Nearest Neighbor Searching Under Uncertainty

Wuzhou Zhang *

RIP Proposal

1 Introduction

1.1 Motivation

Nearest neighbor searching, also known as post-office problem, was first posed by Knuth many years ago [16]: Given a set $S$ of $n$ points in $\mathbb{R}^2$, store it in a data structure such that for any query point $q \in \mathbb{R}^2$, we can efficiently find its nearest neighbor $p^* \in S$ satisfying $d(q, p^*) = \min_{p \in S} d(q, p)$.

The exact algorithm first computes the Voronoi diagram, then preprocesses it for point-location queries, and is optimal in $\mathbb{R}^2$. The Voronoi diagram of $S$, denoted by $\text{Vor}(S)$, where $S = \{p_1, p_2, \ldots, p_n\}$ is a set of $n$ distinct points in $\mathbb{R}^2$, is a subdivision of the plane into $n$ cells, one for each point in $S$. The cell of $p_i$ is defined as $\mathcal{V}(p_i) = \{q \mid d(q, p_i) \leq d(q, p_j), \forall 1 \leq j \leq n, j \neq i\}$, where $d(p, q)$ denotes the Euclidean distance between points $p$ and $q$ in $\mathbb{R}^2$. This problem has applications in various areas including pattern recognition, statistical classification, computer vision, databases, coding theory and data compression, etc. Data uncertainty is intrinsic and inevitable in various applications, e.g., due to measurement errors, uncertainty exists in sensor and RFID data [11, 19]. Therefore, it has both theoretical and realistic significance to investigate nearest neighbor searching under uncertainty.

In this proposal, we consider the nearest neighbor searching problem when the location of each point is represented by a probability density function.

1.2 Related Work

Voronoi diagram offers nearest neighbor searching an optimal solution in $\mathbb{R}^2$ with $O(n \log n)$ preprocessing time, $O(n)$ space and $O(\log n)$ query time.

---

*Wuzhou Zhang, Ph.D Student, Supervised by Pankaj K. Agarwal, Department of Computer Science, Duke University. Email: wuzhou@cs.duke.edu
Unfortunately, if one insists on logarithmic query time, the space and preprocessing time grow exponentially as the dimension grows. Alternatively, if one insists on linear or near-linear storage, the best running time bound for random input is $\min(2^{O(d)}, dn)$, which is essentially linear in $n$ even for small $d$. Best known method, based on ray shooting [3], requires $O((n/m^{1/(d/2)}) \text{polylog } n)$ query time for an $O(m)$-space structure ($n < m < n^{d/2}$). To obtain better performance, many researchers turned to approximate nearest neighbor searching: Given any $\varepsilon > 0$, a point $p$ is a $(1+\varepsilon)$-approximate nearest neighbor of $q$ if $d(q,p) \leq (1+\varepsilon)d(q,p^*)$, where $p^*$ is the actual nearest neighbor. Considering $d$ as a constant, Arya et al. gave an optimal algorithm for approximate nearest neighbor searching [6]: an $O(n)$-space structure, constructed in $O(n \log n)$ time, called the balanced box-decomposition (BBD) tree can find $(1+\varepsilon)$-approximate nearest neighbors in $O(\log n)$ time for any fixed $\varepsilon > 0$. Despite its optimality, the main drawback of this algorithm is that the actual query time is $O((1/\varepsilon)^{d} \log n)$, i.e., the constant factor $(1/\varepsilon)^{d}$ hidden in the big-Oh notation can be quite large for small $\varepsilon$ when $d \geq 3$. Arya et al. [5] also generalized space-time trade-offs for approximate nearest neighbor searching: Given a tradeoff parameter $\gamma$, where $2 \leq \gamma \leq 1/\varepsilon$, there exists a data structure of space $O(n^{\gamma d-1} \log(1/\varepsilon))$ that can answer queries in time $O(\log(n\gamma) + 1/(\varepsilon\gamma)^{(d-1)/2})$. Indyk et al. [4, 13] gave some near-optimal hashing algorithms based on the concept of locality-sensitive hashing (LSH). The key idea is to hash the points using several hash functions to ensure that for each function the probability of collision is much higher for points that are close to each other than for those that are far apart. Then, one can determine approximate nearest neighbors by hashing the query point and retrieving elements stored in buckets containing that point. In the survey about Voronoi diagram [7], both exact and approximate Voronoi diagrams were discussed.

Different models have been proposed for geometric computing on uncertain data: mainly classified into deterministic model and probabilistic model. In deterministic model, each point is assumed to be inside a given region. In [9], Cabello et al. studied the problem of spreading points, i.e., placing $n$ points, each one inside its own, prespecified disk, with the objective of maximizing the distance between the closest pair of them. He showed that the problem of spreading points is NP-hard and gave several approximation algorithms, e.g., in the $L_\infty$ metric, they gave a 2-approximation algorithm running in $O(n^{3/2} \log^2 n)$ time. He also approximated the problem of aligning points [10], i.e., aligning as many points as possible horizontally, vertically, or diagonally, when each point is allowed to be placed anywhere in
its own given region. Löffler et al. [18] studied largest bounding box, smallest diameter, and related problems on imprecise points. They showed that some problems have efficient algorithms and some not. For example, largest smallest bounding box can be computed in $O(n)$ time under the model of squares or discs, while largest width is NP-hard under the model of line segments. Kreveld et al. [20] showed that this uncertainty information can help to do faster triangulation after preprocessing.

Probabilistic model can be further classified into existential model and locational model. In existential model, each point is assumed to appear with certain probability. Suri et al. [14] showed that it is NP-hard to compute the probability that the closest pair distance is less than a given value $l$, even in $\mathbb{R}^2$ and $L_2$ norm, and they gave a linear-space data structure with $O(\log n)$ query time to compute the expected distance of a given query point to its $(1 + \varepsilon)$-approximate nearest neighbor when the dimension $d$ is a constant. They also showed that computing the expected length of the MST or the value of the max-flow is #P-Hard, but that for the MST it can be approximated within $O(\log n)$ factor for metric graphs [15]. In locational model, each point is assumed to be chosen from a known probability distribution. Agarwal et al. [1] provided a probably optimal $O(m^{3/2})$ upper bound for computing the skyline probabilities of all points in $\mathbb{R}^2$, and described some simple, near-linear time approximation algorithms for computing the probability of each point lying on the skyline. They also presented a linear-size index that optimally answers a query in $O(\log n + k)$ time for the fixed-threshold version of range queries on uncertain data [2], and obtained an index for the variable-threshold version of range queries on uncertain data with $O(n \log^2 n)$ space and $O(\log^3 n + k)$ query time.

Nearest neighbor searching under uncertainty is attracting more and more attention in the field of databases [8, 12, 17, 21]. Their methods were based on heuristics and failed to provide theoretical analysis. For example, Kriegel et al. [17] determined appropriate approximations of the object samples by means of clustering in order to improve the query performance, where clustered object representations are stored in R-tree.

## 2 Problem Statement

Let $S = \{P_1, P_2, \ldots, P_n\}$ be a set of $n$ uncertain points in $\mathbb{R}^2$. We assume that each uncertain point $P_i$ is described by a discrete or simple continu-
ous probability density function, e.g., normal distribution. For simplicity’s sake, in the following definitions we assume that each uncertain point $P_i$ is described by a discrete probability density function defined over $k$ discrete points (for an input parameter $k$). Namely, $P_i = \{p_{i,1}, p_{i,2}, \cdots, p_{i,k}\} \subset \mathbb{R}^2$, and the probability of $P_i$ being at location $p_{i,j}$ is $0 \leq w_{i,j} \leq 1$; $\sum_{j=1}^{k} w_{i,j} = 1$.

Given a query point $q$, the expected distance from $q$ to $P_i$ is defined as

$$Ed(q, P_i) = \sum_{j=1}^{k} w_{i,j} d(q, p_{i,j}).$$

The probability that $p_{i,j}$ is the nearest neighbor of $q$ can be formulated as

$$\varphi(q,p_{i,j}) = \prod_{1 \leq t \leq n, t \neq i} w_{i,j} \sum_{u_t \in \eta(u_t)} w_{t,u_t} = \prod_{1 \leq t \leq n, t \neq i} w_{i,j} \left(1 - \sum_{u_t \in \vartheta(u_t)} w_{t,u_t}\right),$$

where $\vartheta(u_t) = \{v \mid d(q, p_{t,v}) < d(q, p_{i,j})\}$ and $\eta(u_t) = \{1, 2, \cdots, k\} - \vartheta(u_t)$. Then the probability that $P_i$ is the nearest neighbor of $q$ is

$$\Phi(q, P_i) = \sum_{j=1}^{k} \varphi(q,p_{i,j}).$$

We are asked to find its expected nearest neighbor, i.e., an uncertain point $P^* \in S$ that is closest to $q$ in expectation,

$$Ed(q, P^*) = \min \{Ed(q, P) : P \in S\},$$

and its nearest neighbor with highest probability, i.e., an uncertain point $P^* \in S$ that is most likely closest to $q$,

$$\Phi(q, P^*) = \max \{\Phi(q, P) : P \in S\}.$$

## 3 Proposed Approaches

For the nearest neighbor in expectation, define the expected Voronoi diagram of $S$, denoted by $\mathcal{EV}(S)$, as a subdivision of the plane into $n$ cells, one for each uncertain point in $S$. The expected cell of $P_i$ is defined as

$$\mathcal{EV}(P_i) = \{q \mid Ed(q, P_i) \leq Ed(q, P_j), \forall 1 \leq j \leq n, j \neq i\}.$$ 

Unfortunately, the bisector of $P_i$ and $P_j$ is not a line any more. For instance, assume we only have two points $P$ and $P'$ with probability distributions: $f_P((1,0)) = f_P((4,1)) = \frac{1}{2}$, $f_{P'}((3,2)) = 1$. We get a curve as the “bisector”:

$$4x^3 + 12x^2y - 49x^2 + 4xy^2 - 94xy + 196x + 12y^2 - 65y^2 + 236y - 272 = 0.$$
(see fig 1a) by solving the equation

\[ \frac{1}{2} \sqrt{(x - 1)^2 + (y - 0)^2} + \frac{1}{2} \sqrt{(x - 4)^2 + (y - 1)^2} = \sqrt{(x - 3)^2 + (y - 2)^2}. \]

The “bisector” is even more complicated if \( f_{P'}((3, 2)) = 1 \) is changed to

\[ f_{P'}((-1, 1)) = \frac{1}{5}, \quad f_{P'}((3, 2)) = \frac{1}{3}, \quad f_{P'}((8, -1)) = \frac{1}{2}. \]

We might need to use techniques shown in [5] to approximate the nearest neighbor. Moreover, the Voronoi cell can be disconnected (e.g., \( f_P((0, 0)) = f_P((4, 0)) = \frac{1}{2}, \quad f_P((2, 1)) = 1, \quad f_{P'}((2, -1)) = 1 \). See figure 1b), thus the complexity of the Voronoi diagram can be quite worse. When the number of Voronoi cells becomes quite large, a question one might ask is that given linear space for approximate the nearest neighbor, can one achieve \( O(\sqrt{n}) \) query time? Fortunately, it can be shown that the bisector of two uncertain points represented by normal distributions is still a line, thus the exact algorithm for nearest neighbor searching in \( \mathbb{R}^2 \) can be employed.

Figure 1: The bisector and Voronoi cell of Voronoi diagram in expectation.

For the nearest neighbor with highest probability, we define the Voronoi diagram with highest probability of \( S \), denoted by HVor(\( S \)), as a subdivision of the plane into \( n \) cells, one for each uncertain point in \( S \). The cell of \( P_i \) with highest probability is defined as

\[ \text{HV}(P_i) = \{ q \mid \Phi(q, P_i) \geq \Phi(q, P_j), \forall 1 \leq j \leq n, \ j \neq i \}. \]

One can show that the complexity of HVor(\( S \)) is \( O(n^4k^4) \). Lower bound of the complexity of HVor(\( S \)) needs to be established. If the exact solution is time-consuming, we might use techniques shown in [5] to approximate it.
References


