WIRELESS COMMUNICATIONS AND MOBILE COMPUTING
Wirel. Commun. Mob. Comput. 2012; **12**:1053–1065
Published online 22 September 2010 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/wcm.1031

RESEARCH ARTICLE

Providing and finding *k*-road-coverage efficiently in wireless sensor networks

Xufei Mao[1]*, Xiaohua Xu[2], ShaoJie Tang[2] and Xiang-Yang Li[2]

[1] Beijing Key Lab of Intelligent Telecommunications Software and Multimedia, BUPT, Beijing, China
[2] Department of Computer Science, Illinois Institute of Technology, Chicago, IL, U.S.A.

ABSTRACT

In this paper, we study k-road-coverage problems in wireless sensor networks (WSNs). Assume there is a 2-dimensional area Ω with a given road map $\mathcal{R} = (V, E)$ where E contains all road segments and V consists of all intersection points on Ω. The first question we study is about 'sensor deployment', *i.e.*, how to deploy a minimum number of sensor nodes on Ω such that each path (each road segment) on $\mathcal{R}$ is k-covered when all sensor nodes have the same sensing range. When sensors can only be deployed in a set of discrete locations, we propose an efficient method with the approximation ratio $6 + \epsilon$ for the special case where $k = 1$ and $O(k)$ generally. If sensors can be deployed in arbitrary locations, we propose an efficient method with the approximation ratio $24 + \epsilon$ when $k = 1$ and $O(k)$ generally. The second question we study is about 'path query', *i.e.*, how to find the k-covered path or k-support path connecting any given source/destination pair of points on the road map $\mathcal{R}$. Basically, given any source/destination pair of points S and D, we present two algorithms which can efficiently find a k-covered path connecting S and D and a k-supported path connecting S and D, respectively. Copyright © 2010 John Wiley & Sons, Ltd.

KEYWORDS
k-coverage; wireless sensor networks; *k*-support; path query

***Correspondence**
Xufei Mao, Beijing Key Lab of Intelligent Telecommunications Software and Multimedia, BUPT, Beijing, China.
E-mail: xmao3@iit.edu

1. INTRODUCTION

Wireless sensor networks (WSNs) have drawn considerable research interests in the last decade. At the same time, many new challenging research related questions have been posed, such as energy efficient routing, data processing, localization, and network diagnosis. Some of these challenging research questions have been successfully addressed, and a number of practical systems have been developed and used in practice. In many WSN applications, we are often required to deploy a bunch of sensor nodes to monitor a given area. For example, when we deploy some fire-alarm sensor nodes to monitor some building, we may require the whole building to be dead angle free. In other words, every place in the building should be in the sensing range of at least one sensor. This kind of problem is called 1-coverage problem in WSN. In some other cases, we may require a (or some) point(s) in the monitored area to be covered by multiple sensors at the same time. For instance, we want to track a moving object (carrying a sensor node) by some other anchor (fixed position) sensor nodes in some given area. As we know, in order to get the accurate location of the moving sensor, we often require the moving sensor to stay in the communication ranges of at least (typically) three anchor sensor nodes without further hardware support (*e.g.*, using angle of antenna). To achieve this, we actually require that every possible path (or say every feasible location) used by this moving sensor should be within the communication range of multiple anchor sensors (at least three in the aforementioned example) at the same time. Generally speaking, in order to maintain a certain quality of service (QoS), we may require every path (or point) in some area to be covered by multiple sensor nodes, *e.g.*, at least k sensor nodes for a general k. This kind of question is called the k-coverage problem. Notice here, a point is said to be covered by a sensor node *iff* the point is falling into the sensing range of the sensor node. A path is said to be k-covered *iff* all the points on this path are k-covered. In addition, we use 'coverage degree' of a point (resp. path) to denote the total number of sensors which cover this point (resp. path).

In this paper, we mainly study two questions. The first question we study is about 'sensor deployment'. Given an area Ω with a road map $\mathcal{R}$, how to deploy sensor nodes in this given area such that all road segments on the road map are k-covered. In other words, any point on any road segment should be in the sensing range of at least k sensor nodes at the same time, *i.e.*, has coverage degree at least k. Clearly, without considering the cost, we may deploy as many sensors as possible at feasible locations by a greedy method until all road segments are k-covered. However, some nodes may be redundant and make no contribution to the coverage since some sensor nodes may share the common sensing region. In other words, even if we remove some sensor node(s) or turn its (their) power off, the coverage property will not change. In this paper, we assume all sensor nodes have same sensing range and our objective is to minimize the total number of sensor nodes needed in order that all road segments on $\mathcal{R}$ are k-covered.

The second question we study is about 'path query', which is based on the following assumption. Assume a bunch of sensor nodes have been deployed on the given area Ω with a road map $\mathcal{R}$. In addition, different sensor nodes may use different power levels thus have different sensing ranges. Given any source/destination pair of points S and D on area Ω, the first case we study is how to find a path connecting S and D such that the resultant path is k-covered.

Clearly, whether a point p is covered by a sensor node v is a '$0/1$' problem, *i.e.*, the indicator returns 1 if p is covered by v; otherwise returns 0. However, from the point of view of QoS, the coverage problem could be further extended. For example, two different points in the same sensing range of a sensor node v may have different 'observability' by v depending on the Euclidean distances between the two points and v. Here, the 'observability' depends on different applications. In the literature, two seemingly contradictory, yet related viewpoints of coverage exist: worst and best case coverage. In the worst case coverage (a.k.a., *minimum breach coverage*), we attempt to quantify the QoS by finding areas of the lowest observability from sensor nodes and detecting breach regions (paths). In best case coverage (a.k.a., *maximum support coverage*), our goal is to find areas of the highest observability from sensors and identifying the best support and guidance regions (paths). In this work, the second case about 'path query' we study is the best case coverage problem in which we simply assume the observability of a point by a sensor node is the Euclidean distance between them. Formally speaking, we study how to find the optimal k-support path (connecting any given source/destination pair of points S and D) on the road map $\mathcal{R}$. Here, a path is called optimal k-support path *iff* it is k-covered and has the minimum k-support among all such k-covered paths. (The concept of k-support will be defined in Section 3 formally.)

To the best of our knowledge, [1,2] and [3] are the only work so far which address the problem of finding an optimal k-support path. In [1], Mehta *et al.* suggested that the worst case k-coverage problem may be addressed by adopting the kth nearest Voronoi diagram. However, no algorithm and theoretical results were given. In Reference [2], Fang *et al.* gave a polynomial time algorithm to identify a k-support path based on binary search and growing disk techniques. Unfortunately, their algorithms cannot guarantee an optimum solution, *i.e.*, the path found by their method may not be the optimum path. Furthermore, they assumed that k is a small constant independent of n. This assumption may reduce the generality of their algorithms since the value of k could be up to the number of sensor nodes n depending on different application and QoS requirements. In one of our previous work [3], we gave a polynomial algorithm to find an optimum k-support path with a general k, *i.e.*, k could be any integer value between 1 and n (the total number of sensor nodes). However, all previous works are based on the assumption that any found path is a feasible path, which is not realistic. Here, we say a path is feasible if it is consisting of the road segments in the given road map. As we know, in most cases, only the paths consisting of the road segments on the road map are feasible due to the physical or other constraints.

The main contributions of this work are as follows. For the 'sensor deployment' problem, when sensors can only be deployed in a set of discrete locations, we give the $(6+\epsilon)$-approximation algorithm to find the minimum number of wireless sensor nodes needed to make all road segments (on the given map) 1-covered. Following this, we present the $O(k)$ approximation algorithm when all road segments are required to be k-covered; when sensors can only be deployed in arbitrary locations, we present an algorithm with the approximation ratio $24+\epsilon$ for $k=1$ and $O(k)$ for a general k. For the second 'path query' problem, we present two efficient algorithms which can find the k-covered path and the k-support path connecting any given source/destination pair of points on the road map when such paths exist. According to the second 'path query' problem, our main idea is based on the kth nearest point (KNP) Voronoi diagram which is not well studied before. Since the properties of KNP Voronoi diagram is quite different from the ordinary Voronoi diagram, in our paper, we first propose an efficient algorithm to generate the KNP Voronoi diagram by combining computational geometry techniques, with graph theoretical and algorithmic techniques. More importantly, we prove and present a number of theoretical results on 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 5 formally.) To the best of our knowledge, these properties studied in this paper about kth nearest-point Voronoi are not known in the literature. In addition, compared with one of our previous work [3], we further reduced the time complexity for obtaining an optimal k-support path from $O(k^2n^2)$ to $O(k^2n\log n)$ where n is the total number of deployed sensors.

The rest of the paper is organized as follows. We first review some related results in Section 2. In Section 3, we define the terms and notations and give the details of the network model used in the paper. We present our two efficient algorithms that solve the 'sensor deployment' related *k*-coverage problems in Section 4. The algorithms for solving 'path query' related problems are discussed in Section 5. We conclude our paper and discuss possible future research topics in Section 6.

2. RELATED WORK

Since the coverage problem can be reduced to disk cover problem (which will be shown in Section 4), we briefly review the recent work [4,5] about the disk cover problem, in which the authors want to deploy some disks (with same radius or not) at some locations on the given area such that all points in the given point set are fully 1-covered. Basically, they studied several different cases, in which the feasible locations of sensor nodes may be a given discrete location set, may be only allowed to be along a line.

In order to evaluate the quality of coverage of the sensor network, Meguerdichian *et al.* [6] formulated the 1-coverage problem under two extreme cases: the best case coverage problem and the worst case coverage problem. They observed that an optimal solution for the best case coverage problem is a path which lies along the edges of the Delaunay triangulation [7,8] and an optimal solution for the worse case coverage problem is a path which lies along the edges of the Voronoi diagram [7,8]. They further proposed centralized optimal algorithms for both problems. Later, Mehta *et al.* [1] improved these algorithms and made them more computationally efficient.

There were some work as well which aimed at solving the 1-coverage problem formulated in Reference [6] in a distributed manner. Li *et al.* [9] showed that the best case coverage path can be constructed by using edges that belong to the relative neighborhood graph (RNG) of the sensor 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. In addition, a distributed algorithm based on RNG was proposed to solve the best case coverage problem. On the other side, Meguerdichian *et al.* [6] implied that a variation of the localized exposure algorithm presented in Reference [8] can be used to solve the worst case coverage problem locally. Another localized algorithm with more practical assumptions was proposed by Huang *et al.* [10].

For the general coverage problem, Huang *et al.* [10] 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. They formulated the problem as a decision problem and proposed a polynomial algorithm which can be easily translated to distributed protocols. In Reference [11], Huang *et al.* further extended this problem to a three-dimensional sensor networks and proposed a solution. The connected *k*-coverage problem was addressed in Reference [12] in which Zhou *et al.* studied the problem of selecting a minimum set of sensors which are connected and each point in a target area 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.* [13] 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*. Kumar *et al.* [14] considered the problem of determining the appropriate number of sensors that are enough to provide *k*-coverage of an area when sensors are allowed to sleep during most of their lifetime. In Reference [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. Santosh *et al.* [14], studies the *k*-coverage problem while concentrating on how to prolong the lifetime of WSNs.

As we have introduced before, [1] and [2] are the only two works which address the problem of finding an optimal *k*-support path. In Reference [1], Mehta *et al.*, suggested that the worst case *k*-coverage problem can be addressed by adopting the *k*th nearest Voronoi diagram. However, no details of the proposed algorithm were given. In Reference [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 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 [3], we designed a centralized polynomial time algorithm which can find an optimum *k*-support path efficiently for general *k*. In this paper, we further improved the upper bound of time complexity for obtaining an optimal *k*-support path.

3. NETWORK MODELS

We assume there is a 2-dimensional area Ω with a known road map $\mathcal{R} = (V, E)$. We consider all paths on $\mathcal{R}$ have been divided into road segments by all intersection points. The node set V contains all intersection points on map $\mathcal{R}$, and the edge set E consists of all road segments, denoted by $\{e_1, e_2, \cdots, e_{|E|}\}$. Here, $|E|$ is the cardinality of edge set E. We further assume that all road segments are line segments. In addition, there is a set of n wireless sensor nodes $U = \{s_1, s_2, \cdots, s_n\}$, which will be deployed and used to monitor Ω. We assume all sensor nodes have the same sensing power $\mathcal{P}$, hence the same sensing range, denoted by r. For simplicity, we assume that all deployed sensor nodes construct a connected WSN.

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

Definition 1. *Given a point p in the field Ω and the set of sensors U, we say p is k-covered if p is in the sensing range of at least k sensor nodes out of U. A path P in Ω is said to be k-covered if all points on P are k-covered.*

Definition 2. *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 3. *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 kth 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.*

We say a path P is *k-covered* with respect to the sensor node set U if every point on P falls in the sensing range of at least k sensor nodes out of U. In addition, we say a path P is the *optimal k-support path* with respect to U if the k-support of P is minimum among all k-covered paths connecting the source/destination pair of points. Notice, in this paper we only consider coverage of the road segments on the map $\mathcal{R}$.

3.1. Questions to be studied

In this paper, we mainly study two questions. The first question is related to 'sensor deployment', *i.e.*, how to deploy sensor nodes in U to Ω such that some special QoS is satisfied. We assume that the sensing ranges of all sensor nodes are same and study how to deploy minimum number of sensor nodes such that all feasible paths (road segments) in $\mathcal{R}$ inside Ω are *k-covered*, *i.e.*, minimize the cardinality of sensor node set U.

The second question we study in this paper is related to 'path query', *i.e.*, how to find a path on the given road map such that the resultant path satisfies some QoS requirements. For the 'path query' question, we assume all sensor nodes have been deployed with fixed geometry position, fixed sensing power (hence fixed sensing range). Notice here, different sensor nodes may be set to use different sensing power during the deployment such that they can have different sensing ranges. However, after being deployed, a sensor node will always use the same sensing power to work and cannot dynamically changing its sensing power. Assume, we have a 2-dimensional area Ω with a road map $\mathcal{R}$ and a WSN with size n has been deployed in Ω. In addition, each sensor node has a fixed position and fixed sensing power (range). The question we study is that given any pair of source/destination points S and D on some road segment of $\mathcal{R}$, how to find a feasible path (only using road segment on $\mathcal{R}$) to connect S and D such that the resultant path is k-covered. As we know, there might exist multiple k-covered paths. Among all of those k-covered paths, the next question we study is how to find an optimal k-support path. Here, we say a path P connecting S and D is an optimal k-support path if the k-support of P is minimum among all such paths.

4. DEPLOYING WIRELESS SENSOR NETWORK

In this section, we study how to deploy sensor nodes such that every path (road segment) in the road map $\mathcal{R}$ is k-covered. In other words, every point on any road segment of $\mathcal{R}$ is falling into the sensing ranges of at least k sensor nodes. We assume all sensor nodes have same sensing power $\mathcal{P}$, thus have the same sensing range r. Under this assumption, our objective is to select a minimum number of sensors such that every path in $\mathcal{R}$ is k-covered.

We first prove that the problem is NP-hard. When sensors can only be deployed in a set of discrete locations, we then present a $(6+\epsilon)$-approximation algorithm for the special case where $k = 1$; Following that, we can achieve a solution with an approximation ratio of $O(k)$ for a general k. When sensors can only be deployed arbitrarily, we present a $(24+\epsilon)$-approximation algorithm when $k = 1$; following that, we can achieve a solution with an approximation ratio of $O(k)$ for a general k.

4.1. The NP-hardness of 1-coverage problem

First we show that even for the special case $k = 1$, the problem (that is 1-coverage problem) is NP-hard. Clearly, this is sufficient to prove the NP-hardness of k-coverage problem.

Theorem 1. *It is NP-hard to find a placement of sensors with minimum number of deployed sensors such that all road-segments are covered by at least* 1 *sensor.*

Proof. We will prove the NP-hardness of this problem by construction of a reduction from a geometric set cover problem. First, we consider the following special geometric set cover problem, which was shown to be NP-hard in Reference [16]: covering a given set of points in the plane with the smallest number of disks from a given set of disks.

Next, we show that our problem is reducible from the geometric set cover problem. Recalling that, in our problem, we want each road segment to be covered by at least 1 sensor node. Clearly, the geometric set cover problem is a special case of our problem, in which we consider a segment as a point. For example, we assume there is any instance (*e.g.*, point set $P = \{p_1, p_2, \cdots, p_a\}$) of geometric set cover problem. Next, we draw a line segment with arbitrarily small length ϵ_1 to cross each point in the given instance. Clearly, the resultant segment set is an instance of our problem. Therefore, if we have a polynomial time algorithm which can find a minimum size of disks to cover all segments, we can directly use the resultant set of disks to cover all the given points in the geometric set cover problem. This finishes the proof. ■

4.2. Solutions with discrete feasible locations

We first study the case when sensors can only be deployed in a set of discrete *m* locations $Z = \{z_1, z_2, \cdots, z_m\}$. Here, a location z_i, $1 \le i \le m$ is a point in the 2-D dimension with coordinates (X_{z_i}, Y_{z_i}). As a preliminary step, we first show that making all road segments 1-covered is equivalent to making some points from the segments 1-covered when the sensor nodes can only be deployed in a given set *Z* of discrete *m* locations. As we know, the coverage area of a sensor node *u* is a disk centered at *u* with radius *r*. If we draw a circle centered at each point out of all feasible discrete *m* locations *Z* with radius *r*, each road segment will be divided into $\le 2m + 1$ subsegments. So, we will have $O(m|E|)$ subsegments in total. Assume the subsegments set is $L = \{l_1, l_2, \cdots, l_b\}$, where *b* is the total number of subsegments where $b = O(m|E|)$. Clearly, if we can deploy sensors such that all *b* subsegments are 1-covered, we are done. Let $P = \{p_1, p_2, \cdots, p_b\}$, where p_i is the middle point of the subsegment l_i. Next we will show that making all these $O(m|E|)$ subsegments *L* 1-covered is equivalent to making $O(m|E|)$ points *P* 1-covered by the following Lemma 2.

Lemma 2. *When the feasible locations of all the sensor nodes are in a given set of discrete m locations Z, L is 1-covered if and only if the set of middle points P is 1-covered.*

See Appendix for the proof of Lemma 2. Based on Lemma 2, we reduce the problem of covering all subsegments to the problem of covering all middle points of all subsegments by the same sensor set when the feasible locations of all sensors are in a given set of discrete *m* locations. Note that by mapping sensors to disks with radius *r* and furthermore scaling the radius of mapped disks from *r* to 1, we reduce our problem to an equivalent problem, which is known as *Discrete Unit Disk Cover (DUDC)* problem. The DUDC problem can be described as this: given a set $\mathcal{P}$ of points in the plane, and a set $\mathcal{D}$ of unit disks of fixed locations, the DUDC problem is to find a minimum-cardinality subset $\mathcal{D}' \subset \mathcal{D}$ that covers all points of $\mathcal{P}$. Next, we will give a $(6 + \epsilon)$-approximation algorithm for DUDC.

Our algorithm employs double partition and divide-and-conquer techniques similar to those employed in Reference [17]. Double partition means that we first partition the plane into big blocks, each with size $t\mu \times t\mu$, where $\mu = \frac{\sqrt{2}}{2}$ and *t* is a large integer constant. Then we partition each block into t^2 small squares, each with size $\mu \times \mu$. The process of double partition is illustrated in Figure 1.

After double partition, our algorithm for solving DUDC can be divided into three phases:

(1) Solve DUDC in a $t\mu \times t\mu$ block, this means all the points to be covered are contained in the block.
(2) Combine the sub-solutions of DUDC in all blocks obtained in the first phase to get a solution in the plane.

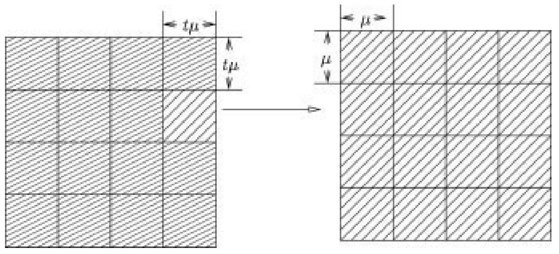

Figure 1. Double partition: Partition plane into blocks and further partition block into squares.

(3) Using shifting strategy to get a set of solutions in the plane similarly. Among all solutions in the plane, we return the one with minimum size as the final solution.

The detail of our algorithm for DUDC in the plane is shown in Algorithm 1.

Algorithm 1 Solve DUDC in a plane.

```
Input: $\mathcal{P}$ and $\mathcal{D}$.
Output: DUDC in a plane.
1: (Double partition) Partition the whole plane into blocks of size $t\mu \times t\mu$, then partition each block into squares with size $\mu \times \mu$, where $\mu = \sqrt{2}/2$.
2: Calculate DUDC for each block that contains points and merge the solutions together to form a solution for the whole plane. See appendix (subsection B) for our approach solving DUDC.
3: Move each block one square along its diagonal direction.
4: Repeat Step 2 for this new partition to update the solution if any better solution is found.
5: Repeat Step 3 for $t$ times, and output the final solution.
```

Notice that, the reason for us to choose μ as $\sqrt{(2)}/2$ in Algorithm 1 is that the diameter of the resultant squares is 1 such that any unit disk (we used in the paper) with center inside the square can cover all the points in the square. Finally, we show the approximation ratio of our algorithm for DUDC in a plane.

Theorem 3. *For any constant* ϵ, *by setting* $m = O(\frac{1}{\epsilon})$, *Algorithm* 1 *always outputs a disk set with size bounded by* $(6 + \epsilon) \cdot |OPT|$, *where* OPT *is the optimum solution and* $|OPT|$ *is the size of* OPT.

Proof. By Lemma 13 (in Appendix B), every disk in the optimal solution OPT can be counted at most six times for solving DUDC in a block. However it may be counted more in the boundary region of a block. When we shift the whole block many times, for any disk in OPT, it would be counted at most six times in most cases. Since we return the one with minimum size as the final solution, we can achieve a solution with size bounded by $(6 + \epsilon) \cdot |OPT|$ for any small constant $\epsilon > 0$. ■

4.3. Solutions with continuous feasible locations

In this subsection, we will solve the case that when the feasible locations for all sensor nodes are continuous, *i.e.*, sensors nodes can be deployed anywhere on the 2-dimensional area Ω. Basically, we will prove that our Algorithm 1 can provide a constant approximation solution with ratio $24+\epsilon$ when the feasible locations of sensors are continuous. Our main idea is that, we partition area Ω into small equilateral triangles and make the continuous feasible locations of sensors discrete.

First, we assume the area Ω is a square for simplicity. Actually, even Ω is not a square strictly, our results will not be affected. We partition area Ω into small equilateral triangles with edge length r. By doing this, we can show that only using some fixed discrete locations, our solution is still 'good', *i.e.*, has a constant approximation ratio. We first prove the following Lemma 4.

Lemma 4. *Given a square area Ω with side length L, we partition Ω into small equilateral triangles with edge length r, any sensor node inside Ω can be replaced by at most* 4 *sensor nodes such that the coverage degree of any road segment will not decrease. We assume L is multiple of r for simplicity.*

Proof. Clearly, after partition, we have totally $O((\frac{L}{r})^2)$ small triangles with side length r as shown in Figure 2(a).

Consider any equilateral triangle $\triangle ABC$ as shown in Figure 2(b), we will show that any sensor E located within $\triangle ABC$ can be replaced by at most four sensors located at four intersection points such that the coverage degree of any road segment will not decrease.

Assume the center of the triangle $\triangle ABC$ is O, then we can divide $\triangle ABC$ into three small triangles $\triangle OAB$, $\triangle OBC$ and $\triangle OCA$. Assume E is located within $\triangle OBC$, then disk centered at E with radius r can be entirely contained in the union of four disks centered at intersection points A, B, C, D with radius r. In other words, any (part of) road segment covered by E will also be covered after we use four sensors located only on the intersection points. The proof is similar when E is located within $\triangle OAB$ or $\triangle OCA$. ■

Based on Lemma 4, the following Lemma 5 is straightforward.

Lemma 5. *Assume OPT is the optimal feasible location set (with minimum cardinality, saying $|OPT|$), at which deploying sensor nodes can cover all the given points in Ω. We can always construct another feasible location set (with size at most $4|OPT|$) only using the intersection points of the partition such that deploying sensor nodes at the location in this set will cover all the given points as well.*

Next, we are ready to show that even when the feasible locations of sensor node are continuous, we can still get a good solution within $24+\epsilon$ times of the optimal solution by constructing a discrete feasible location set for all the sensor nodes.

Our main idea is as follows. We partition area Ω into small equilateral triangles with edge length r. Assume the set of all intersection points (*i.e.*, the vertices of all triangles) in the partition is $\mathcal{B}$. We show that the solution of Algorithm 1 (with $\mathcal{B}$ being the given discrete feasible location set) is no greater than a constant times of the optimal solution when the feasible locations for all sensor nodes are continuous.

Theorem 6. *When the feasible locations for all sensor nodes are continuous, the solution of Algorithm* 1 *using $\mathcal{B}$ as the given discrete feasible location set is within $24+\epsilon$ times of the optimal solution.*

Proof. Assume the optimal solution is OPT with cardinality $|OPT|$ when the feasible locations of all sensor nodes are continuous. By Lemma 5, we can always have

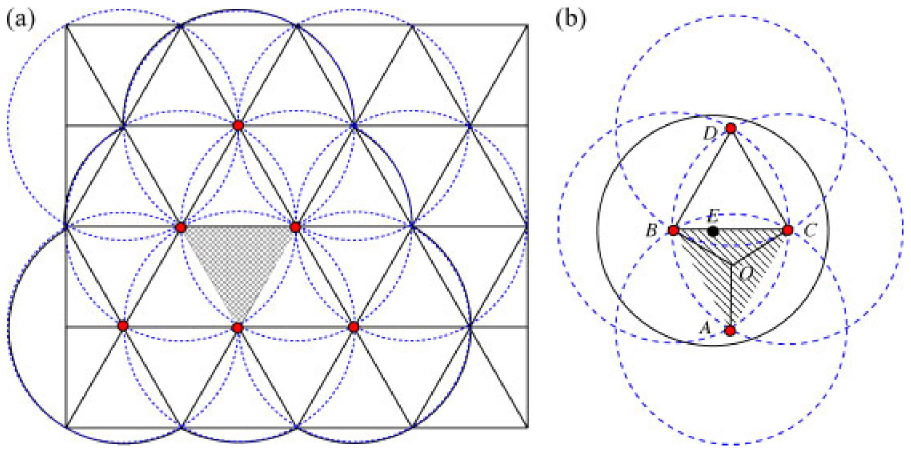

Figure 2. (a) Partition the area into equilateral triangles, use sensor nodes at the intersection points of the partition only. (b) Any sensor node (black node) inside the shaded equilateral triangle can be replaced by at most four red sensor nodes such that the coverage degree of any road segment will not decrease.

another solution $\mathcal{S}_1$ only using intersection points such that $|\mathcal{S}_1| \leq 4|OPT|$. Next, we assume the optimal solution of DUDC problem (using $\mathcal{B}$ as the given discrete feasible location set) is OPT_1. By Theorem 3, we can achieve a solution $\mathcal{S}_2$ that satisfies $|\mathcal{S}_2| \leq (6 + 0.25\epsilon)|OPT_1|$. Then we have

$$\begin{aligned}|\mathcal{S}_2| &\leq (6 + 0.25\epsilon)|OPT_1| \leq (6 + 0.25\epsilon)|\mathcal{S}_1| \\ &\leq (6 + 0.25\epsilon)4|OPT|.\end{aligned}$$

This finishes the proof. ■

Theorem 7. *For k-coverage problem, we can find a solution with O(k)-approximation.*

Proof. Since we have an algorithm with $O(1)$-approximation for 1-coverage problem. If we repeat the executions of the algorithm for k times, the union of all the returned results clearly yields a valid solution for k-coverage problem with the approximation ratio at most $O(k)$. ■

It will be interesting if we can find a $O(\log k)$-approximation (or $O(1)$ approximation) algorithm for DUDC problem for general k.

5. PATH QUERY ON THE ROAD MAP

In this section, we study the 'path query' related questions. Given a 2-dimensional area Ω with a road map $\mathcal{R}$, there are n wireless sensor nodes $\{u_1, u_2, \cdots, u_n\}$ deployed in Ω. Assume the position (x_i, y_i) and sensing range r_i of each sensor node u_i are known and fixed. We further assume different sensor nodes may have different sensing ranges, for example, we use r_i to denote the sensing range of the i^{th} sensor node u_i.

The question we study in this section is how to find the k-covered and k-supported paths using all feasible road segments on $\mathcal{R} = (V, E)$ to connect a given source/destination pair of points S and D. Here, E contains all road segments and V consists of all intersection points on Ω. See Figure 3 for illustration.

5.1. *K*-covered path query

Clearly, given any source/destination point pair S and D on the road map, to find a k-covered path connecting S and D, we only need to find a path such that every point on this path is covered by (falling in the sensing range of) at least k sensor nodes. Our main idea is as follows: (1) For each wireless sensor node $u_i : 1 \leq i \leq n$, we draw a circle centered at u_i with radius r_i. Clearly, the area Ω will be divided into subareas by all the circles we drew and the boundary of Ω. Here we do not consider those subareas outside Ω. (2) We give each subarea a rank which is equal to the number of sensor nodes that cover this subarea. For example, if a subarea is not covered by any sensor node, its rank will be 0. If a subarea falls into the sensing ranges of k sensor nodes, its rank

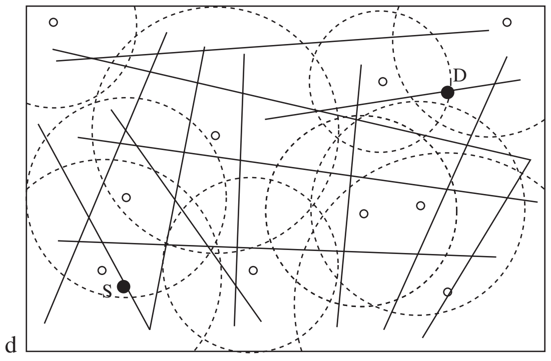

Figure 3. An example of path query. Black nodes denote the source point S and destination point D, respectively. Black curves indicate feasible paths on the road map $\mathcal{R}$. White nodes and dash circle denote the deployed sensor nodes and their corresponding sensing range.

will be k. (3) Next, we check each road segment $e \in E(\mathcal{R})$ one by one. If e goes into or go through some subarea(s) whose rank is less than k, we remove e from $E(G)$. In other words, we only keep road segments which are totally contained in some subareas with rank at least k. (4) From all remaining road segments, we can simply find a path to connect the source/destination pair S and D (by Breath First Search method or Depth First Search method) if such path exists. See Algorithm 2 for details. Clearly, the first step takes $O(n)$ time. The second step takes $O(n^2)$ time, which can be proved by a mathematical induction method. Since we have to check each road segment, the time complexity for the third step is $O(|E|)$. The last step will use $O(|E| + |V|)$ by a Breath First Search method. Hence, the total time complexity of Algorithm 2 is $O(n + n^2 + |E| + |E| + |V|) = O(n^2 + |E| + |V|)$, which means our algorithm can find a solution in polynomial time.

5.2. Optimal *k*-support path query

Recalling that a path P (connecting S and D) is called an optimal k-support path if the k-support of path P is the minimum one among all k-supported paths which connect S and D.

As we know, any path P connecting S and D on $\mathcal{R}$ consists of one or more road segments and the k-support of P is determined by the maximum k-support among all those road segments. Hence, we can compute the k-support of any such path P as long as we can compute the k-support of all road segments constructing P. Before we go through the details of the algorithm computing an optimal k-support path, we first introduce the concept of *kth Nearest-Point Voronoi Diagram* which can be used to compute the k-support for the road segments efficiently.

5.2.1. The *k*th nearest-point Voronoi diagram.

Given a set of identical sensor nodes U distributed in the field Ω, we assign a geometry point p in the field to the

Algorithm 2 Finding *k*-covered path

Input: Source/destination pair S and D on the given road map $\mathcal{R} = (V, E)$ with deployed sensor nodes set U.
Output: A *k*-covered path connecting S and D.

```
1: for Each sensor node $u_i : 1 \leq i \leq n$ with location $(x_i, y_i)$ and sensing range $r_i$ do
2:     Draw a circle centered at $u_i$ with radius $r_i$.
3: end for
4: $\Omega$ will be divided into small subareas by the drawn circles and the boundary of $\Omega$
5: for Each subarea do
6:     Set its rank to be equal to the number of sensor nodes which cover this subarea.
7: end for
8: for Each road segment $e \in E(\mathcal{R})$ do
9:     Remove it from $E(\mathcal{R})$ if $e$ goes through (into) some subarea(s) with rank smaller than $k$.
10: end for
11: Run BFS (or DFS) algorithm starting from $S$ among all road segments remaining in $E(\mathcal{R})$;
12: Return the first path from $S$ to $D$ if such path exist; else return FALSE;
```

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, for any point p in the area, it has one or more *k*th nearest sensor nodes in U. Basically, 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 $\mathbb{V}_k = \{V_k(u_1), \ldots V_k(u_n)\}$, which forms a tessellation. We call the tessellation $\mathbb{V}_k$ *the kth nearest-point Voronoi diagram* generated by U, and the region $V_k(u_i)$ the *k*th nearest-point Voronoi region of node u_i. From now on, we call *k*th nearest-point Voronoi diagram *KNP Voronoi diagram* for simplicity.

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 Reference [18]. Here we call each independent polygon *kth 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 as the *owner* of $C_k(u_i)$. Notice, the source point S and destination point T also have owners depending on the KNP Voronoi cells they belong to. If S (or D) are existing on one of edges of some KNP Voronoi cell, which means S(or D) can have two or more same distance k^{th} nearest sensor nodes, we randomly choose one such sensor node as the owner of S (or D).

The edge (including the boundary of area Ω) of each cell is called *kth nearest-point Voronoi edge* (KNP Voronoi edge) and the intersections of all KNP Voronoi edges are called *kth nearest-point Voronoi vertex* (KNP Voronoi vertex). From now on, when we say KNP Voronoi diagram, we mean the KNP Voronoi diagram with respect to the sensor node set U if there is no confusion.

Clearly, one of important properties of KNP Voronoi diagram is that all points inside a KNP Voronoi cell have the same *k*th nearest sensor node out of U. Hence, we can compute the *k*-support of each road segment by combining its geometry location with KNP Voronoi diagram in Algorithm 4. (Actually, we will show later that we may not need to compute the *k*-support of each road segment in order to get the optimal *k*-support path.)

Algorithm 3 Computing KNP Voronoi Diagram

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

```
1: Compute $U$'s order-$k$ Voronoi diagram;
2: for Each order-$k$ Voronoi cell $C_{ok}(U_i^{[k]})$ do
3:     Compute the farthest point Voronoi diagram using corresponding $k$ sensor nodes in $U_i^{[k]}$;
4: end for
5: for Each KNP Voronoi edge $e$ do
6:     if If two neighbor polygons which share $e$ belong to same sensor node $u_i$ then
7:         Merge these two polygons into one polygon which still belongs to $u_i$;
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 its KNP Voronoi cell;
12: end for
```

The main idea to compute the optimal *k*-support path is as follows. Firstly, we use the same way (described previously) to rank all the subareas. Secondly, we use Algorithm 3 to compute the KNP-Voronoi diagram G in time $O(k^2 n \log n)$. Thirdly, we compute the *k*-support for each road segment e if e is completely contained inside some subareas with rank no less than k by Algorithm 4. We further assign the weight of each such road segment with its *k*-support. Finally, we find a minimum weight path to connect S and D. Here the minimum weight path means the maximum weight of all its partial road segments are minimum.

Since our algorithm is based on the KNP Voronoi diagram, we present the method to compute the KNP Voronoi diagram with respect to sensor node set U first.

5.2.2. Compute the KNP Voronoi diagram.

Before we present our algorithm to compute the KNP Voronoi diagram with respect to 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 $U_i^{[k]}$.*

Definition 5 (The Farthest Point Voronoi Diagram). *The farthest point Voronoi diagram is a special case of kth nearest-point Voronoi diagram, when $k = n - 1$. It is a partition of the plane into polygons such that points in each polygon have the same farthest sensor node 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 8. *For any point p in the plane Ω, it belongs to sensor node $u_i \in U$'s KNP Voronoi cell $C_k(u_i)$ 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 $U_i^{[k]}$ are the k closest sensor nodes to 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 kth nearest sensor nodes of p.

On the other hand, if sensor node u_i is the point p's kth nearest sensor node, p must be located in some $C_{ok}(U_i^{[k]})$ where $U_i^{[k]}$ is the union of p's $1_{\text{st}}, 2_{\text{nd}} \ldots, (k-1)_{\text{th}}$ 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 8, we propose the following method (Algorithm 3) 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 Reference [19].
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 one 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 KNP Voronoi region. Finally, we get U's KNP Voronoi diagram G.

See detailed Algorithm 3 for illustration.

We first show that the time complexity of Algorithm 3 is $O(k^2 n \log n)$ by the following Lemma 9.

Lemma 9. *The time complexity of Algorithm 3 which is used to compute the KNP Voronoi diagram is $O(k^2 n \log n)$.*

Proof. First, using the algorithm given in Reference [19], we can compute the order-k Voronoi diagram of n sensor nodes within time $O(k^2 n \log n)$. 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 $O(k \log k)$ for each order-k Voronoi cell (proven in Reference [20]). Since there are $O(nk)$ order-k Voronoi cells (results showed in Reference [19], the time complexity of the second step is $O(k \lg k) \times O(nk) = O(k^2 n \log k)$. 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 10) and each merge operation uses constant time $O(1)$. So the time complexity for Algorithm 3 is $O(k^2 n \log n) + O(k^2 n \log k) + O(k^2 n) = O(k^2 n \log n)$. This finishes the proof. ■

Lemma 10. *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 References [19] and [18], 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 3 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. ■

Clearly, the KNP Voronoi diagram G will have intersections with some road segments of $\mathcal{R}$. Next, we use the same method to mark the sensing range of each sensor node and rank all subareas. Obviously, we only need to compute the k-support of road segments which are completely contained inside some subareas with rank no less than k. Then we use Algorithm 4 to compute the k-support for each such road segment based on the KNP Voronoi diagram. Clearly, the time complexity of Algorithm 4 is bounded by the possible number of partial segments that an input road segment can be divided into. Since any road segments can be divided into at most $O(k^2 n)$ (here, $O(k^2 n)$ is the total number of all KNP Voronoi Edges as proved in Lemma 10) partial segments, the time complexity is $O(k^2 n)$.

After we got the k-support of each road segment which is completely contained in some subarea(s) with rank at least k. We compute a minimum weight path connecting S and D by considering the k-support as the weight for each road segment. For example, we can run Dijkstra's algorithm by maintaining the maximum weight on a path instead of accumulating all weights on a path.

Algorithm 4 Computing k-support for a road segment

```
Input: KNP Voronoi diagram $G$ and road segment $e$.
Output: $k$-support of road segment $e$.
1: if $e$ is contained completely inside of some subarea(s) with rank at least $k$ then
2:     Assume $e$ has been divided by some KNP Voronoi edges into a bunch of $g$ partial segments $\{e_1, e_2, \cdots, e_q\}$.
3:     for Each partial segment $e_i : 1 \le i \le q$ do
4:         If $e_i$ is contained in KNP Voronoi cell $C_j$, find the point $a$ on $e_i$ which is furtherest away from the owner (sensor node $u$) of $C_j$. See Figure 4 for illustration. Let the $k$-support of $e_i$ be equal to $|ua|$.
5:     end for
6:     Set the $k$-support of $e$ to be equal to $\max_{i=1}^{q}$ $k$-support of $e_i$;
7: else
8:     Set the $k$-support of $e$ to be $\infty$;
9: end if
```

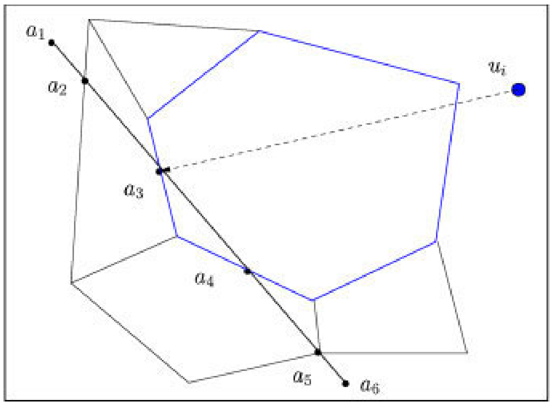

Figure 4. An example to compute k-support for road segment a_1a_6. Road segment a_1a_6 has been divided into five partial paths by KNP Voronoi edges, a_1a_2,a_2a_3,a_3a_4,a_4a_5,a_5a_6. Blue sensor node u_i is the owner of the KNP Voronoi cell with blue KNP Voronoi edges. The k-support of segment a_3a_4 is equal to the Euclidean distance of the dash line since in this case, a_3 is the point on segment a_3a_4 which is furtherest away from u_i.

6. CONCLUSION

In this paper, we studied several questions about k-coverage problem in WSNs. First, we studied how to deploy sensors on a given 2-dimensional area with a road map such that all road segments on the map are k-covered with minimum number of sensors. Then we studied how to find a k-cover path and an optimal k-support path connecting a given pair of source and destination points. For the first 'sensor deployment' problem, when sensors can only be deployed in a set of discrete locations, we give the $(6 + \epsilon)$-approximation algorithm to find the minimum number of wireless sensor nodes needed to make all road segments (on the given map) 1-covered. Following this, we present the $O(k)$ approximation algorithm when all road segments are required to be k-covered; when sensors can be deployed in arbitrary locations, we present an algorithm with the approximation ratio $24 + \epsilon$ for $k = 1$ and $O(k)$ for a general k. For the second 'path query' problem, we present two efficient algorithms which can find the k-covered path and the k-support path connecting any given source/destination pair of points on the road map when such paths exist.

There are still many open questions about k-coverage problem. For example, for deploying sensors to make all paths k-covered, whether there exists an algorithm with an $O(\log k)$-approximation ratio or even a constant approximation ratio. Also, for deploying sensors, we may want to optimize some other objectives, such as the total power or maximum power of all sensors.

APPENDIX A

Proof of Lemma 2. We first prove that covering P is necessary for covering all segments L. This is obvious since any point on a 1-covered subsegment is 1-covered when L is 1-covered.

Next, we prove that covering P is sufficient for covering all segments L. Recall that the resultant subsegment set L is obtained by drawing a circle centered at each feasible location (totally m locations). This means that each subsegment is either entirely covered by a sensor node or totally outside of the sensing range of a sensor node. Let us consider one subsegment l with middle point l_{mp}. See Figure 5 for illustration. Clearly, l cannot have any intersection (except the two end points) with any circle which is the sensing boundary of some sensor. Otherwise, l will be further divided into smaller subsegments such that l cannot be in the subsegment set. Hence, if the middle point l_{mp} of l is covered by any sensor, the entire l will be covered by this sensor as well. This finishes the proof. ■

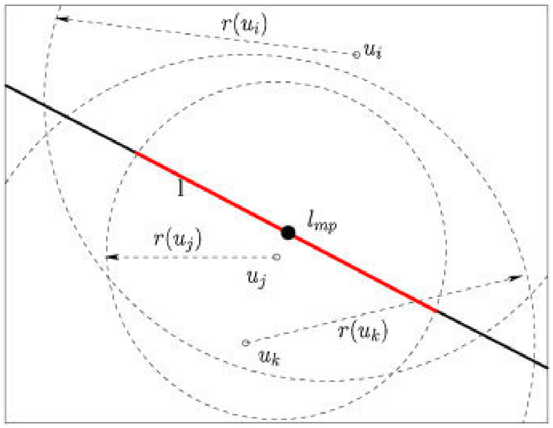

Figure 5. Red curve denotes the subsegment l. l cannot have any intersection with any circle. White node denotes the sensor nodes and the dash circles are the corresponding sensing boundaries. In this example, different sensors may have different sensing range.

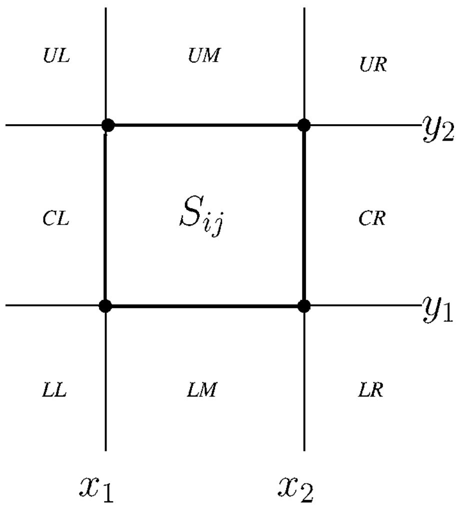

Figure 6. Partition of outside of a $\mu \times \mu$ square into eight regions.

APPENDIX B

SOLVING DUDC IN A BLOCK

In this subsection, we present our algorithm for DUDC in a $t\mu \times t\mu$ block B. We begin with some terms and notations.

Since block B consists of t^2 squares of size $\mu \times \mu$, we denote them as $S_{ij}, i, j \in [t]$. S_{ij} is the square in the ith order from left to right and in the jth order from up to down. All squares S_{kj} together form a horizontal strip (note as s_k^x, $j \in [t]$). Thus, block B contains t horizontal strips, $s_1^x, \cdots, s_m^x$. Similarly, block B contains t vertical strips, $s_1^y, \cdots, s_m^y$, where s_k^y is formed by combination of all squares S_{ik}, $i \in [t]$. Let D_{ij} and P_{ij} denote the set of disks and points lying in S_{ij}, respectively.

For each square S_{ij}, we divide its outside into eight regions $UL, UM, UR, CL, CR, LL, LM, LR$ as shown in Figure 6. Denote by $Left = UL \cup CL \cup LL$, $Right = UR \cup CR \cup LR$, $Up = UL \cup UM \cup UR$, $Down = LL \cup LM \cup LR$. Assume the four lines forming S_{ij} are $x = x_1, x = x_2, y = y_1, y = y_2$.

Then we briefly describe the idea for solving DUDC in a block.

(1) Guessing the covering pattern. Assume the optimum solution is OPT. For each square S_{ij}, we have:
   - $d \in OPT \cap D_{ij} \neq \emptyset$. Since the disk radius is 1 and the diameter of every square is 1, any disk d from $OPT \cap D_{ij}$ can cover S_{ij} entirely. Thus d can cover all points P_{ij}.
   - $OPT \cap D_{ij} = \emptyset$. In this case, P_{ij} are covered by disks out of the square S_{ij}. By Lemma 12, we can use up to four points to separate points in P_{ij} into two groups, one group can be covered by disks only from the Up and Down region of the square, and the other can be covered by disks only from the Left and Right region of the square S_{ij}.

   Thus, we can guess the covering pattern of OPT for each square S_{ij} by enumeration of all possibilities.

(2) Solving DUDC over strips. Once we guess a pattern, we can decompose the problem into problem in strips. We solve DUDC for t horizontal strips s_j^x. Similarly, we solve DUDC for t vertical strips s_j^y. We combine the $2m$ solutions and use $OPT \cap D_{ij}$ as the solution for this pattern. We then output the minimum solution over all possible enumerating patterns.

Lemma 11 ([17]). *Suppose $p \in P_{ij}$ is a point in S_{ij} which can be covered by a disk $d \in LM$. We draw two lines p_l and p_r, which intersect $y = y_1$ by angle $\pi/4$ and $3\pi/4$. Then the shadow P_{LM} surrounded by $x = x_1, x = x_2, y = y_1$, p_l and p_r can also be covered by d. Similar results can be hold for shadow P_{UM}, P_{CL} and P_{CR}, which can be defined with a rotation.*

Then, we give the definition of sandglass and a lemma which can be used to separate P_{ij} into two groups, with one can be covered by disks from $Up \cup Down$ and the other by disks from $Left \cup Right$.

Definition 6 ([17] Sandglass). *If D is a disk set covering P_{ij} and $D \cap S_{ij} = \emptyset$, then there must exist a subset $P_M \subset P_{ij}$ which can only be covered by disks from UM and LM (we can set $P_M = \emptyset$. if there is no such points). Select $P_{LM} \subset P_M$, the disks that can be covered by disks from LM, draw p_l and p_r line for each $p \in P_{LM}$. Select the leftmost p_l and rightmost p_r and form a shadow. Symmetrically, choose P_{UM} and form a shadow. The union of the two shadows form a sandglass region $Sand_{ij}$ of S_{ij}.*

Lemma 12 ([17]). *Suppose D is a disk set covering P_{ij}, and $Sand_{ij}$ are chosen as in Definition 6. Then any points in $Sand_{ij}$ can be covered by disks only from neighbor region $Up \cup Down$, and disks from $S_{ij} \setminus Sand_{ij}$ can be covered by disks only from neighbor region $Left \cup Right$.*

Then, we state Algorithm 5 for solving DUDC in a block similar to Reference [17].

Algorithm 5 DUDC in a block.

```
Input: a set of points $\mathcal{P}$ in the block and $\mathcal{D}$ covering $\mathcal{P}$.
Output: DUDC in the block.
1: For each $S_{ij}$, choose its sandglass or select a disk $u \in D_{ij}$ inside the square.
2: If for a square $S_{ij}$, a disk $u$ is chosen, then remove all points in $P_{ij}$. We also remove all other points outside of square $S_{ij}$ covered by $u$.
3: For each horizontal strip $s_i^x$, calculate an optimum DUDC for the remaining points in the sandglass of $s_i^x$.
4: For each vertical strip $s_j^y$, calculate optimum DUDC for the union of points in the sandglass of $s_j^y$.
```

Lemma 13. *Algorithm 5 can find a 6-approximation solution for DUDC in a $t\mu \times t\mu$ block.*

Proof. According to the definition of OPT, for any square in the block, it is either covered by a disk in this square, or covered by some disks outside of it. So during the enumeration process in Algorithm 5, once the covering pattern is guessed correctly, for any disk that is used in the pattern to cover the square containing this point,it is selected and then deleted by the algorithm in step 1, hence is only used once.

Consider the second case when we calculating DUDC for t horizontal strips, it is used at most three times. For the horizontal strips, the analysis is the same. By adding the horizontal and vertical strips up, for any disk, it could be counted at most six times totally.

The two solutions together have size no more than $6 \cdot OPT$, so Algorithm 5 gives a solution with size no more than $6 \cdot OPT$ when it guesses the pattern correctly. Since the algorithm enumerates all possible covering patterns, and takes the minimum solution, the lemma holds. ■

REFERENCES

1. Mehta D, Lopez M, Lin L. Optimal coverage paths in *ad-hoc* sensor networks. In *IEEE International Conference on Communications, ICC'03*, 2003; **1**.
2. Fang C, Low C. Redundant coverage in wireless sensor networks. In *IEEE International Conference on Communications, ICC'07*, 2007; 3535–3540.
3. Tang S, Mao X, Li X. Optimal k-support coverage paths in wireless sensor networks. In *Proceedings of the 2009 IEEE International Conference on Pervasive Computing and Communications, Vol. 00*, 2009.
4. Alt H, Arkin E, Bronnimann H, *et al*. Minimum-cost coverage of point sets by disks. In *Proceedings of the twenty-second annual symposium on Computational geometry*, 2006; ACM, 458.
5. Carmi P, Katz M, Lev-Tov N. Covering points by unit disks of fixed location. *Algorithms and Computation*, **4835**: 644–655.
6. Meguerdichian S, Koushanfar F, Potkonjak M, Srivastava M. Coverage problems in wireless *ad hoc* sensor networks. In *IEEE INFOCOM*, 2001; **3**: 1380–1387.
7. De Berg M, Cheong O, Van Kreveld M, Overmars M. *Computational Geometry: Algorithms and Applications*. Springer-Verlag New York Inc: New York, 2008.
8. O'Rourke J. *Computational Geometry in C*. Cambridge University Press: New York, NY 10013, USA. 1998.
9. Li X, Wan P, Frieder O. Coverage in wireless *ad hoc* sensor networks. *IEEE Transactions on Computers* 2003; **52**(6): 753–763.
10. Huang C, Tseng Y. The coverage problem in a wireless sensor network. *Mobile Networks and Applications* 2005; **10**(4): 519–528.
11. Huang C, Tseng Y, Lo L. The coverage problem in three-dimensional wireless sensor networks. In *IEEE Global Telecommunications Conference, GLOBECOM'04*, 2004; **5**.
12. Zhou Z, Das S, Gupta H. Connected k-coverage problem in sensor networks. In *Proceedings of 13th International Conference on Computer Communications and Networks, ICCCN*, 2004; 373–378.
13. Xing G, Wang X, Zhang Y, Lu C, Pless R, Gill C. Integrated coverage and connectivity configuration for energy conservation in sensor networks. *ACM Transactions on Sensor Networks (TOSN)* 2005; **1**(1): 72.
14. Kumar S, Lai T, Balogh J. On k- coverage in a mostly sleeping sensor network. *Wireless Networks* 2008; **14**(3): 277–294.
15. Wan P, Yi C. Coverage by randomly deployed wireless sensor networks. *IEEE/ACM Transactions on Networking (TON)*, SI, 2006; **14**: 2669.
16. Hochbaum D, Maass W. Approximation schemes for covering and packing problems in image processing and VLSI. *Journal of the ACM (JACM)* 1985; **32**(1): 136.
17. Huang Y, Gao X, Zhang Z, Wu W. A better constant-factor approximation for weighted dominating set in unit disk graph. *Journal of Combinatorial Optimization* 2009; **18**(2): 179–194.
18. Okabe A, Boots B, Sugihara K, Chiu S. Spatial tessellations: concepts and applications of Voronoi diagrams (POD). *Européen des systèmes automatisé* 2009; **43**: 672.
19. Lee D. On *k*-nearest neighbor Voronoi diagrams in the plane. *IEEE Transactions on Computers* 1982; **100**(31): 478–487.
20. Skyum S. A Sweepline Algorithm for Generalized Delaunay Triangulations. *Technical Report DAIMI PB-373*, CS Dept. Aarhus University.

AUTHORS' BIOGRAPHIES

Xufei Mao is an Assistant Professor in Computer Science Department, Beijing University of Posts and Telecommunications. He holds Ph.D. (2010) degree in Computer Science from Illinois Institute of Technology. He received M.S. (2003) and Bachelor degree from Northeastern University and Shenyang University of Technology, respectively. His research interests include design and analysis of algorithms concerning wireless networks, network security, etc. Topics include Coverage problems in sensor network, Top-k Query, Capacity (Throughput) study, Channel Assignment, Link Scheduling, and Tiny OS programming etc.

Xiaohua Xu is a Computer Science Ph.D. student at Illinois Institute of Technology. He received B.S. from ChuKochen Honors College of Zhejiang University, P.R.China, 2007. His research interests include algorithm design and analysis, optimization in mesh network, energy efficiency and security in wireless network. He is an IEEE student member.

ShaoJie Tang has been a Ph.D. student of Computer Science Department at the Illinois Institute of Technology since 2006. He received B.S. degree in Radio Engineering from Southeast University, China, in 2006. His current research interests include algorithm design and analysis for wireless *ad hoc* networks, wireless sensor networks, and online social networks.

Xiang-Yang Li (M'99, SM'08) has been an Associate Professor (since 2006) and Assistant Professor (from 2000 to 2006) of Computer Science at the Illinois Institute of Technology. He received M.S. (2000) and Ph.D. (2001) degree at Department of Computer Science from University of Illinois at Urbana-Champaign. He received the Bachelor degree at Department of Computer Science and Bachelor degree at Department of Business Management from Tsinghua University, China, both in 1995. His research interests span wireless sensor networks, game theory, computational geometry, and cryptography and network security. He served as a co-chair of ACM FOWANC 2008 workshop, a co-chair of AAIM 2007 conference, a TPC co-chair of WTASA 2007, and TPC members of a number of conferences such as ACM MobiCom, ACM MobiHoc, IEEE INFOCOM, IEEE ICDCS. He serves as an Editor of 'IEEE Transitional on Parallel and Distributed Systems (TPDS)', from 2010; an Editor of 'Networks: An International Journal' from 2009, and Advisory Board of '*Ad Hoc* & Sensor Wireless Networks: An International Journal', from 2005. He was a guest editor of special issues for 'ACM Mobile Networks and Applications', 'IEEE Journal on Selected Areas in Communications', and several other journals. He published a monograph 'Wireless *Ad Hoc* and Sensor Networks: Theory and Applications', in June 2008 by Cambridge University Press. He also co-edited the following books 'Encyclopedia of Algorithms', by Springer publisher, as the area editor for mobile computing.