possible values of $\kappa(S)$ are distributed as ℓ, $(1+\varepsilon)\ell$, $(1+\varepsilon)^2\ell$, etc. Then the $(d_{\min}, d_{\max}, 1+\varepsilon)$-jump J is of size $\Omega(n^2)$. Moreover, for guaranteeing the correctness, it seems that we cannot "skip" any element in J. However, as one will realize later, such an extreme situation can never happen. Recall that we are working on a weighted tree and the $O(n^2)$ possible values of $\kappa(S)$ are indeed the pairwise distances of the vertices of the tree. As such, these values are not arbitrary, and our insight here is to exploit the underlying properties of the distribution of these values.

Let $e_1, \ldots, e_{n-1}$ be the edges of T where e_i has the length (weight) w_i. Assume $w_1 \leq \cdots \leq w_{n-1}$. We define an index set $I = \left\{ m : \sum_{i=1}^{m-1} w_i < w_m \right\}$. Suppose $I = \{m_1, \ldots, m_k\}$ where $m_1 < \cdots < m_k$. Note that $m_1 = 1$. For convenience, we set $m_{k+1} = n$. We design our threshold probability queries as follows. Let J_i be the $(w_{m_i}, s_i, 1+\varepsilon)$-jump where $s_i = \sum_{j<m_{i+1}} w_j$, and $J = J_1 \cup \cdots \cup J_k$. Suppose $J = \{\ell_1, \ldots, \ell_{|J|}\}$ and set $\ell_0 = 0$. Similarly to the previous case, we do $|J|$ threshold probability queries using the thresholds $\ell_1, \ldots, \ell_{|J|}$, and compute

$$E = \sum_{i=1}^{|J|} C_{\geq \ell_i}(S) \cdot (\ell_i - \ell_{i-1})$$

as an approximation of $\mathbf{E}[\kappa(S)]$. We first verify the correctness, i.e., $E \leq \mathbf{E}[\kappa(S)] \leq (1+\varepsilon)E$. The fact $E \leq \mathbf{E}[\kappa(S)]$ can be easily verified. To see the inequality $\mathbf{E}[\kappa(S)] \leq (1+\varepsilon)E$, we define a piecewise-constant function $h : \mathbb{R}^+ \cup \{0\} \to [0,1]$ as

$$h(\ell) = \begin{cases} C_{\geq \ell_i}(S) & \text{if } (1+\varepsilon)\ell_i < \ell \leq (1+\varepsilon)\ell_{i+1}, \\ 0 & \text{if } \ell > (1+\varepsilon)l_{|J|}, \\ 1 & \text{if } \ell = 0. \end{cases}$$

Then it is clear that $(1+\varepsilon)E = \int_0^\infty h(\ell)d\ell$. We claim that $\int_0^\infty h(\ell)d\ell \geq \int_0^\infty C_{\geq \ell}(S)d\ell$, hence we have $\mathbf{E}[\kappa(S)] \leq (1+\varepsilon)E$. Note that the jumps $J_1, \ldots, J_k$ are disjoint and each of them contains a consecutive portion of the sequence $\ell_1, \ldots, \ell_{|J|}$. Furthermore, if ℓ_i and ℓ_{i+1} belong to different jumps, then there is no possible value of $\kappa(S)$ within the range (ℓ_i, ℓ_{i+1}), i.e., $C_{\geq \ell}(S)$ is constant when $\ell \in [\ell_i, \ell_{i+1})$. With this observation, it is not difficult to verify that $h(\ell) \geq C_{\geq \ell}(S)$ for any $\ell \geq 0$. Consequently, we have $\mathbf{E}[\kappa(S)] \leq (1+\varepsilon)E$, which implies the correctness of our method. Now the only thing remaining is to bound the number of the threshold probability queries, which we show in Lemma 4.

Lemma 4. *For each jump J_i, we have $|J_i| = O(\varepsilon^{-1}(m_{i+1} - m_i))$. As a result, the total number of the threshold probability queries, $|J|$, is $O(\varepsilon^{-1}n)$.*

Indeed, the above method can be extended to a much more general case, in which the stochastic dataset S is given in any metric space $\mathcal{X}$ (not necessarily a tree space). In this case, one can still define the threshold probability $C_{\geq \ell}(S)$ as well as the expected closest-pair distance $\mathbf{E}[\kappa(S)]$ in the same fashion. Our conclusion is the following.

Theorem 3. *Given a set S of n stochastic points in a metric space $\mathcal{X}$, one can $(1+\varepsilon)$-approximate the expected closest-pair distance of S, $\mathbf{E}[\kappa(S)]$, via $O(\varepsilon^{-1}n)$ threshold probability queries.* (See [17] for a proof.)

For the expected closest-pair distance in tree space, we can eventually conclude the following by plugging in our algorithm in Section 2.1 for computing $C_{\geq \ell}(S)$.

Corollary 1. *Given a tree space $\mathcal{T}$ represented by a weighted tree T with t vertices and a set S of n stochastic points in $\mathcal{T}$, one can compute a $(1+\varepsilon)$-approximation for the expected closest-pair distance of S, $\mathbf{E}[\kappa(S)]$, in $O(t + \varepsilon^{-1}\min\{tn^2, n^3\})$ time.*

3 The most-likely nearest-neighbor search problem

In this section, we study the k most-likely nearest-neighbor (k-LNN) search in a tree space. Again, let $\mathcal{T}$ be a tree space represented by a t-vertex weighted tree T and $S = \{a_1, \ldots, a_n\} \subset \mathcal{T}$ be the given stochastic dataset where the point a_i has an existence probability π_{a_i}. The k-LNN search problem can be defined as follows. Let $q \in \mathcal{T}$ be any point. For each $a_i \in S$, define $NNP_q(a_i)$ as the probability that the nearest-neighbor of q in a realization of S is a_i. Clearly, the nearest-neighbor of q in a realization is a_i iff a_i is in the realization and any point closer to q is not in the realization. Therefore, we have

$$NNP_q(a_i) = \pi_{a_i} \cdot \prod_{x \in \Gamma}(1 - \pi_x),$$

where $\Gamma = \{x \in S : dist(q, x) < dist(q, a_i)\}$. Given a query point $q \in \mathcal{T}$, the goal of the k-LNN search is to report the k-LNN of q, which is a k-sequence $(a_{i_1}, \ldots, a_{i_k})$ of points in S such that $NNP_q(a_{i_1}) \geq \cdots \geq NNP_q(a_{i_k}) \geq NNP_q(a_j)$ for all $j \notin \{i_1, \ldots, i_k\}$. For convenience, we assume $NNP_q(a_i) \neq NNP_q(a_j)$ for any $q \in \mathcal{T}$ and $a_i \neq a_j$ so that the k-LNN of any query point $q \in \mathcal{T}$ is uniquely defined.

A standard tool for nearest-neighbor search is the Voronoi diagram. In stochastic setting, we seek the most-likely Voronoi diagram (LVD), the concept of which is for the first time introduced in [14]. The k-LVD partitions the query space into connected cells such that points in the same cell have the same k-LNN. See [17] for an example (in color) of 1-LVD in a tree space.

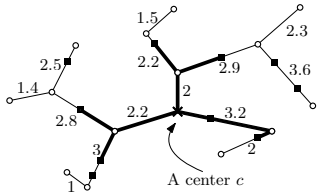

Fig. 3. A degree-3 center involving 5 points.

3.1 The size of the tree-space LVD

We use $\Psi_{\mathcal{T}}^S$ to denote the k-LVD of S on $\mathcal{T}$, i.e., the collection of the cells. Formally, $\Psi_{\mathcal{T}}^S$ can be defined as follows. For any k-sequence $\eta = (a_{i_1}, \ldots, a_{i_k})$, let Ψ_η be the set of the connected components of the subspace $\{q \in \mathcal{T} : \eta \text{ is the } k\text{-LNN of } q\}$. Then $\Psi_{\mathcal{T}}^S$ is the union of Ψ_η over all possible η. Clearly, the size of $\Psi_{\mathcal{T}}^S$ significantly influences the space efficiency of the LVD-based algorithm for k-LNN search. Let $m_{ij} \in \mathcal{T}$ be the "midpoint" of a_i and a_j, i.e., the midpoint of the path between a_i and a_j in $\mathcal{T}$. It is easy to see that the k-LNN only changes nearby these $\binom{n}{2}$ midpoints. However, this does not immediately imply that the size of $\Psi_{\mathcal{T}}^S$ is bounded by $O(n^2)$. The reason is that $O(n^2)$ points do not necessarily decompose $\mathcal{T}$ into $O(n^2)$ pieces (cells), unless these points are located only in the interiors of the edges. Note that throughout this section, we do not make any spatial assumption about the midpoints. In other words, it is allowed that different midpoints occupy the same location in $\mathcal{T}$, and some midpoints are located at the vertices of T. The reason why we allow this is explained in [17]. It is not surprising that even in such a general setting, the size of $\Psi_{\mathcal{T}}^S$ is still bounded by $O(n^2)$. We will see this later as a direct corollary of a technical result (Lemma 5).

Definition 3. *For any two midpoints m_{ij} and $m_{i'j'}$, we define $m_{ij} \equiv m_{i'j'}$ iff m_{ij} and $m_{i'j'}$ have the same location in $\mathcal{T}$ and $dist(a_i, m_{ij}) = dist(a_j, m_{ij}) = dist(a_{i'}, m_{i'j'}) = dist(a_{j'}, m_{i'j'})$. Clearly, $\equiv$ is an equivalence relation over the midpoints. We call the equivalence classes (under $\equiv$) **centers** of S and use $[m_{ij}]$ to denote the center that contains m_{ij}. A stochastic point $a_i \in S$ is said to be **involved** by a center c if $c = [m_{ij}]$ for some j. The **degree** of a center c, denoted by $deg(c)$, is defined as the number of the connected components of $\mathcal{T} \backslash \hat{c}$ that contain at least one point involved by c, where $\hat{c}$ denotes the point in $\mathcal{T}$ corresponding to c, and each such component is called a **branch** of c. A center c is said to be **critical** if $\hat{c}$ is not in the interior of any cell $C \in \Psi_{\mathcal{T}}^S$ and there exists at least one point involved by c that is in the k-LNN of $\hat{c}$.* (See Fig. 3 for an intuitive illustration of a center.)

Lemma 5. *Let Γ be the set of the critical centers and $\xi = \sum_{c \in \Gamma} deg(c)$. Then $|\Psi_{\mathcal{T}}^S| \leq \xi + 1$.*

The above lemma immediately gives us the $O(n^2)$ upper bound for the size of $\Psi_{\mathcal{T}}^S$. Indeed, a center c of S contains at least $\Omega(deg(c) \cdot m)$ midpoints, where m is the number of the points involved by c, so $\xi + 1$ is at most $O(n^2)$. Unfortunately, this upper bound is tight, following from the $\Omega(n^2)$ worst-case lower bound for

the size of the 1-dim 1-LVD given by [14] (note that the 1-dim LVD is a special case of the tree-space LVD). Surprisingly, we show that, if we make reasonable assumptions for the existence probabilities of the stochastic points or consider the average case, the size of $\Psi_{\mathcal{T}}^S$ is significantly smaller. Our results are:

- If the existence probabilities of all points in S are *constant-far from* 0, i.e., there is a fixed constant $\varepsilon > 0$ such that $\pi_{a_i} \geq \varepsilon$ for all $a_i \in S$, then the size of the k-LVD $\Psi_{\mathcal{T}}^S$ is $O(kn)$. Note that this assumption about the existence probabilities is natural and reasonable. In applications, an extremely small existence probability means the data point is highly unreliable. Such a point can be considered as a noise and removed from the dataset.
- The average-case size of the k-LVD $\Psi_{\mathcal{T}}^S$ is $O(kn)$. For the average-case analysis we assume that the existence probabilities of the points in S are i.i.d. random variables drawn from any fixed distribution (e.g., the uniform distribution among $[0, 1]$). In other words, we consider the expectation of $|\Psi_{\mathcal{T}}^S|$ when $\pi_{a_1}, \ldots, \pi_{a_n}$ are such random variables. The interesting point is that the $O(kn)$ upper bound is totally independent of the structure of $\mathcal{T}$ and the locations of the stochastic points. The randomness is only applied to the existence probabilities in our average-case analysis.

To prove these bounds requires new ideas. By Lemma 5, to bound the size of $\Psi_{\mathcal{T}}^S$, it suffices to bound the degree-sum of the critical centers. Intuitively, if a center c is far from the points it involves (compared with other points in S), then c is less likely to be critical, as the c-involved points are less likely to be in the k-LNN of $\hat{c}$. Along with this intuition, we define the following.

Definition 4. *For any center c, the **diameter** of c, denoted by $diam(c)$, is defined as the distance from $\hat{c}$ to the c-involved points. Let $A \subset \mathcal{T}$ be a finite set. We define the **depth** of c with respect to A as $dep_A(c) = |\{x \in A : dist(x, c) < diam(c)\}|$, i.e., the number of the points in A which are closer to c than the c-involved points.*

Our idea here is to first bound the "contribution" (degree-sum) of the "shallow" centers, and then further bound the degree-sum of the critical ones. Specifically, we investigate in Lemma 6 the degree-sum of the *d-shallow centers* of S, i.e., the centers of depth less than d with respect to S.

Lemma 6. *For $1 \leq d \leq n - 1$, the degree-sum of the d-shallow centers of S is at most $8dn$.*

Now we are ready to prove the $O(kn)$ bound for $|\Psi_{\mathcal{T}}^S|$ under the "constant-far from 0" assumption about the existence probabilities.

Lemma 7. *If the existence probabilities of the points in S are constant-far from 0, then a center of S is critical only if it is $O(k)$-shallow.*

Theorem 4. *If the existence probabilities of the points in S are constant-far from 0, then the size of the k-LVD $\Psi_{\mathcal{T}}^S$ is $O(kn)$.*

Proof. Suppose the existence probabilities $\pi_{a_1}, \ldots, \pi_{a_n}$ are constant-far from 0. Lemma 7 shows that all the critical centers of S are $O(k)$-shallow. By further applying Lemma 6, the degree-sum of the critical centers is $O(kn)$. Finally, by Lemma 5, the size of $\Psi_{\mathcal{T}}^{S}$ is $O(kn)$. □

To prove the bound for the average-case size requires more efforts. Let f be a *fixed* probability distribution function whose support is in $(0, 1]$ and μ be the supremum of the support of f. Define two constants $\mu_0 = \mu/(1+\mu)$ and $\lambda = 1 - \int_{-\infty}^{\mu_0} f(x)dx$. Clearly, if X is a random variable drawn from f, then $\lambda = \Pr[X > \mu_0]$. Note that λ is always positive by definition. The following lemma clarifies the meaning of μ_0.

Lemma 8. *Suppose $\pi_{a_1}, \ldots, \pi_{a_n}$ are i.i.d. random variables drawn from f. For any center c of S, the event "c is critical" does **not** happen if there are k (distinct) points $a_{i_1}, \ldots, a_{i_k}$ in S closer to $\hat{c}$ than the c-involved points such that $\pi_{a_{i_1}}, \ldots, \pi_{a_{i_k}}$ are all greater than μ_0.*

Theorem 5. *The average-case size of $\Psi_{\mathcal{T}}^{S}$ is $O(kn)$, given that the existence probabilities of the points in S are i.i.d. random variables drawn from a **fixed** distribution.*

Proof. Suppose the existence probabilities $\pi_{a_1}, \ldots, \pi_{a_n}$ are drawn independently from f. Lemma 8 implies that, if c is a center of S with $dep_S(c) = d \geq k$, then

$$\Pr[c \text{ is critical}] \leq u_d = \sum_{i=0}^{k-1} \binom{d}{i} \lambda^i (1-\lambda)^{d-i}.$$

Then by applying Lemma 5, we have

$$\mathbf{E}[|\Psi_{\mathcal{T}}^{S}|] \leq \sum_{c} \Pr[c \text{ is critical}] \cdot deg(c) \leq \sum_{c \in H_k} deg(c) + \sum_{d=k+1}^{n-1} \sum_{c \in H_d} (u_{d-1} - u_d) deg(c),$$

where H_d is the set of the d-shallow centers of S. Observe that

$$u_{d-1} - u_d = \binom{d-1}{k-1} \lambda^k (1-\lambda)^{d-k}.$$

Based on this and Lemma 6, we further have

$$\mathbf{E}[|\Psi_{\mathcal{T}}^{S}|] \leq 8kn + 8n \sum_{d=k+1}^{n-1} \binom{d-1}{k-1} \lambda^k (1-\lambda)^{d-k} d.$$

Note that

$$\sum_{d=k+1}^{n-1} \binom{d-1}{k-1} \lambda^k (1-\lambda)^{d-k} d = k \left(\frac{\lambda}{1-\lambda}\right)^k \sum_{d=k+1}^{n-1} \binom{d}{k} (1-\lambda)^d.$$

By an induction argument on k, it is not difficult to see that

$$\sum_{d=k+1}^{n-1} \binom{d}{k} (1-\lambda)^d < \sum_{d=k}^{\infty} \binom{d}{k} (1-\lambda)^d = \frac{(1-\lambda)^k}{\lambda^{k+1}}.$$

Finally, by combining the inequalities, $\mathbf{E}[|\Psi_{\mathcal{T}}^{S}|] \leq 8kn + \frac{8kn}{\lambda} = O(kn)$. □

3.2 Constructing LVD and answering queries

In this section, we show how to construct the k-LVD $\Psi_{\mathcal{T}}^S$ and use it to answer k-LNN queries. Let $e_1, \ldots, e_{t-1}$ be the edges of T. Assume each edge e_i has a specified "start point" s_i (which is one of its two endpoints) and the query point q is specified via a pair (i, δ), meaning the point on e_i with distance δ to s_i.

We first explain the data structure used for storing the k-LVD $\Psi_{\mathcal{T}}^S$ and answering queries. The LVD data structure is simple. First, it contains $|\Psi_{\mathcal{T}}^S|$ arrays (called *answer arrays*) each of which stores the k-LNN answer of one cell of $\Psi_{\mathcal{T}}^S$. This part takes $O(k|\Psi_{\mathcal{T}}^S|)$ space. In addition to that, we also need to record the structure of $\Psi_{\mathcal{T}}^S$. For each edge e_i of T, we use a sorted list L_i to store the "cell-decomposition" of e_i. Specifically, the intersection of each cell $C \in \Psi_{\mathcal{T}}^S$ and e_i is an "interval" (may be empty). These intervals are stored in L_i in the order they appear on e_i. Note that this part takes $O(t+|\Psi_{\mathcal{T}}^S|)$ space. Indeed, if an edge is decomposed into p pieces (intervals) by $\Psi_{\mathcal{T}}^S$, then it at least entirely contains $(p-2)$ cells of $\Psi_{\mathcal{T}}^S$ (so we can charge these $(p-2)$ pieces to the corresponding cells and the remaining two pieces to the edge). Therefore, the total space of the LVD data structure is $O(t + k|\Psi_{\mathcal{T}}^S|)$. To answer a query $q = (i, \delta)$, we first do a binary search in the list L_i to know which cell q locates in, and then use the answer array corresponding to the cell to output the k-LNN of q directly. The query time is clearly $O(\log |\Psi_{\mathcal{T}}^S| + k)$.

Next, we consider the construction of the LVD data structure. The first step of the construction is to compute all the centers of S and sort the centers in the interior of each edge e in the order they appear on e. We are able to get this done in $O(t + n^2 \log n)$ time (see [17]). After the centers are computed and sorted, we begin to construct the LVD data structure. Choose a vertex v of T. Starting at v, we do a walk in $\mathcal{T}$ along with the edges of T. The walk visits each edge of T exactly twice and finally goes back to v; see Fig. 4. During the walk, we maintain a (balanced) binary search tree for $NNP_x(a_1), \ldots, NNP_x(a_n)$ w.r.t. the current location x. By exploiting this BST, we can work out the cell-decomposition of each edge e_i (i.e., the sorted list L_i) at the first time we visit e_i in the walk. Specifically, we track the k-LNN when walking along with e_i, which can be obtained by retrieving the k largest elements from the BST. Whenever the k-LNN changes, a new cell of $\Psi_{\mathcal{T}}^S$ is found, so we need to create a new answer array to store the k-LNN information. Also, we need to update the sorted list L_i. In this way, after we go through e_i (for the first time), L_i is correctly computed. At the second time we visit e_i, we do nothing but maintain the binary search tree. When we finish the walk and go back to v, the construction of the LVD data structure is done. Clearly, in the process of the walk, we only need to maintain the binary search tree and retrieve the k-LNN when we arrive at (resp., leave from) a center of S from (resp., to) one of its branches. With a careful implementation and analysis (see [17]), we can complete the entire walk

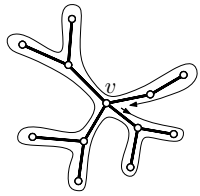

Fig. 4. A walk in tree visiting each edge exactly twice.

and hence the entire LVD structure in $O(t + n^2 \log n + n^2 k)$ time. Combined with the bounds in Section 3.1, we then have the following results.

Theorem 6. *Given a tree space $\mathcal{T}$ represented by a t-vertex weighted tree and a set S of n stochastic points in $\mathcal{T}$, one can construct in $O(t + n^2 \log n + n^2 k)$ time an LVD data structure to answer k-LNN queries in $O(\log n + k)$ time. The LVD data structure uses worst-case $O(t+kn^2)$ space and average-case $O(t+k^2 n)$ space. Furthermore, if the existence probabilities of the points in S are constant-far from 0, then the LVD data structure uses worst-case $O(t + k^2 n)$ space.*

References

1. Agarwal, P., Aronov, B., Har-Peled, S., Phillips, J., Yi, K., Zhang, W.: Nearest neighbor searching under uncertainty II. In: Proc. of the 32nd Sympos. on PODS. pp. 115–126. ACM (2013)
2. Agarwal, P., Cheng, S.W., Yi, K.: Range searching on uncertain data. ACM Transactions on Algorithms 8(4), 43 (2012)
3. Agarwal, P., Har-Peled, S., Suri, S., Yıldız, H., Zhang, W.: Convex hulls under uncertainty. In: Algorithms-ESA, pp. 37–48. Springer (2014)
4. Agarwal, P., Kumar, N., Sintos, S., Suri, S.: Range-max queries on uncertain data. In: Proc. of the 35th SIGMOD/PODS. pp. 465–476. ACM (2016)
5. Chen, J., Feng, L.: Efficient pruning algorithm for top-k ranking on dataset with value uncertainty. In: Proc. of the 22nd CIKM. pp. 2231–2236. ACM (2013)
6. Fakcharoenphol, J., Rao, S., Talwar, K.: Approximating metrics by tree metrics. ACM SIGACT News 35(2), 60–70 (2004)
7. Fink, M., Hershberger, J., Kumar, N., Suri, S.: Hyperplane separability and convexity of probabilistic point sets. In: Proc. of the 32nd SoCG. ACM (2016)
8. Ge, T., Zdonik, S., Madden, S.: Top-k queries on uncertain data: on score distribution and typical answers. In: Proc. of the 2009 SIGMOD. pp. 375–388. ACM (2009)
9. Huang, L., Li, J.: Approximating the expected values for combinatorial optimization problems over stochastic points. In: Intl. Colloquium on Automata, Languages, and Programming. pp. 910–921. Springer (2015)
10. Kamousi, P., Chan, T., Suri, S.: Stochastic minimum spanning trees in Euclidean spaces. In: Proc. of the 27th SoCG. pp. 65–74. ACM (2011)
11. Kamousi, P., Chan, T., Suri, S.: Closest pair and the post office problem for stochastic points. Computational Geometry 47(2), 214–223 (2014)
12. Kumar, N., Raichel, B., Suri, S., Verbeek, K.: Most likely Voronoi Diagrams in higher dimensions. In: LIPIcs-Leibniz International Proceedings in Informatics. vol. 65. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2016)
13. Löffler, M., van Kreveld, M.: Largest and smallest convex hulls for imprecise points. Algorithmica 56(2), 235–269 (2010)
14. Suri, S., Verbeek, K.: On the most likely Voronoi Diagram and nearest neighbor searching. In: ISAAC. pp. 338–350. Springer (2014)
15. Suri, S., Verbeek, K., Yıldız, H.: On the most likely convex hull of uncertain points. In: Algorithms–ESA, pp. 791–802. Springer (2013)
16. Xue, J., Li, Y., Janardan, R.: On the separability of stochastic geometric objects, with applications. In: Proc. of the 32nd SoCG. ACM (2016)
17. Xue, J., Li, Y.: Stochastic closest-pair problem and most-likely nearest-neighbor search in tree spaces. arXiv:1612.04890 (2016)