Best and Worst-Case Coverage Problems for Arbitrary Paths in Wireless Sensor Networks

Chunseok Lee*, Donghoon Shin*, Sang Won Bae† and Sunghee Choi*
*Department of Computer Science, Korea Advanced Institute of Science and Technology, Daejeon, Korea
†Department of Computer Science, Kyonggi University, Suwon, Korea
Email: *{stonecold, dhshin, sunghee}@tclab.kaist.ac.kr, †swbae@kgu.ac.kr

***Abstract*—The best-case and the worst-case coverage were proposed originally for a single source and destination pair in a sensor network. In this paper, we propose a new coverage measure of the sensor network considering arbitrary paths. Surprisingly, this new measure captures both the best-case and the worst-case coverage of the sensor network simultaneously, enabling us to evaluate the given network in a global viewpoint. Accordingly, we pose the evaluation and the deployment problems; the former is to evaluate the new coverage measure of a given sensor network, and the latter is to find an optimal placement of k additional sensor nodes to improve the coverage for a given positive integer k. We present several algorithms solving the problems that are either centralized or localized with theoretical proofs and simulation results, showing that our algorithms are efficient and easy to implement in practice while the quality of outputs is guaranteed by formal proofs.**

Our algorithms are based on an interesting relation between our new coverage measure and a certain quantity of a point set, called the bottleneck, which has been relatively well studied in other disciplines. In doing so, we prove that a maximal support path can always be found in the minimum spanning tree; this is another contribution of ours.

I. Introduction

Thanks to improvement of wireless sensor technology, lots of practical applications of wireless sensor networks continue to emerge [1]–[3]. Especially, wireless sensor networks provide a promising infrastructure for surveillance and monitoring applications such as fire monitoring [4], [5] and intruder detection [6], [7]. By these applications, the coverage problem has been posed as one of the fundamental problems in wireless sensor networks [8]–[11]. Coverage characterizes the ability or the quality of a sensor network in monitoring its sensing field.

In this work, we deal with the *path-based coverage* of wireless sensor networks. Meguerdichian et al. [12] introduced two path-based coverage problems: the *best*-case and the *worst*-case coverage problems, dealing with *maximal support paths* and *maximal breach paths*, respectively.

The *maximal support path* between a source and a destination is a path such that the maximum distance from every point on it to the sensors is minimized. Suppose that several robots are assigned to a job of monitoring an area. While they are traveling to collect information in the area, they can communicate with the wireless sensor nodes to transmit information or to be assigned a new job. In this case, one would expect the robots to be as close as possible to the sensor nodes while moving in the field by several reasons; for example, saving power consumption of sensors or watching the robots by the sensors. In this situation, the maximal support path is the best path for the robots. Meguerdichian et al. [12] presented a polynomial time algorithm finding a maximal support path connecting two given points by exploiting the Delaunay triangulation. The problem of deploying additional sensors to improve a maximal support path was also considered in [12]–[15].

The *maximal breach path* is defined as a path between a source and a destination which maximizes the minimum distance from every point on it to the sensors. For example, imagine that an intruder of the network would want to traverse the sensor field while being as far as possible to the sensors to avoid detection. In this case, the maximal breach path is the worst path for the detector. Meguerdichian et al. [12] showed that the maximal breach path can be found in the Voronoi diagram. The deployment problem to improve the maximal breach path was also considered by Gau et al. [16] and Duttagupta et al. [17].

However, previous works mostly considered only the case of a given pair of source and destination, and thus would not provide a global look at the given sensor network, just giving a narrow view restricted to the specified path. In the above example, one might manage several robots and the robots might have different source-destination pairs individually; even the destinations of the robots could be currently undecided but specified afterwards. Moreover, for the worst-case coverage, it is unrealistic to assume that we always know the source and destination points of an intruder: if we knew, we could just place the sensors to the known source and destination points to minimize breach. This motivates us to extend the concept of the best-case and the worst-case coverage to consider arbitrary paths in the network. In this paper, we introduce new coverage measures, called the *support* and *breach* of *a sensor network* considering *arbitrary* maximal support paths and maximal breach paths, respectively,

978-1-4244-7489-9/10/$26.00 ©2010 IEEE

depending on the distribution of the sensor nodes, not on a specific pair of points or a path connecting them.

We consider two problems with respect to the support and the breach of the sensor network: the evaluation problem is to compute the value of the support/breach of a given sensor network, and the deployment problem is to find optimal locations of k additional sensors to improve the support/breach the most for a given integer k. A remarkable result is that for arbitrary paths the two measures for the best-case and the worst-case coverage coincide. Thus, unlike previous results which require different schemes for the best-case or worst-case coverage, for our new coverage measure, a unified algorithm suffices for both.

We show that our evaluation and deployment problems surprisingly correlate with the *bottleneck spanning tree* and the *bottleneck Steiner tree*, which have been well studied in Operations Research and Computational Geometry [18]–[21]. Consequently, the support/breach of a given network of n sensors can be computed in $O(n \log n)$ time in a centralized manner or in $O(n \log n)$ communication cost bits in a localized manner, and the deployment problem is shown to be NP-hard even to approximate within approximation ratio $\sqrt{2}$. In doing so, we also prove that a maximal support path can always be found in the minimum spanning tree of the given network. This is of independent interest, providing us a much easier way to handle and maintain maximal support paths in a localized environment. Moreover, this improves the result of Li et al. [22] which states that a maximal support path can be found in the relative neighborhood graph, a supergraph of the minimum spanning tree.

In summary, the main contributions of our paper are as follows. First, we define a new coverage measure for sensor networks and show that it captures both the best-case and worst-case coverages for arbitrary paths. Second, we show how to obtain the value of this new coverage measure of a given network. This can be done efficiently either using a centralized algorithm or a localized algorithm. Third, we show that this new coverage measure is closely related to the bottleneck spanning tree, and from this observation give a new algorithm for deploying additional sensors to improve the value of the coverage measure with a guaranteed performance.

The rest of the paper is organized as follows: In Section II, we briefly introduce some preliminaries including geometric notations and the bottleneck spanning/Steiner tree. In Section III, we newly formulate path coverage problems with respect to arbitrary paths. In Section IV, we formally prove that a maximal support path can be found in the minimum spanning tree of the given sensor network. Based on this new discovery, we prove that our problems can be transformed to finding a bottleneck spanning/Steiner tree and present algorithms solving the evaluation and the deployment problems with theoretical guarantees. Some simulation results are given in Section V and finally we conclude the paper with discussion in Section VI.

II. Preliminaries

A. Knowledge from Computational Geometry

Coverage problems in wireless sensor networks are mostly discussed in geometric models; thus, most known algorithms are based on common knowledge from Computational Geometry. Here, we briefly introduce several geometric structures that are frequently referred.

The *Delaunay triangulation* $DT(S)$ of the set S of points in the plane $\mathbb{R}^2$ is a triangulation such that the circumcircle of each triangle contains no point of S in its interior. The Delaunay triangulation is also well known as a dual structure of the *Voronoi diagram*. The Voronoi diagram $VD(S)$ of the points S is a subdivision of the plane into *Voronoi regions*. The Voronoi region of $p \in S$ is the set of points that are closer to p than the others in S. These two structures $DT(S)$ and $VD(S)$ have been intensively studied in Computational Geometry and are widely used in many areas. For more details about the Voronoi diagram and the Delaunay triangulation, we refer to a survey paper by Aurenhammer and Klein [23].

Though the Delaunay triangulation $DT(S)$ well spans all the nodes in S, it is known that $DT(S)$ is hard to construct in a localized way [22]. Thus, several subgraphs were proposed. The *relative neighborhood graph* $RNG(S)$ consists of all edges uv such that the intersection of two circles centered at u and at v with radius $\|uv\|$ contains no other points in S in its interior [24], [25]. The *(Euclidean) minimum spanning tree* $MST(S)$ of a point set S is a tree spanning all points in S in the plane that minimizes the sum of the (Euclidean) lengths of its edges. Following their definitions, it is not difficult to show the following relation

$$MST(S) \subseteq RNG(S) \subseteq DT(S);$$

that is, $MST(S)$ is a subgraph of $RNG(S)$ and $RNG(S)$ is a subgraph of $DT(S)$ [26]. The subgraphs $MST(S)$ and $RNG(S)$ can be constructed efficiently in $O(n \log n)$ time [24] in a centralized manner or in communication cost $O(n \log n)$ bits in a localized manner [22], [27].

B. Bottleneck Spanning/Steiner Tree

A *bottleneck spanning tree* of S is a tree spanning all points in S that minimizes the *maximum* of the lengths of its edges. We call the length of a longest edge in a bottleneck spanning tree of S the *bottleneck of the set* S, denoted by $btn(S)$. Notice that the bottleneck $btn(S)$ of S is only dependent on S, not on how to construct a spanning tree. It is easy to see that there may be many bottleneck spanning trees for a set S and the minimum

spanning tree $MST(S)$ is also a bottleneck spanning tree [21].

A *(Euclidean) bottleneck Steiner tree* of S with k Steiner points is a tree spanning S and k additional nodes that minimizes its longest edge by selecting optimal locations of k nodes. The (Euclidean) bottleneck Steiner tree problem is described as follows: given n points in the plane and a positive integer k, find a bottleneck Steiner tree with at most k Steiner points. Unlike the classical Steiner tree problem where the total length of the Steiner tree is minimized, this problem asks a Steiner tree where the maximum of the edge lengths is minimized and the Steiner points in the resulting tree can be chosen in the whole plane $\mathbb{R}^2$. These problems, and their variations, have known applications in VLSI layout [28], multi-facility location, and wireless communication network design [29].

The bottleneck Steiner tree problem is known to be NP-hard to approximate within ratio $\sqrt{2}$ [29]. This means that it is impossible to get an approximate solution within ratio $\sqrt{2}$ in polynomial time unless P=NP. The best known upper bound on approximation ratio is 1.866 by Wang and Li [30]. There has been some effort on devising an exact algorithm for finding the locations of k Steiner points [19], [31]. To the best knowledge of the authors, there is no known exact algorithm with exponential running time yet. Very recently, Bae et al. [21] presented an $O(n \log n)$ time exact algorithm when $k = 1$ and proved this is best possible.

III. Problem Formulation

We assume that the sensor nodes are given as a set S of n points in a two-dimensional domain Ω, which is a bounded and convex subset of $\mathbb{R}^2$. This guarantees that the convex hull of the set S of sensors is completely contained in the domain Ω.

A. Previous Works for Best-case/Worst-case Coverage

To formulate our problem, we need to introduce results from previous works. We first define the distance $dist(x, A)$ between a point $x \in \Omega$ and a set $A \subset \Omega$ of points as follows:

$$dist(x, A) = \min_{y \in A} \|xy\|,$$

where $\|xy\|$ denotes the Euclidean distance between x and y. Then, we define the *support* of a point set by another set as follows.

Definition 1. *The* support *of a point set $A \subset \Omega$ by another set $B \subset \Omega$ is*

$$support(A, B) = \max_{x \in A} dist(x, B).$$

Since any path Π contained in Ω can be viewed as a set of points in Ω, now we can discuss the distance between a path Π and the set S of sensors with respect to the support defined above. For a path Π, the support $support(\Pi, S)$ of Π by the sensors S is also called the *support* of Π [14].

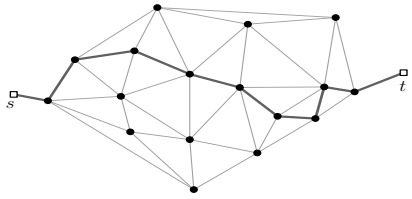

Fig. 1. A maximal support path (thick segments) from s to t can be found by using the Delaunay triangulation (thin gray segments) of the given set of sensors.

Definition 2. *For fixed $s, t \in \Omega$, a path Π connecting s and t is called a* maximal support path *in the sensor field if Π achieves the possible minimum support. That is,*

$$support(\Pi, S) = \min_{\text{any path } \Pi' \text{ between } s \text{ and } t} support(\Pi', S).$$

Also, this value is called the support distance *between s and t, denoted by $support(s, t)$.*

For maximal breach paths, we first define the breach between two sets of points as follows.

Definition 3. *The* breach *of a point set $A \subset \Omega$ by another set $B \subset \Omega$ is*

$$breach(A, B) = \min_{x \in A} dist(x, B).$$

In other words, $breach(A, B)$ represents the minimum Euclidean distance between two sets of points. Then, for a given source and destination pair, the maximal breach path is defined as a path with the maximum breach.

Definition 4. *For fixed $s, t \in \Omega$, a path Π connecting s and t is called a* maximal breach path *in the sensor field if Π achieves the possible maximum breach. That is,*

$$breach(\Pi, S) = \max_{\text{any path } \Pi' \text{ between } s \text{ and } t} breach(\Pi', S).$$

The *best-case/worst-case coverage problem* asks to find a maximal support/breach path connecting two given points in the domain, respectively. Meguerdichian et al. [12] proposed efficient algorithms to find a maximal support/breach path by exploiting the Delaunay triangulation and the Voronoi diagram of the given sensors, respectively. Figs. 1 and 2 illustrate a maximal support path and a maximal breach path, which goes through the Delaunay triangulation and the Voronoi diagram of the sensors S, respectively. Li et al. [22] gave a formal proof of the correctness of the algorithm by Meguerdichian et al. and further showed that the relative neighborhood graph is enough to find a maximal support path.

Theorem 1 (Li et al. [22])**.** *Given a set S of sensors in the domain Ω and any pair of two points $s, t \in \Omega$, there*

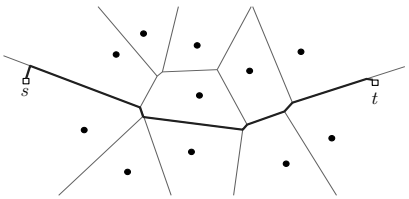

Fig. 2. A maximal breach path (thick segments) from s to t can be found by using the Voronoi diagram (thin gray segments) of the given set of sensors.

is a maximal support path Π *from* s *to* t *such that (1)* Π *connects* s *to its closest sensor* $u_s \in S$ *and* t *to its closest sensor* $u_t \in S$ *and (2) the sub-path of* Π *from* u_s *to* u_t *goes along only edges of the relative neighborhood graph of* S.

The *coverage deployment problem* asks to find the deployment locations for additional k sensor nodes so that the support/breach between two given points is improved the most. For the maximal support paths, Meguerdichian et al. [12] mentioned a simple heuristic method: for a given pair of points, it finds a maximal support path and places sensors on the longest segment of the path. Recently, Hou et al. [15] proposed an optimal placement algorithm to improve the maximal support paths using Dijkstra's shortest path algorithm.

For the maximal breach paths, Gau et al. [16] proposed two approaches to improve the worst-case coverage by adding k additional sensors. One is by exploiting the dual relationship between the worst-case and the best-case coverage and the other is using a genetic algorithm. But, their algorithms do not guarantee an optimal placement. Duttagupta et al. [17] proposed a measure called the average maximal breach considering maximal breach paths connecting Voronoi vertices. But, their average maximal breach does not represent the maximum breach of the sensor network.

B. New Measure for Best-case/Worst-case Coverage considering Arbitrary Paths

Here, we extend the support and breach of a given pair of source and destination to consider any *arbitrary* path by defining two new measures for the quality of service of a given sensor network, called the *support* and *breach* of the *sensor network*.

Definition 5. *The* support *of the set* S *of sensors in the domain* Ω*, denoted by* $support(S)$*, is the maximum value among the support distances of all pairs of points belonging to* S*. That is,*

$$support(S) = \max_{u,v \in S} support(u, v).$$

Since a maximal support path connecting $s, t \in \Omega$ can be obtained by finding the closest sensors $u \in S$ from s and $v \in S$ from t and a maximal support path between u and v by Theorem 1, $support(S)$ well represents the coverage of the network S, giving a guaranteed quantity of support distances between any pair of two points. Unlike the coverage problems studied previously [12], [14], [15], [22] where only a single path is considered, this new quantity gives us a way of viewing the given network in a more global viewpoint by considering arbitrary maximal support paths.

The breach of the set S of sensors is defined similarly, but, here we need to define all possible paths more carefully since we cannot use the set of sensors as candidates for source and destination points. Thus, for the maximal breach paths, we consider all paths such that sensors lie on both sides of the paths. In other words, we do not consider the paths so that all the sensors in S lie completely on one side of the path. This is a reasonable assumption since we are interested in detecting an intruder that traverses the sensing field. We say that a path *crosses the sensing field* Ω if the path has at least one sensor in both sides.

Definition 6. *The* breach *of the set* S *of sensors in the domain* Ω*, denoted by* $breach(S)$*, is the maximum breach among all possible paths crossing* Ω*:*

$$breach(S) = \max_{\text{any path } \Pi' \text{ crossing } \Omega} breach(\Pi', S).$$

In the worst-case coverage, we are interested in finding the most vulnerable area of the sensor network from an intruder. Since $breach(S)$ represents the maximal breach among paths that cross the sensing field, it gives the measure of how vulnerable the network is, considering arbitrary paths.

This paper considers the evaluation and deployment problems with respect to $support(S)$ or $breach(S)$ of the given sensor network: The *evaluation problem* is to compute the values $support(S)$ or $breach(S)$. The *deployment problem*, for given integer $k > 0$, is to find a set Q of k locations for additional sensors to minimize $support(S \cup Q)$ or $breach(S \cup Q)$, the support or breach of $S \cup Q$, respectively.

IV. Algorithms and Theoretical Guarantees

A. Minimum Spanning Trees for Maximal Support Paths

To compute the support and breach of the sensor network, we use the minimum spanning tree. Li et al. [22] showed that a maximal support path between any two points in Ω can be found in the relative neighborhood graph of the sensors S, as stated in Theorem 1. Here, we first show that the minimum spanning tree which is a subset of the relative neighborhood graph is sufficient to find maximal support paths.

Theorem 2. *Given a set* S *of sensors in the domain* Ω *and any pair of two points* $s, t \in \Omega$*, there is a maximal support path* Π *from* s *to* t *such that (1)* Π *connects* s *to*

its closest sensor $u_s \in S$ and t to its closest sensor $u_t \in S$ and (2) the sub-path of Π from u_s to u_t goes along only edges of the minimum spanning tree $MST(S)$.

Proof: From Theorem 1 by Li et al. [22], we know that there is a maximal support path that connects s to u_s and t to u_t. Thus, we can focus only on the case where two endpoints are sensors u_s and u_t in S.

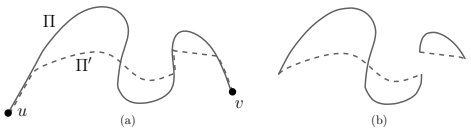

Fig. 3. Illustration to the proof of Theorem 2. (a) Two paths Π and Π' between u and v; (b) The set $X = (\Pi \cup \Pi') \setminus (\Pi \cap \Pi')$ consists of several cycles.

The proof is done by contradiction. Suppose that the theorem is false and thus there exist $u, v \in S$ such that the path Π between u and v in $MST(S)$ has strictly larger coverage distance than another path Π'. Consider the symmetric difference X of two paths Π and Π', that is, $X = (\Pi \cup \Pi') \setminus (\Pi \cap \Pi')$. Since both are paths connecting u and v, the set X is a union of several cycles. See Fig. 3. Furthermore, X is not empty since $\Pi \neq \Pi'$ and thus consists of at least one cycle. Take one of those cycles that contains the longest edge in Π, say C. (Since $support(\Pi, S) > support(\Pi', S)$, the longest edge of Π must appear in X.) Note that C can be seen as a union of two parts, $C \cap \Pi$ and $C \cap \Pi'$. By our selection of C, we also have $support(C \cap \Pi, S) = support(\Pi, S) > support(\Pi', S) \geq support(C \cap \Pi', S)$. Letting e be the longest edge in Π, we get $\frac{1}{2}\|e\| = support(e, S) = support(C \cap \Pi, S)$. On the other hand, $support(C \cap \Pi', S) = \max_{e' \in C \cap \Pi'} support(e', S) = \max_{e' \in C \cap \Pi'} \frac{1}{2}\|e'\|$. Therefore, we have $\|e\| = 2support(C \cap \Pi, S) > 2support(C \cap \Pi', S) = \max_{e' \in C \cap \Pi'} \|e'\|$, implying that e is the longest edge in cycle C but e is contained in $MST(S)$. This is a contradiction to so-called Red Rule for the minimum spanning tree; in any cycle of a given graph, the edge with maximum length must be excluded out from the minimum spanning tree of the graph [32]. Hence, we conclude that there does not exist such a path Π', completing the proof. ∎

The minimum spanning tree $MST(S)$ can be built locally in optimal communication cost $O(n \log n)$ bits and in the same time bound [27] since it is known that its super graph $RNG(S)$ can be computed in communication cost $O(n \log n)$ bits [22]. Moreover, once $MST(S)$ is maintained, a maximal support path can be easily retrieved since there is a unique path in it between any pair of sensors in S.

In a theoretical point of view, Theorem 2 further implies that $MST(S)$ is the minimal graph in which one can find the maximal support paths; since $MST(S)$ is a spanning tree of S, any graph with less edges than $MST(S)$ cannot be exploited as a topology of the sensor network.

B. The Evaluation Problem

The evaluation problem is to compute the support $support(S)$ and the breach $breach(S)$ of the given sensor network. Theorem 2 naturally implies an important relation between the bottleneck $btn(S)$ of S and the support $support(S)$ of S.

Using the dual relationship between the maximal support paths and the maximal breach paths, we can show that the breach $breach(S)$ of a set S of sensors is actually the same as the support $support(S)$ of S.

Lemma 1. *For any set S of n sensors distributed in the domain Ω, we have*

$$support(S) = breach(S) = \frac{1}{2} btn(S).$$

Proof: For any $u, v \in S$, let $\Pi(u, v)$ be the path between u and v in $MST(S)$. By Theorem 2, we know that $\Pi(u, v)$ is a maximal support path connecting u and v, and we have

$$support(\Pi(u, v), S) = \max_{e \in \Pi(u,v)} \frac{1}{2}\|e\| \leq \frac{1}{2} btn(S),$$

since $MST(S)$ is a bottleneck spanning tree of S. The equality is realized when we take u, v as the sensors incident the longest edge in $MST(S)$. Hence, we have $support(S) = \frac{1}{2} btn(S)$.

Let Π' be a maximal breach path of the given sensors. Then Π' satisfies that $breach(S) = breach(\Pi', S)$ and sensors are partitioned into two groups by the path. If any edge e is removed from $MST(S)$, two partitioned trees T_1, T_2 remain and its minimum distance is equal to $\|e\|$ by the property of the minimum spanning tree. Since Π' crosses Ω, it should pass through at least one edge of $MST(S)$. Also, Π' should passes through the middle of edges to maximize $breach$. Thus, we obtain $breach(S) = \frac{1}{2} btn(S)$ when Π' passes thourgh the longest edge of $MST(S)$. ∎

We obtain the following theorem from the above lemma.

Theorem 3. *Given a set S of n sensor nodes in the domain Ω, $support(S)$ and $breach(S)$ of S can be computed locally in $O(n \log n)$ communication cost bits and time.*

Proof: By Lemma 1, we are done by computing a bottleneck spanning tree of S and traversing every edge of the tree to compute the bottleneck $btn(S)$ of S. Recall that the minimum spanning tree $MST(S)$ is a bottleneck spanning tree. In the two-dimensional space Ω, an optimal construction of $MST(S)$ in a localized manner is well known by Awerbuch [27], proving the claimed communication cost and time. ∎

C. The Deployment Problem

For a given positive integer k, the deployment problem asks a set Q of k points in the domain Ω which minimizes the $support(S \cup Q)$ or $breach(S \cup Q)$ of the sensor network including n given sensor nodes and k additional ones. Since the support and the breach of a given set of sensors are the same, one deployment algorithm can optimize both quantities at the same time.

Lemma 1 implies that our deployment problem is equivalent to the bottleneck Steiner tree problem, which finds k additional points Q minimizing the bottleneck $btn(S \cup Q)$, as described in Section II. Therefore, the hardness results of the bottleneck Steiner tree problem by Wang and Du [29] are inherited into our problem.

Theorem 4. *The deployment problem minimizing the support or breach of given n sensors plus k additional ones is equivalent to the Euclidean bottleneck Steiner tree problem with at most k Steiner points. Therefore, the problem is NP-hard even to approximate within approximation ratio $\sqrt{2}$.*

In other words, it is impossible to get a polynomial-time ratio-$\sqrt{2}$ approximation algorithm for our deployment problem unless P=NP.

On the other hand, several known algorithms for the bottleneck Steiner tree problem can be exploited. Wang and Li [30] presented an approximation algorithm with ratio 1.866. This gives the currently best known upper bound on approximation ratio but the algorithm is hard to implement and inefficient, taking $O(n^3k^3)$ time.

In this paper, we propose a new 2-approximation algorithm. To understand it, we first briefly describe two previous algorithms; an optimal exact algorithm for k=1 [21] and an approximation algorithm by Wang and Du [29].

1) Exact Algorithm for $k = 1$: Here, we introduce the exact algorithm for the deployment problem when $k = 1$, that is, when we want to deploy a single sensor node.

Let $e_1, \ldots, e_{n-1}$ be the edges of $MST(S)$ in the order that their lengths are not increasing. Since $MST(S)$ is a bottleneck spanning tree of S, we have $btn(S) = \|e_1\|$, the length of the longest edge in $MST(S)$. In order to have a strictly better bottleneck with a new node, we must be able to remove the longest edge e_1 by the additional sensor. Intuitively, the algorithm removes the longest edge in $MST(S)$ by adding a node but it does not always put a new node on the edge e_1. Sometimes, the newly added node with some new edges can replace the longest and the second longest edges e_1, e_2, simultaneously, or even the third e_3. Fig. 5(c) shows an example where a new node q and new edges incident to q replace e_1 and e_2, simultaneously. But, it is shown in [21] that it is impossible to remove all of $e_1, \ldots, e_5$ simultaneously by one newly added node.

In order to minimize $btn(S \cup \{q\})$, it is obvious that

```
1:  Algorithm EXACT(S)
2:      Compute the minimum spanning tree MST(S)
3:      Sort the edges of MST(S) by their lengths
4:      Let e_1, ..., e_{n-1} be the edges in the order
5:      for c=1 to 4 do
6:          Remove e_1, ..., e_c from MST(S)
7:          Let T_1, ..., T_{c+1} be the c+1 resulting subtrees
8:          Color the points in T_i by color i for each i
9:          Find the smallest color spanning disk D of the T_i
10:         q_c ← the center of D
11:     end for
12:     q ← one minimizing btn(S ∪ {q_i}) among q_1, ..., q_4
13:     return q
14: end Algorithm
```

Fig. 4. Exact deployment algorithm when $k = 1$

effort to remove e_i without removing e_{i-1} is useless. Hence, we are left with only four cases; we could remove $e_1, \ldots, e_c$ by an optimal solution for $c = 1, \ldots, 4$. The main idea of the algorithm is to check all these four cases. Our algorithm does the following for $c = 1, \ldots, 4$: (1) Remove $e_1, \ldots, e_c$ from $MST(S)$ to get $c + 1$ subtrees $T_1, \ldots, T_{c+1}$, (2) find a smallest disk D_c containing at least one point from each subtree T_i, and then its center q_c is a candidate of our solution. (See Fig. 5.)

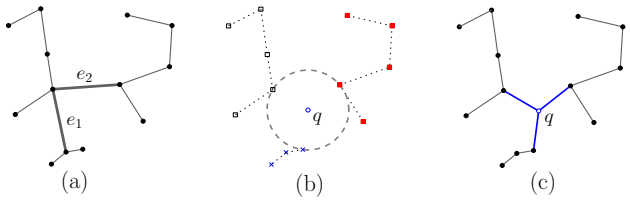

Fig. 5. Illustration to how Algorithm EXACT works. (a) Compute $MST(S)$. (b) Remove the longest and the second longest edges, e_1 and e_2 (when $c = 2$), color the points in each of the resulting subtrees, and compute the smallest color spanning disk (dashed gray circle). (c) Return the center q of the disk; this also shows an additional sensor node can remove e_1 and e_2, simultaneously.

Now, the problem is transformed to finding a smallest disk D_c containing at least one point in each subtree T_i. This can be tackled by algorithms computing the *smallest color spanning disk* [20], [33]; given points colored by one in a color set $C = \{1, \ldots, c\}$, the smallest color spanning disk is a smallest disk that contains at least one point for each color in C. The smallest color spanning disk has been studied in Computational Geometry with applications in multicommodity facility location, and there exists a simple algorithm that finds the disk in $O(cn \log n)$ time when we are given n total points in the plane with c colors [20].

The pseudo-code of the algorithm, named EXACT, is described in Fig. 4. Also, Fig. 5 illustrates how Algorithm EXACT works.

Lemma 2. *Algorithm* EXACT *correctly finds an optimal deployment for a single sensor node in $O(n \log n)$ time.*

1: **Algorithm** WANG-DU(S, k)
2: Compute the minimum spanning tree $MST(S)$
3: Set $k_e \leftarrow 0$ and $l_e \leftarrow \|e\|$ for edge e in $MST(S)$
4: Build the max-heap of (l_e, e) for all e in l_e values
5: Pop the maximum (l_{e_M}, e_M) from the max-heap
6: Set $k_{e_M} \leftarrow k_{e_M} + 1$ and $l_{e_M} \leftarrow \|e\|/(k_{e_M} + 1)$
7: Insert (l_{e_M}, e_M) into the max-heap
8: Repeat Steps 5–7 k times
9: For each edge e, locate k_e sensors evenly on e
10: **return** the positions of k added sensors
11: **end Algorithm**

Fig. 6. Modified version of the algorithm by Wang and Du for deploying $k \geq 1$ sensor nodes

Proof: The correctness of Algorithm EXACT for the bottleneck Steiner tree problem is proved in our companion paper [21]. By Theorem 4, an optimal location q minimizing the bottleneck also minimizes the support.

For the time complexity, computing $MST(S)$ and sorting its edges takes $O(n \log n)$ time and finding the smallest color spanning disk takes $O(cn \log n)$ time [33]. Fortunately, c runs from 1 to 4 only, so the total time complexity settles down to $O(n \log n)$. ■

2) Algorithm by Wang and Du: The algorithm by Wang and Du [29] finds k locations for additional sensors in a greedy way. It also starts with the minimum spanning tree $MST(S)$: for each edge $e = (u, v)$ in $MST(S)$, if we add k_e sensors on e, then the length of the longest edge between u and v has the minimum value $\|e\|/(k_e + 1)$, where $\|e\|$ is the Euclidean length of edge e. We denote $l_e = \|e\|/(k_e + 1)$ and initially set $k_e = 0$ and $l_e = \|e\|$ in the algorithm. The basic idea is to add one sensor on edge e of $MST(S)$ with the largest l_e value at a time. After adding a sensor on e, we update $k_e \leftarrow k_e + 1$ and l_e accordingly. We repeat this until k sensors are all added and at last fix their positions; for each edge e in $MST(S)$, locate newly added sensors evenly on e. The original version of this algorithm takes $O(kn + n \log n)$ time but we modify it to have a slightly better time bound by using a heap structure. This algorithm, named WANG-DU, is described as in Fig. 6, and Fig. 7 shows some examples of how Algorithm WANG-DU finds locations for additional sensor nodes.

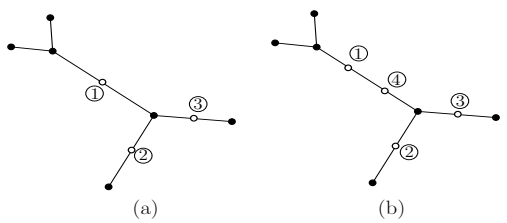

Fig. 7. Two examples of execution of Algorithm WANG-DU; The results of (a) WANG-DU(S, 3) and (b) WANG-DU(S, 4) for 6 given sensors S. The numbers in circle indicate the iteration of Algorithm WANG-DU when the node is added.

This simple algorithm guarantees the quality of its output theoretically.

Lemma 3. *Algorithm* WANG-DU *returns a set Q of locations for k additional sensors such that*

$$support(S \cup Q) \leq 2support(S \cup Q_{opt}),$$

where Q_{opt} is a set of optimal locations for k additional sensors. Its running time is at most $O((n + k) \log n)$.

Proof: Since Algorithm WANG-DU is a 2-approximation algorithm for the bottleneck Steiner tree problem [29], the same approximation ratio holds for the support or breach value. Steps 5–7 are executed k times. Since he max-heap can be constructed in $O(n \log n)$ time and each operation on it takes at most $O(\log n)$ time [34], the claimed time complexity $O((n + k) \log n)$ is shown. ■

3) NEW *Algorithm:* We propose a new deployment algorithm based on two algorithms, EXACT and WANG-DU. Each algorithm described above has its own limitation: Algorithm EXACT works only for $k = 1$ and Algorithm WANG-DU finds locations of additional sensor nodes only on the edges of the minimum spanning tree $MST(S)$ of S. Hence, by combining these two algorithms, we could find better solutions in practice. As aforementioned, Algorithm WANG-DU adds new nodes incrementally in a greedy way, while Algorithm EXACT finds the location of a new sensor node on an edge of $MST(S)$ when there is no better solution that is not on the edges of $MST(S)$; in that case EXACT(S) returns the same location as WANG-DU(S, 1). Thus, we basically execute Algorithm EXACT iteratively and if its output improves the bottleneck reasonably much, then we accept the output; otherwise, we simulate Algorithm WANG-DU. The new algorithm, named NEW, is described in Fig. 8.

Since Algorithm NEW finds the same location as WANG-DU or even better by EXACT at every iteration, it returns a solution at least as good as that of WANG-DU(S, k).

Lemma 4. *Let Q be the output of* NEW*(S, k) and Q' the output of* WANG-DU*(S, k) for any set S of sensors and any positive integer k. Then, we have*

$$support(S \cup Q) \leq support(S \cup Q').$$

The time complexity of Algorithm NEW *is at most $O(k(n+k) \log(n + k))$.*

Proof: Recall that Algorithm EXACT finds the smallest color spanning disk internally and returns its center as q. Thus, the edges incident to q that are newly added in T' have the same length a. When the test at Step 12 is passed, we decide q to be included in the output Q. Since, otherwise, it performs the same procedure as Algorithm WANG-DU, the only possible way to get a

1: **Algorithm** NEW(S, k)
2: Set $Q \leftarrow \emptyset$ and $P \leftarrow \emptyset$
3: Compute $MST(S)$ and set $T \leftarrow MST(S)$
4: Set $k_e \leftarrow 0$ and $l_e \leftarrow \|e\|$ for edge e in T
5: Sort the l_e in decreasing order to get a list L
6: Execute EXACT($S \cup P \cup Q$)
7: Let q be the added node and T' be the resulting tree
8: Let a be the length of the edges incident to q in T'
9: Let d be the degree of q in T'
10: Let $A = T \setminus T'$ and $B = T' \setminus T$
11: Let i be an integer such that $L[i+1] < a \leq L[i]$
12: **if** $d \geq 3$ and $\sum_{e \in A} k_e = 0$ and $k \leq i - 1$ **then**
13: Set $Q \leftarrow Q \cup \{q\}$
14: Modify T to be $MST(S \cup Q)$
15: For each $e \in B$, set $k_e \leftarrow 0$ and $l_e \leftarrow \|e\|$
16: **else**
17: Find the edge e of T with the maximum l_e value
18: Set $q \leftarrow$ any point on e and $P \leftarrow P \cup \{q\}$
19: Set $k_e \leftarrow k_e + 1$ and $l_e \leftarrow \|e\|/(k_e + 1)$
20: Reorganize k_e points in $P \cap e$ evenly on e
21: **end if**
22: If $k > 1$, set $k \leftarrow k - 1$ and go to Step 5
23: Set $Q \leftarrow Q \cup P$
24: **return** Q
25: **end Algorithm**

Fig. 8. Proposed algorithm

different solution is getting into Steps 13–15 by satisfying the conditions that (1) $d \geq 3$, (2) for any edge $e \in A$ removed by addition of q, $k_e = 0$, and (3) $k \leq i - 1$, where i is an integer such that the length a of the edges in B incident to q in T' is larger than the $(i+1)$-th largest l-value $L[i+1]$ and smaller than or equal to the i-th largest l-value $L[i]$.

Now, assume that at some iteration all these conditions are fulfilled. Let $l(j)$ denote the j-th largest l-value, that is, $L[j]$ at this stage. If we do not accept q to be included into the output Q, we are left with only at most $i - 1$ nodes to be added; thus, for any $1 \leq m \leq i - 1$, the bottleneck value will be at least $l(m+1)$ after m iterations of Algorithm WANG-DU. However, if we accept q, the bottleneck value reduces to at most $l(d) \leq l(3)$ at once, and the following $m - 1$ iterations of Algorithm WANG-DU afterwards result in the bottleneck value at most $l(m+1)$ for $1 \leq m \leq i - 2$ and at most a for $m = i - 1$ since $l(i + 1) < a \leq l(i)$. This implies that accepting q with the conditions leads to a better solution than that by Algorithm WANG-DU.

About the time complexity: For the j-th iteration of the algorithm, we spend at most $O((n + j) \log(n + j))$ time. Summing $O((n+j) \log(n+j))$ over $j = 1, \ldots, k$ results in $O(k(n + k) \log(n + k))$ ∎

Therefore, Algorithm NEW guarantees at least ratio-2 approximation and is expected to find even a better solution in practice. Note that in practice the number k of deploying sensor nodes is usually smaller than the number of sensor nodes distributed currently in the domain. In this case, the asymptotic behavior of the running time of Algorithm NEW becomes $O(kn \log n)$.

V. SIMULATION RESULTS

We empirically validate our deployment algorithms presented in Section IV. In the case of $k = 1$, that is, if we want to deploy a single sensor node, we have an efficient algorithm, EXACT, that returns an optimal location. In the case of $k \geq 2$, we execute Algorithm NEW.

The experimental configuration is as follows: We take the square with side length 400 units as our sensor field, or the domain Ω and we spread initial sensor nodes S uniformly at random within Ω. The number of initial nodes is between 10 and 100. In this simulation, we deploy up to 4 additional nodes to the given configuration S in Ω by executing our deployment algorithms, and then evaluate two quantities; *support/breach improvement* and *improved pair ratio*. The support/breach improvement is obtained from two support/breach values taken before and after deployment. Computing the support/breach value is done simply by finding the longest edge in the minimum spanning tree by Theorem 3. The improved pair ratio is the proportion of pairs of points in the domain Ω whose maximal support path has been improved after deploying additional sensor nodes. To empirically measure the improved pair ratio, we randomly generated 500 pairs of a source s and a destination t in the domain Ω and counted the number of pairs such that the support distance $support(s, t)$ gets improved. For the given number n of initial sensors, we repeated each experiment 100 times and took the average values. However, we do not measure the improved pair ratio for the maximal breach paths, since the breach of the sensor network is defined to find the most vulnerable point of the network, i.e., the maximum breach, and to improve it, not to decrease breaches of many maximal breach paths.

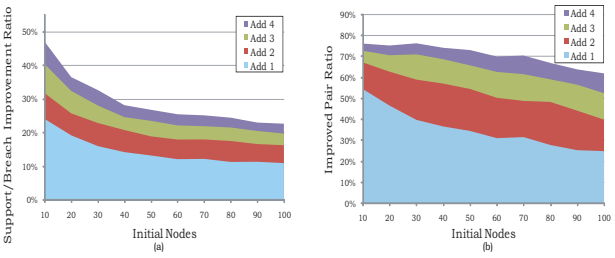

Fig. 9. (a)Support improvement ratio after each sensor addition as we increase the number of initial nodes (b)The ratio of pairs among 500 pairs whose support value is improved after each sensor addition as we increase the number of initial nodes

In the first stage of our experiment, we show how much our algorithm improves the sensor network as the number of additional sensor nodes increases. Fig. 9(a) shows the support/breach improvement when we add up to 4 additional sensor nodes by executing Algorithms EXACT

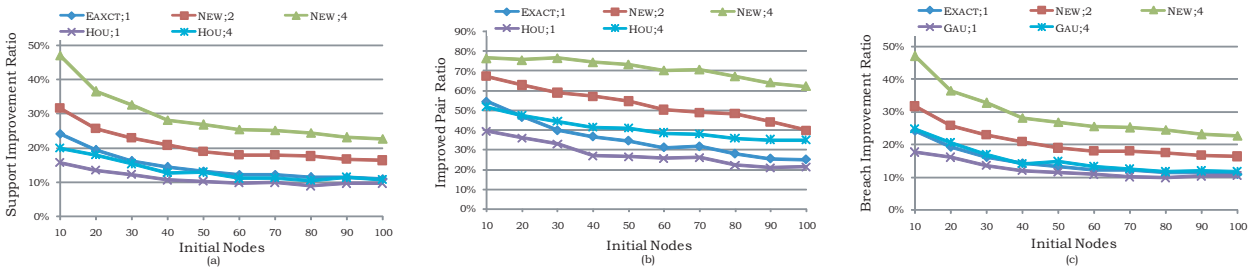

Fig. 10. Comparisons (a) Support improvement ratio compared with HOU (b) Improved pair ratio compared with HOU (c) Breach improvement ratio compared with GAU

or NEW. As the number of initial sensors increases, the difference between the edge lengths of $MST(S)$ becomes smaller due to higher density. Thus, the support/breach improvement degrades gradually. More precisely, the behavior of the graphs in Fig. 9(a) shows rapid decreasing in the beginning but gets stabilized soon after 70 or 80 initial nodes. Notice that a single addition results in more than 10% support improvement even among a large number of initial nodes. Moreover, adding more sensor nodes, we achieved remarkably better improvement; deploying only four sensor nodes gives more than 25% support improvement among 100 initial nodes.

As our algorithms are designed to improve the support of the sensor network, considering arbitrary paths, it is expected that a large portion of paths would be affected and improved. Fig. 9(b) shows such an expected improvement for arbitrary maximal support paths. The graphs of the improved pair ratio in Fig. 9(b) are more or less constantly decreasing compared to those of the support improvement; this is because as initial nodes increases more, deploying a fixed number of sensors affects relatively narrower areas of the whole domain Ω. Nonetheless, Fig. 9(b) shows significant results; among 100 initial nodes, adding a single node by Algorithm EXACT improves more than 25% of maximal support paths and adding four nodes by Algorithm NEW improves almost 60% of maximal support paths.

In the second stage, we compare our algorithms with previously known methods of deploying additional sensor nodes. For the best-case coverage, we choose the method by Hou et al. [15], named HOU hereafter, that deploys sensors optimally for a given maximal support path. For the worst-case coverage, we choose the method by Gau et al. [16], named GAU hereafter, which finds the dual maximal support path and deploys additional sensors uniformly along the maximal support path from the longest edges.

Since the previous algorithms are designed to improve the given maximal support/breach path, not the support/breach of the sensor network, the comparison would not be proper unless we give *good* maximal support/breach paths as input. In this experiment, we gave a maximal support path with the largest support distance, determining $support(S)$, as input of Algorithm HOU and a maximal breach path with the largest breach distance, determining $breach(S)$, as input of Algorithm GAU, in order to obtain their best possible performance.

Figs. 10(a) and 10(b) show the support improvement and the improved pair ratio after executing EXACT(S), NEW(S,2), NEW(S,4), HOU(S,1), and HOU(S,4). Observe that in both cases deploying two nodes by NEW is shown to be even better than deploying four nodes by HOU. Moreover, in the support improvement, deploying only a single node by EXACT shows a similar performance to deploying four nodes by HOU. Adding four nodes by NEW exhibits the outstanding quality of deployment of NEW in both of the support improvement and the improved pair ratio. For breach improvement, we have similar results. Fig. 10(c) shows that deploying two nodes by NEW is better than deploying four nodes by GAU.

VI. DISCUSSIONS AND CONCLUSIONS

We proposed a new coverage measure, namely, the support/breach of a wireless sensor network, considering arbitrary maximal support/breach paths. This provides a global way of evaluating a given network, not restricted by a specific pair of points or a path connecting them. We also posed the evaluation and deployment problem dealing with this new measure, and presented efficient distributed and centralized algorithms, supported by theoretical guarantees. Another contribution of our work is showing that the minimum spanning tree is enough to find a maximal support path between any pair of points in the domain. Based on this new discovery, we proved the evaluation and the deployment problems can be transformed into the bottleneck spanning/Steiner tree problem. Bringing known results from Computational Geometry and Operations Research, we introduced efficient algorithms in both theoretical and practical viewpoints with formal proofs and simulation results.

ACKNOWLEDGMENT

This research was supported by Ministry of Culture, Sports and Tourism(MCST) and Korea Culture Content Agency(KOCCA) in the Culture Technology(CT) Research & Development Program 2009. Work by S.W. Bae was supported by the Contents Convergence Software Research Center funded by the GRRC Program of Gyeonggi Province, South Korea.

REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Comput. Netw.*, vol. 38, no. 4, pp. 393–422, 2002.

[2] G. Werner-Allen, K. Lorincz, M. Welsh, O. Marcillo, J. Johnson, M. Ruiz, and J. Lees, "Deploying a wireless sensor network on an active volcano," *IEEE Internet Computing*, vol. 10, no. 2, pp. 18–25, 2006.

[3] K. Chintalapudi, T. Fu, J. Paek, N. Kothari, S. Rangwala, J. Caffrey, R. Govindan, E. Johnson, and S. Masri, "Monitoring civil structures with a wireless sensor network," *IEEE Internet Computing*, vol. 10, no. 2, pp. 26–34, 2006.

[4] L. Yu, N. Wang, and X. Meng, "Real-time forest fire detection with wireless sensor networks," *Wireless Communications, Networking and Mobile Computing, 2005. Proceedings. 2005 International Conference on*, vol. 2, pp. 1214–1217, Sept. 2005.

[5] C. Hartung, R. Han, C. Seielstad, and S. Holbrook, "Firewxnet: a multi-tiered portable wireless system for monitoring weather conditions in wildland fire environments," in *MobiSys '06*, 2006, pp. 28–41.

[6] I. Onat and A. Miri, "An intrusion detection system for wireless sensor networks," *Wireless And Mobile Computing, Networking And Communications, 2005. (WiMob'2005), IEEE International Conference on*, vol. 3, pp. 253–259 Vol. 3, Aug. 2005.

[7] A. P. R. da Silva, M. H. T. Martins, B. P. S. Rocha, A. A. F. Loureiro, L. B. Ruiz, and H. C. Wong, "Decentralized intrusion detection in wireless sensor networks," in *Q2SWinet '05*, 2005, pp. 16–23.

[8] S. L. F Ye, G Zhong and L. Zhang, "Energy efcient robust sensing coverage in large sensor networks," UCLA, Tech. Rep., 2002.

[9] H. Gupta, S. R. Das, and Q. Gu, "Connected sensor cover: self-organization of sensor networks for efficient query execution," in *MobiHoc '03*, 2003, pp. 189–200.

[10] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill, "Integrated coverage and connectivity configuration in wireless sensor networks," in *SenSys '03*, 2003, pp. 28–39.

[11] G. Wang, G. Cao, and T. F. L. Porta, "Movement-assisted sensor deployment," *IEEE Transactions on Mobile Computing*, vol. 5, no. 6, pp. 640–652, 2006.

[12] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. Srivastava, "Coverage problems in wireless ad-hoc sensor networks," *INFOCOM 2001*, vol. 3, pp. 1380–1387 vol.3, 2001.

[13] S. Megerian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava, "Worst and best-case coverage in sensor networks," *IEEE Transactions on Mobile Computing*, vol. 4, no. 1, pp. 84–92, 2005.

[14] Y.-T. Hou, T.-C. Lee, C.-M. Chen, and B. Jeng, "Node placement for optimal coverage in sensor networks," in *Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, 2006.

[15] Y.-T. Hou, C.-M. Chen, and B. Jeng, "An optimal new-node placement to enhance the coverage of wireless sensor networks," *Wireless Networks*, 2009.

[16] R.-H. Gau and Y.-Y. Peng, "A dual approach for the worst-case-coverage deployment problem in ad-hoc wireless sensor networks," in *Mobile Adhoc and Sensor Systems*, 2006.

[17] A. Duttagupta, A. Bishnu, and I. Sengupta, "Optimisation problems based on the maximal breach path measure for wireless sensor network coverage," in *proceedings of ICDCIT*, 2006, pp. 27–40.

[18] D. Du, L. Wang, and B. Xu, "The Euclidean bottleneck Steiner tree and Steiner tree with minimum number of Steiner points," in *Proc. of COCOON 2001*, ser. LNCS, vol. 2108, 2001, pp. 509–518.

[19] J. L. Ganlet and J. S. Salowe, "Optimal and approximate bottleneck Steiner trees," *Oper. Res. Lett.*, vol. 19, pp. 217–224, 1996.

[20] D. P. Huttenlocher, K. Kedem, and M. Shrir, "The upper envelope of Voronoi surfaces and its applications," *Discrete Comput. Geom.*, vol. 9, pp. 267–291, 1993.

[21] S. W. Bae, C. Lee, and S. Choi, "On exact solutions to the Euclidean bottleneck Steiner tree problem," in *Proc. 3rd Annu. Workshop Algo. and Comput. (WALCOM)*, ser. LNCS, vol. 5431, 2009, pp. 105–116.

[22] X.-Y. Li, P.-J. Wan, and O. Frieder, "Coverage in wireless ad hoc sensor networks," *IEEE Transactions on Computers*, vol. 52, no. 6, pp. 753–763, 2003.

[23] F. Aurenhammer and R. Klein, "Handbook of computational geometry," J.-R. Sack and J. Urrutia, Eds. Elsevier, 2000.

[24] J. Jaromczyk and G. Toussaint, "Relative neighborhood graphs and their relatives," *Proceedings of the IEEE*, vol. 80, no. 9, pp. 1502–1517, Sep 1992.

[25] K. J. Supowit, "The relative neighborhood graph, with an application to minimum spanning trees," *J. ACM*, vol. 30, no. 3, pp. 428–448, 1983.

[26] F. P. Preparata and M. I. Shamos, *Computational Geometry*. Springer-Verlag, 1985.

[27] B. Awerbuch, "Optimal distributed algorithms for minimum weight spanning tree, counting, leader election, and related problems," in *STOC '87*, 1987, pp. 230–240.

[28] C. Chiang, M. Sarrafzadeh, and C. Wong, "A powerful global router: based on Steiner min-max trees," in *Proc. IEEE Int. Conf. CAD*, 1989, pp. 2–5.

[29] L. Wang and D.-Z. Du, "Approximations for a bottleneck Steiner tree problem," *Algorithmica*, vol. 32, pp. 554–561, 2002.

[30] L. Wang and Z. Li, "An approximation algorithm for a bottleneck k-Steiner tree problem in the Euclidean plane," *Inform. Process. Lett.*, vol. 81, pp. 151–156, 2002.

[31] M. Sarrafzadeh and C. Wong, "Bottleneck Steiner trees in the plane," *IEEE Trans. Comput.*, vol. 41, no. 3, pp. 370–374, 1992.

[32] R. E. Tarjan, *Data Structures and Network Algorithms*. SIAM, 1983.

[33] M. Abellanas, F. Hurtado, C. Icking, R. Klein, E. Langetepe, L. Ma, and V. Sacristan, "The farthest color Voronoi diagram and related problems," in *Proc. 17th European Workshop Comput. Geom.*, 2001, pp. 113–116.

[34] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. MIT Press, 2001.