Journal of Algorithms and Computation

journal homepage: http://jac.ut.ac.ir

On the expected weight of the theta graph on uncertain points

Behnam Iranfar*[1] and Mohammad Farshi†[2]

[1,2]Combinatorial and Geometric Algorithms Lab, Department of Mathematical Sciences, Yazd University, Yazd, Iran

ABSTRACT

Given a point set $S \subset \mathbb{R}^d$, the θ-graph of S is as follows: for each point $s \in S$, draw cones with apex at s and angle θ and connect s to the point in each cone such that the projection of the point on the bisector of the cone is the closest to s. One can define the θ- graph on an uncertain point set, i.e. a point set where each point s_i exists with an independent probability $\pi_i \in (0, 1]$. In this paper, we propose an algorithm that computes the expected weight of the θ-graph on a given uncertain point set. The proposed algorithm takes $O(n^2\alpha(n^2, n)^{2d})$ time and $O(n^2)$ space, where n is the number of points, d and θ are constants, and α is the inverse of the Ackermann's function.

ARTICLE INFO

Article history:
Received 19, August 2019
Received in revised form 15, April 2020
Accepted 14 May 2020
Available online 01, June 2020

Keyword: uncertain points , expected weight

AMS subject Classification: 68W05, 68W40

1 Introduction

In many applications, such as sensor databases, mobile object tracking, computer vision or location-based services, the existence or location of the data is uncertain, but we can use statistical information. There are several models for uncertain data. Some of them

*biranfar@gmail.com
†Corresponding author:M. Farshi. Email: mfarshi@yazd.ac.ir

assign an area to each point which represents the area that the point resides, but the exact position of the point in its corresponding area is unknown. In the tuple model of uncertain data, each input point has a fixed location but it only exists probabilistically. The input is a pair (S, Π) such that $S = \{s_1, s_2, \dots, s_n\}$ is a set of n points (sites) in $\mathbb{R}^d$, and $\Pi = \{\pi_1, \pi_2, \dots, \pi_n\}$ is a probability vector with the interpretation of that sites.
The geometric structures on uncertain data are also interesting because they have wide application in solving problems. Theta graphs have many applications, including wireless networking [4], motion planning [6], real-time animation [8], and minimum-spanning tree construction [19]. The properties of a geometric graph influence the time and space complexity of the problem that uses the structure. For example, if one wants to construct a network on a set of points, then the total cost of the network, i.e. the sum of all edge weights of the graph, is an important parameter. For uncertain data, the properties of the structures on the data are not deterministic, but one can compute the expected properties of the structures on uncertain data, see for example [9, 13].

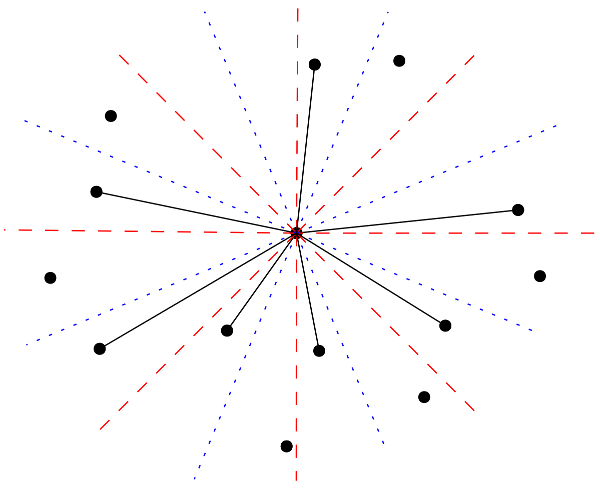


Figure 1: One step of constructing θ-graph, for $\theta = \pi/4$. Dashed red lines are borders of cones and dotted blue lines are ℓ_c.

For building the θ-graph for $S \subset \mathbb{R}^d$ and a given angle θ, divide the space around each point $p \in S$ into a set of cones $\mathcal{C}$ of maximum angle θ. The cones constructed by rotating the horizontal line through p around the point p by angle $i\theta$, where i is a natural number between 0 and $2\pi/\theta$. For each cone $c \in C$ we consider a line ℓ_c that passes through p and inside the cone c. Then for each cone $c \in \mathcal{C}$, the point p is connected to a point $q \in S$ that lies in cone c and minimizes the Euclidean distance between p and the projection of q into the line ℓ_c of c, see Figure 1. For details see [14, Chapter 4]. The visualization of the algorithm is available at [7], and one can see the steps of computing the θ-graph of a point set given by the user.
In this paper, we study the problem of computing the expected weight of the θ-graph of a given uncertain points [6, 10, 11]. The problem is very complicated and to the best of our knowledge, no theoretical results have been reported. We propose an algorithm that com-

putes the expected weight of the θ-graph on a given uncertain point set in $O(n^2\alpha(n^2, n)^{2d})$ time and $O(n^2)$ space, where n is the number of points, d and θ are constants, and α is the inverse of the Ackermann's function.

1.1 Related work.

Uncertainty has been studied in some articles in computational geometry. In 2013, Suri *et al.* [16] have studied the most likely convex hull under the uncertain points and showed that the most likely convex hull under the point model (tuple model) can be computed in $O(n^3)$ time in $d = 2$ dimension, but it is NP-hard for $d \geq 3$. Agarwal *et al.* [2], in 2014, have studied the problem of computing probability of query point lying inside the convex hull, and their results have included both approximation and exact algorithm for given uncertain points, their main result is an $O(n^d)$-time algorithm for computing the probability of convex hull membership. Xue *et al.* [18] investigated several computational problem related to the stochastic convex hull. For diameter, they established the first deterministic 1.633-approximation algorithm with a complexity polynomial of time in both n and d. In 2015, Zhang [20] have formulated two different nearest neighbors on uncertain points: the expected nearest neighbor, where the expected distance between each input point and a query point has been considered, and the probabilistic nearest neighbor.
Agarwal *et al.* [1] studied answering rang query over uncertain data. They build the index on a collection of point S in $\mathbb{R}$ (each represented by its probability density function) with linear or near-linear size, where can report all point of S that lie in given interval I with probability at least τ in logarithmic time. In [3], many algorithms were presented for building an index on S so that for a d-dimensional query rectangle ρ, the expected maximum value or the most-likely maximum value in ρ can be computed quickly.

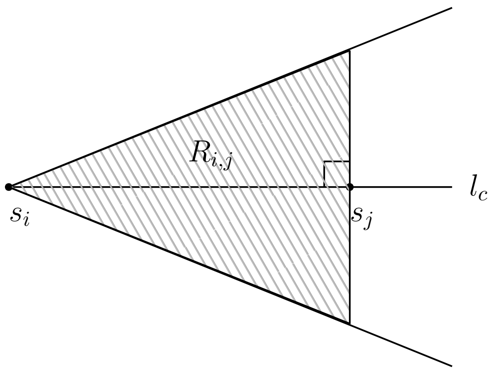
2 The expected weight of the θ-graph

In this section, we will describe an algorithm for computing the expected weight of the θ-graph (EWTG) of n points in $\mathbb{R}^d$ under uncertainty. The algorithm is similar to the algorithm of building the (deterministic) θ-graph.
Let (S, Π) denote the uncertain points in d-dimensional space. For computing EWTG, we must calculate portability of existing each edge and multiply it to its length. In other words,

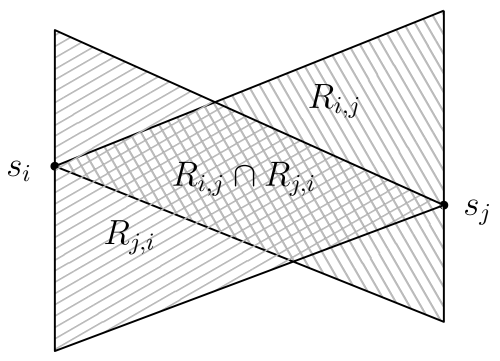
$$\mathrm{EWTG}(S) = \sum_{s_i, s_j \in S,\ i<j} |s_i s_j| \times \pi_{i,j}, \tag{1}$$

where $\pi_{i,j}$, for all $s_i, s_j \in S$, is the probability of having the edge (s_i, s_j) in the θ-graph.
Consider two points $s_i, s_j \in S$, $i \neq j$ in $\mathbb{R}^d$. Let c be the cone with apex s_i that include s_j. We add a half plane to c, where this half plane determine by s_j and orthogonal vector $(-1) \times l_c$, we denote this region by $R_{i,j}$ (see Fig. 2(a)).

Observation 2.1. *The edge between two points s_i and s_j exists if and only if*

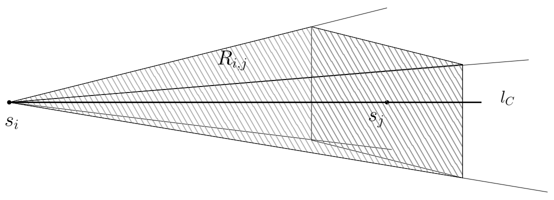


(a)

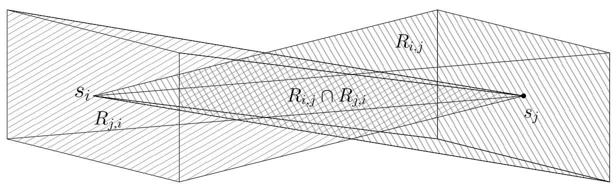


(b)

Figure 2: Illustration of $R_{i,j}$ and $R_{i,j} \cap R_{j,i}$ in the plane.



(a)



(b)

Figure 3: Illustration of $R_{i,j}$ and $R_{i,j} \cap R_{j,i}$ in 3-dimensional.

1. s_i and s_j exist.

2. There is no point in $R_{i,j}$ or $R_{j,i}$ (see Fig. 2(b)).

More formally,

$$\pi_{i,j} = \pi_i \times \pi_j \times \left[\prod_{s_m \in R_{i,j}} \overline{\pi}_m + \prod_{s_m \in R_{j,i}} \overline{\pi}_m - \prod_{s_m \in R_{i,j} \cap R_{j.i}} \overline{\pi}_m \right], \tag{2}$$

where $\overline{\pi}_m = (1 - \pi_m)$. Note that the last sentence subtracted because the points in $R_{i,j} \cap R_{j,i}$ considered twice in the previous terms of the equation.
By Equations (1) and (2), we have

$$\text{EWTG}(S) = \sum_{s_i, s_j \in S,\ i<j} \left[|s_i s_j| \times \pi_i \times \pi_j \times \left[\prod_{s_m \in R_{i,j}} \overline{\pi}_m + \prod_{s_m \in R_{j,i}} \overline{\pi}_m - \prod_{s_m \in R_{i,j} \cap R_{j.i}} \overline{\pi}_m \right] \right]. \tag{3}$$

First, we describe an algorithm for computing

$$\sum_{s_i, s_j \in S,\ i<j} \left[|s_i s_j| \times \pi_i \times \pi_j \times \left[\prod_{s_m \in R_{i,j}} \overline{\pi}_m + \prod_{s_m \in R_{j,i}} \overline{\pi}_m \right] \right]. \tag{4}$$

For each cone $c \in \mathcal{C}_\kappa$, points are sorted based on the orthogonal projection onto l_c. Obviously, for each two points $s_i, s_j \in S$, if $i \geq j$, then $R_{i,j}$ is empty, so we only need to compute $R_{i,j}$, for $i < j$. Consider a cone $c_i \in \mathcal{C}_\kappa$. Let c_{i,s_j} be the cone c_i transfered to a point $s_j \in S$.

Algorithm 1:

```
Input  : Uncertain point set $(S, \Pi)$
Output: $\sum_{s_i,s_j \in S,\ i<j} \left[|s_i s_j| \times \pi_i \times \pi_j \times \left[\prod_{s_m \in R_{i,j}} \overline{\pi}_m + \prod_{s_m \in R_{j,i}} \overline{\pi}_m\right]\right]$
1  $Sum = 0$;
2  for any cones $c_i$, $1 \leq i \leq \kappa$ do
3      Project all points to $l_{c_i}$;
4      Sort the points based on its position on $l_{c_i}$;
5      for $j = 1$ to $n - 1$ do
6          $\mu = \pi_j$;
7          for $k = j + 1$ to $n$ do
8              if $s_k \in c_{s_j}$ then
9                  $Sum = Sum + \mu \times \pi_k \times |s_j s_k|$;
10                 $\mu = \mu \times \overline{\pi}_k$;
11 return $Sum$;
```

Lemma 2.2. *Algorithm 1 computes the first part of Equation (4) in $O(\kappa n^2)$ time and $O(n)$ space.*

Proof. Algorithm 1 spends $O(n)$ time for line 3 and $O(n \log n)$ time for line 4 and they are repeated $O(\kappa)$ times. It also spends $O(1)$ time for lines 8 to 10, and these are repeated $O(\kappa n^2)$ times. So, Algorithm 1 requires $O(\kappa n^2)$ time. We only save n (projection) points, probability vector, Sum and μ. So, the algorithm needs $O(n)$ space. □

Now, we describe an algorithm for computing the last part of Equation (4), i.e.

$$\sum_{s_i,s_j \in S,\ i<j} \left[|s_i s_j| \times \pi_i \times \pi_j \times \prod_{s_m \in R_{i,j} \cap R_{j,i}} \overline{\pi}_m\right]. \tag{5}$$

Definition 2.3. *(Partial sum query) Given a d-dimensional array A with n entries from a semigroup, a partial sum query problem is the problem of given a d-dimensional query rectangle $\gamma = [a_1, b_1] \times \cdots \times [a_d, b_d]$, it computes*

$$\sigma(A, \gamma) = \sum_{(x_1,x_2,\ldots,x_d) \in \gamma} A[x_1, x_2, \ldots, x_d].$$

The partial-sum query problem is a special case of the classic orthogonal range searching problem.

To convert computing $\prod_{s_m \in R_{i,j} \cap R_{j,i}} \overline{\pi}_m$ to a partial-sum query problem, we do the following:

1. Any region $R_{i,j} \cap R_{j,i}$ is converted to a d-dimensional rectangle.
2. The formula $\prod_{s_m \in R_{i,j} \cap R_{j,i}} \overline{\pi}_m$ converts to an operator in a semigroup.

Each cone of $\mathcal{C}_\kappa$ have $f = 2^{d-1}$ faces. Let d_i^j be the line passing through the origin that is orthogonal to the j-th face of the cone c_i. We define $\mathcal{D} = \{d_i^j \ : \ for\ 1 \leq i \leq \kappa\ and\ 1 \leq j \leq f\}$. These lines define a coordinate system that can be used to compute $\prod_{s_m \in R_{i,j} \cap R_{j,i}} \overline{\pi}_m$. or converting each region $R_{i,j} \cap R_{j,i}$, for $1 \leq i, j \leq n$ and $i \neq j$, to a rectangle, we only need to project all of point in S onto lines in $\mathcal{D}$. New coordinates have at most

$$f \times \kappa = 2^{d-1} \times 2dm^{d-1} = O(2^d d^{(d+1)/2} (\pi/\theta)^{d-1})$$

dimensions.
We assign the value $\overline{\pi}_i$ to $A[s_i]$, for all $s_i \in S$. This partial-sum query can be answered by choosing the semigroup to be $(\mathbb{R}, \times)$, where $\times$ denotes the standard real number multiplication.

Algorithm 2:

```
Input  : Uncertain point set $(S, \Pi)$
Output: $\sum_{s_i, s_j \in S,\ i<j} \left[ |s_i s_j| \times \pi_i \times \pi_j \times \prod_{s_m \in R_{i,j} \cap R_{j,i}} \overline{\pi}_m \right]$
1 $Sum = 0$;
2 for $i = 1$ to $n$ do
3     Transform point $s_i$ to the new coordinate system;
4 for $i = 1$ to $n - 1$ do
5     for $j = i + 1$ to $n$ do
6         $\sigma(A, \gamma)$ = the partial-sum for $\gamma = R_{i,j} \cap R_{j,i}$;
7         $Sum = Sum + |s_i s_j| \times \pi_i \times \pi_j \times \sigma(A, \gamma)$ ;
8 return $Sum$;
```

Theorem 2.4. *[5] Given a semigroup of n variables in $\mathbb{R}^d$ and $k \geq 14^d$, there is a scheme that computes any d-dimensional rectangle query in $O(\alpha(kn, n)^d)$ time. Preprocessing this scheme is used to k cells per variable and can be constructed in time proportional to its size.*

The function $\alpha(.,.)$ is the inverse of Ackermann's function defined by Tarjan [17].

Lemma 2.5. *Algorithm 2 computes Equation (5) in $O(n^2 \alpha(n^2, n)^{2^d d^{(d+1)/2} (\pi/\theta)^{d-1}})$ time and $O(n^2)$ space.*

Proof. Algorithm 2 uses $O(n)$ time for line 3, and the second loop repeated $O(n^2)$ times. Line 5 uses $O(\alpha(n^2,n)^{2^d d^{(d+1)/2}(\pi/\theta)^{d-1}})$ time and therefore this line uses $O(n^2\alpha(n^2,n)^{2^d d^{(d+1)/2}(\pi/\theta)^{d-1}})$ time, which dominates the time complexity of the other parts of the algorithm.
Algorithm 2 has to save n points, array A and the partial-sum queries. Since it performs a query for each edge, the algorithm needs $O(n^2)$ space. □

Theorem 2.6. *Let S be a set of n uncertain points in $\mathbb{R}^d$. The expected weight of the θ-graph on S can be computed in $O(n^2\alpha(n^2,n)^{2^d d^{(d+1)/2}(\pi/\theta)^{d-1}})$ time using $O(n^2)$ space.*

3 Improve the running time of the algorithm

In Section 2, for building set $\mathcal{C}$ of cones, we consider a hypercube $H = [-1,1]^d$ with $2d$ faces. The each face of H is partitioned to $(d-1)$-dimensional hypercubes with side length $\frac{2}{m}$, where m is $\left\lceil \sqrt{\frac{2(d-1)}{1-\cos\theta}} \right\rceil$. Each partition is called a subhypercube. These cones are not *simplicial* because of subhypercubes were having $2d-1$ vertices define them.
In this section, we want to decrease the number of dimensional partial-sum query and improve the running time complexity of Algorithm 2. First, we partition the cones to simplex-cones such that each cone has d faces. Then, we put the each region generated by the simplex-cones to a group represented by a simplex-cone.
For example, if we partition the cone c_{s_i} that contains point s_j to two cones, then the region $R_{i,j}$ is partitioned to two region $R^1_{i,j}$ and $R^2_{i,j}$ (see Fig. 4).

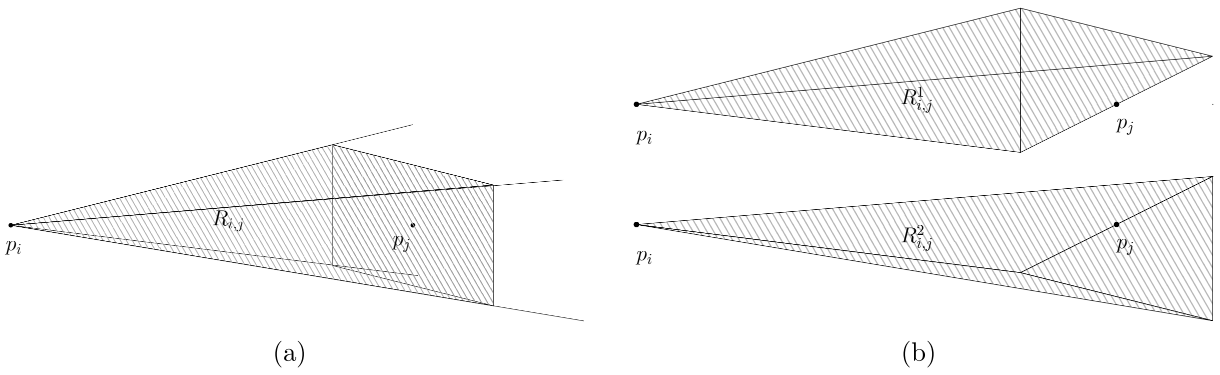


Figure 4: Illustration of $R_{i,j}$, $R^1_{i,j}$ and $R^2_{j,i}$ in $\mathbb{R}^3$.

Simply, it is observed that (see Fig. 5)

$$R_{i,j} \cap R_{j,i} = (R^1_{i,j} \cap R^1_{j,i}) \cup (R^1_{i,j} \cap R^2_{j,i}) \cup (R^2_{i,j} \cap R^1_{j,i}) \cup (R^2_{i,j} \cap R^2_{j,i})$$

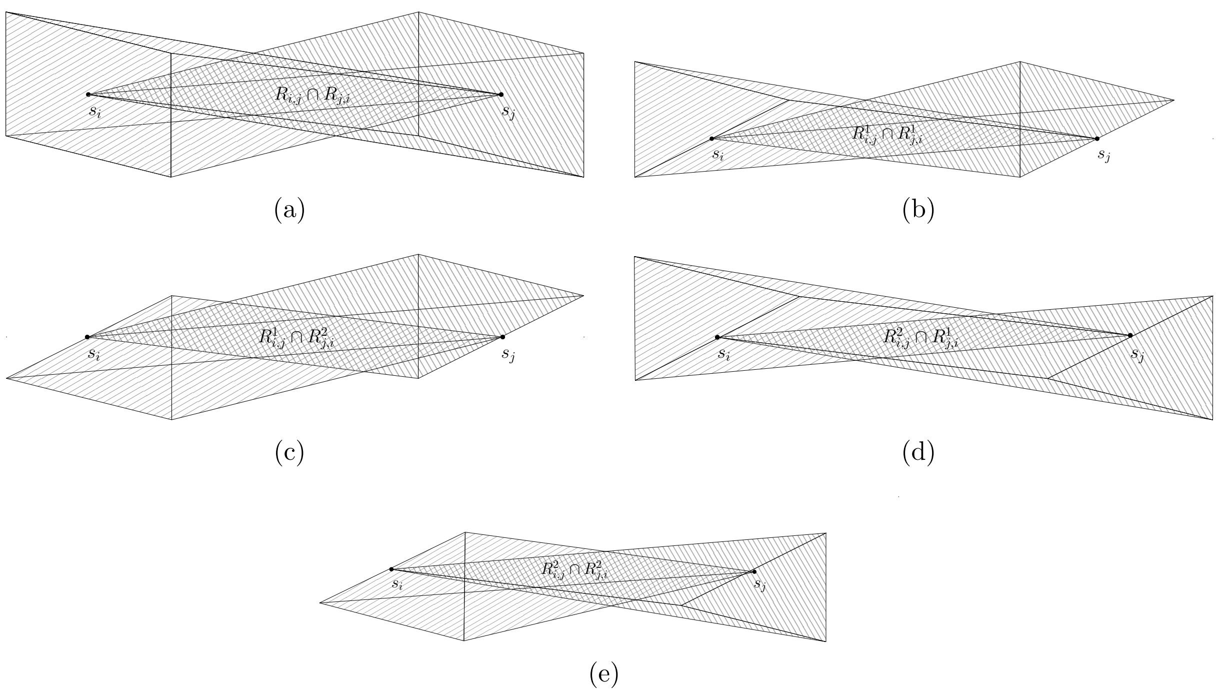

s_i
$R_{i,j} \cap R_{j,i}$
s_j
(a)
s_i
$R^1_{i,j} \cap R^1_{j,i}$
s_j
(b)
s_i
$R^1_{i,j} \cap R^2_{j,i}$
s_j
(c)
s_i
$R^2_{i,j} \cap R^1_{j,i}$
s_j
(d)
s_i
$R^2_{i,j} \cap R^2_{j,i}$
s_j
(e)

Figure 5: Illustration of $R_{i,j}$ and $R_{i,j} \cap R_{j,i}$ in the plane.

and thus

$$\begin{aligned}\prod_{s_m \in R_{i,j} \cap R_{j,i}} \overline{\pi}_m &= \left[\prod_{s_m \in R^1_{i,j} \cap R^1_{j,i}} \overline{\pi}_m\right] \times \left[\prod_{s_m \in R^1_{i,j} \cap R^2_{j,i}} \overline{\pi}_m\right] \times \\ &\times \left[\prod_{s_m \in R^2_{i,j} \cap R^1_{j,i}} \overline{\pi}_m\right] \times \left[\prod_{s_m \in R^2_{i,j} \cap R^2_{j,i}} \overline{\pi}_m\right].\end{aligned}$$

Generally, a d-dimensional hypercube can be triangulation into $d!$ d-simplices with disjoint interiors [12].

Let $V = \{v^0, v^1, \ldots, v^\beta\}$, $0 \leq \beta \leq d$, be a set of $\beta + 1$ points in $\mathbb{R}^d$. If the vectors $v^i - v^0$, $1 \leq i \leq \beta$, are linearly independent, then the convex hull of V is called a β-simplex.

Consider the collection $\mathcal{C}_\kappa = \{c_1, c_2, \ldots, c_k\}$ of cones in Section 2. Each cone c_i in $\mathcal{C}_\kappa$ generated by V_i, where V_i is the vertex set of a $(d-1)$-dimensional hypercube that is contained in one of the $2d$ hypercube $x_1 = 1$, $x_1 = -1$, $x_2 = 1$, $x_2 = -1$, …, $x_d = 1$, $x_d = -1$. This hypercube can be triangulated into $(d-1)!$ many $(d-1)$-simplices $\Delta^1_i, \Delta^2_i, \ldots, \Delta^{(d-1)!}_i$, that are all contained in the same hyperplane as V_i. We define

$$\mathcal{C}_{\kappa_s} = \{c^j_i : for\ 1 \leq i \leq \kappa, 1 \leq j \leq (d-1)!\},$$

where $\kappa_s = 2d!\lceil\sqrt{2(d-1)/(1-\cos\theta)}\rceil^{d-1}$. Since $\kappa = d^{(d+1)/2}(\pi/\theta)^{d-1}$, we have

$$\kappa_s \leq \kappa d^{d-1} = d^{(3d-1)/2}(\pi/\theta)^{d-1}.$$

The collection $\mathcal{C}_{\kappa_s}$ consist κ_s simplicial cones that cover $\mathbb{R}^d$, and that all have their apex at the origin. If d is a constant, then $\mathcal{C}_\kappa$ can be constructed in $O(1/\theta^{d-1})$ time and consists of $\kappa = O(1/\theta^{d-1})$ cones with disjoint interiors [14].
Since the collection $\mathcal{C}_{\kappa_s}$ has at most $d^{(3d-1)/2}(\pi/\theta)^{d-1}$ cones, if we only consider the collection of the vector perpendicular to every face, then we have at most $(d^{(3d-1)/2}(\pi/\theta)^{d-1})^2$ different group regions $R^k_{i,j} \cap R^l_{j,i}$, for all $s_i, s_j \in S, i < j$ and $1 \le k, l \le (d-1)!$.

Observation 3.1. *Let s_i and s_j be two points in S. We have*

$$\prod_{s_m \in R_{i.j} \cap R_{j,i}} \overline{\pi}_m = \prod_{k=1}^{(d-1)!} \left[\prod_{l=1}^{(d-1)!} \prod_{s_m \in R^k_{i.j} \cap R^l_{j,i}} \overline{\pi}_m \right].$$

Similar to Section 2, we will convert $\prod_{s_m \in R^k_{i,j} \cap R^l_{j,i}} \overline{\pi}_m$ to a partial-sum query. First, all regions $R^k_{i,j} \cap R^l_{j,i}$, for all $s_i, s_j \in S$, $i < j$ and $1 \le k, l \le (d-1)!$, are grouped base of lines through the origin that are orthogonal to its face, we denote the i-th group of regions by G_i and the number of groups by q. Next, for each region, let $D_1, D_2, \ldots, D_f$ be the lines through the origin that are orthogonal to the face of this region, where f is equal to $2d$. These lines define the coordinate system that can be used to compute $\prod_{s_m \in R_{i,j} \cap R_{j,i}} \overline{\pi}_m$. We define the function *Source*() such that

$$Source(R^k_{i,j} \cap R^l_{j,i}) = R_{i,j} \cap R_{j,i},$$

for any $1 \le i < j \le n$ and $1 \le k, l \le (d-1)!$. We preprocess every group for computing partial-sum query.

Lemma 3.2. *Algorithm 1 computes Equation (5) in $O(n^2 \alpha(n^2, n)^{2d})$ time and $O(n^2)$ space.*

Proof. Since we have at most $(d^{(3d-1)/2}(\pi/\theta)^{d-1})^2$ different group regions $R_{i,j} \cap R_{j,i}$, for all $s_i, s_j \in S, i < j$, grouping these regions takes

$$O(d^{(3d-1)/2}(\pi/\theta)^{d-1})^2 \log(d^{(3d-1)/2}(\pi/\theta)^{d-1})$$

time. Since d and θ are constants, grouping these regions takes $O(1)$ time. Preprocessing each group of the region takes

$$\sum_{i=1}^{q} [O(n) + O(Q_j)] = q \times O(n) + \sum_{i=1}^{q} O(Q_j) = O(qn) + O((d-1)!^2 n^2) = O(n^2)$$

time, where Q_j is the number of regions that lie in the j-th group. Finally, computing all partial-sum queries take $O((d-1)!^2 n^2 \alpha(n^2, n)^{2d})$ time. Therefore, Algorithm 1 takes $O(n^2 \alpha(n^2, n)^2 d)$ time. Space complexity of Algorithm 1 is similar to Algorithm 2. □

From Lemma 2.2 and Lemma 3.2, the following theorem is obtained.

Theorem 3.3. *Let S be a set of n uncertain points in $\mathbb{R}^d$. The expected weight of the θ-graph can be computed in $O(n^2 \alpha(n^2, n)^{2^d})$ time and $O(n^2)$ space.*

Algorithm 3:

```
1  htb Input : Uncertain point set $(S, \Pi)$
   Output: $\sum_{s_i, s_j \in S,\ i<j} \left[ |s_i s_j| \times \pi_i \times \pi_j \times \prod_{s_m \in R_{i,j} \cap R_{j,i}} \overline{\pi}_m \right]$
2  for $i = 1$ to $n - 1$ do
3      for $j = i + 1$ to $n$ do
4          for $k = 1$ to $(d - 1)!$ do
5              for $l = 1$ to $(d - 1)!$ do
6                  Group regions $R^k_{i,j} \cap R^l_{j,i}$ base of lines through the origin that is
                   orthogonal to its faces;
7          $E[R_{i,j} \cap R_{j,i}] = \pi_i \times \pi_j \times |s_i s_j|$;
8  for $i = 1$ to $q$ do
9      for $j = 1$ to $n$ do
10         Transform point $s_j$ to new coordinates base of lines through the origin that
           is orthogonal to the face of $G_i$ ;
11     preprocess $G_i$ for partial-sum query ;
12     for every region $\gamma \in G_i$ do
13         compute partial-sum query $\sigma(A, \gamma)$;
14         $E[Source(\gamma)] = E[Source(\gamma)] \times \sigma(A, \gamma)$;
15 $Sum = 0$;
16 for $i = 1$ to $n - 1$ do
17     for $j = i + 1$ to $n$ do
18         $Sum = Sum + E(R_{i,j} \cap R_{j,i})$
19 return $Sum$;
```

4 Conclusion

In this paper, we studied the θ-graph of a set of n points in uncertain points in tuple model. We proposed an algorithm to compute the expected weight of θ-graph in $O(n^2\alpha(n^2, n)^{2d})$ time using $O(n^2)$ space, where θ and d are constants. There are some interesting problems to be pursued. One of them is computing the expected weight of other spanners on uncertain points, since the θ-graph is a t-spanner for a t which is a function of θ.

References

[1] Agarwal, P. K., Cheng, S.-W., and Yi, K. Range searching on uncertain data. *ACM Transactions on Algorithms (TALG) 8*, 4 (2012), 43.

[2] Agarwal, P. K., Har-Peled, S., Suri, S., Yıldız, H., and Zhang, W. Convex hulls under uncertainty. In *European Symposium on Algorithms* (2014), Springer, pp. 37–48.

[3] Agarwal, P. K., Kumar, N., Sintos, S., and Suri, S. Range-max queries on uncertain data. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems* (2016), ACM, pp. 465–476.

[4] Alzoubi, K., Li, X.-Y., Wang, Y., Wan, P.-J., and Frieder, O. Geometric spanners for wireless ad hoc networks. *IEEE Transactions on parallel and Distributed Systems 14*, 4 (2003), 408–421.

[5] Chazelle, B., and Rosenberg, B. Computing partial sums in multidimensional arrays. In *Proceedings of the fifth annual symposium on Computational geometry* (1989), ACM, pp. 131–139.

[6] Clarkson, K. Approximation algorithms for shortest path motion planning. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing* (1987), pp. 56–65.

[7] Farshi, M., and Hosseini, S. H. Visualization of Geometric Spanner Algorithms, `http://cs.yazd.ac.ir/cgalg/AlgsVis/`.

[8] Fischer, M., Lukovszki, T., and Ziegler, M. Geometric searching in walkthrough animations with weak spanners in real time. In *European Symposium on Algorithms* (1998), Springer, pp. 163–174.

[9] Goldreich, O., and Ron, D. Approximating average parameters of graphs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Springer, 2006, pp. 363–374.

[10] Keil, J. M. Approximating the complete euclidean graph. In *Scandinavian Workshop on Algorithm Theory* (1988), Springer, pp. 208–213.

[11] Keil, J. M., and Gutwin, C. A. Classes of graphs which approximate the complete euclidean graph. *Discrete & Computational Geometry 7*, 1 (1992), 13–28.

[12] Kuhn, H. W. Some combinatorial lemmas in topology. *IBM Journal of research and development 4*, 5 (1960), 518–524.

[13] Morin, P., and Verdonschot, S. On the average number of edges in theta graphs. In *2014 Proceedings of the Eleventh Workshop on Analytic Algorithmics and Combinatorics (ANALCO)* (2014), SIAM, pp. 121–132.

[14] Narasimhan, G., and Smid, M. *Geometric spanner networks.* Cambridge University Press, 2007.

[15] Russel, D., and Guibas, L. Exploring protein folding trajectories using geometric spanners. In *Biocomputing 2005.* World Scientific, 2005, pp. 40–51.

[16] Suri, S., Verbeek, K., and Yıldız, H. On the most likely convex hull of uncertain points. In *European Symposium on Algorithms* (2013), Springer, pp. 791–802.

[17] Tarjan, R. E. Efficiency of a good but not linear set union algorithm. *Journal of the ACM (JACM) 22*, 2 (1975), 215–225.

[18] Xue, J., Li, Y., and Janardan, R. On the expected diameter, width, and complexity of a stochastic convex hull. *Computational Geometry 82* (2019), 16–31.

[19] Yao, A. C.-C. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM Journal on Computing 11*, 4 (1982), 721–736.

[20] Zhang, W. *Geometric computing over uncertain data.* PhD thesis, Duke University, 2015.