Nearest-Neighbor Search Under Uncertainty

Boris Aronov* John Iacono* Khadijeh Sheikhan*

Abstract

We study the problem of Nearest-Neighbor Searching under locational uncertainty. Here, an uncertain query or site consists of a set of points in the plane, and their distance is defined as distance between the two farthest points within them. In L_∞ metric, we present an algorithm with $O(n \log^2 n + s)$ expected preprocessing time, $O(n \log n)$ space, and $O(\log^2 n + k)$ query time, where s is the total number of site points, n is the number of sites, and k is the size of the query. We also propose a $\sqrt{2}$-approximation algorithm for the L_2 version of the problem.

1 Introduction

In this paper, our focus is on *Nearest-Neighbor (NN) Searching Under Uncertainty*. In the basic version of the NN problem, one wants to preprocess a set of site points in the plane, so that the closest one to a query point can be found efficiently.

For a brief survey on NN searching under uncertainty, refer to [18]. Two models of uncertainty have been considered in the literature. In the *existential model*, which we will not address, a site has a specified location, but it appears with a given probability and otherwise is not present at all. The model that we are interested in is the *locational model*, where a site and/or a query consists of more than one point, for example a region or a finite set of points representing possible locations of the uncertain point. An application of this model is location-based services where the data is imprecise. The distance of an uncertain query from an uncertain site is defined as an aggregate function of distances of points within them, such as maximum, minimum, or average of all possible distances.

For most of this paper, $d(.,.)$ is the L_∞ distance, and sites and queries are uncertain; an uncertain site or query consists of a finite set of possible locations in the plane, i.e., points, and the distance between them is defined as the maximum of all possible distances (see Figure 1).

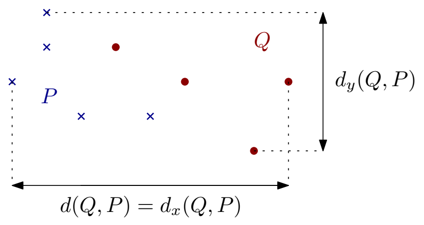

Figure 1: Distance of an uncertain query Q (red dots) from an uncertain site P (blue crosses).

Throughout this paper, for simplicity of presentation, we assume general position, which here means no two points share their x- or y-coordinates.

Problem statement We define the *distance* $d(P, Q)$ between two compact point sets P, Q in the plane by

$$d(P, Q) = \max_{p \in P, q \in Q} d(p, q).$$

Given a set $\mathcal{P} = \{P_1, P_2, ..., P_n\}$ of n uncertain points $P_i \subseteq 2^{\mathbb{R}^2}$, with $s = \sum_{P_i \in \mathcal{P}} |P_i|$, construct a data structure $D(\mathcal{P})$, such that queries of the form $\mathrm{NN}_{D(\mathcal{P})}(Q)$, for $Q \subseteq 2^{\mathbb{R}^2}$, $k = |Q|$, that return $\arg\min_i d(P_i, Q)$, can be answered efficiently.

Motivation Our definition of distance between a query and a site as the largest separation between any two representatives of the respective sets can be motivated by the following slightly artificial example: Each site represents the set of possible locations of an ambulance. The query represents the set of possible locations of a 911 caller. The answer to the query is the ambulance that is closest to the caller, using the worst-case combination of the positions of the ambulance and the caller. In other words, it minimizes the worst-case response time given the uncertain locations.

Our results We first propose an algorithm to find the nearest site in $O(\log n + k)$ time, using quadratic space and $O(n^2 \log n + s)$ expected preprocessing time. Then we improve the space requirement to near linear by adding a logarithmic factor to the query time and propose a data structure smoothly interpolating between the two extremes. The nearest neighbor in L_∞ metric is also a $\sqrt{2}$-approximate solution for the L_2 version of the

*NYU Tandon School of Engineering, Brooklyn, NY, 11201, USA, {boris.aronov,iacono,khadijeh}@nyu.edu. Work of B.A. on this paper has been partially supported by NSF Grants CCF-11-17336, CCF-12-18791, and CCF-15-40656, and by BSF grant 2014/170. Work by K.S. on this paper has been supported by NSF Grants CCF-12-18791 and CCF-13-19648. Work of J.I. on this paper has been supported by NSF Grant CCF-13-19648.

problem. Finally, we propose a more efficient algorithm to find a $\sqrt{2}$-approximate answer to the L_2 version.

2 Related Work

The nearest-neighbor problem has been studied extensively in the literature. A well-known approach is subdividing the plane into cells, each consisting of points with the same nearest neighbor. This subdivision of the plane is called a *Voronoi Diagram*. A Voronoi diagram is then preprocessed for point location, in order to answer NN queries. This diagram has been extensively studied for different types of sites such as segments or polygons, using different metrics and also in higher dimensions [15].

NN searching under uncertainty has been studied in a variety of settings. In the case where queries are points but sites are uncertain, modified versions of Voronoi Diagram are proposed. An *Uncertain Voronoi Diagram* is a subdivision of space into regions, so that all the points in each region have the same *set* of possible nearest neighbors [4]. Zhang et al. propose the notion of *Possible Voronoi cell (PV-cell)* [17]. The PV-cell of a site is the region where that site has positive probability of being the nearest neighbor. Evans et al. introduce another version of Voronoi diagram where each cell contains those points guaranteed to be closest to a particular site [6].

When distance from a query point to an uncertain site is defined as the distance to the site's farthest point, it is equal to the *Hausdorff* distance from the query point to the site, so NN searching can be done using Hausdorff Voronoi Diagrams (HVD). Papadopoulou proves that the size of the HVD is $O(n^2)$ where n is the total number of vertices on the convex hulls of the sites. She provides a plane sweep algorithm to construct it [14]. If the convex hulls of sites are disjoint then the HVD's size is $O(n)$ and can be computed in $O(n \log^3 n)$ time [5]. By performing a point-location query in the HVD, the nearest uncertain site can be found in poly-logarithmic time.

Agarwal et al. cover different cases of uncertainty in their work [2]. In their setting a site or query point is specified as a probability density function (pdf) and the goal is to find the *Expected Nearest Neighbor (ENN)* which is the site with minimum expected distance to the query. Under squared Euclidean distance, they prove that if the pdf of each site has description complexity at most k, the *Expected Voronoi Diagram (EVD)* has linear size and can be computed in $O(n \log n + nk)$ time. Thus, an ENN query can be answered in $O(\log n)$ using point location in the EVD. Using rectilinear metrics and assuming each site has a discrete pdf consisting of k points, they provide an algorithm to answer queries in $O(\log^3(kn))$ time by doing a point location query. For the Euclidean distance they construct an ε-approximation of the EVD (ε-EVD), and the ε-approximation of the ENN (ε-ENN) can be reported in $O(\log(n/\varepsilon))$. In another work [1], Agarwal et al. propose an algorithm to find those sites that can be the NN with probability greater than a threshold and an algorithm to report the point that has the maximum probability of being the NN.

Many results on the NN problem use branch-and-bound pruning techniques. These methods mostly use R-trees to index the sites, try to prune nodes, and use heuristics to make the process more efficient, but there is no guarantee that it runs asymptotically faster than linear-time brute-force algorithm, in the worst case.

In *Aggregate Nearest-Neighbor Searching* an aggregate or group query consists of a finite set of points, and distance is an aggregate function of all the distances, such as their sum, maximum, or average. This type of queries can be viewed as an equivalent to our variant of uncertain queries. Dealing with sum version of aggregate queries under Euclidean distance, Papadias et al. use R-trees to create an index on the set of sites [12]. They propose several empirical algorithms using this data structure. They also provide a modification of those algorithms to work efficiently for disk-resident queries. In another work [13], previous algorithms are modified to cover other variants of the aggregate NN problem such as the sum, max, and min versions. Again, input sites are indexed using R-trees, algorithms are evaluated by experiments, and no worst-case analysis is provided.

To answer the aggregate-max NN query on a set of n sites in the plane, Wang provides algorithms for L_1 and L_2 metrics that give exact answers in sub-linear time [16]. For the L_1 version, he builds a linear-size data structure in $O(n \log n)$ time that answers a query of size k in $O(\log n + k)$ time. For L_2, constructing the data structure takes $O(n \log n)$ time and $O(n \log \log n)$ space and a query can be answered in $O(k\sqrt{n} \log^{O(1)} n)$ time. He also proposes another data structure for L_2, which takes $O(n^{2+\varepsilon})$ time and space and guarantees $O(k \log n)$ query time.

As an example of uncertainty for both queries and sites, Lian et al. introduce *Probabilistic Group Nearest Neighbor (PGNN)*, which are aggregate NN queries in uncertain data sets [10]. Their approach is reducing the search space by proposing pruning methods. They demonstrate the efficiency of their algorithm experimentally, with no analysis provided. Our results are closely related to the PGNN problem, except that we provide preprocessing and query time analysis.

3 Exact Nearest Neighbor for L_∞

First we observe that we can find the nearest neighbor using only the axis-parallel bounding box of uncertain sets; the exact position of points within the box are not needed (see Figure 2). Therefore, after computing the bounding boxes, neither preprocessing nor query time will depend on the sizes of sites or queries.

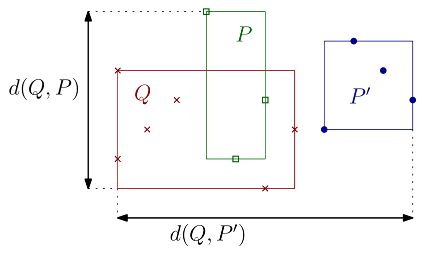

Figure 2: Distance of two uncertain points only depends on their axis-aligned bounding boxes.

Lemma 1 *For any two compact sets P, Q in the plane, $d(P,Q) = d(b(P), b(Q))$, where $b(\cdot)$ denotes the axis-parallel bounding box of a set.*

Proof. Since $P \subset b(P)$, $Q \subset b(Q)$, and the distance between sets is defined as the *longest* interpoint distance, we have $d(b(P), b(Q)) \geq d(P,Q)$.

Now suppose $d(b(P), b(Q)) = |p_x - q_x|$, for $p \in b(P)$, $q \in b(Q)$; p, q must lie on bounding box boundaries, for otherwise $|p_x - q_x|$ can be increased. By definition of a bounding box, there exist $p' \in P$, $q' \in Q$ with $p'_x = p_x$, $q'_x = q_x$. Thus $d(b(P), b(Q)) = |p'_x - q'_x| \leq d(P,Q)$. □

So, our problem is reduced to the nearest-neighbor problem for axis-parallel boxes. For the remainder of this paper, we will assume that a site or query is given by its axis-aligned bounding box, specified by its four coordinates. First we will define properties of a query rectangle based on these four values.

For a given query rectangle $Q(x_1, x_2, y_1, y_2)$, its center is $\odot = \odot(Q) = (x_\odot, y_\odot) = ((x_1+x_2)/2, (y_1+y_2)/2)$. The width and height are defined as $\Delta_x = \Delta_x(Q) = x_2 - x_1$ and $\Delta_y = \Delta_y(Q) = y_2 - y_1$, respectively, as shown in Figure 3. We also define $\Delta = \Delta(Q) = |\Delta x - \Delta y|$ to be

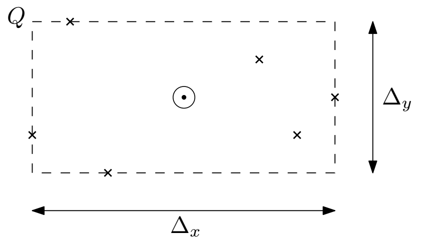

Figure 3: Center, width, and height of a query.

the absolute difference between the height and width of Q.

Without loss of generality, we assume that width of Q is greater than or equal to the height, so $\Delta = \Delta_x - \Delta_y \geq 0$; a symmetric structure will handle the complementary case. We partition the set of sites into left and right subsets, named $\mathcal{P}_\ell$ and $\mathcal{P}_r$, based on the position of their center relative to the vertical line through $\odot$. When $\Delta_x < \Delta_y$, 'right' and 'left' will be replaced by 'top' and 'bottom' in all of the arguments.

Observation 1 *The left edge $e_\ell = e_\ell(P)$ of any rectangle site $P \in \mathcal{P}_\ell$ is sufficient to compute the distance from the query Q; in other words, $d(P,Q) = d(e_\ell, Q)$. A symmetric statement holds for $\mathcal{P}_r$.*

So sites can be replaced by either vertical or horizontal segments, based on the width and height of the query, and their dimension decreases by one. See Figure 4. Next we explain how to effectively decrease the dimension of a query by replacing it with a point.

Lemma 2 *The nearest site in $\mathcal{P}_\ell$ to the rectangle Q, is the same as the nearest site to its center $\odot$ shifted by $\Delta/2$ to the right.*

Proof. First, we compute the distance of a site $P \in \mathcal{P}_\ell$ from a query Q using the query's center, width, and height:

$$\begin{aligned} d(P,Q) &= d(e_l, Q) = \max(d_x(e_l, Q), d_y(e_l, Q)) \\ &= \max(d_x(e_l, \odot) + \frac{\Delta_x}{2}, d_y(e_l, \odot) + \frac{\Delta_y}{2}). \end{aligned}$$

We assumed $\Delta_x \geq \Delta_y$, thus

$$d(P,Q) = \frac{\Delta_y}{2} + \max(d_x(e_l, \odot) + \frac{\Delta}{2}, d_y(e_l, \odot)).$$

Since e_l is to the left of $\odot$, if we shift $\odot$ to the right by $\Delta/2$, to the point $\odot' = (x_\odot + \Delta/2, y_\odot)$, then

$$\begin{aligned} d(P,Q) &= \frac{\Delta_y}{2} + \max(d_x(e_l, \odot'), d_y(e_l, \odot')) \\ &= \frac{\Delta_y}{2} + d(e_l, \odot') \end{aligned}$$

So, the distance from Q differs from the distance from $\odot'$ by $\Delta_y/2$. This proves that the nearest site to the rectangle Q among those in $\mathcal{P}_\ell$, is the same as the nearest one to the point $\odot'$. □

Now we query in $\mathcal{P}_\ell$ and $\mathcal{P}_r$ separately, using an appropriately shifted center of Q, to find the nearest site in each set. Then we can compare their distance from Q to find the real nearest neighbor.

Theorem 3 *The nearest site to a query Q can be found in $O(\log n)$ time, using $O(n^2)$ space and $O(n^2 \log n)$ expected preprocessing time.*

Proof. Given a set of n vertical segments, we need the closest to a point. Assuming general position, the segments are pairwise disjoint and our definition of the distance (a variant of the Hausdorff metric) satisfies the axioms of an *Abstract Voronoi Diagram* [11]. Therefore

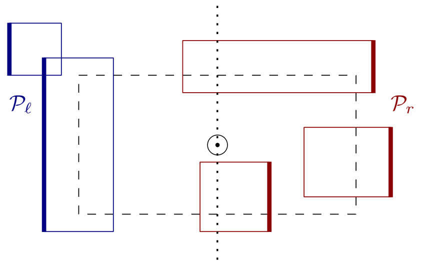

Figure 4: Dividing the sites into sets $\mathcal{P}_\ell$ and $\mathcal{P}_r$. The highlighted edges are sufficient for measuring the distance from the query.

the diagram has linear size and can be constructed in $O(n \log n)$ expected time using linear space [8]. We then preprocess it for logarithmic-time point location.

We focus on finding the nearest site in $\mathcal{P}_\ell$. A site in $\mathcal{P}_\ell$ is a site with the x-coordinate of its center smaller than $\odot_x$. We presort the sites in $\mathcal{P}$ according to the x-coordinate of their centers and construct the Voronoi diagram of every prefix, for a total of n diagrams with total size $O(n^2)$, in expected time $O(n^2 \log n)$. When answering a query, we binary search using $\odot_x$ to find the Voronoi diagram corresponding to $\mathcal{P}_\ell$ and query in the diagram with $\odot'$ to find the nearest site. We repeat this process for $\mathcal{P}_r$, then compare the results in constant time and return the better of the two answers. Answering a query involves a constant number of binary searches and point locations, so the query time is $O(\log n)$. $\square$

Theorem 4 *The nearest site to a query Q can be found in $O(\log^2 n)$ time, using $O(n \log n)$ space and $O(n \log^2 n)$ expected preprocessing time.*

Proof. To answer a query with center $\odot$, we need to find the nearest left (right) edge of sites the x-coordinate of whose center is less (greater) than $\odot_x$, which is a prefix (suffix) of sites sorted by their centers. This problem is an example of a *decomposable searching problem* introduced by Bentley and Saxe [3], i.e., to find NN in some set, we can decompose it into smaller sets, search for NN in each subset, then compare the results to find the real NN.

Using this property we can improve the preprocessing time, by creating a hierarchical data structure of sites sorted by x-coordinate of their centers. We construct a binary tree where each internal node stores a list of sites in its subtree, which is equivalent to sites whose centers belong to some canonical range of x values. At each node we construct the Voronoi diagram of this list, separately for the left and right edges. Given query Q, we perform a query using its center, to obtain the $O(\log n)$ nodes whose subtrees are a decomposition of $\mathcal{P}_\ell$ and $\mathcal{P}_r$. We perform a query with corresponding $\odot'$ in each of these Voronoi diagrams, and compare them to find the real nearest neighbor. This way the preprocessing takes $O(n \log^2 n)$ expected time and $O(n \log n)$ space, but the query time will be $O(\log^2 n)$. $\square$

If we replace the binary tree by an m-ary one in the data structure, for any $2 \leq m \leq n$, we obtain a trade-off between query and preprocessing costs. Each node has m children; for $1 \leq i \leq m$, we store Voronoi diagram of sites of the union of the subtrees of its first (and last) i children, $O(m)$ diagrams at each node. A query takes $O(\log_m n \log m) = O(\log n)$ time to find those $O(\log_m n)$ nodes that cover P_ℓ or P_r (at most one of each at each tree level), and we need to do point location in $O(\log_m n)$ nodes and each takes $O(\log n)$. So the total query time is $O(\log_m n \log n)$. Each site is stored at $O(\log_m n)$ nodes, and at each node there are $O(m)$ copies of it. So the size of the data structure is $O(mn \log_m n)$, and it takes $O(mn \log_m n \log n)$ expected time to construct it. If we set $m = n^{1/\delta}$, for any $1 \leq \delta \leq \log n$, we obtain a data structure of size $O(\delta n^{1+1/\delta})$ in $O(\delta n^{1+1/\delta} \log n)$ expected preprocessing time and each query takes $O(\delta \log n)$ time.

4 Approximate Nearest Neighbor for L_2

In the Euclidean metric, finding the exact answer to nearest-neighbor problem under uncertainty is more complicated than in rectilinear metric. In order to obtain fast queries, we consider approximating the answer. In the case where the query is uncertain (aggregate queries), but each site is a point, Li et al. provided a $\sqrt{2}$-approximation answer [9]. We generalize this result for the version of the problem where both queries and sites are uncertain.

In the following theorem, we show that an uncertain query Q can be represented by a single point $\odot$, the center of its minimum enclosing circle, so that the distance of Q from the site nearest to $\odot$ is an approximation of the distance to the true nearest neighbor.

Theorem 5 *When querying a set of uncertain sites with an uncertain query Q, if C is the minimum enclosing circle of Q with radius r and with center at $\odot$, P the nearest site to $\odot$, and P^* the nearest site to Q, then $l = d(Q, P)$ will be a $\sqrt{2}$-approximation of $l^* = d(Q, P^*)$.*

Proof. Let d and d^* be the distance of $\odot$ from P and P^*, respectively. Let z be the farthest point in P^* from $\odot$ (so that $d(z, \odot) = d^*$) and AB be the diameter of C orthogonal to $\odot z$. Connect z to $\odot$ and extend it so that it hits the circle at E (see Figure 5). By the triangle inequality,

$$l = d(Q, P) \leq d + r.$$

Since C is a minimum enclosing circle, there should be a query point q on $\frown AEB$, and $\angle z \odot q \geq \pi/2$. Therefore,

$$d(q, z) \geq \sqrt{d^{*2} + r^2}.$$

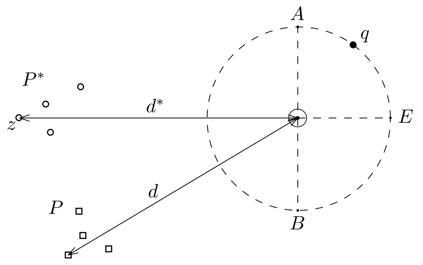

Figure 5: The nearest site to the center of an aggregate query is an approximate nearest neighbor.

By definition

$$l^* = d(Q, P^*) \geq d(q, z),$$

and since P is the nearest site to $\odot$

$$d^* \geq d.$$

As a result,

$$l^* \geq \sqrt{d^2 + r^2}.$$

If both d and r are equal to zero, since $l \leq d + r$, l is also zero, so $l \leq \sqrt{2}l^*$ holds. Otherwise, $\sqrt{d^2 + r^2} \neq 0$ and

$$l \leq \frac{d + r}{\sqrt{d^2 + r^2}} l^* \leq \sqrt{2}l^*.$$

□

The challenge here is finding the nearest uncertain site to the point $\odot$. Since this distance is equivalent to Hausdorff distance we can find the nearest site by performing a point location query in the Hausdorff Voronoi Diagram which takes poly-logarithmic time. The preprocessing time and the size of the diagram depends on how separated the sites are [5, 14]. In the case where convex hulls of uncertain points are disjoint, the diagram can be created in $O(s \log^3 s)$ time using linear space [7].

5 Conclusion

We studied NN searching with uncertain sites and queries. Under L_∞ metric, we provided two algorithms to find the exact NN. There is a trade-off between the preprocessing and query time as shown in Table 1. One obvious open problem is to establish a lower bound on the query time when using, say, linear space.

For L_2 version of the problem, we presented a $\sqrt{2}$-approximation algorithm. Is there is an algorithm with sublinear query time to find the exact nearest neighbor? Moreover, there is no lower bound on query time for the Euclidean metric.

Preproc. time	DS Size	Query time
$O(n^2 \log n + s)$	$O(n^2)$	$O(\log n + k)$
$O(n \log^2 n + s)$	$O(n \log n)$	$O(\log^2 n + k)$
$O(mn \log_m n \log n + s)$	$O(mn \log_m n)$	$O(\log_m n \log n + k)$
$O(\delta n^{1+1/\delta} \log n + s)$	$O(\delta n^{1+1/\delta})$	$O(\delta \log n + k)$

Table 1: The trade-off between preprocessing and query time.

References

[1] P. K. Agarwal, B. Aronov, S. Har-Peled, J. M. Phillips, K. Yi, and W. Zhang. Nearest-neighbor searching under uncertainty II. *ACM Trans. Algorithms*, 13(1):3:1–3:25, 2016.

[2] P. K. Agarwal, A. Efrat, S. Sankararaman, and W. Zhang. Nearest-neighbor searching under uncertainty. In *PODS*, pages 225–236, 2012.

[3] J. L. Bentley and J. B. Saxe. Decomposable searching problems I. Static-to-dynamic transformation. *Journal of Algorithms*, 1(4):301–358, 1980.

[4] R. Cheng, X. Xie, M. L. Yiu, J. Chen, and L. Sun. UV-diagram: A Voronoi diagram for uncertain data. In *26th International Conference on Data Engineering*, pages 796–807. IEEE, 2010.

[5] O. Cheong, H. Everett, M. Glisse, J. Gudmundsson, S. Hornus, S. Lazard, M. Lee, and H.-S. Na. Farthest-polygon Voronoi diagrams. *Computational Geometry*, 44(4):234–247, 2011.

[6] W. Evans and J. Sember. Guaranteed Voronoi diagrams of uncertain sites. In *20th Canadian Conference on Computational Geometry*, pages 207–210, 2008.

[7] J. Iacono, E. Khramtcova, and S. Langerman. Searching edges in the overlap of two plane graphs. *CoRR*, abs/1701.02229, 2017.

[8] R. Klein. Abstract Voronoi diagrams and their applications. *Computational Geometry and its Applications*, 333:148–157, 1988.

[9] F. Li, B. Yao, and P. Kumar. Group enclosing queries. *IEEE Transactions on Knowledge and Data Engineering*, 23(10):1526–1540, 2011.

[10] X. Lian and L. Chen. Probabilistic group nearest neighbor queries in uncertain databases. *IEEE Transactions on Knowledge and Data Engineering*, 20(6):809–824, 2008.

[11] J. Pach and M. Sharir. The upper envelope of piecewise linear functions and the boundary of a region enclosed by convex plates: Combinatorial analysis. *Discrete & Computational Geometry*, 4(1):291–309, 1989.

[12] D. Papadias, Q. Shen, Y. Tao, and K. Mouratidis. Group nearest neighbor queries. In *20th International Conference on Data Engineering*, pages 301–312, 2004.

[13] D. Papadias, Y. Tao, K. Mouratidis, and C. K. Hui. Aggregate nearest neighbor queries in spatial databases. *ACM Transactions on Database Systems (TODS)*, 30(2):529–576, 2005.

[14] E. Papadopoulou. The Hausdorff Voronoi diagram of point clusters in the plane. *Algorithmica*, 40(2):63–82, 2004.

[15] G. Voronoï. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Deuxième mémoire. Recherches sur les parallélloèdres primitifs. *Journal für die reine und angewandte Mathematik*, 134:198–287, 1908.

[16] H. Wang. Aggregate-max nearest neighbor searching in the plane. In *Canadian Conference on Computational Geometry*, pages 71–76, 2013.

[17] P. Zhang, R. Cheng, N. Mamoulis, M. Renz, A. Züfle, Y. Tang, and T. Emrich. Voronoi-based nearest neighbor search for multi-dimensional uncertain databases. In *29th International Conference on Data Engineering*, pages 158–169, 2013.

[18] W. Zhang. Nearest-neighbor searching under uncertainty. Master's thesis, Department of Computer Science, Duke University, 2012.