Evaluating Coverage Quality Through Best Covered Pathes in Wireless Sensor Networks

ShaoJie Tang*, XuFei Mao†, Xiang-Yang Li*, Guojun Dai‡
*Dept. of Computer Science, Illinois Institute of Technology, Chicago, IL
†Beijing Key Lab of Intelligent Telecommunications Software and Multimedia, BUPT, Beijing, China
‡Institute of Computer Application Technology, Hangzhou Dianzi University, Hangzhou, China
Emails: {stang7,xmao3}@iit.edu, xli@cs.iit.edu, daigj@hdu.edu.cn

***Abstract*— Coverage quality is one critical metric to evaluate the Quality of Service (QoS) provided by wireless sensor networks. In this paper, we address maximum support coverage problem (a.k.a. best case coverage) in wireless sensor networks. Most of the existing work assume that the coverage degree is 1, *i.e.* every point on the resultant path should fall within the sensing range of at least one sensor node. Here we study the k-coverage problem, in which every point on the resultant path is covered by at least k sensors while optimizing certain objectives. We present tackle this problem under both centralized and distributed setting. The time complexity is bounded by $O(k^2 n \log n)$ where n is the number of deployed sensor nodes. To the best of our knowledge, this is the first work that presents polynomial time algorithms that find optimal k-support paths for a 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 found path is the optimum one under a certain quality measurement. For example, we assume there is working environment which is monitored by a wireless sensor network. When some emergency happens, the sensor network should provide safe path(s) which can guide the users leaving from the working place to some safe exit(s). In this scenario, the path should be close to some sensor(s) such that the situation along the path can be monitored well. Let us consider another scenario. Assuming there are some soldiers who have to sneak through some area. Unfortunately, the enemies have deployed a bunch of sensor nodes which are able to detect the movement of objects in this area. Clearly, the soldiers should move forwarder following a path which is furthest away from each sensor. This kind of question is generally called a *coverage* problem in wireless sensor networks. In other words, coverage is a measure of quality of service (QoS) of a sensor network to some extent.

In best case coverage (a.k.a., *maximum support coverage*), our goal is to find areas of high observability from sensors and identify the best support and guidance regions (paths). In this paper, we are interested in designing algorithms addressing the following question when a wireless sensor network is given: Finding a path connecting a source point S and a destination point D inside the given area, which maximizes the smallest observability of all points along the path. This is called *best coverage problem* [9]. The observability of a point p depends on different applications. Most of the existing work [7]–[9] focus on the 1-coverage problem, *i.e.*, any point on the region (path) falls in the sensing range of at least one sensor. The observability of a point p is simply the shortest Euclidean distance from p to the set of sensors U, *i.e.*, $\min_{v \in U} \|pv\|$.

In this paper, we consider a more general case of the coverage problem in wireless sensor networks. Given a set U of sensors deployed in a 2-dimensional region Ω, we want to determine how well the area Ω is k-covered. In this case, the observability of a point p is the shortest Euclidean distance from p to the k-th nearest sensor out of U. To the best of our knowledge, [10], [2] and [14] are the only work so far which address the problem of finding an optimal k-covered path. In [10], Mehta *et.al* suggested that the worst case k-coverage problem may be addressed by adopting the k-th nearest-point Voronoi diagram. However, no algorithm and theoretical results were given. In [2], 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 guarantee to find the optimum solution. This is because the binary search method in their algorithm may not return the optimal k-support path when k-th distance of some point on the path is not integer. Furthermore, they assumed that k is a given constant which may reduce the generality of their algorithm. This is because the value of k could be up to the number of sensor nodes n depending on different applications and QoS requirements. In one of our previous work [14], we presented a centralized polynomial algorithm to find optimum k-support path with more general k, *i.e.*, k could be any integer value between 1 and n (the total number of sensor nodes). In this paper, we first improve the time complexity needed to find a optimal k-support path, and further present a distributed algorithm which can find am optimal k-support path efficiently.

The main contributions of our paper are as follows. Since the properties of k-th nearest point Voronoi diagram are quite different from the ordinary Voronoi diagram and are not well studied before, we first propose an efficient algorithm to generate the KNP Voronoi diagram by combining computational geometry techniques with graph theoretical and algorithmic

978-1-4577-0103-0/11/$26.00

techniques. More importantly, we prove and present a number of theoretical results about the properties of KNP Voronoi diagram. 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 $O(n)$. (We give the definitions of KNP-Voronoi edge and KNP Voronoi cell in Section III formally.) To the best of our knowledge, these theoretical results presented in this paper about the properties of KNP Voronoi diagram are not known in the literature. Secondly, we present polynomial time algorithms that find optimum k-support in both centralized manner and distributed manner. Previous studies assume that all sensor nodes have unbounded sensing ranges, we also studied the other case that the sensing range of each sensor node is bounded.

II. Related Work

In order to evaluate the quality of coverage of the sensor network, Meguerdichian *et.al* [9] formulated the 1 coverage problem under two extreme cases: the best case coverage (maximum support) problem and the worst case coverage (minimum breach) problem. In [9] the authors observed that an optimal solution for the maximum support problem is a path which lies along the edges of the Delaunay triangulation [1] [12], and an optimal solution for the minimum breach problem is a path which lies along the edges of the Voronoi diagram [1] [12]. They further proposed centralized algorithms for both problems. Later, Mehta *et.al* [10] improved these algorithms and made them more computational efficient.

There were some other work which aimed at solving the 1 coverage problem formulated in [9] in a distributed manner. Li *et.al* [7] showed that the maximum support path can be constructed by only using edges of the relative neighborhood graph (RNG) of the sensor node set. They attempted to address best case 1 coverage problem in distributed manner. This is an improvement since the RNG is a subgraph of the Delaunay triangulation and can be constructed locally. On the other side, Meguerdichian *et.al* [9] implied that a variation of the localized exposure algorithm presented in [12] can be used to solve the worst case coverage problem locally. Another localized algorithm with more practical assumptions was proposed by Huang *et.al* [3].

For the general coverage problem, Huang et.al [3] studied the problem of determining if the area is sufficiently k-covered, *i.e.*, every point in the target area is covered by at least k sensors. In [3], the authors formulated the problem as a decision problem and proposed a polynomial algorithm which can be easily translated to distributed protocols. They further extended this problem to three-dimensional sensor networks and proposed the solution in [4]. The connected k-coverage problem was studied and addressed in [17] by Zhou *et.al*. They studied the problem of selecting a minimum set of sensors which are connected and each point in a target region is covered by at least k distinct sensors. They gave both a centralized greedy algorithm and a distributed algorithm for this problem and showed that their centralized greedy algorithm is near-optimal. Xing *et.al* [16] explored the problem concerning energy conservation while maintaining both desired coverage degree and connectivity. They studied the integrated work between the coverage degree and the connectivity and proposed a flexible coverage configure protocol.

Some studies focused on the relationship between the coverage degree k, the number of sensors n and the sensing radius r of sensor nodes. Kumar *et.al* [5] studied the problem of determining the appropriate number of sensors that are enough to provide k-coverage of a region under the condition that sensors are allowed to sleep during most of their lifetime. In [15], Wan *et.al* 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, [10], [2] and [14] are the only work which aims to find an optimal k-covered path. In [10], Mehta *et al.*, suggested that the worst case k-coverage problem can be addressed by adopting the KNP Voronoi diagram. However, no details of the proposed algorithm were given. In [2], Fang *et.al* gave a polynomial time algorithm to identify a k-covered path based on binary search and growing disk techniques. Unfortunately, the time complexity of their algorithm can not be bounded if an optimum solution is required. Furthermore, they assume that k is some constant which may reduce the generality of their algorithm. In one of our previous work [14], we designed a centralized polynomial time algorithm which can find optimum k-support path efficiently for general k. In this work, we further improved the time complexity of our centralized algorithms and proposed a distributed algorithm to solve this problem.

III. Preliminaries

In this section, we formally define the questions to be studied and some terms which will be used throughout the paper.

A. Problem Formulation

We assume that there is a set of n identical wireless sensor nodes $U = \{u_1, u_2, ..., u_n\}$ deployed inside a continuous two-dimensional field Ω (a simple 2-dimensional polygon), where the location (x_i, y_i) of each sensor node u_i is known. In addition, we assume all sensor nodes have enough sensing range such that it can sense any point in Ω. However, the sensing ability (observability) of a sensor node for a point depends on the Euclidean distance between them. We further assume all sensor nodes are stationary and construct a connected wireless sensor network. A pair of points S and D are given as the source point and the destination point in the given area Ω. Generally speaking, the sensing ability of a sensor node depends on the Euclidean distance between an object (point) and the sensor node's location. In this paper, we use Euclidean distance as the measurement of QoS. Notice that we did not take the length of possible resultant paths into consider in this paper.

Before we formulate the questions to be studied in this paper, we introduce some definitions which will be used later.

Definition 1: Given a point v in the field Ω and the set of sensors U, the *k-th distance* of v, with respect to U, denoted as $\ell_k(v, U)$, 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 of all points on P. In other words, $S_k(P) = \max_{p \in P} \ell_k(p, U)$ where p is a point on path P.

In this paper we study the following question:

Problem 1: **Optimal 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 (KNP Voronoi Diagram)

Given a set of identical sensor nodes U deployed in the 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. In other words, if the sensor node $u_i \in U$ is the k-th nearest sensor node of point p, the point p is assigned to sensor node u. As a result, we obtain a collection of regions associated with sensor nodes in U, denoted by $\mathbb{V}_k = \{V_k(u_1), ...V_k(u_n)\}$, which forms a tessellation. We call the tessellation $\mathbb{V}_k$ *the k-th nearest-point Voronoi diagram* (KNP Voronoi diagram) with respect to U, and the region $V_k(u_i)$ the k-th nearest-point Voronoi region of node u_i. In other words, all points inside region $V_k(u_i)$ have the same k-th nearest sensor node u_i out of set U. Notice that $V_k(u_i)$ may be disconnected and composed by several independent polygons as shown in [11]. Here we call each independent polygon *k-th nearest-point Voronoi cell* (KNP Voronoi cell) of node u_i and use $C_k(u_i)$ to denote it, in addition, we simply call u_i is the **owner** of $C_k(u_i)$. Notice, the source point S and destination point D also have owners depending on the KNP Voronoi cells they belong to. If S(resp. D) are existing on one of edges of some KNP Voronoi cell, which means S(resp. D) can have two or more k-th nearest sensor nodes, we randomly choose one such sensor node as the owner of S(resp. D).

The edge (including the boundary of area Ω) of each KNP Voronoi cell is called *k-th nearest-point Voronoi edge* (KNP Voronoi edge) and the intersections of all KNP Voronoi edges are called *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 a KNP Voronoi edge is defined as the point (on this edge) which has the minimum Euclidean distance to its owner (k-th nearest sensor node).

Notice that for each KNP Voronoi edge, it has one and only one perfect support location.

C. Difference Between KNP Voronoi Diagram and k-order Voronoi Diagrams

Most of our results are based on KNP Voronoi Diagram. To make the paper easy to be understood, we briefly discuss the differences between the following two concepts, KNP Voronoi Diagram and k-order Voronoi diagrams. As we have presented, the KNP Voronoi Diagram of a set of sensor nodes U partitions the plane into cells such that all points in the same cell have the same k-th nearest sensor node out of set U. On the other hand, k-order Voronoi Diagrams [1] of a set of sensor nodes U partitions the plane into cells such that all points in the same cell have the same set (maybe in different distance orders) of k nearest sensors out of U. For example, given a set of three sensor nodes $U = \{1, 2, 3\}$ falling in a 2-dimensional plane, the 2-order Voronoi diagram could be what Fig. 1(a) shows. All points in the left blue half plane have same 2 closest sensor node set $(1, 2)$ and the right half green plane contains all points which have 2 closest sensor node $(2, 3)$. In this example, the 2 closest sensor node set $(1, 3)$ does not have corresponding polygon. Fig. 1(b) shows the 2^{nd} nearest-point Voronoi diagram with respect to the same sensor node set $U = \{1, 2, 3\}$. Two blue rectangles contain all points whose second closest sensor node is 2. From now on, we call k-th nearest point Voronoi Diagram as KNP Voronoi Diagram for simplicity.

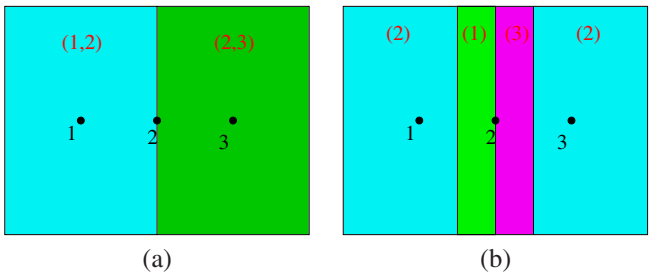


Fig. 1. (a) 2-order Voronoi diagram. (b) 2^{nd} nearest-point Voronoi diagram

Now we are ready to present our polynomial time algorithms computing the optimal k-support path within $O(k^2 n \log n)$ time. Given a set of sensor nodes U and the continuous field Ω, our algorithm consists of two phases. In the first phase, we use Algorithm 1 in Section IV to compute the KNP Voronoi diagram G in time $O(k^2 n \log n)$. During the second phase, we construct a new weighted graph G' based on G and then compute the optimal k-support path on G', which can be done in time $O(k^2 n \log n)$. Since our algorithms for both coverage problems (best case coverage and worst case coverage) are base on the KNP Voronoi diagram, we present the method to compute the KNP Voronoi diagram with respect to sensor node set U first.

IV. Compute the KNP Voronoi Diagram

Before we present our algorithm to compute the KNP Voronoi diagram with respect to the sensor node set 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 in set $U^{[k]}$. Each polygon is named *order-k Voronoi cell* $C_{ok}(U_i^{[k]})$ corresponding to a set of k sensor nodes $U_i^{[k]}$. (Illustrated in Fig. 1(a).)

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

Next we give a lemma which will be used to prove the correctness of our algorithm.

Lemma 1: For any point p in the plane Ω, $p \in C_k(u_i)$ if and only if p is located in some order-k Voronoi cell $C_{ok}(U_i^{[k]})$ where $u_i \in U_i^{[k]}$ and u_i is p's farthest sensor node among all k sensor nodes in $U_i^{[k]}$.

Proof: We prove this lemma in both directions. According to the definition of the order-k Voronoi diagram, we know that if point p is located in $C_{ok}(U_i^{[k]})$, all sensor nodes in $U_i^{[k]}$ belong to the k closest sensor nodes set of v. Clearly, if some sensor node $u_i \in U$ is the farthest sensor node from point p among all sensor nodes in $U_i^{[k]}$, u_i must be the k-th nearest sensor nodes of p.

Now we prove the other direction. If sensor node u_i is point p' k-th nearest sensor node, p must be located in some $C_{ok}(U_i^{[k]})$ where $U_i^{[k]}$ is the union of p' $1^{st}, 2^{nd}..., (k-1)^{th}$ nearest sensor nodes. Obviously, u_i is the farthest sensor node among all k sensor nodes in $U_i^{[k]}$. This finishes the proof. ■

Based on Lemma 1, we propose the following method (Algorithm 1) to compute the KNP Voronoi diagram with respect to sensor node set U. The main idea of our method is as follows.

1) Compute the order-k Voronoi diagram of given sensor nodes set U using the algorithm given in [6].
2) For each order-k Voronoi cell $C_{ok}(U_i^{[k]})$, we compute the farthest Voronoi diagram of its corresponding k sensor nodes $U_i^{[k]}$, and for each sensor node $u_i \in U_i^{[k]}$ return the corresponding farthest Voronoi cell as a part of u_i's KNP Voronoi cell.
3) For each sensor node u_i, we merge the partial cells computed above into one KNP Voronoi cell if they share one edge. We union all those KNP Voronoi cells as u_i's k-th nearest point Voronoi region. Finally, we get U's KNP Voronoi diagram G.

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

Since the deployed wireless sensor nodes are considered to be stationary, to compute the KNP Voronoi diagram is one-time work and the Alg. 1 can be finished on any wireless sensor node as a centralized manner.

Algorithm 1 Computing KNP Voronoi Diagram

Input: Set of sensor nodes U.
Output: U's KNP Voronoi Diagram G.

```
Compute U's order-k Voronoi diagram;
for Each order-k Voronoi cell C_ok(U_i^[k]) do
    Compute the farthest point Voronoi diagram using corresponding k sensor nodes in U_i^[k];
end for
for Each KNP Voronoi edge e do
    if If two neighbor polygons which share e belong to same sensor node u_i then
        Merge these two polygons into one polygon;
    end if
end for
for Each sensor node u_i do
    Return the union of all polygons belongs to u_i as its KNP Voronoi cells;
end for
```

V. Best Case Coverage: Optimal k-Support Path

In this section, we address the optimal k-support path problem, *i.e.*, to find a path (connecting the source point S and destination point D in the given area Ω) which has the minimum k-support. We first give an polynomial time algorithm to compute the optimal k-support path and further prove that its time complexity is $O(k^2 n \log n)$.

A. Preliminaries

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

Theorem 2: 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 KNP Voronoi diagram. For any given path P_1 (inside region Ω) connecting S and D, we decompose it into a set of partial paths by the intersections of P_1 and all KNP Voronoi edge gone through by P_1. For example, if P_1 comes into some KNP Voronoi cell by point a and leave this KNP Voronoi cell by point b subsequently, we use p_{ab} to denote this part of path. Clearly, each partial path will be entirely contained in some KNP Voronoi cell $C_k(u_i)$. Here, p_{ab} is also considered as "contained" by some KNP Voronoi cell $C_k(u_i)$ if p_{ab} is overlapping with some KNP Voronoi edges of $C_k(u_i)$. Notice that, P_1 may come into and leave some KNP Voronoi cell more than once, we consider them as different partial paths. Essentially, we will prove that using line segment ab to replace the original partial path p_{ab} will lead to a no worse result, *i.e.*, the k-*support* of this line segment is no greater than that of p_{ab}.

Notice that $S_k(p_{ab})$ is no smaller than $\max\{\ell_k(a,U), |\ell_k(b,U)\}$ where $\ell_k(p,U)$ is the k-th distance of point a. In addition, line segment ab is entirely contained in a disk centered at u_i with radius $\max\{\ell_k(a,U), |\ell_k(b,U)\}$ such that $S_k(ab)$ is no more than $\max\{|u_ia|, |u_ib|\}$. Thus, the k-support of line segment ab ($\max\{\ell_k(a,U), \ell_k(b,U)\}$) already achieves the lower bound. See Figure 2 for illustration. Because the $S_k(P_1)$ is equal to the maximum one among all

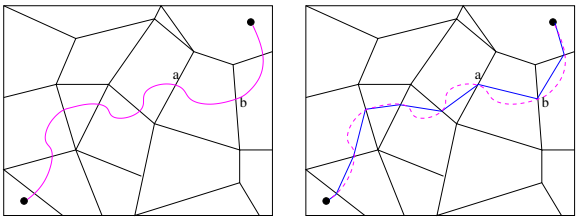


Fig. 2. Given any path, if we replace every partial path in each Voronoi cell by a line segment, we get a new path on the right hand side whose k-support is no larger than the original path.

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 greater than $S_k(P_1)$. This finishes the proof. ■

Based on Theorem 2, we further prove the following two theorems.

Theorem 3: Based on any given path P_1 connecting source node S and destination node D, we can construct another path P_3 consisting of only line segments whose end points are perfect support location of the KNP Voronoi edges such that

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

Proof: Firstly, we mark the perfect support location of each KNP-Voronoi edge. Notice that two or more KNP Voronoi edges may have the same perfect support location. Secondly, for each KNP Voronoi cell $\mathcal{C}_j$, we sorted all perfect support location points of all KNP Voronoi edges belong to $\mathcal{C}_j$ by their k-th distance including the source point S or the destination point D or both (if S or D falls into $\mathcal{C}_j$). Notice, if S(resp. D) is existing on some KNP Voronoi edge which is shared by two KNP Voronoi cells, we let S(resp. D) belong to any one of cells. If two KNP Voronoi edges of $\mathcal{C}_j$ share the same perfect support location, we consider them as a single perfect location.

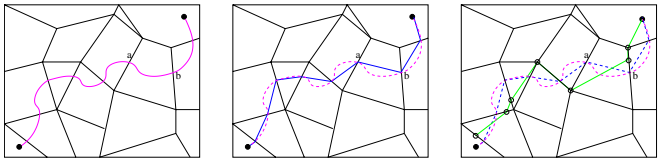
For each KNP Voronoi cell, assume the sorted perfect support locations list is $\{p_1, p_2, \cdots, p_m\}$, we use line segments to connect each adjacent pair of points in the list.

Assume P_1 comes into KNP Voronoi cell $\mathcal{C}_j$ through KNP Voronoi edge e_1 and leave KNP Voronoi cell $\mathcal{C}_j$ subsequently through KNP Voronoi edge e_2 and intersection points on e_1 and e_2 are p_1' and p_2' respectively. We use $P(p_1', p_2')$ to denote this part of path P_1. We further assume the perfect support location of e_1 and e_2 are p_1 and p_2 respectively. We use $P(p_1, p_2)$ to denote the path consisting of line segments which connect p_1 and p_2. Next, we show that using line segments from p_1 to p_2 to replace $P(p_1', p_2')$, the k-support will not increase. First, the k-support of part of path $P(p_1', p_2')$ is not smaller than the bigger k-th distance of p_1' and p_2'. Next, according to the definition of perfect support location, we know that the k-th distance of p_1 (resp. p_2) will be no greater than that of p_1' (resp. p_2'). Remember that the k-support of all the line segments (connecting p_1 and p_2) which are used to replace the part of path $P(p_1', p_2')$ will be no greater than the bigger k-distance of p_1 and p_2 since we sorted all perfect support locations before we connect them.

■

Theorem 4: There is one optimal k-support path consisting of only line segments whose end points are located at the perfect support locations of the KNP Voronoi edges.

Proof: This theorem is straightforward from Theorem 2 and 3.



(a) (b) (c)

Fig. 3. Given any path, we can always find a new path based on the conversion shown above such that the new constructed path is composed by only line segments with their end points located on its KNP Voronoi edge's perfect support location, and the k-support of the new constructed path is no greater than the original path.

Given any minimum k-support path, we decompose it into partial paths, each of which is entirely contained in some KNP Voronoi cell. If every partial path is already a line segment with its end points located at the perfect support location of the corresponding KNP Voronoi edge, it is done. If not, as shown before, by replacing each partial path by one or more line segment whose end points are only those perfect support locations of some KNP Voronoi edges, we can get another path whose k-support is no larger than the given minimum k-support path, see Figure 3.$(a) \rightarrow (b) \rightarrow (c)$ illustrates this process. It implies that the new constructed path is another optimal k-support path. This finishes the proof. ■

B. Compute the Minimum k-Support Path

After getting the KNP Voronoi diagram G with respect to U by Algorithm 1, we present our algorithm to compute the optimal k-support path based on Theorem 4.

As shown in Theorem 4, there must exist one minimum k-support path consisting of only line segments and all of these line segments' end points are located on the perfect support location of some KNP Voronoi edges. Clearly we only need to consider all the paths that using only line segments connecting the perfect support locations of the KNP Voronoi edges. Among them, the one with the minimum k-support must be one of the desired optimal k-support path.

First, we construct a new graph G' based on KNP Voronoi diagram G as follows:

1) For each KNP Voronoi edge e in G, we add a new node v' (to $V[G']$) whose location is the perfect support location

of e. Notice, when multiple KNP Voronoi edges have the same perfect support location, we only add one node. In other words, we establish a one-to-one (or many-to-one) mapping between each kNP Voronoi edge e in G and each node $v' \in V[G']$. Additionally, we add another two nodes S' and D' to $V[G']$ where S' (resp. D') has the same location as S (resp. D). Here, S' and D' will be considered as new source point and destination point in G'.

2) Next, We assign weight to each node in $V[G']$ as follows. If node $v' \in V[G']$ is S' or D', the weight of v' (denoted by $w(v')$) is equal to the k-th distance of S or D in G (*i.e.*, $w(v') = \ell_k(S, U)$ if $v' = S$ or $w(v') = \ell_k(D, U)$ if $v' = D$). Otherwise, $w(v')$ is equal to the k-th distance of the perfect support location of edge $e \in G$ where $e \in G$ is corresponding to $v' \in G'$ in the one-to-one (or many-to-one) mapping.
3) Next, we divide all nodes in $V[G']$ into groups. A bunch of nodes belong to one group iff their mapping KNP Voronoi edges (or S, T) in G belong to the same KNP Voronoi cell. Notice, different groups may contain same node(s) since multiple KNP Voronoi edges may mapping to the same node in G'. For each group of nodes, we sort all nodes by their weights in decreasing (or increasing) order. After that we add an edge to $E[G']$ between each pair of adjacent two nodes in the sorted list. Finally, we assign each edge $u'v' \in E[G']$ with weight $w(u', v')$ which is equal to $\max\{w(u'), w(v')\}$.

Next, we use Algorithm 2, which originates from Dijkstra's shortest path algorithm, to find a minimum weight path P' in G' to connect S' and D'. Here, a path P' (connecting S' and D') in G' is called a minimum weight path iff the maximum weight of all its edges is minimized among all such paths in G'. Finally, we get one minimum k-support path P in G by directly applying P' to G.

Algorithm 2 The Optimal 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-support[v'] ← infinity
3:    previous[v'] ← undefined
4: end for
5: k-support[S'] ← w(S)
6: Q ← the set of all nodes in graph G'
7: while Q is not empty do
8:    u' ← node in Q with smallest k-support
9:    remove u' from Q
10:   for Each neighbor v' of u': do
11:      alt ← max{w(u', v'), k-support[u']}
12:      if alt < k-support[v'] then
13:         k-support[v'] ← alt
14:         previous[v'] ← u'
15:      end if
16:   end for
17: end while
18: Return P'
```

C. Correctness

Theorem 5: Given any two source/destination pair of points S and D, the path P returned by algorithm 2 is an optimal k-support path.

Proof: First we know that graph G' is connected by the following observation. Every two adjacent KNP Voronoi cells (which share a KNP Voronoi edge) share the same perfect support location on this KNP Voronoi edge. Clearly, there is at least one path connecting any two vertices in G' since every KNP Voronoi cell contains a connected line graph which connected all perfect support location inside this KNP Voronoi cell.

Next, Based on Theorem 4, we know that among all paths consisting of only line segments whose endpoints located only at some KNP Voronoi edges' perfect support locations, the one with the minimum k-support must be the optimal k-support path. Next, we 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 which connect the perfect support locations of the KNP Voronoi edges it crosses. Next, what we need to show is that P's k-support is indeed minimized among all such paths.

As we know, $S_k(P)$ is the maximum k-support among all P' line segments, so P is a optimal k-support path iff it minimized the maximum k-support among all its line segments. Remember that the weight of each edge uv, $w(uv)$ denotes the bigger k-distance of the perfect support location of two KNP Voronoi edges (mapping to u' and v' respectively). Clearly, $w(uv)$ is the lower bound of k-support for any path which goes across these two KNP Voronoi edges (mapping to u' and v' respectively). Since P' computed by Algorithm 2 minimized the maximum weight among its edges, this implies that the final path P we computed has the minimum k-support among all possible paths. We can conclude that P is an optimal k-support path. ■

D. Time Complexity Analysis

In this subsection, we prove that the time complexity of our algorithm is $O(k^2 n \log n)$ which is better than the work in [14]. As shown before, our algorithm is composed by two phases, one is to compute the KNP Voronoi diagram, and the other one is to compute the optimal k-support path based on the KNP Voronoi diagram. We will analyze the time complexity of these two phases respectively.

We first give a bound of the time complexity for the first phase,

Lemma 6: The time complexity of Algorithm 1 which is used to compute the KNP Voronoi diagram is $O(k^2 n \lg n)$.

Proof: First, using the algorithm given in [6], we can compute the order-k Voronoi diagram of n sensor nodes within time $O(k^2 n \lg n)$. 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 (proven in [13]). Since there are $O(nk)$ order-k Voronoi cells (results showed in [6], the time complexity of the second step is $O(k \lg k) \times O(nk) = O(k^2 n \lg k)$. In the third step, we may do some merge operations for each KNP Voronoi edge, this operation will cost $O(k^2 n)$ time since there are at most $O(nk^2)$ KNP Voronoi edges in KNP Voronoi diagram (which will be proved in Lemma 7) and each merge operation uses constant time $O(1)$. So the time complexity for Algorithm 1 is $O(k^2 n \lg n) + O(k^2 n \lg k) + O(k^2 n) = O(k^2 n \lg n)$. This finishes the proof. ■

Next, we show the time complexity for the second part. Since the time complexity of the second part is determined by the number of KNP Voronoi cells, the number of KNP Voronoi edges per KNP Voronoi cell etc., we first prove some properties of KNP-Voronoi diagram.

Lemma 7: 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 $O(k^2 n)$ and the number of edges of each KNP Voronoi cell is $O(n)$.

Proof: From the results in [6] and [11], 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 $O(k)$. So the total number of KNP Voronoi edges computed from Algorithm 1 is $O(nk) + O(nk) \times O(k) = O(k^2 n)$.

Next we use a simple construction approach to show 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 we further call the open half-plane (defined by the sectors which does not contain u_i) 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 8: The time complexity of our algorithm to compute the optimum k-support path based on KNP Voronoi diagram is $O(k^2 n \log n)$.

Proof: According to the rules we used to construct G' from G as shown previously and from Lemma 7, we can prove that • *There are at most $O(k^2 n)$ vertices (nodes) in G':* This is because there are at most $O(k^2 n)$ KNP Voronoi edges in G and each vertex in G' has one (or more) corresponding KNP Voronoi edge(s) in G;

• *The total number of edges in G' is at most $O(k^2 n)$:* Remember that we divided all nodes in G' into groups. For each node v' in each group, v' will only to connect to at most two other adjacent nodes. This is because each KNP Voronoi edge in G will be only shared by at most two KNP Voronoi cells. Hence, the total number of edges in G' will no more than 4 times of the total number of KNP Voronoi edges in G which is $O(k^2 n)$.

Consequently, the 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 in G'. So, the time complexity of Algorithm 2 is $O(k^2 n \log n)$. Hence, the overall time complexity of our method is $O(k^2 n \log n)$. This finishes the proof. ■

Combining Lemma 6 and Lemma 8 together, we have the following theorem.

Theorem 9: The time complexity of our algorithm to compute an optimum k-support path is $O(k^2 n \log n)$.

VI. Distributed Algorithm for Computing the Optimal k-Support Path

In this section, we present our distributed algorithm to compute the optimal k- support path after getting the KNP Voronoi diagram with respect to U by Algorithm 1. First, we let each sensor node record its owned KNP-Voronoi cells, including the geometry information of each edge of each KNP-Voronoi cell. Here we assume that each KNP Voronoi edge is assigned an unique identity. Remembering that, each KNP Voronoi edge can belong to one or two KNP Voronoi cells such that each KNP Voronoi edge can have one or two owners. Next, we present our distributed algorithm to compute the optimal k-support path based on Theorem 4.

First, we construct a new graph G' based on KNP Voronoi diagram G in a distributed manner. Our main idea is to let each sensor node u_j $(1 \le j \le n)$ construct a new graph $G_{\mathcal{C}_i}^{u_j}$ locally for each of its own KNP-Voronoi cells $\mathcal{C}_i$. And G' will be the union of all such new constructed local graphs, *i.e.*,

$$G' = \bigcup_{u_j \ owns \ \mathcal{C}_i} G_{\mathcal{C}_i}^{u_j} : \forall u_j \in U$$

For each cell $\mathcal{C}_i$ owned by each sensor node u_j, u_j will construct a local graph for cell $\mathcal{C}_i$ as follows,

- Initially, the vertex set of $V(G_{\mathcal{C}_i}^{u_j})$ and the link set of $E(G_{\mathcal{C}_i}^{u_j})$ will be empty.
- For each KNP Voronoi edge e belong to $\mathcal{C}_i$ (owned by u_j), u_j will add a corresponding point v' to $G_{\mathcal{C}_i}^{u_j}$. Let v' use the perfect support location of e as its location and let the weight of v' (denoted by $w(v')$) be equal to the k-th distance of the perfect support location on edge $e \in E(G)$. By the way, we let the node v' has the same unique ID as its corresponding edge.
- If the source point S or destination point D (or both) is owned by u_j, u_j will add a corresponding new source point S' or new destination point D' (or both) to $G_{\mathcal{C}_i}^{u_j}$ as well. Let S' (resp. D') has the same location as S (resp. D) and let the weight of S' (resp. D') be equal to the k-th distance of S (resp. D) in graph G.
- Sort all vertices of $G_{\mathcal{C}_i}^{u_j}$ in decreasing (or increasing) order by their weights. Add an edge between each pair of adjacent two vertices in the sorted list to $E(G_{C_i}^{u_i})$. In other words, graph $G_{\mathcal{C}_i}^{u_j}$ is a line graph.

- Assign weight to each newly added edge e', $w(e') = \max\{w(v'_1), w(v'_2)\}$ where v'_1 and v'_2 are two end points of e'.

Clearly, the graph $G' = \bigcup_{u_j \ owns \ \mathcal{C}_i} G^{u_j}_{\mathcal{C}_i} : \forall u_j \in U$ is a connect planar graph. Then we run a distributed minimum weight path algorithm (slightly changes from work in [18]) on graph G' to compute the minimum weighted path connecting S' and D' since all sensor nodes construct a connect network and they can exchange information via one- or multi-hop. A path is said to be a minimum weighted path if the maximum weight of all its partial paths is minimized among all such paths connecting S' and D'.

Then, we use Algorithm 3, which originates 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, by applying P' to graph G directly, we got an optimum k-support path P.

Algorithm 3 Distributed Optimum k-Support Path Algorithm

Input: KNP Voronoi diagram based on sensor node sets U, G, source node S, destination node D.
Output: Minimum k-support Path Connecting S and D.

1: $V[G'] \leftarrow \emptyset$; $E[G'] \leftarrow \emptyset$;
2: **for** Each sensor node $u_i \in U$ **do**
3: **for** Each KNP-Voronoi cell $\mathcal{C}_j$ owned by u_i **do**
4: Use Alg. 4 to distributively construct a new graph $G^{u_i}_{\mathcal{C}_j}$.
5: $V[G'] = V[G'] \bigcup V[G^{u_i}_{\mathcal{C}_j}]$. (The vertices with same ID will be merge to one single vertex after union.)
6: $E[G'] = E[G'] \bigcup E[G^{u_i}_{\mathcal{C}_j}]$
7: **end for**
8: **end for**
9: Run distributed minimum weighted path algorithm (slightly changes from the work in [18]) on G' to get an minimum weight path P' connecting S' and D'.
10: Return P'

Algorithm 4 Distributed Constructing New Graph G' Algorithm

Input: Sensor node $u_i \in U$ and a KNP-Voronoi cell $\mathcal{C}_j$ owned by u_i
Output: New constructed graph $G^{u_i}_{\mathcal{C}_j}$.

1: $V[G^{u_i}_{\mathcal{C}_j}] \leftarrow \emptyset$; $E[G^{u_i}_{\mathcal{C}_j}] \leftarrow \emptyset$;
2: **for** Each KNP-Voronoi edge $e \in \mathcal{C}_j$ **do**
3: u_i adds a new vertex v' at the perfect support location point of e.
4: Assign v' with the same unique ID of e
5: Assign the k-th distance of the perfect support location of e as the weight of vertex v' ($w(v')$).
6: **end for**
7: **if** u_i owns S or D (or both) **then**
8: Add new source vertex S' or new destination vertex D' (or both) at the same location of S or D's.
9: Let the weight of S' (resp. D') in $G^{u_i}_{C_i}$ be equal to the k-th distance of S (resp. D) in G.
10: **end if**
11: Sort all new added vertices by their weights. Add an edge between each adjacent pairs of vertices in the sorted list.
12: Let the weight of each new added edge be equal to the bigger weight of its two end vertices.

VII. Discussion

In this paper, we studied the case that all sensor nodes have unbounded sensing ranges. In other words, we only use the Euclidean distance between a point p and a sensor node u to measure the sensing level of p by u. Actually, we also studied the other case that each sensor node has a bounded sensing range. In addition, different sensor nodes may have different sensing ranges depending on the sensing power used. Due to the space limited, we only introduce our main idea of how to find optimal k-support path.

When the sensing range of a sensor node is bounded, we simply assume the valid sensing range of a sensor node u is inside a circle centered at u with radius $r(u)$ (the sensing radius of node u). Actually, the sensing range of a sensor node u could have an arbitrary shape depending on the geometry situation (like some barriers) near u. For example, when a sensor node u is deployed in a city, the sensing range of u may affected by tall buildings around it. However, this will not affect our results. In addition, we still use the Euclidean distance to measure the sensing level for any point within the sensing range of some sensor node. For example, when two points p_1 and p_2 fall in the sensing range of sensor node u, we say point p_1 is better supported by u iff $|up_1| < |up_2|$ and vice versa. Here, $|up_1|$ (resp. $|up_2|$) is the Euclidean distance between u and p_1 (resp. p_2).

To compute the optimal k-support path in the given region Ω, we first marked the sensing range of each sensor node, *i.e.*, we draw a circle centered at u with radius $r(u)$ for each sensor node u. The whole area will be partitioned into some subareas whose boundary consists of (part of) those circles and the original boundary of Ω. Assume there are B subareas and we use a_i to denote one of subareas, clearly we have $\bigcup_{i=1}^{B} a_i = \Omega$. Next, we rank each subarea by the number of sensor nodes which are able to sense it. For example, if a subarea falls into the sensing range of k sensor nodes, its rank will be k. We use $R(a_i)$ to denote the rank of subarea a_i. Assume S and D are within in subarea a_i and a_j respectively. Here, i may be equal to j. Clearly, the k-support path connecting source/destination pair points S and D exists iff the following two conditions are satisfied: firstly, both a_i and a_j have ranks at least k; secondly a_i and a_j are connected by a bunch of subareas with rank at least k. Otherwise, the k-support path will not exist. To find an optimal k-support path, we first remove all subareas with rank lower than k. Second, we run Alg. 2 to find the optimal k-support path. The difference from the case when the sensing

range is not bounded is that the input area are all remaining subareas, instead of Ω.

VIII. Conclusion

In this paper, we proposed polynomial time algorithms (both centralized and distributed) for best case k-coverage problems in wireless sensor networks. Our algorithms can efficiently find a path connecting two points in a sensor network with the worst observability (*i.e.*, minimizing the maximum observability of all points on the path). We proposed a number properties for KNP Voronoi diagram as well. These properties may be of independent interests. As an interesting future work, we would like to design algorithms that can address the coverage problem when the sensing abilities of sensors are heterogeneous and the coverage of a point p is not simply based on the Euclidean distance between p and some special sensor node.

IX. Acknowledgement

The research of authors is partially supported by NSF CNS-0832120, NSF CNS-1035894, National Natural Science Foundation of China under Grant No. 60828003, No.60773042, the Natural Science Foundation of Zhejiang Province under Grant No.Z1080979, program for Zhejiang Provincial Key Innovative Research Team, program for Zhejiang Provincial Overseas High-Level Talents (One-hundred Talents Program).

References

[1] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. Computational geometry: Algorithms and applications. *Spinger, New York*, 1997.

[2] C. Fang and C.P. Low. Redundant coverage in wireless sensor networks. In *Proceedings of IEEE ICC*, 2007.

[3] C. Huang and Y. Tseng. The coverage problem in a wireless sensor network. In *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, 2003.

[4] C. Huang, Y. C. Tseng, , and L. Lo. The coverage problem in threedimensional wireless sensor networks. In *Proceedings of IEEE GLOBECOM*, 2004.

[5] S. Kumar, T. Lai, and J. Barlogh. On k-coverage in a mostly sleeping sensor network. In *Proceedings of MobiCom pp. 144-158.*, 2004.

[6] Der-Tsai Lee. On k-nearest neighbor voronoi diagrams in the plane. *IEEE Trans. Comput.*, 1982.

[7] X. Li, P. Wan, and O. Frieder. Coverage in wireless ad-hoc sensor networks. *IEEE Transaction on Computers, vol. 52, no. 6, pp. 753-763*, Jun. 2003.

[8] 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.

[9] S. Meguerdichian, F. Koushanfar M. Potkonjak, and M. B. Srivastava. Coverage problems in wireless ad-hoc sensor networks. In *Proc. INFOCOM 139-150*, 2001.

[10] DP Mehta, MA Lopez, and L. Lin. Optimal coverage paths in ad-hoc sensor networks. In *IEEE International Conference on Communications, 2003. ICC'03*, volume 1, 2003.

[11] Atsu. Okabe, B. Boots, K. Sugihara, and S.N. Chiu. Spatial tessellations. *Wiley Series In probability and Statics*, 2000.

[12] J. O Rourke. Computational geometry in c. *Cambridge University Press, New York*, 1998.

[13] S. Skyum. A sweepline algorithm for generalized delaunay triangulations. *Technical Report, DAIMI PB-373, CS Dept., Aarhus University,*, 1991.

[14] Shaojie Tang, Xufei Mao, and Xiang-Yang Li. Optimal k-support coverage paths in wireless sensor networks. In *IQ2S Workshop*, 2009.

[15] P. Wan and C. Yi. Coverage by randomly deployed wireless sensor networks. *IEEE TIT, vol. 52, pp. 2658-2669*, 2006.

[16] 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. 36-72*, 2005.

[17] Z. Zhou, S. Das, and H. Gupta. Connected k-coverage problem in sensor networks. In *Proceedings of ICCCN*, 2004.

[18] Shan Zhu and Garng M. Huang. A new parallel and distributed shortest path algorithm for hierarchically clustered data networks. *IEEE Transactions on Parallel and Distributed System*, 1998.