An Energy-Efficient and Distributed Cooperation Mechanism for k-Coverage Hole Detection and Healing in WSNs

Chenxi Qiu, Haiying Shen, *Senior Member, IEEE*, and Kang Chen

Abstract—Present approaches to achieve k-coverage for Wireless Sensor Networks still rely on centralized techniques. In this paper, we devise a distributed method for this problem, namely Distributed VOronoi based Cooperation scheme (DVOC), where nodes cooperate in hole detection and recovery. In previous Voronoi based schemes, each node only monitors its own critical points. Such methods are inefficient for k-coverage because the critical points are far away from their generating nodes in k-order Voronoi diagram, causing high cost for transmission and computing. As a solution, DVOC enables nodes to monitor others' critical points around themselves by building local Voronoi diagrams (LVDs). Further, DVOC constrains the movement of every node to avoid generating new holes. If a node cannot reach its destination due to the constraint, its hole healing responsibility will fall to other cooperating nodes. The experimental results from the real world testbed demonstrate that DVOC outperforms the previous schemes.

Index Terms—Wireless sensor networks, coverage, voronoi diagram

1 Introduction

Among numerous challenges confronted in designing protocols for WSNs, the coverage problem stands out as one of the most critical issues. Some WSN applications such as environment/ocean monitoring and animal tracking require to cover each point of the target region. However, even if we initially can deploy sensor nodes to make the entire target region fully covered, the nodes may die due to battery drain or environmental causes, which may generate coverage holes in the region. Also, nodes may deviate from their initially assigned positions due to uncontrollable factors (e.g., motion of ocean waves), leaving some areas uncovered [1]. Coverage holes reduce the ability of WSNs to detect events and network reliability. Therefore, it is crucial to equip sensor nodes with efficient hole detection and recovery capabilities to ensure full coverage of the target region.

In this paper, we study the hole detection and recovery strategies for k-coverage problems in WSNs. A target field is termed k-covered ($k \geq 1$) if every point in the target field is in the sensing ranges of at least k nodes. A k-coverage hole is a continuous area in the target field comprised of points that are covered by at most $k-1$ sensors. The problem of k-coverage is motivated by robustness concerns as well as protocol requirements. For example, triangulation based localization protocols require at least three sensors to localize an object, and hence requires every point in the target to be at least 3-covered [2]. Also, an aircraft might to need to deploy redundant sensors to provide fault tolerance since sensors are prone to failures, especially when the target region is inaccessible [3]. In the other word, if any point in the monitored area is monitored by at least k sensors, proper operation of the network can still be ensured, even if some sensors fail.

Previous schemes [4], [5] for k-coverage hole detection generate a high time complexity in a large-scale WSN with a large number of nodes. Also, their centralized method makes them not feasible in large-scale WSNs because it burdens the central node while sensor nodes have limited energy and computation capacity, thus easily generating bottlenecks. Besides hole detection, hole recovery is a key issue in the coverage problem. Numerous hole recovery schemes [4], [6], [7], [8], [9], [10] use sensor movement to improve network coverage. In these schemes, sensor nodes equipped with mobile platforms move around after the initial deployment. However, most of these schemes use centralized methods, which are not feasible in large-scale WSNs due to the aforementioned reasons. Also, they assume that accurate location information of each node is known, which are impractical for WSNs in some cases [11].

Some previous schemes [12], [13] use distributed approaches to build 1-order Voronoi diagrams (VDs) for 1-coverage detection. A VD is composed of numerous Voronoi cells, each of which has one sensor called *generating node* residing in it. All points within a Voronoi cell are closer to their generating node in the cell than to those in other cells. Thus, a Voronoi cell is fully covered if a generating node covers all of its Voronoi cell's vertices. However, it would

- *C. Qiu is with the College of Information Science and Technology, Pennsylvania State University, State College, PA 16801. E-mail: czq3@psu.edu.*
- *H. Shen is with the Computer Science Department, University of Virginia, Charlottesville, VA 22904. E-mail: hs6ms@virginia.edu.*
- *K. Chen is with the Department of Electrical and Computer Engineering, Southern Illinois University, Carbondale, IL 62901. E-mail: kchen@siu.edu.*

Manuscript received 29 Mar. 2016; revised 1 July 2017; accepted 22 Oct. 2017. Date of publication 1 Nov. 2017; date of current version 3 May 2018. (Corresponding author: Haiying Shen.)
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TMC.2017.2767048

be very energy-expensive to directly extend these schemes for the k-order coverage problem because each sensor node requires much more location information for building k-order VD and must monitor its distant critical points.

Our work aims to solve two formidable challenges: 1) how can k-coverage holes be efficiently detected in a distributed manner, and 2) how can new holes be avoided while healing current holes? Accordingly, we propose a Distributed VOronoi based Cooperation scheme (DVOC) based on mathematical models for k-coverage in WSNs. In DVOC, nodes cooperate in hole detection and recovery by node movement, which significantly saves energy by reducing message transmission and avoiding generating new holes during node movement.

We first introduce a definition of the kth generating (nearest) node of a critical point (i.e., Voronoi vertex), so that as long as each kth generating node covers its critical point, the whole area is k-covered. We also introduce a warm up approach to find a point's kth generating node from a set of nodes. Thus, simply enabling Voronoi vertex to be monitored by its kth nearest node can achieve k-coverage hole detection. However, their long monitoring distance would lead to high transmission cost. To this end, we propose DVOC, in which each node builds *local k-order VD (LVD)* that enables it to find its nearby critical point's generating nodes and check if the generating nodes cover this critical points. Thus, nodes cooperate in monitoring each other's critical points and informing the generating nodes of uncovered critical points. We have proven that DVOC never misses any holes if the accuracy of the LVD is guaranteed, and it alleviates the transmission burden for each node significantly. "Accuracy" here means the degree that the constructed VD approximates the actual VD. Further, in order to avoid generating new holes during node movement, we mathematically identify the safe area of each node, where the node should be located to avoid new hole generated. If a node cannot reach its destination due to the constraint, its hole healing responsibility falls to another kth generating node. Compared with previous schemes, DVOC costs less for both transmission and mechanical movement; since DVOC can avoid oscillating movement with its cooperation mechanism, it converges more rapidly than previous movement schemes. Specially, DVOC consists of three components.

1) *Distributed k-coverage checking*: each node first collects location information from its neighbor nodes, and builds a LVD, which allows it to find the critical points of others around itself within the diagram. Then, nodes cooperate in monitoring nearby critical points for each other. When a node detects a hole, it informs the generating nodes to move towards the hole.
2) *Safe area identification*: By transforming the k-coverage problem to the problem that whether the radius of circumcircle of each k-order Delaunay triangle (DT) [14] formed by each critical point's three generating nodes is smaller than the radius of nodes' sensing ranges, we calculates each node's safe area.
3) *Movement-based hole healing strategy*: When a sensor is informed to move to a point, it calculates its *safe area* using the information of its critical points retrieved from nearby nodes. Once the sensor finds itself unable to reach the destination due to the constraint of its safe area, its hole healing responsibility falls to the point's another generating node (sensor). Thus, nodes cooperate with each other in movement-based hole healing in order not to generate new holes.

The rest of this paper is organized as follows. Section 2 presents related work. Section 4 introduces the models for the coverage problem and presents the movement-assisted scheme for detecting and healing coverage holes. Section 6 presents a performance evaluation of DVOC in comparison with several previous schemes. The final section concludes with a summary of contributions and discussions on further research work.

2 Related Work

Over the past years, intensive research efforts have been devoted to the study of WSNs' coverage problems (including *hole detection* [4], [5], [6], [7], [15] and *hole healing*) in WSNs [1], [4], [6], [7], [8], [9], [10], [16], [17], [18], [19], [20], [21], [22].

2.1 Hole Detection Strategies

One category of schemes for k-coverage is called *sensing border based methods* [5], [15], in which each node verifies the coverage of its vicinity by checking if its border is completely k-covered by other nodes. The target field is k-covered iff the sensing border of every internal node is k-covered. Huang et al. [5] realized this scheme in a centralized manner with a time complexity of $O(nm\log m)$, where m is the maximum number of sensing regions that can intersect the sensing region of a particular sensor and n is the number of nodes in the network, while Bejerano et al. [15] realized it in a distributed manner with a time complexity $O(b^3)$, where b denotes the number of sensors in the vicinity for each node. Though the sensing border based scheme can detect holes in both centralized and decentralized ways, the computing cost is too high in the worst-case [4] ($O(n^2\log n)$ in centralized algorithm and $O(n^3)$ in decentralized algorithm).

Another type of approaches, called VD based methods, has been served as more efficient strategies to solve WSNs' coverage problems [4], [6], [7]. For example, the ordinary VD (also called 1-order VD) is used to detect 1-coverage hole in WSNs [6], [7]. A VD is composed of numerous Voronoi cells, each of which has one sensor called *generating node* residing in it. All points within a Voronoi cell are closer to their generating node in the cell than to those in other cells. Thus, if some points in a Voronoi cell are not covered by their generating node, coverage holes are generated because the points cannot be covered by other sensors. A Voronoi cell is fully covered if a generating node covers all of its Voronoi cell's vertices. Thus, each node only needs to detect whether its vertices are covered. So and Ye [4] extended the 1-order VD scheme [6], [7] to the k-coverage case using a k-order VD in a centralized manner. Similar to 1-VD based method, in k-order VD, each node only needs to detect the vertices of the Voronoi cells which are associated with itself.

A challenge for VD based algorithm is how to build VD distributively and efficiently. Some previous works have introduced distributed algorithms for constructing 1-order VD, which only exploits locality information, rather than broadcasting location information to all nodes in the WSN.

For example, Sharifzadeh and Shahabi [12] proposed a method in which a node uses its collected location information of some nodes to build a 1-order VD. Bash and Desnoyers [13] proposed a method to improve the accuracy. The method begins with an initial approximation of a local k-order Voronoi cell at each node based on its neighboring nodes and then leverages geographic routing primitives (e.g., GPSR [23]) to systematically refine the Voronoi cell and verify its correctness. To judge whether its constructed Voronoi cell is accurate, a node only needs to check whether there is a node unknown by itself that is closer to any of the cell's vertices than itself. However, none of previous works propose an algorithm for building a k-order VD in a distributed manner, which requires each node to hold much more location information of other nodes and generates higher energy costs for transmission and computing.

Both sensing border based methods and VD based methods require precise location information of each sensor. There are another category of hole detection approaches, called *homology-based* methods, can detect coverage hole in WSNs via homology without node location information [24], [25], [26], [27], [28], [29], [30], [31], [32]. Specifically, De Silva et al. [25] first proposed a centralized hole detection algorithm via homology. They constructed the Rips complex corresponding to the WSN's communication connectivity and checked the coverage by verifying whether the first homology group of the Rips complex is trivial. Then the above ideas were first implemented in a distributed way by Muhammad et al. in [26], [27]. Muhammad et al. also used the flows of k-Laplacian (a natural generalization of the graph Laplacian) operators on time-varying simplicial complexes to verify dynamic coverage in mobile sensor networks in [32]. Decreusefond et al. [30] applied computational homology to verify coverage in a variety of settings, including static planar coverage, 3-D barrier coverage, and time-dependent sweeping coverage. All these homology-based algorithms can be used to detect coverage holes for WSNs on surfaces, but they do not consider the cases that Rips complex may miss some special coverage holes. Although [24] improves the accuracy of homology-based coverage hole detection, the accuracy still cannot achieve 100 percent. In contrast, the location-based hold detection methods, like VD-based approaches, don't miss coverage holes provided the network connectivity is guaranteed.

2.2 Hole Healing Strategies

In VD based methods, like [6], [7], when a critical point is detected uncovered, the generating node will move towards the critical point to recover the hole. Apart from these, many other methods have also been proposed for coverage detection and recovery. For example, grid quorum-based movement schemes [17], [18], [19], [20], [21] have been used for healing coverage holes. These schemes view the movement-assisted network re-deployment problem as a load balancing problem under the virtual grid model. The schemes partition an entire target region into small grid cells, and the number of nodes in each cell is considered as the load of the cell. The schemes schedule sensor movements in order to achieve a balanced distribution of sensors among the grid cells. A node within a grid cell can communicate with other nodes in its four adjacent cells, and makes movement decisions according to the information from adjacent cells (i.e., wether the cells are overloaded or underloaded). In particular, SMART [20] directs nodes from overloaded cells to underloaded cells and prevents any unnecessary movement. Thus both the total moving distance and the total number of moves can be minimized. However, the moving distance of grid quorum-based movement schemes is comparatively long since these approaches do not aim to heal holes, but tries to balance the distribution of all the nodes. Compared with grid quorum-based movement schemes, VD based movement schemes are more likely to find a shorter path because their target is solely to recover the hole, not to balance the distribution of the hole network. Other movement strategies also have been proposed for solving the coverage problem. One method simulates "virtual force" between sensor nodes [8], [9], [10], [16].

In addition, many WSNs' healing strategies are proposed for some special scenarios. For example, Luo et al. [1] studied the coverage problem on sea surface, where nodes keep moving due to wave of water. They assumed that the entire target region is fully covered initially, and each node dominates a number of interest points which are randomly and uniformly distributed throughout the entire region. When some nodes lose their interest points, their neighbors move to inherit them. Though the above movement-assisted schemes have their own merits., most of these schemes only focus on the 1-coverage case, or require nodes to have a knowledge of location information of all other nodes. Further, none of these schemes can prevent generating new holes during node movement. Our rigorous mathematical analysis proves that DVOC can distributively detect holes more efficiently than previous schemes, and prevent generating new holes when nodes are moving to heal old holes.

TABLE 1
Notations

Notations	Description
Λ	The target region
S	The sensor node set
(x_i, y_i)	The Cartesian coordinate of node i
$\mathcal{D}_i$	The sensing range of s_i
R_s	The sensing radius of each node
$d_{i,j}$	The Euclidean distance between node i and node j
$\mathcal{V}_1(i)$	The 1-order Voronoi cell of node i
$V_k(\mathcal{S})$	The k-order VD of $\mathcal{S}$, where $\mathcal{S}$ is a set of nodes
$V_k^i(\mathcal{N}_i)$	The local k order VD of node i, where $\mathcal{N}_i$ is a set of nodes stored in node i
$B_{i,j}$	Perpendicular bisector of node i and node j
$h_{i,j}$	Use $B_{i,j}$ to divide the target region into two half planes, and $h_{i,j}$ denotes the half plane that contains node i
$\mathcal{G}_v$	The set of k^{th} generating nodes of a Voronoi vertex v
$\mathcal{N}_i$	The node set stored in node i's storage

3 Background: The k-Coverage Problem and The k-Order Voronoi Diagram

Before describing our method, in this section, we first give a brief overview of the k-coverage problem, 1-order VD and k-order VD, as well as the notations and the definitions that will be used throughput this paper. Table 1 lists the notations and their descriptions.

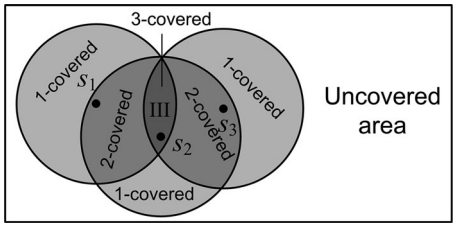


Fig. 1. An example of 1-covered, 2-covered, and 3-covered regions.

The k-coverage problem. Consider a set of senor nodes (or nodes) $S = \{1, 2, \ldots, N\}$ that are distributed over a *target field* Λ, where the position of each node i is represented by its Cartesian coordinate (x_i, y_i). Each node i can sense specified events in its *sensing range* modeled as a disk D_i with radius R_s. We say Λ is k-covered by all nodes S if only if

$$\forall p \in \Lambda,\ \exists \mathcal{S} \subseteq S,\ \text{where } |\mathcal{S}| \geq k \quad \text{s.t. } p \in \bigcap_{i \in \mathcal{S}} D_i. \tag{1}$$

In other words, Λ is k-covered by the set of sensor nodes S when every point in Λ is covered by the sensing ranges of at least k sensors. Finally, the k-Coverage problem is defined *to detect whether a region Λ is k-covered by a given set of nodes S*. We call a point $p \in \Lambda$ a *coverage hole* if p is covered less than k times by S. Fig. 1 shows an example of 1-covered, 2-covered, and 3-covered regions.

The 1-order VD. The 1-order VD of a collection of nodes partitions the plane into *Voronoi cells* (or *cells*) with one node inside each cell [6], [33]. Fig. 2a gives an example of 1-order VD with 8 nodes on the plane, where the grey area is the Voronoi cell of node i. The node located in a cell is called the cell's *generating node*. Every point in a cell is closer to the cell's generating node than to any other nodes. For each cell, we call all its vertices the *critical points* of the cell's generating node. To detect whether the whole cell is covered, the generating node only needs to check whether it covers each of its critical points. If there exists a critical point that cannot be reached by the generating node, then no other node can cover this critical point as the generating node is the closest node to its critical points.

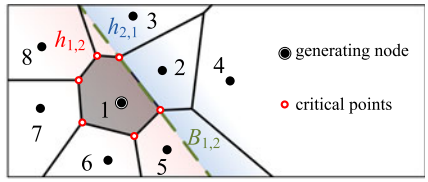
The Voronoi cell for each node can be derived as follows: given any pair of nodes $i, j \in S$, we use their perpendicular bisector $B_{i,j}$ (the green dotted line in Fig. 2a is the perpendicular of node 1 and node 2) to partition the plane into two half-planes: $h_{i,j}$ containing node i and $h_{j,i}$ containing node j. Then, the 1-order Voronoi cell of node i, denoted by $\mathcal{V}_1(i)$, is the intersection of $N-1$ half-planes with all other nodes (the grey part in Fig. 2a is the 1-order Voronoi cell of node 1)

$$\mathcal{V}_1(i) \triangleq \bigcap_{1 \leq j \leq N,\ j \neq i} h_{i,j}. \tag{2}$$

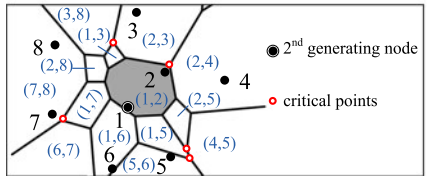
The 1-order VD is composed of the 1-order Voronoi cell of each node in S. Notice that in what follows we use $\mathcal{V}$ and V to represent Voronoi cell and Voronoi diagram, respectively.

The k-order VD. An extension form of the 1-order VD is a k-order VD [34], [35], [36], which partitions the plane into cells such that each cell, say $\mathcal{V}_k(\mathcal{S})$, is associated with a subset of nodes $\mathcal{S}$ ($\mathcal{S} \subset S$ and $|\mathcal{S}| = k$). In particular, $\mathcal{V}_k(\mathcal{S})$ can be calculated by

$$\mathcal{V}_k(\mathcal{S}) \triangleq \bigcap_{i \in \mathcal{S},\ j \in S \backslash \mathcal{S}} h_{i,j}. \tag{3}$$



(a) 1-order VD



(b) 2-order VD

Fig. 2. 1-order VD and 2-order VD.

That is, $\mathcal{V}_k(\mathcal{S})$ is the locus of points closer to all nodes in $\mathcal{S}$ than to any other nodes in $S \backslash \mathcal{S}$. Or equivalently, $\mathcal{S}$ are the k closest nodes to the point in $\mathcal{V}_k(\mathcal{S})$.

For each vertex v of $\mathcal{V}_k(\mathcal{S})$, we order the nodes in $\mathcal{S}$ according to their distance to v (i.e., from the shortest distance to the longer distance). Then, to detect whether $\mathcal{V}_k(\mathcal{S})$ is k-covered, we only need to check whether each vertex of $\mathcal{V}_k(\mathcal{S})$ can be covered by its kth nearest node in $\mathcal{S}$: If each vertex v can be covered by its kth nearest node in $\mathcal{S}$, then v can be covered by other $k-1$ nodes in $\mathcal{S}$; otherwise, v cannot be k-covered, since $\mathcal{S}$ are the k nearest nodes to every point $\mathcal{V}_k(\mathcal{S})$ including v. We call v a *critical point* of its kth nearest node and name the kth nearest node to v by the *kth generating node* of v. In most cases, each vertex has three kth generating nodes and more details will be introduced in Section 5.1.

Consequently, the whole target region can be detected by checking each cell $\mathcal{V}_k(\mathcal{S})$. We use $V_k(S)$ to represent the whole k-order VD, which is the combination of all the k-order Voronoi cells $\mathcal{V}_k(\mathcal{S})$ ($\mathcal{S} \in S$)

$$V_k(S) = \cup_{\mathcal{S} \in S} \mathcal{V}_k(\mathcal{S}). \tag{4}$$

Fig. 2b gives an example of the 2-order VD for 8 nodes, where each cell is associated with two nodes. In particular, the grey cell is associated with node 1 and node 2. Also, there are 5 cells are associate with node 1, labeled by $(1,2)$, $(1,3)$, $(1,5)$, $(1,6)$, and $(1,7)$. The red vertices are the critical points of node 1 and node 1 is the 2nd generating node of those red vertices.

4 System Design

In this part, we will introduce our VD-based hole detection approach. In particular, we will first give a warmup strategy called *PVD based k-coverage checking* in Section 4.1, which is directly developed from the previous methods, and then devise a more efficient approach called *LVD based k-coverage checking* in Section 4.2.

In what follows, we assume that each sensor knows its own location (e.g., by GPS), and the sensor nodes are dense enough to enable the entire network to be well connected. We also assume that sensors can freely move in any direction in the target region, and there is no obstruction area where sensors cannot move in.

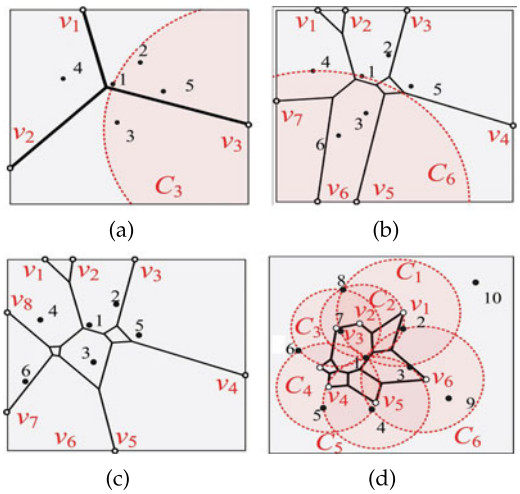


Fig. 3. Construction of 2-order VD.

4.1 PVD Based k-Coverage Checking

Our approach, DVOC, conducts k-order VD construction and k-coverage checking in a distributed manner. A question is how to distribute these workload among sensors. Before introducing LVD, we first introduce a warm up method called *PVD-based k-coverage checking*, which can be simply extended from the distributed method for constructing 1-order VD [13]. Notice that in a k-order VD, a node i may be associated with several cells. We call the combination of these associated cells node i's *partial k-order VD (PVD)* (Definition 4.1), and it requires each node to be responsible for its partial k-order VD in k-coverage checking.

Definition 4.1. *Suppose* $\mathcal{S}_i^1, \ldots, \mathcal{S}_i^{m_k}$ $(m_k = \binom{N-1}{k-1})$ *are the subsets of* S *with cardinality* k *that contain node* i. *The combination of the cells associated with these subsets is called the partial k-order VD (PVD) of node* i, *denoted by*

$$\tilde{V}_k^i(\mathcal{N}_i) = \cup_{l=1}^{m_k} \mathcal{V}_k(\mathcal{S}_i^l). \quad (5)$$

In the previous distributed 1-order VD construction methods [12], [13], each node initializes its tentative cell using a small subset of nodes in its region. The node is the *generating node* of the vertices of its cell. The GPSR routing algorithm [23] always routes a packet to the reachable node with minimum distance. To increase the cell's accuracy, the generating node i checks if there exits a node that is closer to each of the cell's vertices than itself using GPSR; if yes, node i reduces the area of the tentative cell by excluding the closer node from the cell. When each node stops cell modification, the 1-order VD is constructed. Similarly, to construct PVD, each node i first initializes its tentative PVD using the known nodes (including itself) in its storage. For each vertex of its PVD v, node i first locates v's kth generating nodes and then probes nearby nodes using GPSR to check if there is a node is closer to v than the v's current kth generating node. It then adds such nodes into its storage. After the checking of all vertex, node i rebuilds its PVD to enhance the accuracy of the PVD. Since a node's PVD only consists of its nearby nodes, each node only needs to probe its nearby nodes for the PVD construction [12], [13]. Node i stores the vertices and edges for its tentative PVD. This process repeats until node i cannot find closer node to any of its PVD's vertices. After all nodes build their PVD, the k-order VD is constructed.

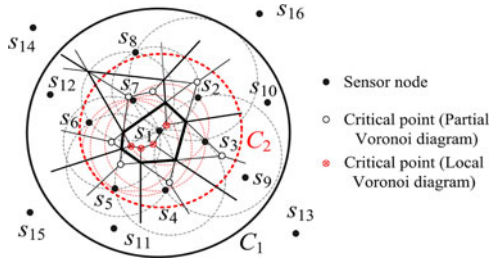


Fig. 4. LVD versus PVD.

Fig. 3 shows an example for the 2-order PVD construction. In Fig. 3a, node 1 modifies its PVD to 3 (b) when a new node 5 is closer to the PVD's vertex v_3 than node 2 and node 3 (node 2 and node 3 are the 2nd generating nodes of v_3). Similarly, in Fig. 3b, node 1 continues to modify its PVD to 3 (c) because a new node 6 is closer to the PVD's vertex v_6 than node 1 and node 4 (node 1 and node 4 are the 2nd generating nodes of v_6). While in Fig. 3d, node 10 will not be observed by node 1 through GPSR as node 10 is not closer to any vertex in the diagram than vertex's 2^{nd} generating nodes.

4.2 LVD Based k-Coverage Checking

Accordingly, rather than relying on one central node to collect the location information of all nodes in the WSN and then build the k-order VD, PVD based k-coverage checking distributes this workload among the nodes by letting each node collect partial location information and build its own k-order PVD in order to build the k-order VD. Fig. 4 gives an example of a 16-node WSN, where node 1 has built its partial 2-order VD with 7 critical points so far. We draw circles with the critical points be the circle centers, and their distances to node 1 be the radius. Then, we draw the smallest circle centered at node 1 that embraces all these circles (denoted by C_1). To guarantee the accuracy of its partial diagram, node 1 only needs to use GPSR to collect the location information of some nodes inside C_1, because nodes outside of C_1 are farther to the critical points than node 1 and these nodes do not affect the PVD construction. However, higher value of k leads to larger C_1 and more location information that node 1 needs to collect by GPSR, leading to a high energy cost.

In addition, we observe that nodes' partial k-order VDs always overlap with each other. In other words, each critical point is known by several nodes rather than only one node. Thus, multiple nodes checking the same critical point leads to duplicated checking and unnecessary energy cost.

Therefore, simply extending the distributed 1-order VD construction method to a k-order VD construction method cannot fully use nodes' location information, especially when k is large. It is crucial to find an energy-efficient method to reduce the probing cost for high accuracy. To this end, we propose a k-coverage checking method based on *local k-order VD (LVD)*.

LVD designates the node i closest to a critical point of another node j to conduct the accuracy checking and k-coverage checking on this point. Thus, LVD shrinks the circle where node i needs to collect location information, hence reducing GPSR probing distances and the number of

probed nodes of each node. It also avoids unnecessary probing cost due to the overlapping.

A challenge in LVD is how to distribute the set of all critical points in the target field to sensors so that each critical point is only assigned to one sensor that is closest to itself. We notice that a node i's 1-order Voronoi cell is mutually exclusive to any 1-order Voronoi cell of other node's location. We then define LVD as the intersection of node i's 1-order Voronoi cell and the k-order VD. Then, the local k-order VD of each node must be mutually exclusive. Also, the combination of LVD form the intact k-order VD, which will be proved later.

Definition 4.2. *We use $\mathcal{N}_i$ to denote the set of location points that node i has collected. Then, the local k-order VD of a node i (denoted by $\hat{V}_k^i(\mathcal{N}_i)$) is defined as the intersection of node i's 1-order Voronoi cell and the k-order VD of $\mathcal{N}_i$, or formally*

$$\hat{V}_k^i(\mathcal{N}_i) \triangleq V_k(\mathcal{N}_i) \cap \mathcal{V}_1(i). \tag{6}$$

where $V_k(\mathcal{N}_i)$ denotes the k-order VD given by $\mathcal{N}_i$. We say a LVD $\hat{V}_k^i(\mathcal{N}_i)$ is accurate *iff $\hat{V}_k^i(\mathcal{N}_i) = V_k^i(S)$.*

Lemma 4.1. *To guarantee node i's LVD's accuracy, there cannot be any node unknown to node i that is closer to any of the cell's vertex than any of the cell's generating nodes.*

Proof. Lemma 4.1 can be easily derived from [13]. □

Below, we present how a node builds its LVD (Algorithm 1). Similar to PVD construction, each node also conducts the accuracy check on its identified critical points of other nodes. Basically, it checks whether there is a node within the circle with the critical point as the center and the distance between the critical point and its generating node as the radius. If yes, it additionally considers the location of the closer node to build a more accurate local k-order VD. Each node i calculates its LVD by Equation (6).

Algorithm 1. Pseudo Code for Building LVD.

input : $\mathcal{N}_i = \{i\}$ `// At beginning node` i `only has its own location;`
output : $\hat{V}_k^i$;
1 **repeat**
2 accuracy $\leftarrow$ TRUE; `// Denote whether the LVD is accurate`
3 Calculate $\hat{V}_k^i$ based on the current $\mathcal{N}_i$ (Equation (6));
4 **for** *each* $v \in \hat{V}_k^i$ **do**
5 Find a kth generating node of v, say node j;
6 Use GPSR to probe the nearest node to v in $S/\mathcal{N}_i$, say node l;
7 **if** *node l is closer to v than node j* **then**
8 Add node l to $\mathcal{N}_i$;
9 accuracy = FALSE;
10 **until** accuracy = *TRUE*;

For example, in Fig. 4, node 1 builds its local k-order VD (marked by bold lines in the center). It finds the 6 critical points within the diagram, whose generating nodes are nodes $2-7$. Then, node 1 only needs to probe nodes within circle C_2, which is the smallest circle that embraces all the circles; each having a critical point as the center and its distance with its generating node as the radius. When node 1 finds that a critical point cannot be reached by its kth generating nodes' sensing ranges, node 1 informs the generating nodes. Compared to the PVD-based scheme, the LVD-based scheme reduces the number of critical nodes that should be checked by node 1, and reduces the probing scope from circle C_1 to C_2. In the following, we proves the correctness of our LVD-based scheme in Lemma 4.2.

Lemma 4.2. *The union of all nodes' LVDs forms the VD and no critical points will be missed if the accuracy of every LVD is guaranteed.*

Proof. Assuming the accuracy of every local k-order VD is guaranteed, by definition we can retrieve

$$\begin{aligned}\cup_{i=1}^{N}\hat{V}_k^i(\mathcal{N}_i) &= \cup_{i=1}^{N}\hat{V}_k^i(S) = \cup_{i=1}^{N}(V_k(S)\cap\mathcal{V}_1(i)) \\ &= V_k(S)\cap\big(\cup_{i=1}^{N}\mathcal{V}_1(i)\big) = V_k(S)\cap\Lambda = V_k(S)\end{aligned}$$

□

Time complexity. We then compare the running time of building PVD and LVD. Here we use the k-order VD construction algorithm in [35] with time complexity $O(k^2(N\log N))$, where N denotes the number of nodes in S. Due to limitations of space, we do not introduce the details of this algorithm. In this algorithm, whenever a new node location is added to $\mathcal{N}_i$, node i needs to execute the algorithm one more time for accuracy checking. We denote c as the number of nodes' locations that node i needs to collect to build a PVD. After the cth node's location has been probed, the algorithm for building PVD is finished. The running time equals

$$\sum_{u=1}^{c} O\big(k^2 u\log u\big) = O\big(k^2 c^2\log c\big), \tag{7}$$

and each node needs to store the location information of $O(c)$ neighbor nodes in its storage. Similarly, the running time of LVD can be calculated as $O(k^2e^2\log e)$ and the space complexity is $O(e)$, where e denotes the number of nodes' locations that node i needs to collect for the accuracy checking of the diagram. Since e is always less than c, the running time of k-order LVD construction scheme is less than that of the k-order PVD construction scheme. Thus, LVD is more computation and energy efficient, and hence is more practical than the previous schemes.

5 Hole Healing Strategy Using Nodes' Mobility

In this section, we will introduce how to heal a hole when the hole is detected. Specifically, we first introduce a concept called *safe area* to constrain nodes' mobility to prevent new holes being generated in Section 5.1. Based on this concept, we then describe our Cooperative movement-based hole healing strategy in Section 5.2.

5.1 Safe Area Identification

In DVOC, after nodes build their own k-order LVDs, they collaborate to detect whether the vertices of the cells in the diagram (i.e., critical points) are k-covered by finding out if each vertex is covered by its kth generating node. In previous works such as VOR [6], a sensor node simply moves towards the Voronoi vertex that is uncovered. However,

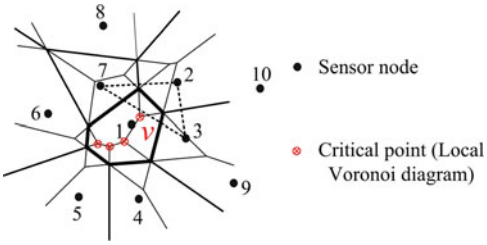


Fig. 5. 1-order DT in 2-order VD

such node movements might cause some areas to become uncovered, and several iterations might be needed to converge when a new hole cannot be covered by some other nodes. Therefore, we introduce a method to enable a node to identify its safe area that it should not move out. If the node moves out of the safe area, it no longer covers its critical point. First, we give the definition of k-order Delaunay triangle and its property (Theorem 5.1):

Definition 5.1. *(k-order Delaunay triangle (DT) [14]) Let S be a set of sensor nodes's locations in the plane. For nodes i, j, $l \in S$, a triangle $\triangle_{i,j,l}$ is a k-order DT if the circle through nodes i, j, and l has at exactly k nodes of S inside.*

Theorem 5.1. *If v is a vertex of a k-order Voronoi cell, then v must be the intersection of the bisectors $B_{i,j}$ and $B_{i,l}$, and the kth generating nodes of v are nodes i, j, and l [4].*

According to Theorem 5.1, for any vertex v of a k-order Voronoi cell, v has at least three kth generating nodes, say i, j and l, and v is the center of the circle of $\triangle_{i,j,l}$. Also, in the circle of $\triangle_{i,j,l}$, there must exist $(k-1)$ nodes nearer to the center of circle than the triangle's three vertices, which means that the circle must contains exactly $(k-1)$ nodes. Therefore, by connecting the three kth generating nodes of any critical point in a node's LVD, we can always get a $(k-1)$-order DT.

Algorithm 2. Hole Detection by Node i.

input : $\hat{V}_k^i$
output : $[\mathcal{C}_{\text{hole}},\ \mathcal{G}_{\text{hole}}]$
1 // $\mathcal{C}_{\text{hole}}$ is the set of critical points uncovered, $\mathcal{G}_{\text{hole}}$ is the set of $\mathcal{C}_{\text{hole}}$'s generating nodes;
2 **for** *each critical point v in $\hat{V}_k^i$* **do**
3 Find v's kth generating nodes $\mathcal{G}_v$;
4 **for** *each node $j \in \mathcal{G}$* **do**
5 **if** *node j cannot cover v* **then**
6 Add v to $\mathcal{C}_{\text{hole}}$;
7 Add node j to $\mathcal{G}_{\text{hole}}$;

If the kth generating nodes' sensing ranges covers their critical point, then the radius of their $(k-1)$-order Delaunay triangle is smaller than or equal to their sensing range. Consequently, the problem that whether every point of target region is k-covered can be transformed to the problem that whether the radius of each of such $(k-1)$-order DTs is smaller than the sensing ranges of the kth generating nodes. For example, in Fig. 5, node 1 builds its 2-order LVD, which contains v and its 2nd generating nodes are nodes 2,3 and 7. The $\triangle_{2,3,7}$ is a 1-order DT which center v.

In order to make the three kth generating nodes (i.e., three vertices of a triangle) to cover their critical point (i.e., the center of the circle), from the law of sine and cosine, we can derive that the relationship among the radius of triangle's circumscribed circle (denoted as r) and triangle's three edges (denoted as $d_{i,j}$, $d_{i,l}$ and $d_{j,l}$). That is,

$$r = \frac{d_{i,j}d_{i,l}d_{j,l}}{\sqrt{\left(d_{i,j}^2 + d_{j,l}^2 + d_{i,l}^2\right)^2 - 2\left(d_{i,j}^4 + d_{j,l}^4 + d_{i,l}^4\right)}} \tag{8}$$

According to Equation (8), we can retrieve that, any vertex of a k-order Voronoi cell can be covered by its kth generating nodes iff the relationship of its $(k-1)$-order DT's three edges satisfy the following condition

$$R_s \geq \frac{d_{i,j}d_{i,l}d_{j,l}}{\sqrt{\left(d_{i,j}^2 + d_{j,l}^2 + d_{i,l}^2\right)^2 - 2\left(d_{i,j}^4 + d_{j,l}^4 + d_{i,l}^4\right)}}. \tag{9}$$

Definition 5.2. *(*Safe area*) Consider a $(k-1)$-order DT $\triangle_{i,j,l}$, where nodes i and j are not moving, the* safe area *of node l for $\triangle_{i,j,l}$ is defined as the area, where node l can be located to guarantee that the triangle's circumcenter is k-covered (satisfying Equation (9)).*

According to Definition 3.6 above, we calculate node l's *safe area* for $\triangle_{i,j,l}$. According to Equation (9), we can get that the location of node l should satisfy one of the following two equations, where (x_l, y_l), (x_i, y_i) and (x_j, y_j) represent the Cartesian coordinates of nodes l, i and j respectively

$$(x_l - x' + f_x)^2 + (y_l - y' + f_y)^2 \leq R_s \tag{10}$$

$$(x_l - x' - f_x)^2 + (y_l - y' - f_y)^2 \leq R_s \tag{11}$$

where $x' = \frac{x_i + x_j}{2}$, $y' = \frac{y_i + y_j}{2}$, $f_x = |x_j - x_i|\sqrt{(\frac{R_s}{d_{i,j}})^2 - \frac{1}{4}}$, and $f_y = |y_j - y_i|\sqrt{(\frac{R_s}{d_{i,j}})^2 - \frac{1}{4}} \times \frac{x_i - x_j}{y_j - y_i}$. Note that a node can be the generating node for multiple critical points. When node l needs to move to cover a hole (i.e., an uncovered critical point), it should not move out of its safe area of another covered critical point, that is, its location (x_l, y_l) should satisfy Equations (10) and (11). Then, we achieve Proposition 5.2, which presents a necessary condition to k-cover a point.

Proposition 5.2. Let $d_{i,j}$ denote the distance between any pair of two vertices of a $(k-1)$-order DT, then $d_{i,j} \leq 2R_s$ is a necessary condition to k-cover the center of the circumcircle of the triangle.

Proof. Suppose for the sake of contradiction that $d_{i,j} > 2R_s$, then x_l and y_l have no solutions in Equations (10) and (11), implying that $d_{i,j}$ should be no larger than $2R_s$. □

According to Equations (10) and (11), we know that a generating node must know the other two generating nodes of this critical point in order to know its safe area for their critical point. Hence, when a node i notifies the three generating nodes that their critical point is not covered, it also tells them the location information (x, y) of the critical point and the other two generating nodes. Based on Equations (10) and (11), the node limits its movement within its safe area when it is moving.

Now, we consider a more complicated scenario, where the safe area of a sensor belongs to multiple $(k-1)$-order

DTs. Suppose node i is in a number of $(k-1)$-order DTs denoted by $DT_{i,1}, \ldots, DT_{i,m}$ and the corresponding *safe area* of node i in the triangles are $A_{i,1}, \ldots, A_{i,m}$, where m is the number of the triangles. Then, the *united safe area* A^i_{united} for node i is defined by the intersection of $A_{i,1}, \ldots, A_{i,m}$:

$$A^i_{\text{united}} = \bigcap_{j=1}^{m} A_{i,j}. \tag{12}$$

We also define *coverage rate* of a WSN as the ratio of the area k-covered by given sensor nodes to the area of the entire target region of the WSN. Then, we have the following conclusion:

Lemma 5.1. *The coverage rate of a WSN increases monotonically with node movements if the nodes do not break the united safe area constraint.*

Proof. Consider that when a node, say node i, moves towards one hole, the area size of this hole is decreased; whereas if node i stays in the united safe area it originally is located in, no new hole is generated. Thus, the coverage rate is increased monotonically with node movements. □

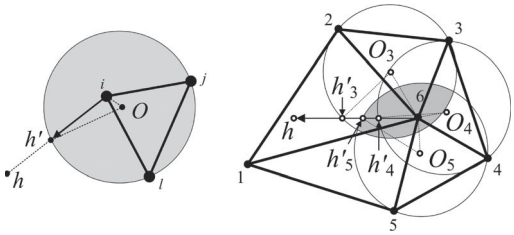
Lemma 5.1 indicates that by obeying the constraint of the safe area, DVOC is guaranteed to converge if the density of sensor nodes is high enough.

5.2 Cooperative Movement-Based Hole Healing Strategy

In Section 5.1, we introduce a concept, called safe area, to constrain nodes' mobility to prevent new holes being generated. Then, based on this concept, we propose a cooperation mechanism that can heal coverage holes while preventing new holes being generated during node movements in this section. Here, we let $S_{\text{m}} \subseteq S$ denote the set of nodes that can move, and we call the nodes in S_{m} *mobile nodes*. Basically, a mobile node tries to stay in its safe area when moving to cover its critical point. If it cannot move to its destination, it asks other mobile generating nodes of the same critical point to move towards the hole for compensation. In the following part, we will introduce the detail of the approach.

As we mentioned in Section 4.2, when a sensor node detects that there exists a critical point v beyond the sensing range of its kth generating nodes, it will send a notification to all the three generating nodes. If there exists at least one mobile generating node, then the mobile generating node(s) is required to move towards v to heal the hole; otherwise, each generating node, say node j, needs to forward the notification to a nearby mobile node that is further away from v than node j to fix the coverage hole.

Let node i be the mobile node that is required to move to heal the coverage hole. Then node i figures out its destination h to cover the hole, where h is the nearest point that node i needs to move to make its sensing range reach its uncovered critical point. Here, the shortest path is the line that connects node i and node i's uncovered critical point. To prevent generating new holes, node i needs to calculate the safe area A^i_{united} for all of its critical points. To this end, node i needs to identify all of its critical points by building a PVD, and then consults its surrounding nodes for the information of its critical points and their generating nodes. Notice that the operation of building PVD and calculating safe area is only required when the coverage hole is discovered.



(a) Single safe area (b) United safe area

Fig. 6. Calculating the movement destination according to safe area.

After calculating the safe area, node i figures out the distance it needs to move towards h. Let h' be the intersection of the line and A^i_{united}. If h is inside A^i_{united}, node i directly moves to it. Otherwise, the destination is h'. We use Fig. 6 to better illustrate how to calculate the movement distance. In Fig. 6a, we first consider a single safe area, where h is the critical point of nodes i, j and l, and the gray part is the safe area of node i. When h is outside of node i's single safe area of $\triangle_{i,j,l}$, the movement distance $d_{i,h'}$ is given by

$$d_{i,h'} = \sqrt{d^2_{O,h'} - d^2_{O,i} + 2d_{O,i}d_{O,h'}\cos\angle h'kO}. \tag{13}$$

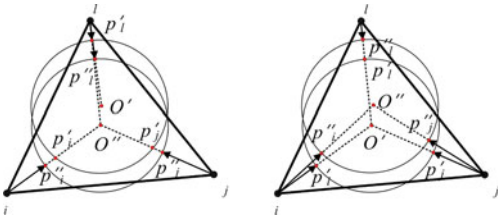
We then extend the case of single safe area to the united safe area of $A_{i,1}$, $A_{i,2}$, $A_{i,3}, \ldots, A_{i,m}$, as Fig 6b shows. Here, node 6 is the sensor node informed to move to point h to recover a hole. Each circle in the figure is the safe area of node 6 in each triangle, and the gray area is the united safe area of node 6, A^i_{united}. The movement range of node 6 should be within the united safe area A^i_{united}. That is, the maximum moving distance should be no larger than d_{6,h'_l}. The distance that node 6 should move can be calculated by

$$d_{\max} = \min\{d_{6,h'_l}\} \quad (l = 1, 2, 3, \ldots, 5). \tag{14}$$

Accordingly, node 6 is unable to each the identified destination due to the constraint of its safe area. To tackle this problem, we propose a cooperation mechanism as follows.

We call the nodes associated with the same critical point *partners*; we can find that there are always three or more partners sharing one critical point. When a critical point is detected as a coverage hole, all the partners sharing this critical point will move towards to the critical point. However, a node might not be able to reach the calculated destination due to the constraint of safe area. In the cooperation mechanism, when a node cannot move enough distance to reach its critical point, its partners are required to move to heal the hole. Below, we discuss two cases: (1) when one node cannot move to its destination, and (2) when two nodes cannot move to their destinations, due to the constraint of safe areas (i.e., avoiding generating new holes).

In Fig. 7a, O' is the critical point of the three generating nodes l, i and j. Nodes l, i, j are notified that O' is not k-covered, then nodes l, i, j calculate that their destinations are p'_l, p'_i and p'_j and the uncovered critical point can be covered. However, due to the safe area constraint in Equation (14), node l finds that it cannot reach p'_l, but can only arrive at p''_l. Node l then sends a message to inform its partners, nodes i and j, to move more in order to cover the



(a) When 1 node cannot move to its destination (b) When 2 nodes cannot move to the destinations

Fig. 7. Cooperative movement.

critical point. Node l needs to calculate the destinations of nodes i and j for achieving the coverage, denoted by p_i' and p_j'. That is, p_l', p_i', and p_j' must satisfy Equation (9). Making a circle going through p_l'' with the radius R_s, the circumcenter O'' of the circle can be determined. The intersections of the circle and two lines connecting O'' with nodes i and j respectively are p_i'' and p_j''. Then, node l informs nodes i and j to move to p_i'' and p_j'' respectively.

Algorithm 3. Movement Scheme of Node i.

input : $[\mathcal{C}_{\text{hole}},\ \mathcal{G}_{\text{hole}}]$
output : p_i // The destination of node i
1 Initiate t by 0;
2 **repeat**
3 **for** *each* $v \in \mathcal{C}_{\text{hole}}$ **do**
4 Notify all three v's kth generating nodes that v is outside of their sensing range. If there is no mobile kth generating nodes, notify the nodes that is further away from v than the kth generating nodes;
5 After the last movement, listen for a period and store the first message it receives, say MSG;
6 **if** *MSG.type* $=$ MOV **then**
7 $P \leftarrow$ MSG.loc;
8 **if** $P \notin A^i_{\text{united}}$ **then**
9 $P' \leftarrow$ ReDest(A^i_{united}, P) // ReDest recalculates the destination of the node when node's original destination is outside the safe area;
10 Move to P';
11 $P'' \leftarrow$ ParDest(i, j, k) //ParDest recalculates the destinations of the node's partners;
12 newMSG.type $\leftarrow$ HELP;
13 newMSG.loc $\leftarrow P''$;
14 Send newMSG to node j (or node k);
15 **if** *MSG.type* $=$ HELP **then**
16 **if** MSG.loc $\notin A^i_{\text{united}}$ **then**
17 $P'' \leftarrow$ ParDest(i, j, k);
18 newMSG.type $\leftarrow$ FORCE;
19 newMSG.loc $\leftarrow P''$;
20 Send newMSG to node j;
21 **if** *MSG.type* $=$ FORCE **then**
22 Move to MSG.loc;
23 Increase t by 1;
24 **until** $t \leq$ TTL

Fig. 7b shows an example that two nodes cannot reach their destinations. In the figure, when two nodes, i and j, find that destinations p_i' and p_j' are outside their safe areas and stop at points p_i'' and p_j'', they inform their partner node i. node i recalculates its path and moves to p_l''. To locate p_l'', we make the circle to go through p_i'' and p_j'' with the radius R_s and then identify its circumcenter O''. The intersection between the circle and the link connecting O'' and node l is p_l''.

TABLE 2
Messages' Explanation in Algorithm 3

Symbol	Explanation
type = MOV	Require the node to move
type = HELP	Let the node ask help from other nodes
type = FORCE	Force the node to move to a given destination
loc	The destination's location

It is possible that all three partners cannot move enough distances to recover the hole; in this case, the three nodes are forced to move out of their safe areas to their calculated destinations, and stay at those points for a period of time, which is predetermined according to the network condition in order to prevent oscillations. Generally the new hole will be healed by some other nodes if the density of the network is high. Actually, in a dense sensor scenario, the possibility that all the three partners cannot reach their destinations is very low. That is, the cooperation mechanism can prevent generating new coverage holes in most cases. In addition, when multiple non-overlapping areas ask node i to perform different movements to restore k-coverage, node i only moves to the one that first sends the request. After moving to the destination, the node will stay for a period to prevent oscillations (as Fig. 8 shows).

Algorithm 3 describes the detailed progress of movement conducted by each node and Table 2 lists the explanation of different types of messages in Algorithm 3. In Algorithm 3, $\text{Dest}(v)$ is used to calculate the target point of node i if its critical point v is uncovered, $\text{ReDest}(A^i_{\text{united}}, P)$ is used to calculate the new destination point of node i if P (target point) is outside node i's safe area A^i_{united}, and $\text{parDest}(p_i, p_j, p_k)$ is used to calculate the destination of p_j if p_i cannot reach its original destination. All these three functions are calculated by Equations (10), (11) and (13). To avoid the interferences between VD building (Algorithm 1), hole detection (Algorithm 2) and hole healing (Algorithm 3) (for example what happens when one sensor is executing the first algorithm while a neighboring sensor is executing the third algorithm), all the nodes are synchronized at the beginning of each operation. In addition, we set a TTL (Time To Live) for Algorithm 3 to prevent oscillations.

6 Performance Evaluation

In this section, we present the experimental results of DVOC in comparison with typical VORonoi-based algorithm (VOR) [6], [13]. Also, we compare DVOC with the other two typical movement-assist schemes for full coverage in WSNs: Scan-based Movement-Assisted Sensor Deployment (SMART) [20] and Sea Surface Coverage (SSC) [1] in Matlab on GENI Orbit testbed [37], [38]:

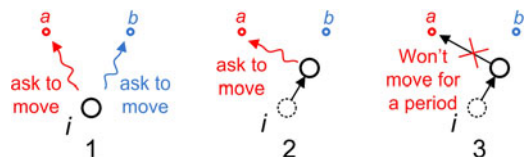
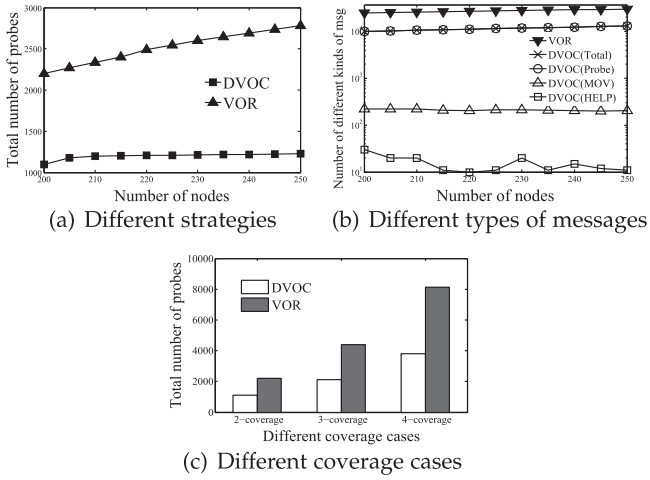


Fig. 8. When a node has to fix two coverage holes a and b.

1) *VOR*. VOR divides the entire target field into cells and each node covers one cell. When a node detects a coverage hole in its cell, it moves towards the farthest Voronoi vertex to cover the hole. In our experiment, VOR is improved to handle k-coverage in a distributed fashion by using an incremental algorithm [13] to compute and maintain the Voronoi cell at each sensor node and using GPSR to verify the correctness of the cell [12].
2) *SMART*. SMART is a grid quorum-based movement scheme that divides the entire field to virtual grids. Each grid has a head, which communicates with other heads to identify overloaded and underloaded grids and directs the movement of nodes to balance the load (i.e., the number of sensors) of all the grids.
3) *SSC*. SSC distributes a number of "interest points" randomly and uniformly throughout the entire target field, and assigns a sensor to each of these points to cover it. When some nodes lose their "interest points", other nearby nodes move to inherit the points. Nodes can only move in four directions. Since the target region is not fully covered at the beginning in our simulation environment, we assume that every "interest point" is assigned to the node nearest to it.

The testbed uses a large two-dimensional grid of 400 802.11 radio nodes, which can be dynamically interconnected into specified topologies. Since GENI-Orbit testbed has limited number of nodes (less than 400 nodes), here we only take the 2-coverage as an example for the k-coverage in GENI Orbit testbed. We also evaluate the performance of different strategies in 3-coverage and 4-coverage using simulation. The target field is a $400\ \mathrm{m} \times 400\ \mathrm{m}$ area. In 2-coverage, the number of sensors was varied from 200 to 250. In 3-coverage and 4-coverage, the number of nodes are set by 300 and 400, respectively. The radius of the sensing range and the transmission range of each sensor are set by 45 m and 70 m, respectively. We set each packet size by 56 bytes (TinyOS 2.x) [39], and each mobile sensor node consumes 27.96 Joule per meter in moving. The energy consumption of transmission is in the order of 100×10^{-9} Joule per bit [1]. We assume that the energy contained in each sensor's battery is 1,000 Joule, and like [1], [6], [13], we set the proportion of mobile nodes in WSNs by 100 percent by default. We measured the following metrics:

1) *Total number of probes*. both DVOC and VOR require each node to probe the locations of other nodes to build a VD. This metric reflects the transmission cost for both schemes.
2) *Number of messages*. Because DVOC needs nodes to transmit messages to each other for cooperation after k-order VD is built, simply comparing the total number of probes is not sufficient for transmission cost measurement. Hence, (includes messages for probing, informing other nodes to move, and asking help from partners)
3) *Total moving distance*. this is defined as the sum of the moving distances of all nodes for healing holes. It reflects the delay and energy cost of node movement in healing holes. This metric can measure the energy consumption since the cost of physical movement is the main energy consumption for sensor node [20].
4) *Coverage rate*. We distribute 500,000 points uniformly throughout the entire field, and the coverage rate equals the percent of covered points. This metric reflects the schemes' performance in achieving full coverage [6].

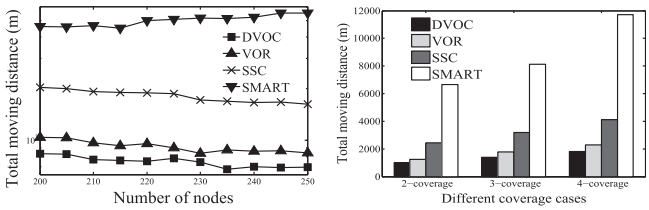


(a) Different strategies (b) Different types of messages
(c) Different coverage cases

Fig. 9. Transmission cost.

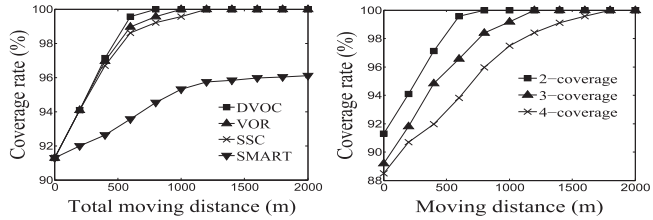
6.1 Experimental Results

Transmission cost. Figs. 9a and 9b show the transmission costs between DVOC and VOR measured by the total number of probes and messages, respectively. Both schemes require that each node probes the locations of its surrounding nodes for building a VD. The difference is that under DVOC, every node needs to send messages when finding a hole or moving to heal it. The number of messages of DVOC includes the messages for probing (Probe), informing other nodes to move to heal holes (MOV), and asking for help from partners (HELP). We did not count the number of messages forcing partner to move (FORCE) since the number of such messages is extremely small. For VOR, only the messages for probing are included. From both Fig. 9 ab, we find that in both metrics, DVOC is significantly superior to VOR. The total number of messages in VOR (all for probing) is about 2 (range from 1.936 to 2.176) times of that of DVOC, and the number of messages transmitted in DVOC is only about half of VOR's (range from 0.512 to 0.546). We also measured the proportion of different types of messages in DVOC; on average, probe messages constitute 98.60 percent (this explain why in Fig. 9b the curve of the number of probes and the curve of the total number of messages are very close); MOV messages constitute 1.21 percent and HELP messages constitute 0.19 percent. The reason why DVOC has higher efficiency than VOR in terms of transmission cost is that DVOC uses a cooperation mechanism based on a local k-order VD, which requires less location information than the partial k-order VD used in VOR. Admittedly, DVOC still makes nodes communicate with each other after the VD is built, but such communication cost is very small compared with the cost for probing location information. As confirmed by Fig. 9b, the transmission cost for MOV and HELP constitute only a small percentage of the total communication.



(a) Different strategies (b) Different coverage cases

Fig. 10. Total moving distance.



(a) Different strategies (b) Different coverage cases

Fig. 11. Efficiency of node movements.

Fig. 9c compares the total number of probes between DVOC and VOR in 2-coverage (200 nodes), 3-coverage (300 nodes) and 4-coverage (400 nodes) in simulation. From the figure, we can observe that both DVOC and VOR have more probes in 3-coverage and 4-coverage than in 2-coverage. On the other hand, in 3-coverage and 4-coverage, the system has more nodes, i.e., respectively 300 and 400 nodes, allocated in the target region, which requires more LVDs to be built.*Total Moving Distance*. Fig. 10a shows the *total moving distance* versus the number of nodes in different schemes. From the figures, we find that DVOC has the shortest *total moving distance*. On average, the *total moving distances* of VOR, SMART, and SSC are respectively 1.25, 2.08 and 8.65 times as large as that of DVOC. SSC generates a much longer total moving distance than VOR and DVOC, since SSC is a cell-based algorithm where every node is only able to move in four directions and cannot move along a straight line to the target. In contrast, VOR enables nodes to move in all directions, leading to a shorter total moving path than SSC. Also, we can observe that SMART generates a significantly longer total moving distance than the others. This is because the main task of SMART is to balance the distribution of sensor nodes throughout the entire target region in order to achieve full coverage. Even when a region has no holes, nodes need to move to achieve balance. Both DVOC and VOR use VDs for hole detection and healing. The difference between DVOC and VOR's movement patterns is that DVOC is a cooperation based algorithm where each node needs to calculate whether it will move out of its *safe area* when informed to recover holes. If the destination is outside of its safe area, the node stops moving and asks for partners' help in covering the hole. Though it is possible that all the partners cannot move to their destinations, the cooperation mechanism in DVOC can prevent new holes from being generated in most cases.

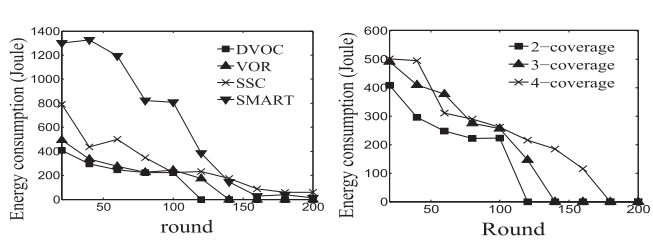
Fig. 10b compares the total moving distance of different strategies in 2-coverage, 3-coverage and 4-coverage in the simulation, from which we observe that the total moving distance of 3-coverage and 4-coverage is higher than that of 2-coverage. It is because that when the dimension of coverage is higher, the coverage holes detected are usually farther from their generating nodes, which requires longer distance to move to heal the hole.

Efficiency of Node Movements. In order to show the efficiency of node movement in achieving full coverage, Fig. 11a shows the *coverage rate* versus *the total moving distance* in VOR, SSC, SMART and DVOC when the number of sensors equals 200. From the figure, we observe that DVOC achieves full coverage (coverage rate $> 99.99\%$) more rapidly than the other three schemes. In particular, DVOC achieves 99.99 percent coverage rate when the total moving distance is 900 m, while nodes in VOR, SMART and SSC move over 1,000 m to achieve full coverage. This is because DVOC avoids generating new holes during node movement. Lemma 5.1 has indicated that under DVOC, almost every movement can increase the coverage rate, which helps DVOC converge more rapidly than the other schemes. In addition, we observe that SMART achieves full coverage very slowly, as the objective of SMART is to balance the distribution of nodes, indicating that many movements contribute to the balanced sensor distribution, and the hole size does not decrease rapidly at the beginning.

We also compare the efficiency of node movement of DVOC in 2-coverage, 3-coverage and 4-coverage in simulation in Fig. 11b. From the figure, we find that DVOC achieves full coverage less rapidly than in 3-coverage and 4-coverage than in 2-coverage. As we mentioned before (for Fig. 10c), nodes in higher coverage cases move longer distance to heal holes, implying that smaller area can be recovered per distance unit. Accordingly, the efficiency of node moment of 3-coverage and 4-coverage is lower than that of 2-coverage.

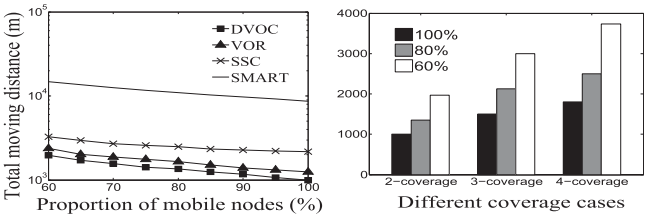
Energy consumption over time. We use round to denote the sequence of each time period in which the destinations of moving nodes are calculated. Figs. 12a and 12b compare the energy consumption over rounds in different algorithms and different coverage cases (i.e., 2-coverage, 3-coverage, and 4-coverage), respectively. In both figures, the energy consumption decreases over time because as more coverage holes are fixed, less mobile nodes are required to move for hole healing. From Figs 12a and 12b, we observe that the overall energy consumption follows: DVOC $<$ VOR $<$ SSC $<$ SMART and 2-coverage $<$ 3-coverage $<$ 4-coverage, which are consistent with the results in Figs. 11a and 11b.

Comparison with different proportion of mobile nodes. In this experiment, we set the proportion of mobile nodes by 100 percent by default. But in some WSN applications (e.g., [40]), only part of nodes are allowed to move. Accordingly, we also ascertain the performance (i.e., moving distance) of our approach when the proportion of mobile nodes varies



(a) Different strategies (b) Different coverage cases

Fig. 12. Energy consumption over time.



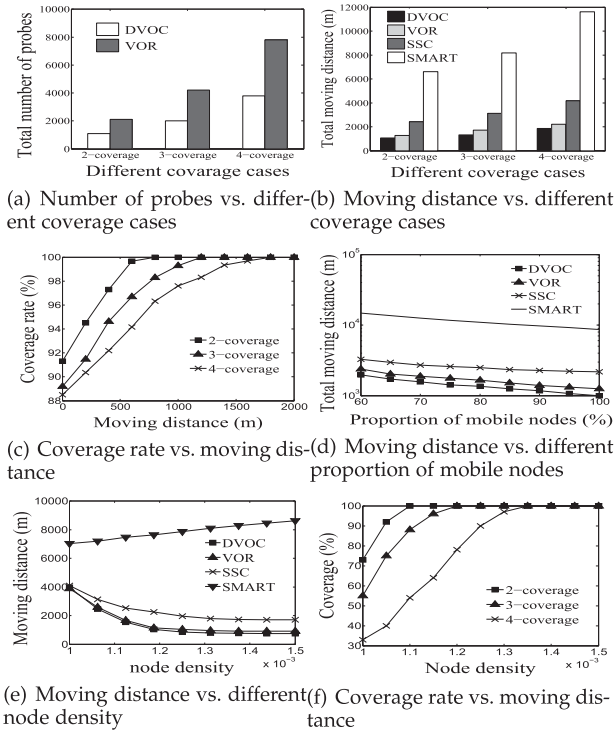
(a) Different strategies (b) Different coverage cases

Fig. 13. Comparison with different proportion of mobile nodes.

from 60 to 100 percent in Fig. 13 ab. Specifically, Fig. 13a compares the total moving distance of all the nodes in different algorithms and Fig. 13b compares the total moving distance in 2-coverage, 3-coverage, and 4-coverage. From both figures, we observe that the total moving distance is decreasing with the increase of the proportion of mobile nodes. It is because that the more static sensor nodes are located in the system, on average the longer distance mobile nodes need to move to fix the coverage holes.

Experiments with spatial Poisson process. In the above experiment, we assume that the nodes are uniformly distributed and the number of nodes is constant in each setting. In what follows, we test the performance of different algorithms under the assumption that sensor nodes are randomly distributed according to a spatial Poisson process [41], and we depict the experimental results in Figs. 14a–14f. We set the node density by $0.00125/\mathrm{m}^2$ by default. Fig. 14a compares the total number of probes of DVOC and VOR, from which we observe that VOR needs more probes than that of DVOC. Fig. 14b shows the total moving distance of different strategies in the cases of 2-coverage, 3-coverage, and 4-coverage, respectively. Fig. 14c compares the efficiency of node movement of DVOC in 2-coverage, 3-coverage and 4-coverage. Fig. 14d shows the total moving distance versus different proportion of mobile nodes. Overall, the experimental results in the setting of the spatial Poisson process (Fig. 14a–14d) are consistent with the results under which nodes are uniformly distributed (Figs. 9c, 10b, 11b, and 13a).

Fig. 14e depicts the total moving distance when the node density varies from 0.001 m^{-2} to 0.0015 m^{-2}. From Fig. 14e, we observe that lower node density leads to longer moving distance. It is because that 1) the less sensor nodes are located in the target region, the more coverage holes need to heal; and 2) when node density is lower, each node's movement is more likely to generate new holes, leading to more movements from the node's partners. Fig. 14f compares the maximum coverage rate that DVOC can achieve in differen coverage cases (i.e., 2-coverage, 3-coverage, and 4-coverage) when the node density is changed from 0.001 m^{-2} to 0.0015 m^{-2}. From Fig. 14f, we observe that DVOC cannot achieve full coverage when the node density is lower than a threshold. The thresholds are 0.00135, 0.0012, and 0.0011 in 2-coverage, 3-coverage, and 4-coverage, respectively. When the node density is too low, DVOC cannot fix all the coverage holes because 1) node movement will generate new holes while moving to heal holes and 2) higher moving distance causes more nodes to die as their power is run out, which probably disconnects the network and leads the generating nodes to be unaware of coverage holes.



(a) Number of probes vs. different coverage cases (b) Moving distance vs. different coverage cases

(c) Coverage rate vs. moving distance (d) Moving distance vs. different proportion of mobile nodes

(e) Moving distance vs. different node density (f) Coverage rate vs. moving distance

Fig. 14. Experiments with spatial Poisson process.

7 Conclusions

In this paper, we propose a Distributed VOronoi-based Cooperation Mechanism for full coverage. In this scheme, each node builds its own *local k-order VD* and helps other nodes to detect holes within their own diagrams. With the guarantee of the accuracy of the local k-order VDs, DVOC can greatly alleviate the burdens on nodes for transmission while ensuring no holes are missed. Further, DVOC uses a cooperation mechanism to prevent generating new holes during node movement, which greatly increases the efficiency of node movements. Experimental results from GENI's ORBIT testbed shows that DVOC has superior performance than previous schemes in terms of energy-efficiency and efficiency of coverage. In our future work, we will study techniques to shorten the moving distance to minimize the sum of all moving paths, i.e., globally optimized paths. We will also apply connectivity based method (or homology based method) to detect k-coverage holes.

Acknowledgments

This research was supported in part by U.S. National Science Foundation grants NSF-1404981, IIS-1354123, CNS-1254006, and CNS-1249603, and Microsoft Research Faculty Fellowship 8300751.

References

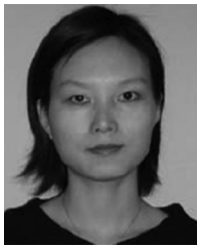
[1] J. Luo, D. Wang, and Q. Zhang, "Double mobility: Coverage of the sea surface with mobile sensor networks," in *Proc. IEEE Conf. Inf. Comput. Commun.*, 2009, pp. 118–126.

[2] O. Tekdas and V. Isler, "Sensor placement for triangulation-based localization," *IEEE Trans. Autom. Sci. Eng.*, vol. 7, no. 3, pp. 681–685, Jul. 2010.

[3] R. Katsuma, Y. Murata, N. Shibata, K. Yasumoto, and M. Ito, "Extending k-coverage lifetime of wireless sensor networks using mobile sensor nodes," in *Proc. IEEE Int. Conf. Wireless Mobile Comput. Netw. Commun.*, 2009, pp. 48–54.

[4] A. M.-C. So and Y. Ye, "On solving coverage problems in a wireless sensor network using voronoi diagrams," in *Proc. 1st Int. Conf. Internet Netw. Economics*, 2005, pp. 584–593.
[5] C.-F. Huang and Y.-C. Tseng, "The coverage problem in a wireless sensor network," in *Proc. 2nd ACM Int. Conf. Wireless Sensor Netw. Appl.*, 2003, pp. 115–121.
[6] G. Wang, G. Cao, and T. F. L. Porta, "Movement-assisted sensor deployment," in *Proc. IEEE Conf. Inf. Comput. Commun.*, 2004, pp. 2469–2479.
[7] A. Ghosh, "Estimating coverage holes and enhancing coverage in mixed sensor networks," in *Proc. Proc. 29th Annu. IEEE Int. Conf. Local Comput. Netw.*, 2004, pp. 68–76.
[8] Y. Zou and K. Chakrabarty, "Sensor deployment and target localization based on virtual forces," in *Proc. IEEE Conf. Inf. Comput. Commun.*, 2003, pp. 1293–1303.
[9] S. Poduri and G. Sukhatme, "Constrained coverage for mobile sensor networks," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2004, pp. 165–171.
[10] A. Howard, M. J. Mataric, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," in *Proc. Int. Symp. Distrib. Autonomous Robot. Syst.*, 2002, pp. 299–308.
[11] D. Niculescu, "Positioning in ad hoc sensor networks," *IEEE Netw.*, vol. 18, no. 4, pp. 24–29, Jul./Aug. 2004.
[12] M. Sharifzadeh and C. Shahabi, "Supporting spatial aggregation in sensor network databases," in *Proc. Int. Symp. ACM Int. Workshop Geographic Inf. Syst.*, 2004, pp. 166–175.
[13] B. A. Bash and P. J. Desnoyers, "Exact distributed voronoi cell computation in sensor networks," in *Proc. 6th Int. Symp. Inf. Process. Sensor Netw.*, 2007, pp. 236–243.
[14] J. Gudmundsson, M. Hammar, and M. van Kreveld, "Higher order delaunay triangulations," *Comput. Geometry: Theory Appl.*, vol. 23, no. 1, pp. 85–95, 2002.
[15] Y. Bejerano, et al., "Simple and efficient k-coverage verification without location information," in *Proc. IEEE Conf. Inf. Comput. Commun.*, 2008, pp. 897–905.
[16] N. Heo and P. Varshney, "Energy-efficient deployment of intelligent mobile sensor networks," *IEEE Trans. Syst. Man Cybern. Part A: Syst. Humans*, vol. 35, no. 1, pp. 78–92, Jan. 2005.
[17] S. Chellappan, X. Bai, B. Ma, D. Xuan, and C. Xu, "Mobility limited flip-based sensor networks deployment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 2, pp. 199–211, Feb. 2007.
[18] Z. Jiang, J. Wu, R. Kline, and J. Krantz, "Mobility control for complete coverage in wireless sensor networks," in *Proc. 37th IEEE Int. Conf. Distrib. Comput. Syst. Workshops*, 2008, pp. 291–296.
[19] G. Wang, G. Cao, T. F. L. Porta, and W. Zhang, "Sensor relocation in mobile sensor networks," in *Proc. IEEE Conf. Inf. Comput. Commun.*, 2005, pp. 2302–2312.
[20] S. Yang, M. Li, and J. Wu, "Scan-based movement-assisted sensor deployment methods in wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 8, pp. 1108–1121, Aug. 2007.
[21] Y. Xu, J. Heidemann, and D. Estrin, "Geography-informed energy conservation for ad hoc routing," in *Proc. 9th Annu. Int. Conf. Mobile Comput. Netw.*, 2001, pp. 70–84.
[22] N. Heo and P. K. Varshney, "An intelligent deployment and clustering algorithm for a distributed mobile sensor network," in *Proc. IEEE Int. Conf. Syst. Man Cybern.*, 2003, pp. 45776–4581.
[23] B. Kar and H. T. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," in *Proc. 9th Annu. Int. Conf. Mobile Comput. Netw.*, 2000, pp. 243–254.
[24] F. Yan, P. Martins, and L. Decreusefond, "Accuracy of homology based coverage hole detection for wireless sensor networks on sphere," *IEEE Trans. Wireless Commun.*, vol. 13, no. 7, pp. 3583–3595, Jul. 2014.
[25] V. de Silva and R. Ghrist, "Coverage in sensor networks via persistent homology," *Algebraic Geometric Topology*, vol. 7, no. 1, pp. 339–358, Apr. 2007.
[26] A. Muhammad and M. Egerstedt, "Control using higher order laplacians in network topologies," in *Proc Int. Symp. Math. Theory Netw. Syst.*, 2006, pp. 24–28.
[27] A. Muhammad and A. Jadbabaie, "Decentralized computation of homology groups in networks by gossip," in *Proc Amer. Control Conf.*, 2007, pp. 3438–3443.
[28] A. Tahbaz-Salehi and A. Jadbabaie, "Distributed coverage verification in sensor networks without location information," *IEEE Trans. Autom. Control*, vol. 55, no. 8, pp. 1837–1849, Aug. 2010.
[29] A. Vergne and L. Decreusefond, "Simplicial homology for future cellular networks," *IEEE Trans. Mobile Comput.*, vol. 14, no. 8, pp. 1712–1725, Aug. 2014.
[30] V. de Silva and R. Ghrist, "Coordinate-free coverage in sensor networks with controlled boundaries via homology," *Int. J. Robot. Res.*, vol. 25, no. 12, pp. 1205-1222, Dec. 2006.
[31] V. D. Silva and R. Ghrist, "Simplicial homology of random configureations," *Int. J. Robot. Res.*, vol. 46, no. 2, pp. 325–347, 2006.
[32] A. Muhammad and A. Jadbabaie, "Dynamic coverage verification in mobile sensor networks via switched higher order laplacians," in *Proc Conf.: Robot.: Sci. Syst. III*, 2007.
[33] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars, *Computational Geometry: Algorithm and Applications*. Berlin, Germany: Springer, 2008.
[34] M. I. Shamos and D. Hoey, "Closest point problems," in *Proc. 16th Symp. Found. Comput. Sci.*, 1975, pp. 151–162.
[35] D. T. Lee, "On k-nearest neighbor voronoi diagram in the plane," in *IEEE Trans. Comput.*, vol. 31, no. 6, pp. 478–487, Jun. 1982.
[36] H. Edelsbrunner, J. O. Rouke, and R. Seidel, "Constructing arrangements of lines and hyperplanes with applications," *SIAM J. Comput.*, vol. 15, no. 2, pp. 341–363, 1986.
[37] GENI project. (2017). [Online]. Available: http://www.geni.net/
[38] Orbit. (2017). [Online]. Available: http://www.orbit-lab.org/
[39] TinyOS. (2011). [Online]. Available: http://www.tinyos.net/tinyos-2.x/doc/html
[40] E. P. Freitas, R. S. Allgayer, M. A. Wehrmeister, C. E. Pereira, and T. Larsson, "Supporting platform for heterogeneous sensor network operation based on unmanned vehicles systems and wireless sensor nodes," in *Proc Intell. Vehicles Symp.*, 2009, pp. 786–791.
[41] A. Guo, Y. Zhong, W. Zhang, and M. Haenggi, "The gauss-poisson process for wireless networks and the benefits of cooperation," in *Proc. IEEE Int. Symp. Inform. Theory*, 2004, pp. 1916–1929.

Chenxi Qiu received the BS degree in telecommunication engineering from Xidian University, China, and the PhD degree in electrical and computer engineering from Clemson University, in 2009 and 2015. He currently is working toward the postdoc degree in the College of Information and Science, Pennsylvania State University, PA. His research interests include cyber security, cyber physical systems, and cloud computing.



Haiying Shen received the BS degree in computer science and engineering from Tongji University, China, and the MS and the PhD degrees in computer engineering from Wayne State University, in 2000, 2004, and 2006, respectively. She is currently an associate professor in the CS Department, University of Virginia. Her research interests include distributed computer systems and computer networks with an emphasis on P2P and content delivery networks, mobile computing, wireless sensor networks, and grid and cloud computing. She was the program co-chair for a number of international conferences and member of the program committees of many leading conferences. She is a 2010 Microsoft Faculty Fellow, a senior member of the IEEE, and a member of the ACM.

Kang Chen received the BS degree in electronics and information engineering from the Huazhong University of Science and Technology, China, the MS degree in communication and information systems from the Graduate University of Chinese Academy of Sciences, China, and the PhD degree in computer engineering from Clemson University, in 2005, 2008, and 2014, respectively. He is currently an assistant professor in the Department of Electrical and Computer Engineering, Southern Illinois University. His research interests include emerging wireless networks and software defined networking.