Construction of the Nearest Neighbor Embracing Graph of a Point Set[1]

M.Y. Chan[1], Danny Z. Chen[2], Francis Y.L. Chin[1], and Cao An Wang[3]

1. Department of Computer Science and Information Systems
The University of Hong Kong, Hong Kong
e-mail: {mychan,chin}@csis.hku.hk

2. Department of Computer Science and Engineering
University of Notre Dome, Notre Dome, IN 46556, USA
e-mail: chen@cse.nd.edu

3. Department of Computer Science
Memorial University of Newfoundland
St. John's, Newfoundland, Canada
e-mail: wang@garfield.cs.mun.ca

Abstract

This paper gives optimal algorithms for the construction of the Nearest Neighbor Embracing Graph (NNE-graph) of a given point set V of size n in the k-dimensional space (k-D) for $k = 2, 3$. The NNE-graph provides another way of connecting points in a communication network, which has lower expected degree at each point and shorter total length of connections with respect to those using Delaunay triangulation. In fact, the NNE-graph can also be used as a tool to test whether a point set is randomly generated or has some particular properties.

We show that in 2-D the NNE-graph can be constructed in optimal $\Theta(n^2)$ time in the worst case. We also present an $O(n \log n + nd)$ time algorithm, where d is the $\Omega(\log\ n)$-th largest degree in the output NNE-graph. The algorithm is optimal when $d = O(\log n)$. The algorithm is also sensitive to the structure of the NNE-graph, for instance when $d = g \cdot (\log n)$, the number of edges in NNE-graph is bounded by $O(gn \log n)$ for any value g with $1 \leq g \leq \frac{n}{\log n}$. We finally propose an $O(n \log n + nd \log d^*)$ time algorithm for the problem in 3-D, where d and d^* are the $\Omega(\frac{\log n}{\log \log n})$-th largest vertex degree and the largest vertex degree in the NNE-graph, respectively. The algorithm is optimal when the largest vertex degree of the NNE-graph d^* is $O(\frac{\log n}{\log \log n})$.

1 Introduction

The Nearest Neighbor Embracing Graph (NNE-graph for short) of a given point set V in k-D is constructed as follows (throughout this paper, Euclidean distance is assumed): *for each point p in V, we connect p to its first nearest neighbor x_1, its second nearest neighbor x_2, and so on by edges until p is contained in the convex hull of these nearest neighbors.* An NNE-graph thus

[1]This work was supported in part by RGC Grant HKU 7019/00E for the first and the third authors, by the US National Science Foundation under Grants CCR-9988468 and CCF-0515203 for the second author, and by NSERC grant OPG0041629 for the fourth author. The work of the second author was partially done while visiting the Department of Computer Science, Hong Kong University of Science and Technology in 2003.

consists of the vertex set V and the edge set of all these connecting edges. For these vertices of V on the boundary of the convex hull, a special treatment will be discussed later in the concluding remarks. The NNE-graph was first introduced in [4] and has very interesting properties. If the point set V is generated by a Poisson process in a plane (i.e., 2-D), it is proved that the expected degree of a typical point in the NNE-graph is 5, in contrast with the expected degree of 6 of a typical point in the Delaunay triangulation [3,7] of V. Thus, the NNE-graph provides another way of connecting points in a communication network, which has lower expected degree at each point and shorter total length of connections. In the other hand, the NNE-graph can also be used as a tool to test whether a point set is randomly generated or has some particular properties. If the expected degree of the NNE-graph of a point set V deviates from 5, it suggests that the point set V exhibits some regularity (if it is lower than 5) or some clustering (if it is higher than 5) [1].

In this paper, we present a worst-case optimal algorithm and two improved algorithms for finding the respected NNE-graph of point set in 2-D and 3-D. The algorithms are based on three key techniques: exponential search based on selection algorithm in [6], fast k nearest neighbor algorithm based on well-separated pair decomposition in [2], and fast point-in-convex hull test based on projection.

In Section 2, we show that the worst case lower bound for construction of the NNE-graph of a given point set V of size n is at least $\Omega(n^2)$ time and the brute force method of construction has a time complexity of $O(n^2 \log n)$. In Section 3, we present a worst-case optimal $\Theta(n^2)$ time algorithm and an improved $O(n \log n + nd)$ time algorithm, where d is the $\Omega(\log n)$-th largest vertex degree in NNE-graph. That is, d is the $(\log n)$-th largest vertex degree or higher in the NNE-graph. This algorithm is optimal when $d = O(\log n)$. The optimality can be seen from a simpler problem: finding the closest pair of n points in 2-D, which takes $\Omega(n \log n)$ time, can be transformed to the corresponding NNE-graph problem. In Section 4, we extend the problem to 3-D and propose an $O(n \log n + nd(\log d^*))$ time algorithm, where d^* and d are the largest vertex degree and the $\Omega(\frac{\log n}{\log\log n})$-th largest vertex degree in the NNE-graph, respectively. The algorithm is optimal when $d^* = O(\frac{\log n}{\log\log n})$. In Section 5, we give some concluding remarks.

2 Lower Bound and a Brute Force Approach

Lemma 1 *The time complexity of any algorithm for constructing the NNE-graph of a given point set V of size n is at least $\Omega(n^2)$ in the worst case.*

Proof: Consider the following point set V of size n: $n-1$ of the points lie on a convex curve facing a distant single point v, with the distance between any two points on the curve shorter than the distance between v and its nearest neighbor on the curve (refer to Figure 1).

In this scenario, it is easy to see that for any point p on the curve, the NNE-graph of V has to include the edge from p to v as well as the edges from p to all the rest of points on the curve. Thus, the NNE-graph must have $O(n^2)$ edges. By this example, the time complexity of any algorithm for constructing the NNE-graph is at least $\Omega(n^2)$ in the worst case. □

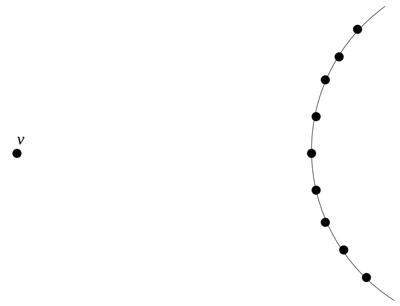

Figure 1: The worst case scenario.

A brute force construction of the NNE-graph can be done as follows. For each point p in V, sort the points in V according to their distances from p, and then check, for successive k starting at 2, whether p is contained inside the convex hull of $\{x_1, x_2, \ldots, x_k\}$, where x_k is the k-th nearest neighbor of p, stopped when p lies inside. This brute force method has a time complexity of $O(n^2 \log n)$.

3 Fast Algorithms for Finding the NNE-graph of Point Set in 2-D

In this section, we present an $O(n \log n + dn)$ time algorithm for computing the NNE-graph on an input set V of n points in 2-D, where d is at least the $(\log n)$-th largest vertex degree or higher in the output NNE-graph (i.e., $\Omega(\log n)$-th). Our algorithm is based on the ideas of exponentially searching the nearest neighbors for each input point and of testing the point-in-convex hull of these nearest neighbors, simply by maintaining a proper convex hull on the boundary of a unit circle. To illustrate the main ideas and key operations of our approach, we first describe a preliminary $\Theta(n^2)$ time algorithm; this algorithm is also used as a subroutine by the final solution. We then give the $O(n \log n + dn)$ time algorithm.

3.1 A Preliminary Algorithm

For a point $p \in V$, let x_j denote the j-th nearest neighbor of p in V ($j \geq 1$), and $X_p^* = \{x_1, x_2, \ldots, x_h\}$ be the set of nearest neighbors of p in V such that the convex hull $CH(X_p^*)$ contains p but the convex hull $CH(X_p^* - \{x_h\})$ does not contain p. To illustrate an $\Theta(n^2)$ time algorithm, it is sufficient to show how to compute the set X_p^* for an arbitrary point $p \in V$ in $O(n)$ time. Clearly, X_p^* can be obtained in $O(n)$ time once point x_h is identified. Our key idea here is to exponentially search for the point x_h (in $O(n)$ time) using the selection algorithm [6].

For each $j = 1, 2, \ldots, n-1$, let $X_p(j)$ denote the set of the first j nearest neighbors of p in V, i.e., $X_p(j) = \{x_1, x_2, \ldots, x_j\}$. The following "decision" question plays the role of the "probing" operation in the exponential search for p: Given a set $X_p(j)$, whether the convex hull $CH(X_p(j))$ contains p?

Instead of computing $CH(X_p(j))$ (in 2-D) explicitly for the decision question, we show below

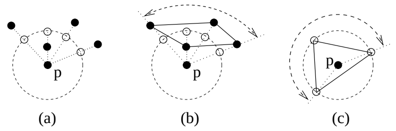

Figure 2: (a) The projection of points on the unit circle C_p (the unfilled little dots are the projected points). (b) If $CH(X_p(j))$ (the solid convex polygon) does not contain p, then the wedge $W_p(X_p(j))$ is less than π. (c) If $CH(X_p(j))$ does contain p, then $W_p(X_p(j))$ is no smaller than π.

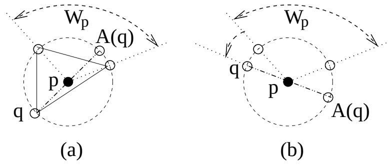

Figure 3: Illustrating the iterative computation of $W_p(X_p(j))$: (a) When a point q is not contained in the maintained wedge W_p but its antipodal point $A(q)$ is contained in W_p; (b) when both q and its antipodal point $A(q)$ are not contained in W_p.

that it is sufficient to compute the smallest wedge $W_p(X_p(j))$ whose apex is p and that contains all points of $X_p(j)$. Note that such a wedge can be represented by two points on the boundary of the unit circle C_p centered at p. In fact, we can project all points of $X_p(j)$ onto the boundary of C_p and view the wedge $W_p(X_p(j))$ as the '1-D convex hull' of the projected points of $X_p(j)$ on the boundary of C_p (see Figure 2(a)).

The following lemma is trivial.

Lemma 2 *For any set $X_p(j)$, p is contained in $CH(X_p(j))$ if and only if $W_p(X_p(j))$ is not smaller than π.*

In order to meet the time bound, it is necessary to decide whether $W_p(X_p(j)) \geq \pi$ in $O(|X_p(j)|)$ $(= O(j))$ time, i.e., $O(1)$ time for each point in $X_p(j)$. It can be verified with the help of Figure 3 that the extrema of $W_p(X_p(j))$ can be maintained in constant time per step.

We are now ready to describe the following lemma.

Lemma 3 *Given a subset $X \subset V$ with $p \in X$, the edges for p in the NNE-graph can be computed in $O(\mid X \mid)$ time.*

Proof: We shall use binary search on the point set X and combine with the decision question on some subset $(x_1, x_2, \ldots, x_i)$ to find the point x_h, where x_h is the closest point to p such that $CH(X_p^*)$ contains p for $X_p^* = (x_1, \ldots, x_h)$. To do this, we compute the median x_m in X in terms of the nearest neighbor to p, and use x_m to split X into 'near' neighbors X_1 and 'far' neighbors X_2 with respect to p. Now, we ask the decision question for each points in X_1. If there is a 'yes' answer, then we discard X_2 and compute the median x'_m in X_1 and split X_1 into 'near' neighbors X_{11} and 'far' neighbors X_{12}. If there is a 'no' answer after exhausting all the points in X_1, then we discard X_1 and compute the median x''_m in X_2 and split X_2 into 'near' neighbors X_{21} and 'far' neighbors X_{22}. In either case, we ask the decision question and continue the binary search on the proper subset by median selection and set split. This binary search continues until point x_h is found. Note that when we ask the decision question on a subset X_{ij}, the data $W_p(X_p(i))$ is available in the previous step, where x_i is the fist point in X_{ij}. Thus, the number of decision questions asked in the entire process is proportional to the size of X by an argument as for binary search. □

Theorem 1 *Given a set V of n points in 2-D, the NNE-graph can be computed in $\Theta(n^2)$ time, which is worst-case optimal.*

3.2 An Improved Algorithm

We show in this subsection how to modify the preliminary algorithm in Subsection 3.1 to obtain an $O(n \log n + dn)$ time solution, where d is at least the $(\log n)$-th largest vertex degree or higher in the output NNE-graph. Note that when the degrees of most vertices of the output NNE-graph (say, $n - c \cdot \log n$ vertices for some positive constant c) are upper-bounded by $O(\log n)$, our algorithm takes no more than $O(n \log n)$ time, and it is optimal.

The improved algorithm has two key differences from the preliminary algorithm:

- It uses Callahan and Kosaraju's k nearest neighbors algorithm [2], instead of the selection algorithm [6], in the binary search process.
- It searches for the $\Omega(\log n)$ nearest neighbors of the NNE-graph by simultaneously processing all points of V in each iteration, instead of the computations for the points in V one after the other sequentially.

The following results are the base of our algorithm.

Lemma 4 *[2] For a set P of n points in l-D for any fixed integer $l \geq 1$, (1) a structure called* **well-separated pair decomposition** *of P can be constructed in $O(n \log n)$ time, and (2) given a well-separated pair decomposition of P, the k nearest neighbors of all points in P can be obtained in $O(kn)$ time for any positive integer k.*

Our improved algorithm works as follows.

1. Construct a well-separated pair decomposition of the 2-D point set V (* which can be done in $O(n \log n)$ time as stated in part (1) of Lemma 4.*); Let $g = 1$
2. Find all the $g \cdot (\log n)$ nearest neighbors of every point in V. (* which can be done in $O(gn \log n)$ time as stated in part (2) of Lemma 4. *)
3. For every point $p \in V$, ask the 'decision question' on the nearest neighbor set $X_p(g \cdot (\log n))$ (* as shown in Subsection 3.1, which take $O(\log n)$ time per point in V.*) If the answer is "yes", then mark the point p with 'the search range for the nearest neighbor set X_p^* is known'; otherwise, 'the search range for X_p^* is not yet known'.

4. If there are only $O(\log n)$ points left in V such that each of them has marked with 'search range is not yet known', then terminate the iterative process; otherwise (more than $O(\log n)$ such points left in V), double the number of nearest neighbor search by letting $g \leftarrow 2g$ and go to Step 2.

   After the termination of this iteration, the n points in V can be partitioned into two types: (1) Those points p whose search range for X_p^* is already known (there are $n - O(\log n)$ of them), and (2) those points p whose search range for X_p^* is still not yet known (there are only $O(\log n)$ of them).

5. For every point p of type (1) ($n - O(\log n)$ points), we perform a binary search to determine X_p^*, which takes $O(g \cdot \log n)$ time. For every point p of type (2) ($O(\log n)$ points), we use the binary search algorithm to determine X_p^*, which takes $O(\log n)$ time. Thus, we find the NNE-graph for V.

The correctness of this algorithm follows from the same observations in Subsection 3.1. The time analysis of this algorithm is as follows. The initialization in Step 1 takes $O(n \log n)$ time (for constructing a well-separated pair decomposition of the point set V). The iterative process in Steps 2, 3 and 4 takes dn time, where we let $d = g \cdot (\log n)$. To see this, note that the number of nearest neighbors to be searched for each point is doubled after Step 4 for next iteration and there are g iterations for $1 \leq g \leq \frac{n}{\log n}$. In Step 5, the determination of X_p^* for each point p of type (1) takes $O(g \cdot \log n)$ time, and there are $n - O(\log n)$ type (1) points in V; the determination of X_p^* for each point p of type (2) takes $O(n)$ time, and there are $O(\log n)$ type (2) points in V. Therefore, the determination of X_p^* for all n points in V takes altogether $O(n \log n + g \cdot (n \log n))$ time. In total, the running time of the algorithm is $O(n \log n + dn)$ for $d = g \cdot \log n$.

Theorem 2 *Given a set V of n points in 2-D, the NNE-graph can be computed in $O(n \log n + dn)$ time, where d is the $\Omega(\log n)$-th largest degree of the NNE-graph.*

4 The 3-D Case

It is possible to extend our 2-D approach to solving the problem on point sets in l-D for any fixed integer $l \geq 3$. In this section, we show this extension to 3-D. In particular, we present an $O(n \log n + dn \log d^*)$ time algorithm for computing the NNE-graph on an input set V of n points in 3-D, where d is the $\Omega(\frac{\log n}{\log \log n})$-th largest degree and d^* is the largest degree in the NNE-graph. The ideas of our 3-D algorithm are very similar to those for the 2-D case (i.e., exponential search and maintaining the convex hull of '2-D' points on the surface of a unit sphere). We first illustrate the main ideas and key operations of our 3-D approach by presenting a preliminary $O(n^2 + m \log d^*)$ time algorithm, where m is the edge number of the output NNE-graph. We then give the $O(n \log n + dn \log d^*)$ time algorithm.

For a point p of V in 3-D, a unit sphere S_p centered at p is used to project the points of $V - \{p\}$ onto the surface of S_p along the rays emitted at p. A *pyramid* $._p(X')$ for a 3-D point set

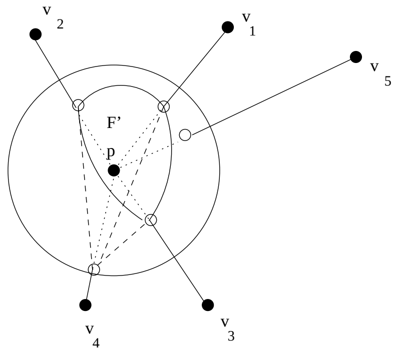

Figure 4: The nearest neighbors of p in 3-D are projected into a unit sphere.

X' with apex p the convex hull of p and X'. Furthermore, the convex hull of the points of X' projected on the surface of the unit sphere S_p is the *base* of .$_p(X')$. Let $CH_{S_p}(X')$ denote that base. .$_p(X')$ is a convex polygonal cone. It is obvious that for any set $X_p(j)$, p is contained in $CH(X_p(j))$ if and only if p is contained in a tetrahedron defined by four vertices of $CH(X_p(j))$. Therefore, we have the following lemma.

Lemma 5 *For any set $X_p(j)$, p is contained in $CH(X_p(j))$ in 3-D if and only if the following holds: There exists a plane P in 3-D that passes through the center p of semi-sphere S_p, such that the (2-D) convex hull $CH(P \cap CH_{S_p}(X_p(j)))$ of the intersection of P and $CH_{S_p}(X_p(j))$ contains p (clearly, the convex hull $CH(P \cap CH_{S_p}(X_p(j)))$ is on the plane P).*

Note that in the degenerate case, p can be contained in a line segment defined by some points of $P \cap CH_{S_p}(X_p(j))$, where P is a plane as defined in Lemma 5.

Again, the decision question on any point $p \in V$ and any nearest neighbor set $X_p(j)$ of p in V is crucial to our exponential search process in 3-D. Lemma 5 seemingly implies that some complicated testing is required, it is actually quite simple to perform the test. In fact, the decision question on p and $X_p(j)$ can be answered in $O(|X_p(j)| \log |X_p(j)|) = O(j \log j)$ time (i.e., $O(\log j)$ time per point in $X_p(j)$).

We first compute the '2-D' convex hull $CH_{S_p}(X_p(j))$ on the surface of the unit sphere S_p. We compute $CH_{S_p}(X_p(j))$ iteratively, as follows. We first take two arbitrary points from $X_p(j)$; if these two points define a line segment that contains p (i.e., the degenerate case), then we are done (i.e., the answer to the decision question on $X_p(j)$ is "yes"). Let $CH_{S_p}(X')$ be the convex hull on the surface of S_p computed so far, with $X' \subseteq X_p(j)$, and consider the next point q from $X_p(j)$. Here, unless otherwise specified, we will not distinguish the point q from its projected point on the surface of S_p. If q is contained in $CH_{S_p}(X')$, then $CH_{S_p}(X')$ is unchanged, i.e., $CH_{S_p}(X' \cup \{q\}) = CH_{S_p}(X')$. Note that checking whether q is in $CH_{S_p}(X')$ can be done easily in $O(\log |CH_{S_p}(X')|)$ time. If q is not contained in $CH_{S_p}(X')$, then there are two cases: (1) The *antipodal point* $A(q)$ of q (i.e., the point on the surface of S_p that is on the ray starting at the center point p of S_p and along the opposite direction of q) is contained in $CH_{S_p}(X')$, and (2) the antipodal point $A(q)$ of q is not contained in $CH_{S_p}(X')$. In Case (1), there are four projected points of $X_p(j)$ (one of which is q) that form a tetrahedron containing p, and thus we know by Lemma 5 that no semi-sphere of S_p can properly contain $CH_{S_p}(X_p(j))$. (Note that checking whether Lemma 5 holds in this situation is simply done by deciding whether $A(q) \in CH_{S_p}(X')$, which can be easily carried out in $O(\log |CH_{S_p}(X')|)$ time.) In Case (2), we update $CH_{S_p}(X')$ by computing the two common tangents between $CH_{S_p}(X')$ and q on the surface of S_p, and let $X' = X' \cup \{q\}$ (note that the resulted $CH_{S_p}(X')$ should not contain the antipodal point $A(q)$ of q). Computing these two common tangents takes $O(\log |CH_{S_p}(X')|)$ time. Therefore, processing each point q in $X_p(j)$ takes $O(\log |X_p(j)|) = O(\log j)$ time. If the iterative process runs through all points of $X_p(j)$ (in $O(j \log j)$ time), then we know that no semi-sphere of S_p can properly contain $CH_{S_p}(X_p(j))$, and hence the decision question has a "no" answer.

The preliminary algorithm for the 3-D case performs a binary search separately for each point p of V, to compute the nearest neighbor set X_p^* using the selection algorithm. Due to the nature of the binary search and the $O(j \log j)$ time bound of the decision question on each point p and its nearest neighbor set $X_p(j)$ of p, it takes $O(n + |X_p^*| \log |X_p^*|)$ time to compute X_p^* for each point p in V; note that $|X_p^*|$ is not larger than the degree of p in the output NNE-graph and $\sum_p X_p^* < 2^m$. Therefore, we have the following result.

Theorem 3 *Given a set V of n points in 3-D, the NNE-graph can be computed in $O(n^2 + m \log d^*)$ time, where m is the edge number and d^* is the largest degree of the output NNE-graph.*

Our improved algorithm for the 3-D case is also similar to the corresponding one for the 2-D case. Our main tool is the 3-D version of Lemma 4: Step 1 establishes a well-separated pair decomposition of point set V in $O(n \log n)$ time. We also set the number of nearest neighbor searched in Step 2 to be $g \cdot \frac{\log n}{\log \log n}$ for $g = 1$ initially.

In the iterative process, Steps 2, 3, and 4 are the same as in 2-D case. The searching of $g \cdot \frac{\log n}{\log \log n}$ nearest neighbor (X_p^*) for each point p in V takes $g \cdot \frac{\log n}{\log \log n} \cdot \log \frac{\log n}{\log \log n}$ time for those type whose search range is already known. In total $O(ng \cdot \frac{\log n}{\log \log n} \cdot \log \frac{\log n}{\log \log n})$ time for all $n - O(\frac{\log n}{\log \log n})$ such points in V. Searching the nearest neighbor set X_p^* for each point p in V whose search range is not yet known takes $O(n + d \cdot \log d^*)$ time, and in total $O(n \cdot \frac{\log n}{\log \log n} + \frac{\log n}{\log \log n} \cdot d \cdot \log d^*)$ time for all $O(\frac{\log n}{\log \log n})$ such points in V, where d^* is the largest degree of the output NNE-graph. Therefore, in summary, we have the following result.

Theorem 4 *Given a set V of n points in 3-D, the NNE-graph can be computed in $O(n \log n + nd \log d^*)$ time, where d is the $\Omega(\frac{\log n}{\log \log n})$-th largest degree of the NNE-graph and d^* is the largest degree of the NNE-graph.*

5 Concluding Remarks

In this paper, we present a worst-case $\Theta(n^2)$ time optimal algorithm for finding the NNE-graph of a set of n points in 2-D. We also present algorithms for finding the NNE-graphs of a point set in both 2-D and 3-D. These algorithms are optimal when the $(\log n)$-th largest vertex degree of the NNE-graph is bounded by $O(\log n)$ in 2-D case and when the $(\frac{\log n}{\log \log n})$-th largest vertex degree of the NNE-graph is bounded by $O(\frac{\log n}{\log \log n})$ in 3-D case.

Thus far, we have ignored points on the boundary of point set V. Points near the boundary cannot be embraced by the convex hull of their neighbors. So, which points near the boundary should be considered as in NNE-graph? For statistical applications, it is important to study the unbiased statistical properties of the NNE-graph of a set of points within a bounded region. Working within a bounded region, our procedure for finding the NNE-neighbors of p, where p is distanced d from the boundary of the given bounded region, can stop when the nearest neighbor being considered is distanced greater than d from p. Another way to deal with the boundary point b is to assign b by degree j and to connect b to j nearest neighbors of b, where j is the average degree of the NNE-graph of the internal point set V. This will be consistent with the connections among the internal points in V.

Our improved algorithms are sensitive to the structure of the NNE-graph. For example, when $d = g \cdot \log n$, the size of the NNE-graph is bounded above by $O(gn \log n)$ for $1 \leq g \leq \frac{n}{\log n}$. However, it cannot bound the size of NNE-graph from below. It is interesting if one can find an $O(n \log n + m)$ time output-sensitive algorithm, where m is the number of edges in the NNE-graph.

References

[1] A.J. Baddeley and B.W. Silverman, A cautionary example on the use of second-order methods for analyzing point patterns, *Biometrics*, 40, pp. 1089-1093, 1984.

[2] P.B. Callahan and S.R. Kosaraju, A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields, *Journal of the ACM*, 42(1), pp. 67–90, 1995.

[3] F. Chin and C.A. Wang, Finding the constrained Delaunay triangulation and constrained Voronoi diagram of a simple polygon in linear time, *SIAM Journal of Computing*, 28, 2, pp. 471-486, 1998.

[4] S.N. Chiu and I.S. Molchanov, A new graph related to the directions of nearest neighbours in a point process, *Advances in Applied Probability*, 35(1), 2003.

[5] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd edition, McGraw-Hill, New York, 2001.

[6] F. Harary, *Graph Theory*, Addison-Wesley, 1971.

[7] F.P. Preparata and M.I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, 1985.