

Optimal k -support Coverage Paths in Wireless Sensor Networks

ShaoJie Tang

Department of Computer Science
Illinois Institute of Technology
Chicago, IL, 60616.
Email: stang7@iit.edu

XuFei Mao

Department of Computer Science
Illinois Institute of Technology
Chicago, IL, 60616.
Email: xmao3@iit.edu

Xiang-Yang Li

Department of Computer Science
Illinois Institute of Technology
Chicago, IL, 60616.
Email: xli@cs.iit.edu

Abstract—Coverage is a fundamental problem in wireless sensor networks since sensors may be spread in arbitrary manner. In this paper, we addressed the k -support coverage problem. Most of the existing works assume that the coverage degree is 1, *i.e.* every point in a certain path falls within the sensing range of at least one sensor node. In this work, we focus on the case when we require that every point on the path is covered by at least k sensors, while optimizing certain objectives. We give an optimal polynomial-time algorithm and prove that the time complexity of our algorithm is $O(k^2n^2)$. To the best of our knowledge, this is the first polynomial time algorithm finding an optimum k -coverage path in sensor networks with general sensing radius and general k .

I. INTRODUCTION

In many wireless sensor network applications, we are often required to find a path from a source point to a destination point such that the path is the best choice under a certain quality measurement. For example, when a sensor network is used to monitor a working environment and some emergency happens, the sensor network should provide a path that will guide the users leaving from a working place to a safe exit. This kind of questions is generally called a *coverage* problem (which is also considered as a measure of quality of service (QoS)) in wireless sensor networks.

In this paper, our goal is to find areas of high observability from sensors and identifying the best support and guidance regions. Given a wireless sensor networks, we are interested in designing algorithms addressing the following question: Finding a path connecting a source point S and a destination point D , which maximizes the smallest observability of all points on the path. It is called the *best coverage problem* [3]; The observability of a point p depend on the applications. Most of the existing work [3], [6], [10] in this area focus on the 1-coverage problem, which means any point on the region falls in the sensing range of at least one sensor. The observability of a point p is simply the shortest distance from p to a sensor. In this paper, we consider a more general case of sensor coverage problem, which is, given a set of sensors deployed in a 2-dimensional region Ω , we would like to determine how well this target area is k -covered. In this case, the observability of a point p is the shortest distance from p to the k nearest sensor. To the best of our knowledge, [4] and [5] are the only two works which address the problem of

finding an optimal k -covered path. In [4], Mehta *et.al* suggest that the worst-case k -coverage problem may be addressed by adopting the k -th nearest Voronoi diagram. However, no any algorithm and theoretical results were given. In [5], Fang *et.al* gave a polynomial time algorithm to identify a k -covered path based on binary search and growing disk techniques. Unfortunately, their algorithms cannot find the optimum solution. Furthermore, they assume that k is a given constant which may reduce the generality of their algorithm.

The main contribution of our paper is that we give provably polynomial time algorithm for k -coverage in sensor networks with general sensing radius and general k . We combined computational geometry techniques, specifically the k -th nearest Voronoi diagram, with graph theoretical algorithmic techniques. Since the properties of k -th nearest point Voronoi diagram is quite different from the ordinary Voronoi diagram and not well studied before, in our paper, we first propose an efficient algorithm to generate the k -th nearest Voronoi diagram. More importantly, we give a number of theoretical results on the properties of k -th nearest point Voronoi diagram which is never addressed before, for example, we show that the total number of KNP-Voronoi edges is $O(k^2n)$ and the number of edges of each KNP-Voronoi cell is $O(n)$.

The rest of the paper is organized as follows. We first review the related results in Section II. In Section III, we define terms and notations used in presenting our algorithms. We present the first polynomial time algorithms that solve the best k -coverage problem efficiently in Section IV. We introduce our extensive simulation results in Section V and conclude our paper in Section VI.

II. RELATED WORK

In order to evaluate the quality of coverage of the sensor network, Meguerdichian *et.al* [3] formulated the coverage problem to be a the best-case coverage (maximum support) problem. They observed that an optimal solution for the maximum support problem is a path which lies along the edges of the Delaunay triangulation [8] [9] and presented centralized algorithms for this problem. Mehta *et.al* [4] improved these algorithms and made them more computational efficient. Li *et.al* [6] showed that the maximum support path can be constructed by using edges that belong to the

relative neighborhood graph(RNG) of sensor set, which is an improvement since the RNG is a subgraph of the Delaunay triangulation and can be constructed locally. In addition, a distributed algorithm based on RNG was proposed to solve the best-case coverage problem. On the other side, Meguerdichian *et.al* [3] implied that a variation of the localized exposure algorithm presented in [8] can be used to solve the worst case coverage problem locally. Another localized algorithm with more practical assumptions was proposed by Huang *et.al* [7].

For the general coverage problem, Huang *et.al* [7] studied the problem of determining if the area is sufficiently k -covered which means every point in the target area is covered by at least k sensors. The extension of this problem to three-dimensional sensor networks was studied and solved in [13], by Huang *et al.* The connected k -coverage problem was addressed in [14] by Zhou *et.al*. They gave a centralized greedy algorithm for this problem which is near-optimal. A distributed algorithm has also been proposed in [14]. Xing *et.al* [15] explored the problem of energy conservation while maintaining both desired coverage degree and connectivity. Some studies focused on the relationship between the coverage degree k , the number of sensors n and the sensing radius r . Kumar *et.al* [16] considered the problem of determining the appropriate number of sensors that are enough to provide k -coverage of a region when sensors are allowed to sleep most of their lifetime. Wan *et.al* [17] analyzed the probability of the k -coverage when the sensing radius or the number of sensors changes while taking the boundary effect into account. To the best of our knowledge, [4] and [5] are the only two works which address the problem of finding an optimal k -covered path. However, as we mentioned before, no details of the proposed algorithm were given in [4] and the time complexity of the algorithm in [5] can not be bounded if optimum solution is required.

III. PRELIMINARIES

A. Problem Formulation

We assume that there is a set of n identical wireless sensor nodes $U = \{u_1, u_2, \dots, u_n\}$ distributed inside a continuous two-dimensional field Ω , where for each sensor node u_i , its location (x_i, y_i) is known. We assume that the field Ω is given as a simple 2-dimensional polygon. A pair of points S and D are given as the source point and the destination point.

Definition 1: Given a point v in the field Ω and the set of sensors U , the **k -th distance** of v , respect to U , denoted as $\ell_k(p, U)$, is defined as the Euclidian distance from v to its k -th nearest sensor node in U .

Definition 2: Given a path P connecting a source point S and a destination point D , the **k -support** of P , denoted by $S_k(P)$, is defined as the *maximum* k -th distance experienced by an agent traversing along P . In other words, $S_k(P) = \max_{p \in P} \ell_k(p, U)$.

In this paper we will study the following question.

Problem 1: Minimum k -support Path (Best Case Coverage) Problem: Given a source point S and destination point

D , find a path P in the field Ω to connect S and D such that $S_k(P)$ is minimized.

B. The k -th Nearest Point Voronoi Diagram

Given a set of identical sensor nodes U distributed in field Ω , we assign a geometry point p in the field to the sensor node $u_i \in U$ if u_i is the k -th nearest sensor node from p . Following this assignment rule, we assign all points in the field to at least one sensor node in U . Resultantly, we obtain a collection of regions associated with sensor nodes in U , denoted by $\mathbb{V}_k = \{V_k(u_1), \dots, V_k(u_n)\}$, which forms a tessellation. We call the tessellation \mathbb{V}_k the *k -th nearest point Voronoi diagram* (KNP-Voronoi diagram) generated by U , and the region $V_k(u_i)$ the k -th nearest-point Voronoi region of node u_i . Notice that $V_k(u_i)$ may be disconnected and composed by several independent polygons as shown in [1]. Here we call each independent polygon *k -th nearest Voronoi cell* (KNP-Voronoi cell) of node u_i and denote it as $C_k(u_i)$. The boundary of each cell as *k -th nearest-point Voronoi edge* (KNP-Voronoi edge) and the intersection of boundary edges as *k -th nearest point Voronoi vertex* (KNP-Voronoi vertex). The following notations will also be used in our algorithms.

Definition 3 (Perfect Support Location): The perfect support location of each KNP-Voronoi edge is defined as the point on this edge which is closest to its corresponding sensor node(s).

Notice that there is one and only one perfect support location which is easy to find on each KNP-Voronoi edge.

IV. BEST CASE COVERAGE: MINIMUM k -SUPPORT PATH

In this section, we address the minimum k -support path problem. We are going to give an algorithm to compute the minimum k -support path and further prove that its time complexity is $O(k^2 n^2)$.

A. Preliminaries

Before we give the algorithm, we first introduce some useful theoretical results which will help us to prove the correctness of our algorithm.

Theorem 1: Based on any given path P_1 connecting source node S and destination node D , we can always construct another (maybe same) path P_2 composed by only a finite number of line segments such that

$$S_k(P_2) \leq S_k(P_1).$$

Proof: Given a set of sensor nodes U and its corresponding k -NP-Voronoi diagram. For any given path P_1 connecting S and D , we decompose it into a set of partial paths such that each partial path p_{ab} (starting at a and ending with b) is entirely contained in some k -NP-Voronoi cell $C_k(u_i)$ (it is also considered as "contained" if it is overlapping with its k -NP-Voronoi edge) with its end points a and b located at $C_k(u_i)$'s k -NP-Voronoi edge. Essentially, we will prove that using line segment ab to replace the original partial path p_{ab} will lead to a no worse result, in other words, the k -support of this line segment is no larger than that of p_{ab} . Notice that $S_k(p_{ab})$ is no

less than $\max\{|u_i a|, |u_i b|\}$ where $|u_i a|$ denotes the Euclidian distance between a and its k -th nearest sensor node u_i . On the other hand, line segment ab will be entirely contained in a disk centered at u_i with radius $\max\{|u_i a|, |u_i b|\}$ such that $S_k(ab)$ is no more than $\max\{|u_i a|, |u_i b|\}$. Thus, the k -support of line segment ab is $\max\{|u_i a|, |u_i b|\}$ which already achieves the lower bound. Hence, using line segment ab to replace the original partial path will lead to a no worse result.

Because the $S_k(P_1)$ is equal to the maximum one among all its partial paths' k -support, after we replace each partial path with its corresponding line segment, the k -support of the new constructed path P_2 must be no larger than $S_k(P_1)$. ■

Theorem 2: Based on any given path P_1 connecting source node S and destination node D , we can construct another path P_3 composed by only line segments, each of these line segments' end points are located on the perfect support location of the KNP-Voronoi edges they are crossing such that

$$S_k(P_3) \leq S_k(P_1).$$

Proof: Given any path P_1 connecting S and D , we first replace each partial path by line segment, Theorem 1 guarantees that this modification will not increase the k -support. Next we will prove that moving the end points of each line segment to the perfect support location of its corresponding KNP-Voronoi edge will not increase the k -support. Here we have two cases needed to address, illustrated by Figure 1.

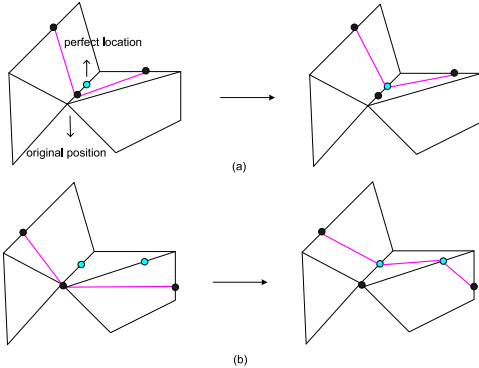


Fig. 1. Dark nodes denote the original position of the end points of a segment in the path. Light nodes denote the perfect support location of the corresponding KNP-Voronoi edge. a) When the original position of the end point is not located at any Voronoi vertex. b) When the original position of the end point is located at some Voronoi vertex.

First, consider the case that the endpoint is not located at some KNP-Voronoi vertex, see Figure 1 (a) for illustration. As shown in the proof of Theorem 1, the k -support of any line segment ab contained in some KNP-Voronoi cell is $\max\{|va|, |vb|\}$, then according to the definition of the perfect support location, we conclude that if we move a or b to the perfect support location of its corresponding KNP-Voronoi edge, the k -support will not increase. It follows that if we move every this kind of end point to its perfect support location, the k -support of any line segment will not increase as long as the movement does not lead to crossing new KNP-Voronoi

cells(generating new line segments). Since we assume that the endpoint is not located at any KNP-Voronoi vertex, then according to the fact that each KNP-Voronoi cell is convex, it is impossible to cross any new cell by moving the end point to the perfect location of its corresponding KNP-Voronoi edge. This finishes the proof of this case.

The second case is that the endpoint is located at some KNP-Voronoi vertex, see Figure 1 (b) for illustration. We first choose one of its adjacent KNP-Voronoi edges as its corresponding KNP-Voronoi edge (since it is located at the intersection of more than one KNP-Voronoi edges) and move it to the perfect support location of that edge as we explained before. However, this movement may lead to crossing some new KNP-Voronoi cells, and new line segments as well as their endpoints will be generated. To handle this case, we move the new generated endpoints to their corresponding perfect support locations until there are no new line segments generated. Now the same proof used in Case 1 can be applied here. Then we can conclude that this chain of movements will not increase the k -support.

Finally, we get a new path P_3 composed by only line segments with all of their end points located on their KNP-Voronoi edge's perfect support location such that $S_k(P_1) \leq S_k(P_3)$. This finishes the proof. ■

Theorem 3: There is one minimum k -support path which is composed by only line segments and all of these line segments' end points are located at the perfect support location of the KNP-Voronoi edges they are crossing.

Proof: This theorem is straight forwarded from previous two lemmas, given any minimum k -support path, we decompose it into partial paths each of which is entirely contained in some KNP-Voronoi cell.

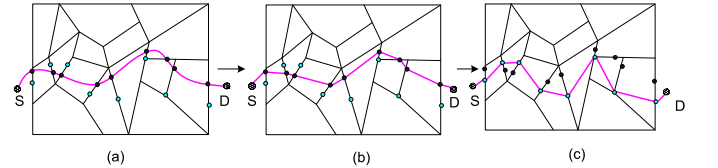


Fig. 2. (a) Original path (b) Replaced by line segments (c) Adjust to perfect support location

If every partial path is already a line segment with its endpoints located at the perfect support location of the corresponding KNP-Voronoi edge, we done. If not, as shown before, by replacing each partial path by a line segment and do some necessary end points movement, we can get another path whose k -support is no larger than the given minimum k -support path, Figure 2. (a) \rightarrow (b) \rightarrow (c) illustrates this process. It implies that the new constructed path is another minimum k -support path. This finishes the proof. ■

B. Our Algorithm

Now we are ready to present our polynomial time algorithm to compute the minimum k -support path within time $O(k^2 n^2)$. Given a set of sensor nodes U and the continuous field Ω , we first use Algorithm 1 to compute its KNP-Voronoi diagram G

in time $O(k^2 n \log n)$. Then we construct a new graph G' based on G and apply Algorithm 2 to G' to compute the minimum k -support path, which can be done in time $O(k^2 n^2)$.

1) *Compute the KNP-Voronoi Diagram:* Before we present our algorithm to compute the KNP-Voronoi diagram for U , we first introduce some new definitions.

Definition 4 (The Order- k Voronoi Diagram): The order- k Voronoi diagram is a partition of the plane into regions such that points in each region have the same k closest sensor nodes $U^{[k]}$. Each polygon is named **order- k voronoi cell** $C_{ok}(U_i^{[k]})$ corresponding to a set of k sensor nodes $U_i^{[k]}$.

Definition 5 (The Farthest Point Voronoi Diagram): The farthest point Voronoi diagram is a special case of k -th nearest point diagram when $k = n - 1$. It is a partition of the plane into polygons such that points in each polygon have the same farthest sensor node. Each polygon is called a **Farthest Voronoi Cell**.

The time complexity of our algorithm is $O(k^2 n \lg n)$ which is proven in the following section.

Algorithm 1 Computing KNP-Voronoi Diagram

Input: Set of sensor nodes U .

Output: U 's KNP-Voronoi Diagram.

- 1: Compute U 's order- k Voronoi diagram
 - 2: **for** each order- k Voronoi cell $C_{ok}(U_i^{[k]})$ **do**
 - 3: Compute the farthest point Voronoi diagram using corresponding k sensor nodes in $U_i^{[k]}$
 - 4: **for** each KNP-Voronoi edge e **do**
 - 5: **if** its two neighbor polygons belong to same sensor node u_i **then**
 - 6: merge these two polygons into one polygon which still belongs to u_i
 - 7: **for** each sensor node u_i **do**
 - 8: Return the union of all polygons belongs to u_i as its KNP-Voronoi Cell
-

2) *Compute the Minimum k -Support Path:* In this subsection, we assume U 's KNP-Voronoi diagram is already generated using Algorithm 1. We then present our algorithm to compute the minimum k -support path based on Theorem 3.

As shown in Theorem 3, since there must exist one minimum k -support path consisting of only line segments whose end points are located on the perfect support location of the KNP-Voronoi edges that they cross, then we only need to consider all the paths that using only line segments connecting the perfect support locations of the KNP-Voronoi edges, among which the one with the minimum k -support must be one of the desired minimum k -support paths.

First, we construct a new graph G' based on k -th nearest point Voronoi diagram G as follows: First, for each KNP-Voronoi edge e in G , we have a corresponding node u' in G' . In other words, we establish a one to one mapping between each KNP-Voronoi edge in G and each node in G' . Additionally, we have another two special nodes S' and D' corresponding to source point S and destination point D .

Furthermore, each node u' is assigned a weight $\omega(v')$ such that $\omega(u') = \text{dist}[e]$ where $\text{dist}[e]$ denotes the Euclidian distance between e 's perfect support location and its corresponding k -th nearest sensor node; for S' (or D'), their weight $\omega(S')$ (or $\omega(D')$) equals to the Euclidian distance between S (or D) and its k -th nearest sensor node.

Second, adding an edge between any two nodes u' and v' in G' if and only if their corresponding KNP-Voronoi edges belong to the same KNP-Voronoi cell in G . For those two special nodes S' and D' , we add an edge between S' (or D') and u' if and only if u' 's corresponding KNP-Voronoi edge e is of the KNP-Voronoi cell that contains node S (or D). And each edge $u'v'$ has weight $\omega(u'v') = \max\{\omega(u'), \omega(v')\}$.

Then, we use Algorithm 2 (modified from Dijkstra's shortest path algorithm) to identify a path P' in G' to connect S' and D' such that the maximum weight among its edges' is minimized. Finally, we get one minimum k -support path P in G by replacing each vertex u' appeared in P' by e' perfect support location in G and use line segments to connect them.

Algorithm 2 The Minimum k -Support Path Algorithm

Input: G' , source node S' , destination node D' .

Output: Minimum k -support Path Connecting S' and D' .

- 1: **for** each vertex v' in graph G' **do**
 - 2: $k\text{-support}[v'] := \text{infinity}$
 - 3: $\text{previous}[v'] := \text{undefined}$
 - 4: $k\text{-support}[S'] := \omega(S)$
 - 5: $Q :=$ the set of all nodes in graph G'
 - 6: **while** Q is not empty **do**
 - 7: $u' :=$ node in Q with smallest $k\text{-support}[]$
 - 8: remove u' from Q
 - 9: **for** each neighbor v' of u' : **do**
 - 10: $alt := \max\{\omega(u', v'), k\text{-support}[u']\}$
 - 11: **if** $alt < k\text{-support}[v']$ **then**
 - 12: $k\text{-support}[v'] := alt$
 - 13: $\text{previous}[v'] := u'$
 - 14: **Return** P'
-

Correctness Proof: Based on Theorem 3, we know that among all of the paths which are composed by only line segments with each of the segments' endpoints located at some KNP-Voronoi edges' perfect support locations, the one who has the minimum k -support must be the minimum k -support path. Next, we only need to show that the path P we finally get indeed satisfies the preceding condition.

Obviously, the path P computed by our algorithm only uses line segments to connect the perfect support locations of the KNP-Voronoi edges it crosses. Next we prove that P 's k -support is indeed minimized among all of those paths. As we know, $S_k(P)$ has the maximum k -support among all its line segments, so P is minimum k -support path iff it minimized maximum k -support among its line segments. Since P' computed by Algorithm 2 minimized the maximum weight among its edges where the weight of each edge $u'v'$ denotes the k -support of the line segment connecting the

perfect support locations of the KNP-Voronoi edges in G mapped by u' and v' . This implies that the final path P we computed has the minimum k -support among all those kind of paths. Therefore, P is one minimum k -support path.

C. Time Complexity Analysis

In this subsection, we prove that the time complexity of our algorithm is $O(k^2n^2)$. As shown before, our algorithm is composed by two parts, one is to compute the KNP-Voronoi diagram, and the other one is to compute the minimum k -support path based on the KNP-Voronoi diagram. In the following part, we will analyze the time complexity of these two parts respectively. We first give a bound of the time complexity for first part,

Lemma 4: The time complexity of Algorithm 1 which is used to compute the KNP-Voronoi diagram is $O(k^2n \lg n)$.

Proof: Using the algorithm proposed in [2], we can compute the order- k Voronoi diagram of n sensor nodes within time $O(k^2n \lg n)$. In the second step, we compute the farthest Voronoi diagram in each order- k Voronoi cell $C_{ok}(U_i^{[k]})$ of the k sensor nodes in $U_i^{[k]}$. This operation will cost $O(k \lg k)$ for each order- k Voronoi cell as proven in [18]. Since there are $O(nk)$ order- k Voronoi cells which has been proven in [2], the time complexity of the second step is $O(k \lg k) \times O(nk) = O(k^2n \lg k)$. In the third step, we may do some merge operations for each KNP-Voronoi edge, this operation will cost $O(k^2n)$ time since there are at most $O(nk^2)$ KNP-Voronoi edges in KNP-Voronoi diagram as shown in Lemma 5 and each merge operation is constant time complexity. So the time complexity for Algorithm 1 is $O(k^2n \lg n) + O(k^2n \lg k) + O(k^2n) = O(k^2n \lg n)$. ■

Next, we show the time complexity of the second part. Since the time complexity of the second part is decided by the number of KNP-Voronoi cells, the number of KNP-Voronoi edges per KNP-Voronoi cell, etc. It is necessary for us to first address some properties of KNP-Voronoi diagram.

Lemma 5: For a set of n sensor nodes U on the field Ω and its KNP-Voronoi diagram G , the total number of KNP-Voronoi edges is $O(k^2n)$ and the number of edges of each KNP-Voronoi cell is $O(n)$.

Proof: We know that the total number of KNP-Voronoi edges in KNP-Voronoi diagram is $O(kn)$ and the number of KNP-Voronoi edges in the farthest Voronoi diagram of k sensor nodes is $O(k)$ which have been proven in [2] and [1] respectively. So the total number of KNP-Voronoi edges computed from Algorithm 1 is $O(nk) + O(nk) \times O(k) = O(k^2n)$. We conclude that the total number of KNP-Voronoi edges of KNP-Voronoi diagram is $O(k^2n)$.

Next we prove that the number of KNP-Voronoi edges of each KNP-Voronoi cell is $O(n)$. For any sensor node $u_i \in U$, we construct the bisectors between u_i and all the other sensor nodes in U and the open half-plane defined by the sectors which does not contain u_i is named farther half-plane. The KNP-Voronoi cell of u_i is the area which is intersected by exactly $k - 1$ farther half-planes. This observation comes from the definition of KNP-Voronoi cell. Since there are no more

than $n - 1$ bisectors for each sensor node, the total number of KNP-Voronoi edges of each KNP-Voronoi cell is no more than $n - 1$. This finishes the proof. ■

Lemma 6: The time complexity of the algorithm to compute the minimum k -support path based on KNP-Voronoi diagram is $O(k^2n^2)$

Proof: From Lemma 5, we can prove that there are at most $O(k^2n)$ vertices in G' . This is because there are at most $O(k^2n)$ KNP-Voronoi edges in G and each vertex in G' has one and only one corresponding KNP-Voronoi edge in G .

Second, the total number of edges in G' is at most $O(k^2n^2)$. For each k -th Voronoi cell in G , we have a complete subgraph in G' by connecting all the vertices whose corresponding boundary edges belong to the same cell to each other, and from Lemma 5, we know that each cell in G can only generate at most $O(n^2)$ edges in G' and there are at most $O(2k^2n/n) = O(k^2)$ such cells. Furthermore, the source node S and D can only generate at most $O(k^2n)$ edges in G' . Thus, the number of edges in G' is at most $O(n^2 \times k^2) + O(k^2n) = O(k^2n^2)$.

Consequently, time complexity on conversion from G to G' is at most $O(k^2n^2)$. Time complexity of Algorithm 2 is $O(|V| \lg |V| + |E|)$ which is same as Dijkstra's shortest path algorithm's, where $|V|$ denotes the number of vertices and $|E|$ denotes the number of edges. From the previous discussion, we know that $|V| = O(k^2n)$ and $|E| = O(k^2n^2)$ in G' , thus, the time complexity of Algorithm 2 is $O(k^2n^2)$. Consequently, the overall time complexity of our method is $O(k^2n^2)$. This finishes the proof of this theorem. ■

Combining Lemma 4 and Lemma 6 together, we have the following theorem.

Theorem 7: The time complexity of our algorithm to compute a minimum k -support path is $O(k^2n^2)$.

V. SIMULATION RESULTS

We have implemented and test our proposed k -support algorithm to find the optimum k -support paths by Matlab R2006a [19]. In this section, we will introduce some sampling results and present an overview and analysis of the application.

A. Experimentation Platform - Sample Results

In our simulation, a set of n wireless sensors is randomly and uniformly deployed in the target square region. The following figures shows the main procedure of our algorithms to find a minimum 3_{rd} -support path in a 500×500 meter² square region where 6 sensor nodes are randomly deployed. Fig. 3(a) shows the 3_{rd} nearest point Voronoi diagram for 6 sensors (stars). We use different colors to distinguish k^{th} nearest point Voronoi cells ($k = 3$ in this example) for different sensors and each 3_{rd} nearest point Voronoi cell has the same color as its corresponding sensor has. The two white nodes inside of the square region denote the position of departure and target points respectively.

As we can see from Fig. 3(a), each sensor could have several corresponding 3 nearest point Voronoi cells. Fig. 3(b) presents the corresponding new graph of Fig. 3(a) induced by our algorithm. As we can see, all boundaries in the original

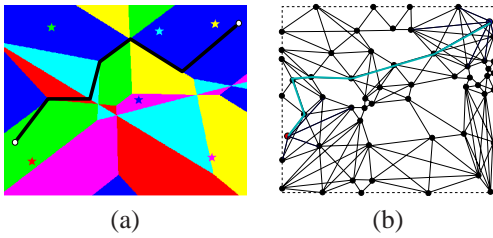


Fig. 3. (a) 3_{rd} nearest point Voronoi cell for 6 sensors in a $500 \times 500 \text{ meter}^2$ square region. Stars denotes sensor nodes which have the same color as their 3_{rd} nearest point Voronoi cell. White points denote the positions of departure point and target point respectively. The final corresponding minimum 3-support path which is marked with black line. (b) Resultant graph. The red and blue points denote the original departure point and target point separately. The dash line is the original boundaries of the square region. The final resultant Min-Max weighted shortest path which is marked with blue line.

3_{rd} nearest point Voronoi cell will become points in the new graph, in addition those points which are corresponding to the boundaries belong to the same single Voronoi cell form a clique. The departure (target) point becomes a new node in the graph and has links to all nodes which are corresponding to the boundaries of the 3_{rd} nearest point Voronoi cell containing the departure (target) point. The final min-max weight path is shown in Fig. 3(b) and the corresponding minimum 3-support path on the original square region is shown in Fig. 3(a).

We also test our algorithms where we deployed different number of sensor nodes and compute optimum k -support paths for different k . In our simulation, we let the number of sensors n increase gradually from 5 to 30 in steps of 5, and find the optimum k -support paths for each $k \in \{1, 3, 5\}$.

Fig. 4 shows that minimum k -support decreases with the increment of the number of sensors.

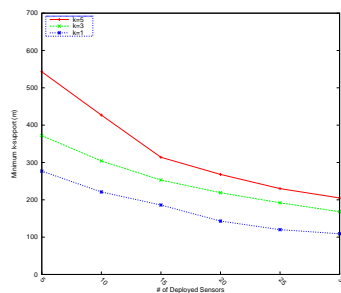


Fig. 4. Results for k -support.

Clearly, this result can be used to estimate the coverage quality if the number of sensors and required coverage degree are given. If the sensing radius for each sensor node is fixed, it is better to deploy more sensor nodes in order to get better coverage quality. On the contrary, if the sensor can adjust its sensing (coverage) radius by power adjustment, the result presented here could be used to estimate the transmission power needed by the sensor such that different requirement (k -support) can be satisfied. In addition, we found that when the number of sensors exceed some threshold value (around 20 in our case), the decreasing trend of curve become slow.

This illustrates that there is a tradeoff between the number of sensor nodes needed and the desired coverage quality if the sensing radius of each sensor node is fixed.

VI. CONCLUSION

In this paper, we proposed polynomial time algorithms for k -coverage problem in wireless sensor networks. Our algorithms can efficiently find a path connecting two points in a sensor network with the best observability (*i.e.*, minimizing the maximum observability of all points on the path). In studying our algorithms, we proposed a number properties for k -nearest point Voronoi diagram. These properties may be of independent interests. We also give the detailed procedure and instances to illustrate our algorithms by giving the extensively simulation results. It is interesting to design algorithms that can address the coverage problem when the sensing abilities of sensors are heterogeneous.

REFERENCES

- [1] ATSU. OKABE, B. BOOTS, K. SUGIHARA AND S.N. CHIU Spatial Tessellations. *Wiley Series In probability and Statics*, 2000.
- [2] DER-TSAI LEE On k -Nearest Neighbor Voronoi Diagrams in the Plane. *IEEE Trans. Comput.*, 1982.
- [3] S. MEGUERDICHIAN AND F. KOUSHANFAR M. POTKONJAK AND M. B. SRIVASTAVA Coverage problems in wireless ad-hoc sensor networks. *Proceedings of IEEE INFOCOM 01*, pp. 139C150, 2001.
- [4] MEHTA, D.P. LOPEZ, M.A. AND L. LIN Optimal coverage paths in ad-hoc sensor networks. *IEEE ICC 03*, pp. 507-511, 2003.
- [5] C. FANG AND C.P. LOW Redundant Coverage in Wireless Sensor Networks. *IEEE ICC*, 2007.
- [6] X. LI, P. WAN, AND O. FRIEDER, Coverage in wireless ad-hoc sensor networks. *IEEE Transaction on Computers*, 52(6), pp. 753-763, 2003
- [7] C. HUANG AND Y. TSENG, The coverage problem in a wireless sensor network. *ACM Intl Workshop on Wireless Sensor Networks and Applications (WSNA)*, 2003.
- [8] M. DE BERG, M. VAN KREVELD, M. OVERMARS, AND O. SCHWARZKOPF, Computational geometry: Algorithms and applications, Springer, New York, 1997.
- [9] J. OROURKE, Computational geometry in C, Cambridge University Press, New York, 1998.
- [10] S. MEGERIAN, F. KOUSHANFAR, M. POTKONJAK, AND M. SRIVASTAVA, Worst and best-case coverage in sensor networks, *IEEE Transaction on Mobile Computing*, vol. 4, no. 1, pp. 84-92, 2005.
- [11] S. MEGUERDICHIAN, S. SLIJEPEVIC, V. KARAYAN, AND M. POTKONJAK, Localized algorithms in wireless ad hoc networks: Location discovery and sensor exposure, *Proceedings of MobiHoc 01*
- [12] L. HUANG, H. XU, Y. WANG, J. WU, AND H. LI, Coverage and exposure paths in wireless sensor networks, *Journal of Computer Science and Technology*, vol. 21, no. 4, pp. 490-495, 2006.
- [13] C. HUANG, Y. C. TSENG, AND L. LO, The coverage problem in three-dimensional wireless sensor networks, *Proceedings of IEEE GLOBECOM 04*, vol. 5, Nov-Dec.
- [14] Z. ZHOU, S. DAS, AND H. GUPTA, Connected k -coverage problem in sensor networks, *Proceedings of ICCCN 04*, pp. 373C378, 2004.
- [15] G. XING, X. WANG, Y. ZHANG, C. LU, R. PLESS, AND C. GILL, Integrated coverage and connectivity configuration for energy conservation in sensor networks, *ACM Transactions on Sensor Networks*, vol. 1, pp. 36C72, Aug 2005.
- [16] S. KUMAR, T. LAI, AND J. BARLOGH, On k -coverage in a mostly sleeping sensor network, *Proceedings of MobiCom 04*, pp. 144C158.
- [17] P. WAN AND C. YI, Coverage by randomly deployed wireless sensor networks, *IEEE Transactions on Information Theory*, vol. 52, pp. 2658C 2669, June 2006.
- [18] S. SKYUM. A sweepline algorithm for generalized Delaunay triangulations. *Technical Report DAIMI PB-373*, CS Dept., Aarhus University, 1991.
- [19] <http://www.mathworks.com/>