Finding Best and Worst *k*-Coverage Paths in Multihop Wireless Sensor Networks

Xufei Mao, *Member*, *IEEE*, Yunhao Liu, *Senior Member*, *IEEE*, Shaojie Tang, *Member*, *IEEE*, Huafu Liu, Jiankang Han, and Xiang-Yang Li, *Senior Member*, *IEEE*

Abstract—Coverage is a fundamental problem in wireless sensor networks (WSNs). From both economic and applicable concerns, designers always would like to provide guaranteed QoS of coverage of WSNs. In this paper, we address two path-coverage problems in WSNs, maximum k-support path coverage (a.k.a. best case coverage) and minimum k-breach path coverage (a.k.a. worst case coverage), in which every point on the desired resultant path is covered by at least k sensors simultaneously while optimizing certain objectives. We present two polynomial-time approaches to find optimal solutions for both maximum k-support coverage problem and minimum k-breach coverage problem. The time complexity of both algorithms are $O(k^2 n \log n)$, where n is the number of deployed sensor nodes and k is the coverage degree. In addition, a number of properties of kth-nearest point Voronoi diagram are presented, which is new to the literature.

Index Terms—Optimum k-coverage, k-support path, k-breach path, wireless sensor networks

1 Introduction

Being a major benchmark when applying wireless sensor networks (WSNs) into real applications, coverage-related problems of WSNs have drawn considerable research interests in the past two decades while posing many new challenging research questions. In most WSNs, two seemingly contradictory, yet related viewpoints of coverage exist: *best case coverage* and *worst case coverage* [10]. Generally speaking, the optimization objective of a best case coverage problem is to make areas, paths or points have much observability from deployed sensors, on the contrary, less observability from deployed sensors when we target at a worst case coverage problem. Here, the observability of an area, a path or a point from a sensor node can be explained as the sensing ability of the sensor to the area, the path or the point. For instance, for a sensor node v equipped with a temperature sensor, the observability of a point p (within the valid coverage range of v) from v indicates the accuracy when we use the reading value of v to denote the temperature of the point p. Generally speaking, the closer v and p are, the more accurate the reading is. When the observability is limited to one sensor node, it is usually called 1-coverage problem; when the observability is determined by k (k is usually, but not limit to a constant) sensor nodes simultaneously, it belongs to the k-coverage problem.

- *X. Mao, Y. Liu and J. Han are with the School of Software and TNLIST, Tsinghua University, #236, East Main Building, Tsinghua University, Haidian District, Bejing 100084, China. E-mail: {xufei, yunhao, jiankang}@greenorbs.com.*
- *S. Tang and X.Y. Li are with the Department of Computer Science, Illinois Institute of Technology, #10, West 31st Street, Chicago, IL, 60616. E-mail: stang7@iit.edu, xli@cs.iit.edu.*
- *H. Liu is with the Department of Computer Science and Technology, Changsha University, #98, Hongshan Road, Kaifu District, Changsha City, Hunan Province 410008, China. E-mail: hfliu9063@163.com.*

Manuscript received 16 Sept. 2011; revised 13 Nov. 2012; accepted 20 Nov. 2012; published online 4 Dec. 2012.
Recommended for acceptance by W. Jia.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2011-09-0660.
Digital Object Identifier no. 10.1109/TPDS.2012.329.

Let us consider the following application scenario which is typically considered as a k-coverage problem. Assuming that there are a group of anchor sensor nodes deployed in some area used to track the movement of a mobile sensor node inside this area. When some fading effects-based approach (e.g., using received signal strength indicator) is used to localize the moving sensor, generally, we need at least three anchor sensors that are able to detect (communicate with) the moving sensor node directly at any time without further hardware assistance in a real-time manner. In addition, since the QoS of observability of the moving sensor node from an anchor node is kind of proportional to the euclidean distance between them (Fading Effects' property), it is better to restrict the longest distance between the moving sensor and any one of three anchor nodes during its movement. In other words, the QoS is kind of limited by the furthest node (the third-nearest node in this case) such that we would like to bound the longest distance between the mobile node and its third-nearest anchor node all the time during its movement. This problem belongs to the family of best case coverage problems, i.e., *maximum support coverage problem*, more precisely, *maximum k-support coverage problem* where k is a critical parameters (or say coverage degree) determining the QoS of coverage. Let us take another application scenario as an example. Assuming there are some soldiers who have to sneak through some area. Unfortunately, the enemies have deployed a bunch of sensor nodes that are able to detect the movement of objects in this area. Clearly, the soldiers should move forwarder following a path which is keeping away from deployed sensor nodes. This type of problems belongs to the family of worst case coverage whose objective is to decrease the observability from sensor nodes.

In this paper, we aim to solve the following two k-coverage path problems:

1. *optimum k-support path problem*. Finding a path connecting any given source/destination pair of points S and T inside the given area, which

maximizes the smallest observability of all points along the path [10]; and

2. *optimum k-breach path problem*. Finding a path connecting any given source/destination pair of points S and T inside the given area, which minimizes the largest observability of all points along the path.

Here, the definition of *observability* of a point p depends on different applications. For instance, some existing work [8], [9], [10] focus on the 1-coverage problem and assume that the observability of a point p is simply the shortest euclidean distance from p (on the region or path) to the set of sensors U, i.e., $\min_{v \in U} \|pv\|$. In this paper, we consider the more general case, k-coverage problems, i.e., given a set U of sensors deployed in a two-dimensional region Ω, we would like to determine how well the area Ω is k-covered through investigating the observability of paths connecting any given source/destination pairs of points. In this case, the observability of a point p from the sensor node set U is defined as the shortest euclidean distance from p to the kth-nearest sensor out of U. From now on, we use distance to denote euclidean distance when there is no confusion.

To the best of our knowledge, [11], [3] and [15] are the only work so far which aim at addressing the problem of finding an optimal k-covered path. In [11], Mehta et al. suggested that the worst case k-coverage problem may be addressed by adopting the kth-nearest point Voronoi diagram. However, no algorithm and theoretical results were given. In [3], Fang and Low gave a polynomial time algorithm to identify a k-covered path based on binary search and growing disk techniques. Unfortunately, their algorithms cannot guarantee an optimum solution. This is because the binary search method in their algorithm may not return the optimal k-support path when kth-distance of some point on the path is not integer. Furthermore, they assumed that k is a given constant, which reduces the generality of the algorithm because the value of k could be up to the number of sensor nodes n depending on different applications and QoS requirements. Clearly, the 1-coverage problem is a special case of the proposed problem in the paper, where $k = 1$.

The main contributions of our paper are as follows:

1. We present and prove a number of theoretical results about the properties of kth-Nearest Point (kth-NP) Voronoi diagram, which are new to the literature and have independent interests.
2. We further propose an efficient algorithm to generate the kth-NP Voronoi diagram by combining computational geometry techniques with graph theoretical and algorithmic techniques.
3. We design and implement centralized polynomial time algorithms, which are able to find optimum k-support and k-breach paths in sensor networks with general k. A distributed algorithm is further given for the k-support path problem. Both our algorithms run in polynomial time $O(k^2 n \log n)$, where n is the number of wireless sensor nodes deployed and k is the coverage degree.
4. To the best of our knowledge, this is the first work that presents polynomial time algorithms finding optimal k-support paths and optimal k-breach paths for a general k.

The rest of the paper is organized as follows: In Section 2, we define terms, notations, and problems studied, following which we detail the procedure of computing the kth-NP Voronoi diagram of any given sensor node set U. We present the polynomial time algorithms that solve the optimum k-support path problem efficiently in Section 3, and further propose the distributed algorithm that can obtain an optimal k-support path in Section 4. Section 5 describes the details solving the optimum k-breach path problem efficiently. We give the running procedure of our algorithms with some simulation results in Section 6. We review the some related work in Section 7 and conclude our paper in Section 8. In the supplementary file (Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2012.329), we further discuss the methods to find both k-support and k-breach paths when each sensor node has (different) bounded sensing range, provide detailed proofs of theorems and lemmas and give more simulation results, respectively.

2 Preliminaries

In this section, we formally define the terms, notations used throughout the paper, and give the formal definitions of questions to be studied.

2.1 Problem Formulation

Assume that there is a connected WSN consisting of n identical and stationary wireless sensor nodes $U = \{u_1, u_2, \ldots, u_n\}$ deployed inside a continuous two-dimensional field Ω, where the location (x_i, y_i) of each sensor node $u_i : 1 \leq i \leq n$ is known a priori. Here, the location information of each sensor node could be measured by precise GPS equipments when deployed. Since the localization problem of wireless sensor nodes is out of the scope of this work, we assume the location information of each wireless sensor node is precise.

In addition, we assume that each sensor node v has enough sensing range such that it is able to observe any point p in Ω where the observability of p from v depends on the distance $\|vp\|$ between them. Generally speaking, the sensing ability of a sensor node to a point has monotone decreasing property with the increment of the distance between an object (point) being sensed and the sensor node's location. In this paper, we use euclidean distance as the measurement of QoS. Before we formulate the questions to be studied in this paper, we introduce some definitions which will be used in the paper.

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

Definition 2. *Given a path P connecting a source point S and a destination point T, the **k-support** of P, denoted by $S_k(P)$, is defined as the* maximum *kth-distance of all points on P, i.e., $S_k(P) = \max_{p \in P} \ell_k(p, U)$, where p is a point on the path P.*

Definition 3. *Given a path P connecting a source point S and a destination point T, the* ***k-breach*** *of P, denoted by $B_k(P)$, is defined as the* minimum *kth-distance of all points on P, i.e., $B_k(P) = \min_{p \in P} \ell_k(p, U)$, where p is a point on the path P.*

The main questions studied in the paper are as follows;

Question 1. Optimal k-Support Path (Best Case Coverage) Problem. Given a pair of points S and T in the region Ω, finding a path P inside Ω to connect S and T such that $S_k(P)$ is minimized.

Question 2. Optimal k-Breach Path (Worst Case Coverage) Problem. Given a pair of points S and T in the region Ω, finding a path P inside the field Ω to connect S and T such that $B_k(P)$ is maximized.

In the remaining part of this section, we present some key concepts that are critical to our polynomial solutions.

2.2 The kth-Nearest Point Voronoi Diagram

Definition 4. *Given a set of identical sensor nodes $U = \{u_1, u_2, \ldots, u_n\}$ deployed in the field Ω, we assign a geometry point p in Ω to the sensor node $u_i \in U$ if u_i is the kth-nearest sensor node of p. Following this assignment rule, we assign all points in the field to at least one sensor node in U. As a result, we obtain a collection of regions associated with sensor nodes in U, denoted by $\mathbb{W}_k = \{V_k(u_1), \ldots V_k(u_n)\}$, which forms a tessellation. We call the tessellation $\mathbb{W}_k$ the* ***kth-Nearest Point Voronoi Diagram (kth-NP Voronoi Diagram)*** *of U, and the region $V_k(u_i)$ the* ***kth-NP Voronoi region*** *of node u_i.*

Clearly, according to the definition of kth-NP Voronoi diagram, all points inside region $V_k(u_i)$ have the same kth-nearest sensor node $u_i \in U$. It is worth observing that $V_k(u_i)$ may consist of several independent polygons. Here, we call each independent polygon as the ***kth-NP Voronoi cell*** (denoted by $C_k(u_i)$) of node u_i and name u_i as the ***kth-owner*** (abbreviated to ***owner***) of $C_k(u_i)$. In addition, an edge of some kth-NP Voronoi cell is called a ***kth-NP Voronoi edge*** and the intersection point of any two kth-NP Voronoi edges is named as the ***kth-NP Voronoi vertex***. When some point (e.g., falling on some kth-NP Voronoi edge) has multiple owners, it randomly chooses one of them as the owner.

Definition 5 (Perfect Support Location). *The perfect support location of a kth-NP Voronoi edge is defined as the point with the minimum kth-distance on the edge, i.e., point p is the perfect support location of the kth-NP Voronoi edge e iif $\|pu_i\| = \min_{\forall p \in e}\{\|pu_i\|\}$, where $u_i \in U$ is the owner of e.*

It is worth mentioning that for each kth-NP Voronoi edge, it has one and only one perfect support location.

2.3 Difference between kth-NP Voronoi Diagram and Order-k Voronoi Diagram

Since most of our results are based on kth-NP Voronoi diagram, most properties of which are unknown to the literature, it is helpful to briefly discuss the differences between kth-NP Voronoi diagram and another well-known concept, *order-k Voronoi diagram* [2].

Definition 6 (The Order-k Voronoi Diagram[2]). *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 $U^{[k]}$, where $U^{[k]}$ is a subset of nodes in U with cardinality k. Each polygon is named order-k Voronoi cell $C_{ok}(U^{[k]})$ corresponding to the subset $U^{[k]}$ of U.*

According to the Definition 4 and Definition 6, the kth-NP 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 kth-nearest sensor node $\in U$ while the order-k Voronoi diagram [2] 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. Please refer to the examples shown in Fig. 5 (in Appendix, available in the online supplemental material) for details.

2.4 Compute the kth-NP Voronoi Diagram in Polynomial Time

We first present the method to compute the kth-NP Voronoi diagram with respect to sensor node set U in polynomial time since the kth-NP Voronoi diagram plays an important role in our solutions.

Definition 7 (The Farthest Point Voronoi Diagram). *The farthest point Voronoi diagram is a special case of kth-NP Voronoi diagram when $k = n$. It is a partition of the plane into polygons such that points in the same polygon have the same farthest sensor node u_i out of U with cardinality n. Each polygon is called a* ***farthest Voronoi cell****.*

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

Proof. Please refer to the supplementary file, available online. □

Based on Lemma 1, we propose the following Algorithm 1 with time complexity $O(k^2 n \lg n)$ (proved in Lemma 6) to compute the kth-NP Voronoi diagram of the sensor node set U. The main idea of our method is as follows:

1. Compute the order-k Voronoi diagram of the given sensor nodes set U using the algorithm given in [7].
2. For each order-k Voronoi cell $C_{ok}(U^{[k]})$ (defined in [7]), we compute the farthest Voronoi diagram of its corresponding k sensor nodes set $U^{[k]}$ following the method in [14]; and for each sensor node $u_i \in U^{[k]}$, return the corresponding farthest Voronoi cell as a part of u_i's kth-NP Voronoi cell.
3. For each sensor node u_i, we union any two kth-NP Voronoi cells computed in step 2 as one kth-NP Voronoi cell if they both have the same owner and share at least one edge. After the union operation, we get kth-NP Voronoi diagram G of U.

Now, we are ready to present our polynomial time algorithms computing the optimal k-support path and the optimal k-breach path within $O(k^2 n \log n)$ time.

Algorithm 1. Computing kth-NP Voronoi Diagram.
Input: The set of sensor nodes U.
Output: The kth-NP Voronoi diagram G of U.
1: Compute U's order-k Voronoi diagram;
2: **for** Each order-k Voronoi cell $C_{ok}(U^{[k]})$ **do**
3: Compute the farthest point Voronoi diagram using corresponding k sensor nodes in $U^{[k]}$;
4: **end for**
5: **for** Each kth-NP Voronoi edge e **do**
6: **if** If two polygons having the same owner share e **then**
7: Merge these two polygons into one polygon;
8: **end if**
9: **end for**
10: **for** Each sensor node u_i **do**
11: Return the union of all polygons belongs to u_i as u_i's kth-NP Voronoi cells;
12: **end for**

Given a set of sensor nodes U with cardinality n and the continuous field Ω, our algorithm computing the optimum k-support path mainly consists of two phases. In the first phase, we use Algorithm 1 to compute the kth-NP Voronoi diagram G in time $O(k^2 n \log n)$. During the second phase, we construct a new weighted graph G' based on kth-NP Voronoi diagram $\mathbb{W}_k = \{V_k(u_1), \ldots V_k(u_n)\}$ (output of Algorithm 1) and then compute the optimal k-support path on G' in time $O(k^2 n \log n)$.

3 Best Case Coverage: Optimal k-Support Path

In this section, we address the optimal k-support path problem, i.e., for any source/destination pair of points S and T in the given area Ω, finding a path with minimum k-support among all possible paths connecting S and T in Ω.

3.1 Preliminaries

Before we introduce the main idea of our solutions, we first present some useful theoretical results which are useful for proving the correctness of our algorithms.

Theorem 2. *Given any path P_1 connecting a source node S and a destination node T inside region Ω, we can always construct another (maybe same) path P_2 consisting of only a finite number of line segments such that*

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

Proof. Please refer to the supplementary file, available online. □

Based on Theorem 2, we further prove the following Theorems 3 and 4.

Theorem 3. *Given any path P_1 connecting the source S and the destination T inside of region Ω, we can always find a path P_3 consisting of line segments whose end points are perfect support locations only such that*

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

Proof. Please refer to the supplementary file, available online. □

Theorem 4. *There is one optimal k-support path consisting of only line segments whose end points are located exactly at some perfect support locations of kth-NP Voronoi edges.*

Proof. Please refer to the supplementary file, available online. □

3.2 Compute the Optimum k-Support Path

As shown in Theorem 4, there must exists at least one optimum k-support path connecting the given source/destination pair of points S and T, which consists of only line segments whose end points are the perfect support locations of some kth-NP Voronoi edges. Hence, we can limit the solution space to all paths consisting of only line segments, which connect perfect support locations of kth-NP Voronoi edges. Among all paths consisting of these line segments, the one with the minimum k-support must be one of the desired optimal k-support path. The main idea is as follows; First, we construct a new graph G' based on kth-NP Voronoi diagram G as follows:

1. For each kth-NP Voronoi edge e in kth-NP Voronoi diagram $\mathbb{W}$, we add a new node v' (to $V[G']$) whose location is the perfect support location of e. Notice that when multiple kth-NP Voronoi edges share 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 kth-NP Voronoi edge e in $\mathbb{W}$ and each node $v' \in V[G']$. Finally, we add the source and destination points S and T to $V[G']$.
2. Use the following rules to assign weight to each node in $V[G']$. If node $v' \in V[G']$ is S or T, the weight of v' (denoted by $w(v')$) is equal to the kth-distance of S or T in $\mathbb{W}$ (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 kth-distance of the perfect support location of edge $e \in \mathbb{W}$, where $e \in \mathbb{W}$ is corresponding to $v' \in G'$ in the one-to-one (or many-to-one) mapping.
3. Divide all nodes in $V[G']$ into groups such that a bunch of nodes (including S and T) are in the same group iff their mapping kth-NP Voronoi edges in $\mathbb{W}$ belong to the same kth-NP Voronoi cell. Notice, different groups may contain same node(s) since multiple kth-NP Voronoi edges may map 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 such that we have a simple graph G'.
4. 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 T. Here, the weight of a path is equal to the maximum weight of all edges it contains and a path P' (connecting S to T) in G' is said to be a minimum weight path iff P' has the minimum weight among all paths connecting S to T. Finally, we get one optimum 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 T.
Output: Minimum k-support Path Connecting S to T.

for each vertex v' in graph G'**do**
  k-support[v'] $\leftarrow$ infinity
  previous[v'] $\leftarrow$ undefined
end for
k-support[S'] $\leftarrow w(S)$
$Q \leftarrow$ all nodes in graph G'
while Q is not empty **do**
  $u' \leftarrow$ node in Q with smallest k-support
  remove u' from Q
  for Each neighbor v' of u': **do**
    $alt \leftarrow \max\{w(u', v'), k - support[u']\}$
    if alt < k-support[v'] **then**
      k-support[v']$\leftarrow alt$
      previous[v']$\leftarrow u'$
    end if
  end for
end while
Return P' which is the minimum weighted path connecting S to T

3.3 Correctness

We show the correctness of our algorithm by proving the following theorem.

Theorem 5. *Given any source/destination pair of points S and T inside the region Ω where sensor nodes in set U have been deployed, the path P returned by algorithm 2 is an optimal k-support path.*

Proof. Please refer to the supplementary file, available online. □

Remembering that we have claimed that there are multiple approaches for connecting all perfect support locations inside one kth-NP Voronoi cell such that we may have multiple choices of graph G'. For example, we can connect all perfect support locations in a complete graph manner, i.e., adding an edge between each pair of perfect support locations (assume that the resultant graph is G''). Clearly, running Algorithm 2 on G'' definitely returns an optimum k-support path as well since G'' is a complete graph containing all edges connecting each pair of perfect support locations. The reason for us to connect perfect support locations linearly, i.e., connecting them after sorted, is that we try to reduce the time complexity of our algorithm since the smaller the number of edges in G' is, the less running time is.

3.4 Time Complexity Analysis

In this section, we prove that the time complexity of our algorithm is $O(k^2 n \log n)$, which is better than that of the work [15]. Here, n is the number of deployed sensor nodes and k is the coverage degree. As shown before, our algorithm is composed by two phases, one is to compute the kth-NP Voronoi diagram, and the other one is to find the optimal k-support path based on the solution of the first phase. We analyze the time complexity of these two phases one by one. We first give a bound of the time complexity for the first phase.

Lemma 6. *The time complexity of Algorithm 1 computing the kth-NP Voronoi diagram is $O(k^2 n \lg n)$, where n is the cardinality of the set U containing all deployed sensor nodes and k is the coverage degree.*

Proof. Please refer to the supplementary file, available online. □

Next, we show the time complexity of the second phase. Since the time complexity of the second phase is determined by the number of kth-NP Voronoi cells and the number of kth-NP Voronoi edges, we first prove some properties of kth-NP Voronoi diagram, which is new to the literature.

Lemma 7. *For a set of sensor nodes in U with cardinality n on the field Ω and its kth-NP Voronoi diagram $\mathbb{W}$, the total number of kth-NP Voronoi edges is bounded by $O(k^2 n)$ and the number of edges of each kth-NP Voronoi cell is $O(n)$.*

Proof. Please refer to the supplementary file, available online. □

Lemma 8. *The time complexity of our algorithm to compute the optimum k-support path based on kth-NP Voronoi diagram is $O(k^2 n \log n)$.*

Proof. Please refer to the supplementary file, available online. □

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

Theorem 9. *The time complexity of our algorithm to find an optimum k-support path is $O(k^2 n \log n)$, where n is the cardinality of the set U containing all deployed sensor nodes and k is the coverage degree.*

4 Distributed Algorithm for Computing the Optimal k-Support Path

In this section, we present the distributed algorithm to compute the optimal k- support path after getting the kth-NP Voronoi diagram of U by Algorithm 1. First, we let each sensor node record its owned kth-NP Voronoi cells, including the geometry information of each edge of each kth-NP Voronoi cell. Here, we assume that each kth-NP Voronoi edge is assigned with an unique identity. It is worth observing that every kth-NP Voronoi edge may belong to one or two cells such that it can have multiple owners. The main idea of our distributed algorithm is based on Theorem 4.

First, we construct a new graph G' based on kth-NP Voronoi diagram G in a distributed manner. Our key point is to let each sensor node $u_j (1 \leq j \leq n)$ construct a new graph $G_{\mathcal{C}_i}^{u_j}$ locally for each of its own kth-NP Voronoi cells $\mathcal{C}_i$. And G' is 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 the sensor node u_j, u_j constructs a local graph for cell $\mathcal{C}_i$ by the following steps:

- Initially, the vertex set $V(G_{\mathcal{C}_i}^{u_j})$ and the link set $E(G_{\mathcal{C}_i}^{u_j})$ are initiated to be empty.

- For each kth-NP Voronoi edge e belonging to $\mathcal{C}_i$ (owned by u_j), u_j adds a corresponding point v' to $G^{u_j}_{\mathcal{C}_i}$. Let v' utilize the perfect support location of e as its location and let the weight of v', denoted by $w(v')$, be equal to the k-distance of the perfect support location of edge e. In addition, we let the node v' has the same unique ID as its corresponding edge.
- If the source point S or the destination point T (or both) is owned by u_j, u_j adds the source point S or the destination point T (or both) to $G^{u_j}_{\mathcal{C}_i}$ as well. The weight of S (resp. T) is equal to the kth-distance of S (resp. T) in $\mathbb{W}$.
- Sort all vertices of $G^{u_j}_{\mathcal{C}_i}$ in decreasing (or increasing) order by their weights. Add an edge between each pair of adjacent two vertices in the sorted list to $E(G^{u_i}_{C_i})$. In other words, the graph $G^{u_j}_{\mathcal{C}_i}$ consists of a series of continuous line segments.
- 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 graph. Then, we run a distributed minimum weight path algorithm (modification from work in [19]) on graph G' to compute the minimum weighted path connecting S and T because all sensor nodes construct a connect network and they can exchange information via one- or multihop. A path is said to be a minimum weighted path connecting S and T if the maximum weight of all its line segments is minimized among all paths connecting S and T.

Then, we use Algorithm 3, which originates from Dijkstra's shortest path algorithm, to identify a path P' in G' to connect S and T such that the maximum weight among its edges' is minimized. Finally, by applying P' to graph G directly, we get an optimum k-support path P.

Algorithm 3. Distributed Optimum k-Support Path Algorithm.
Input: the kth-NP Voronoi diagram $\mathbb{W}$ of the sensor node set
U, the source point S and destination point T.
Output: the minimum k-support path connecting S and T.

1: $V[G'] \leftarrow \emptyset$; $E[G'] \leftarrow \emptyset$;
2: **for** Each sensor node $u_i \in U$ **do**
3: **for** Each kth-NP Voronoi cell $\mathcal{C}_j$ owned by u_i **do**
4: Use Algorithm 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 (in [19]) on G' to get an minimum weight path P' connecting S and T.
10: Return P'

Algorithm 4. Distributed Constructing New Graph G'.
Input: Sensor node $u_i \in U$ and a kth-NP 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 kth-NP 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 as that of e;
5: Consider the kth-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 T (or both) **then**
8: Add the source vertex S or the destination vertex T (or both);
9: Let the weight of S (resp. T) in $G^{u_i}_{C_i}$ be equal to the kth distance of S (resp. T) in $\mathbb{W}$.
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 one of its two end vertices' weights.
13: Return $G^{u_i}_{\mathcal{C}_j}$;

5 Worst Case Coverage: Optimal k-Breach Path

In this section, we address the optimum k-breach path problem. By taking advantages of the previous work [10], we prove that an optimal k-breach path lies along the kth-NP Voronoi edges only except the subpath connecting S (resp. T) to some point on some kth-NP Voronoi edge if S (resp. T) lies inside some kth-NP Voronoi cell rather than on some kth-NP Voronoi edge. The first step of finding an optimal k-breach path is to compute the kth-NP Voronoi diagram of the sensor node set U, which can be computed by Algorithm 1.

5.1 Preliminaries

Before we go through the details of the algorithm to solve the optimal k-breach path problem, we give some theoretical results in advance, which illustrate the main idea and the correctness of our algorithm.

Theorem 10. *Given any path P_1 connecting source point S and the destination point T, we can always construct another (maybe same) path P_4 which only uses kth-NP Voronoi edges (neglecting the subpath from S or T to one of edges of the kth-NP Voronoi cell containing S or T) such that*

$$B_k(P_4) \geq B_k(P_1).$$

Proof. Please refer to the supplementary file, available online. □

Obviously, the following theorem immediately follows:

Theorem 11. *There is one maximum k-breach path which lies along the kth-NP Voronoi edges (except the first edge or last edge when S or T is not on some Voronoi edge).*

5.2 Compute the Maximum k-Breach Path

We are now ready to present our algorithm to compute the optimum k-breach path. For any given set of sensor nodes U

and source/destination pair of points S and T, The main idea of our approach is as follows:

1. Use Algorithm 1 to generate kth-NP Voronoi diagram $\mathbb{W}$ of U.
2. Add all kth-NP Voronoi vertices (including S and T) and all kth-NP Voronoi edges to new graph G'.
3. Each kth-NP Voronoi vertex $v \in \mathbb{W}$ is assigned a weight $w(v)$, which is equal to the kth-distance of v, i.e., $\ell_k(v, U)$.
4. If S (resp. T) is inside some kth-NP Voronoi cell $\mathcal{C}_k(u_i)$, rather than on some kth-NP Voronoi edge, we draw a line from u_i to S (resp. T), where u_i is the owner of the kth-NP Voronoi cell $\mathcal{C}_k(u_i)$. We further extend this line segment to some edge of $\mathcal{C}_k(u_i)$ (assuming that the intersection point is a). We add an edge between S (resp. T) and a along with the point a itself to G'. When S (resp. T) exists inside some kth-NP Voronoi cell whose edges contain part of the boundary of Ω, we can add an edge between S (resp. T) to each perfect support location of this cell (explained later).
5. For each edge (u, v) ((u, v) may be a newly added edge or a kth-NP Voronoi edge in G'), we let the weight of (u, v) be equal to the minimum kth-distance among all points on (u, v).
6. Finally, by applying Algorithm 5 to G', we get one optimal k-breach path connecting S to T.

Algorithm 5. Compute the Optimum k-Breach Path.
Input: G, source point S, destination point T.
Output: An optimum k-breach path connecting S to T.

for each vertex v in graph G **do**
  k-breach[v] $\leftarrow$ infinity
  previous[v] $\leftarrow$ undefined
end for
k-breach[S] $\leftarrow w(\mathrm{S})$
$Q \leftarrow$ the set of all nodes in graph G
while Q is not empty **do**
  $u \leftarrow$ node in Q with largest k-breach[]
  remove u from Q
  for each neighbor v of u: **do**
    $alt \leftarrow \min\{w(u, v), k - breach[u]\}$
    if $alt > k$-breach[v] **then**
      k-breach[v] $\leftarrow$ alt
      previous[v] $\leftarrow u$
    end if
  end for
end while
Return P, which is the maximum weighted path connecting S to T

5.3 Correctness

Based on Theorem 11, we know that there is one optimal k-breach path using the kth-NP Voronoi edges only (except the first edge and last edge which are used to connect S to T to some kth-NP Voronoi edge when S or T is not on some kth-NP Voronoi edge). Since the path P computed by our algorithm has maximize the minimum k-breach of it, P must be the optimal k-breach path among all those paths which lie on the kth-NP Voronoi edges.

5.4 Time Complexity Analysis

Compared to the algorithm used to identify the optimal k-support path in the previous section, the procedure of constructing a new graph G' is simpler. The following Theorem 12 gives us an upper bound on the time complexity of our method.

Theorem 12. *The time complexity of our algorithm computing an optimum k-breach path is $O(k^2 n \log n)$.*

Proof. Please refer to the supplementary file, available online. □

6 Running Procedure

In this section, we give the details of entire running procedure of proposed algorithms, including all the intermediate and final results. We implement and test our proposed k-support and k-breach algorithms to find the optimum k-support and k-breach paths by Matlab R2006a [1].

6.1 Experimentation Platform—Sample Results

In the simulation, a set of wireless sensor nodes with cardinality n (n varies from 5 to 20 with step 1) is randomly and uniformly deployed in the target square region with size $900 \times 900\ \mathrm{meter}^2$. We change the coverage degree k from 3 to 10 with step 1 for different cases. For each round, we randomly generate the position of each wireless sensor node. After that, we run Algorithm 1 to obtain order-k Voronoi diagram (intermediate results), based on which we get the corresponding kth-NP Voronoi diagram. (Some detailed examples are given in the Appendix, available in the online supplemental material.)

Next, we show how to find the optimum k-support path and the optimum k-breach path, respectively. Fig. 1 describes a complete procedure for obtaining an optimum 3-support path connecting source/destination pair of S and T in a $900 \times 900\ \mathrm{meter}^2$ square region where 10 wireless nodes have been randomly deployed. Given the area, deployed sensor nodes, and the positions of S and T, we first get the third-NP Voronoi diagram which is shown in Fig. 1a. We use different colors to distinguish third-NP Voronoi cells for different sensors such that a third-NP Voronoi cell has the same color as its owner's. The two black nodes inside of the square region denote the positions of S and T, respectively.

As we can see from Fig. 1a, each sensor could have several corresponding third-NP Voronoi cells. Next, we find the perfect support location for each third-NP Voronoi edge. For all perfect support locations (including S and T) belonging to the same third-NP Voronoi cell, we sort them in decreasing order by their third-distance, and use line segments to connect them one-by-one, thus get a graph G' (shown in Fig. 1b). Here, the node set of G' consists of source/destination pair of vertices S and T, and perfect support locations for all kth-NP Voronoi edges, and the edge set consists of line segments connecting the sorted perfect support locations inside each kth-NP Voronoi cell. As we can see, all boundaries in the original third-nearest point Voronoi cell turn into points in the new graph G' as well. One thing needs to be mentioned that when the boundaries

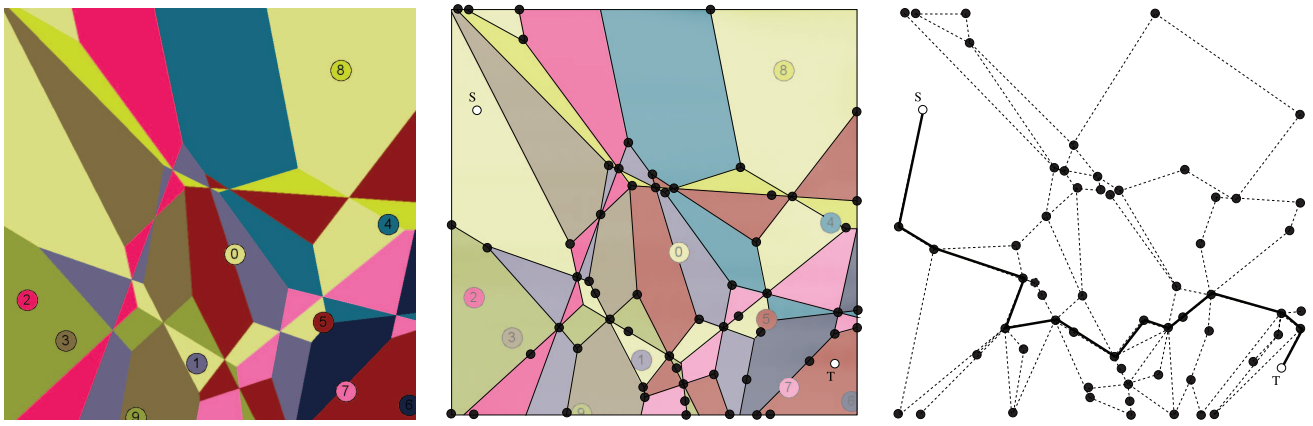


Fig. 1. (a) Third-nearest point Voronoi cell of 10 sensors in a 900×900 meter^2 square region. The IDs of nodes have been circled and filled with different colors. Each sensor node's third-NP Voronoi cell have been filled with the same color as its owner's. Two blace nodes are the source point S and destination point T, respectively. (b) Perfect support locations (indicated by black vertices) for each third-NP Voronoi edge (indicated by solid line segments). (c) Planar graph G' consisting of perfect support locations and edges (dotted line segments) connecting the perfect support locations inside each cell. Solid line denotes the optimum k-support path. ($k = 3$) in this case.

of Ω are not available, our algorithm still gets correct solutions. Then, we assign weights to the edges of graph G' such that the weight of each edge is equal to the bigger one of two third-distance of its two end points'. By considering S and T as the source/destination pair of vertices in graph G', the problem becomes to find a minimum weighted path connecting S and T in G' where the weight of a path is equal to the maximum one among all weights of line segments it contains. By running our k-support path algorithm, we obtain the minimum weight path connecting S and T in graph G', which is shown in Fig. 1c (the solid path). Clearly, this path can be applied to graph G directly, hence it is the final optimal k-support path. It is worth mentioning that there may exist multiple optimum k-support paths depending on the different ways to connect all perfect support locations of a single kth-NP Voronoi cell. In our case, we connect all perfect support locations inside a single kth-NP Voronoi cell by sorted kth-distance of them in decreasing order such that we get a graph G' with fewer edges (a tree inside each cell) to reduce the time complexity of our algorithms. Some other connection methods also can get optimum solutions, for example, we can connect each pair of perfect support locations inside each kth-NP Voronoi cell such as to have a clique.

According to the optimum k-breach path problem, we have proved that there must exist an optimum k-breach path using kth-NP Voroni edges only (Theorem 11). Hence, the first step is still to get the kth-NP Voronoi diagram, which is the same operation as obtaining a k-support path (shown in Fig. 1a') did. Next, we consider the graph G' consisting all kth-NP Voronoi edges and kth-NP Voronoi vertices. Notice that, if S (resp. T) is inside some kth-NP Voronoi cell, we use the method in Section 5.2 to connect S (resp. T) to some point on some kth-NP Voronoi edge, please refer to the Fig. 2a for illustration. Here, since both S and T are falling in some cells whose edges contain part of boundaries of Ω, we use two different ways to handle S and T to illustrate different cases. For instance, when all boundaries of Ω are available, we draw a line from wireless sensor node (assuming with ID 0) to S and further extend the line to the boundary. If all boundaries of Ω are not available, we can add edges connecting the destination

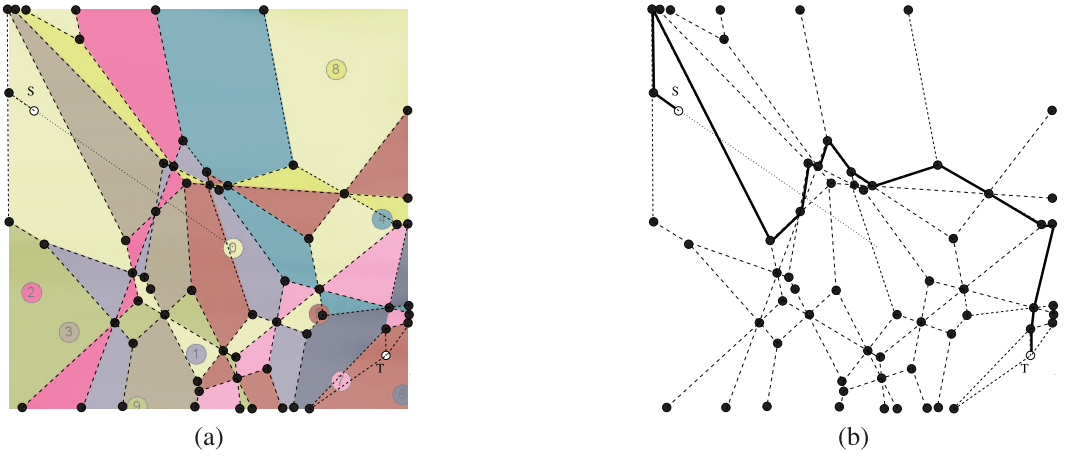


Fig. 2. k-breach path when $k = 3$ and $n = 10$. (a) Graph G' consisting of third-NP Voronoi vertices and edges including S, T, and corresponding edges incident to them. (b) The optimum 3-breach path (solid line segments) based on graph G' shown in dash line segments.

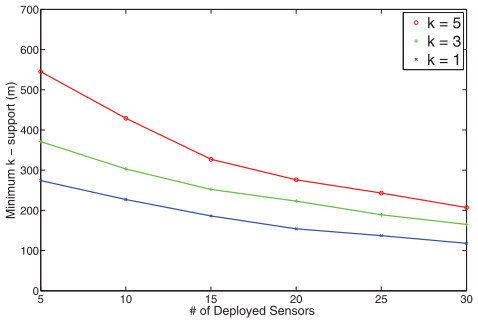


Fig. 3. Results for k-support paths.

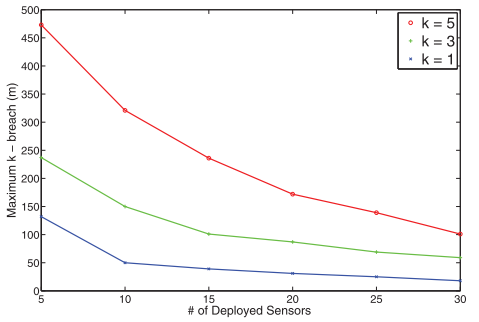


Fig. 4. Results for k-breach paths.

node T to all vertices of the kth-NP Voronoi cell containing T. See Fig. 2b for details. Next, after assigning a weight to each edge following the method in Section 5.2 and applying Algorithm 5 to graph G', we obtain an optimum k-breach path finally (shown in Fig. 2b). As we can see, although our methods for obtaining the k-support path and the k-breach path are based on same kth-NP Voronoi diagram, the intermediate graph G's for them are quite different. For k-support path problem, the intermediate graph G' contains perfect support locations only, each of which is not necessarily a kth-NP Voronoi vertex, on the contrary, any vertex in the intermediate graph G' (when we solve k-breach path problem) is exactly a kth-NP Voronoi vertex except the source point S and the destination point T.

We conduct more simulations and show some typical results as follows: In one set of our simulation, we let the number of sensors n increase gradually from 5 to 30 with step 5, and find the optimum k-breach and k-support paths for each $k \in \{1, 3, 5\}$. The results shown in Figs. 3 and 4 indicate that both optimum k-support and optimum k-breach decrease with the increment of the number of sensors as we expected.

Clearly, this result can be used to estimate the coverage quality when the number of sensors n and the required coverage degree k are given. If the sensing radius for each sensor node is fixed, it is better to deploy more sensor nodes to get better coverage quality. On the contrary, if the sensor can adjust its sensing (coverage) radius by power adjustment, the results presented here could be used to estimate the transmission power needed by sensors such that different requirements (k-support or k-breach) can be satisfied. In addition, we found that when the number of sensors exceeds some threshold value (around 20 in our case), the decreasing trend of the curve in Fig. 3 becomes slowly. 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.

7 Related Work

To evaluate the quality of coverage of the sensor network, Meguerdichian et al. [10] 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 [10], the authors observed that an optimal solution for the maximum support problem is a path which lies along the edges of the Delaunay triangulation [2], [13], and an optimal solution for the minimum breach problem is a path which lies along the edges of the Voronoi diagram [2], [13]. They further proposed centralized algorithms for both problems. Later, Mehta et al. [11] improved these algorithms and made them more computational efficient.

There were some other work which aimed at solving the 1-coverage problem formulated in [10] in a distributed manner. Li et al. [8] 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. [10] implied that a variation of the localized exposure algorithm presented in [13] can be used to solve the worst case coverage problem locally. Another localized algorithm with more practical assumptions was proposed by Huang and Tseng [4].

For the general coverage problem, Huang and Tseng [4] 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 [4], 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 [5]. The connected k-coverage problem was studied and addressed in [18] 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. [17] 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. [6] 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 [16], Wan and Yi 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, [11], [3] and [15] are the only work which aims to find an optimal k-covered path. In [11], Mehta et al., suggested that the worst case k-coverage problem can be addressed by adopting the kth-NP Voronoi diagram. However, no details of the proposed algorithm were given. In [3], Fang and Low 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 cannot 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 [15], we designed a centralized polynomial time algorithm that 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.

8 Conclusion

In this paper, we proposed polynomial time algorithms for two k-coverage problems, i.e., the optimum k-support path problem and the optimum k-breach path problem in wireless sensor networks. Our algorithms can efficiently find a path connecting two points in a given area where a sensor network is randomly deployed with the best observability (i.e., maximizing the minimum observability of all points on the path), and the worst observability (i.e., minimizing the maximum observability of all points on the path). We proposed a number properties for kth-NP Voronoi diagram, which are new to the literature. These properties may be of independent interests.

Acknowledgments

This work is supported by NSFC 61272426, China Postdoctoral Science Foundation funded project under grant No. 2012M510029, NSFC Major Program 61190110, National High-Tech R&D Program of China (863) under grant No. 2011AA010100, and the China 973 Program under Grant No. 2011CB302705. The research of Xiang-Yang Li is partially supported by the US National Science Foundtion (NSF) CNS-0832120, NSF CNS-1035894, NSF ECCS-1247944, National Natural Science Foundation of China under Grant No. 61170216, No. 61228202, and the China 973 Program under Grant No.2011CB302705.

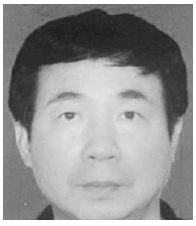
References

[1] http://www.mathworks.com/, 2013.
[2] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications.* Springer, 1997.
[3] C. Fang and C.P. Low, "Redundant Coverage in Wireless Sensor Networks," *Proc. IEEE Int'l Conf. Comm. (ICC),* 2007.
[4] C. Huang and Y. Tseng, "The Coverage Problem in a Wireless Sensor Network," *Proc. ACM Int'l Workshop Wireless Sensor Networks and Applications (WSNA),* 2003.
[5] C. Huang, Y.C. Tseng, and L. Lo, "The Coverage Problem in Three-Dimensional Wireless Sensor Networks," *Proc. IEEE GLOBECOM,* 2004.
[6] S. Kumar, T. Lai, and J. Barlogh, "On k-Coverage in a Mostly Sleeping Sensor Network," *Proc. ACM MobiCom,* pp. 144-158, 2004.
[7] D. Lee, "On k-Nearest Neighbor Voronoi Diagrams in the Plane," *IEEE Trans. Computers,* vol. C-31, no. 6, pp. 478-486, June 1982.
[8] X. Li, P. Wan, and O. Frieder, "Coverage in Wireless Ad-Hoc Sensor Networks," *IEEE Trans. Computers,* vol. 52, no. 6, pp. 753-763, June 2003.
[9] S. Megerian, F. Koushanfar, M. Potkonjak, and M. Srivastava, "Worst and Best Case Coverage in Sensor Networks," *IEEE Trans. Mobile Computing,* vol. 4, no. 1, pp. 84-92, Jan. 2005.
[10] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M.B. Srivastava, "Coverage Problems in Wireless Ad-Hoc Sensor Networks," *Proc. IEEE INFOCOM,* pp. 139-150, 2001.
[11] D.P. Mehta, M.A. Lopez, and L. Lin, "Optimal Coverage Paths in Ad-Hoc Sensor Networks," *Proc. IEEE Int'l Conf. Comm.,* vol. 1, 2003.
[12] A. Okabe, B. Boots, K. Sugihara, and S.N. Chiu, *Spatial Tessellations.* Wiley Series in Probability and Statics. John Wiley & Sons 2000.
[13] J.O. Rourke, *Computational Geometry in c.* Cambridge Univ. Press, 1998.
[14] S. Skyum, "A Sweepline Algorithm for Generalized Delaunay Triangulations," Technical Report DAIMI PB-373, CS Dept., Aarhus Univ., 1991.
[15] S. Tang, X. Mao, and X.Y. Li, "Evaluating Coverage Quality through Best Covered Paths in Wireless Sensor Networks," *Proc. IEEE Int'l Conf. Network Protocols (ICNP),* pp. 99-108, 2011.
[16] P. Wan and C. Yi, "Coverage by Randomly Deployed Wireless Sensor Networks," *IEEE Trans. Information Theory,* vol. 52, no. 6, pp. 2658-2669, June 2006.
[17] 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 Trans. Sensor Networks,* vol. 1, pp. 36-72, 2005.
[18] Z. Zhou, S. Das, and H. Gupta, "Connected k-Coverage Problem in Sensor Networks," *Proc. Int'l Conf. Computer Comm. and Networks (ICCCN),* 2004.
[19] S. Zhu and M. Huang, "A New Parallel and Distributed Shortest Path Algorithm for Hierarchically Clustered Data Networks," *IEEE Trans. Parallel and Distributed System,* vol. 9, no. 9, pp. 841-855, Sept. 1998.

Xufei Mao (M'10) received the bachelor's degree in 1999 and the MS degree in 2003 from Shenyang University of Technology and Northeastern University, respectively. He received the PhD degree in 2010 in computer science from the Illinois Institute of Technology, Chicago. He is a postdoctor at School of Software, Tsinghua University, Beijing, China. He is also an associate researcher in Internet of Things Center, TNLIST, Tsinghua University China. His research interests include wireless ad hoc networks, wireless sensor networks, pervasive computing, mobile cloud computing and game theory, and so on. He is a member of the IEEE.

Yunhao Liu (M'02-SM'06) received the BS degree in automation from Tsinghua University, Beijing, China, in 1995, and the MS and PhD degrees in computer science and engineering from Michigan State University, East Lansing, in 2003 and 2004, respectively. He is a professor with the Tsinghua National Lab for Information Science and Technology, School of Software, and the director of the MOE Key Lab for Information Security, Tsinghua University. He is also a faculty member with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong. He is a senior member of the IEEE.

ShaoJie Tang received the BS degree in radio engineering from Southeast University, P.R. China in 2006. He has been working toward the PhD degree from the Department of Computer Science at Illinois Institute of Technology since 2006. His research interests include algorithm design and analysis for wireless ad hoc networks, wireless sensor networks, and online social networks. He is a member of the IEEE.



Huafu Liu is a professor in the Department of Computer Science and Technology Changsha University China. His primary research interests include the design and implementation of wireless sensor networks, pattern recognition, and so on.

Jiankang Han received the BS and master's degrees in 2009 and 2012, respectively, in computer science from Haishi University, China and Beijing University of Posts and Telecommunications, respectively. He is an engineer of Tsinghua National Lab of Information Science and Technology. His research interests include hardware design and implement of wireless sensor networks.

Xiang-Yang Li (M'99-SM'08) received the bachelor's degree from the Department of Computer Science and the bachelor's degree from the Department of Business Management from Tsinghua University, P.R. China, both in 1995. He received the MS degree in 2000 and the PhD degree in 2001 from the Department of Computer Science, University of Illinois at Urbana-Champaign. He has been an associate professor (since 2006) and an assistant professor (from 2006) of computer science at the Illinois Institute of Technology. He is a senior member of the IEEE.