A Delaunay-based Coordinate-free Mechanism for Full Coverage in Wireless Sensor Networks

Chenxi Qiu, Haiying Shen*, *Senior Member, IEEE*

Abstract—Recently, many schemes have been proposed for detecting and healing coverage holes to achieve full coverage in wireless sensor networks (WSNs). However, none of these schemes aim to find the shortest node movement paths to heal the coverage holes, which could significantly reduce energy usage for node movement. Also, current hole healing schemes require accurate knowledge of sensor locations; obtaining this knowledge consumes high energy. In this paper, we propose a DElaunay-based Coordinate-free Mechanism (DECM) for full coverage. Based on rigorous mathematical analysis, DECM can detect coverage holes and find the locally shortest paths for healing holes in a distributed manner without requiring accurate node location information. Also, DECM incorporates a cooperative movement mechanism that can prevent generating new holes during node movements in healing holes. Simulation results and experimental results from the real-world GENI Orbit testbed show that DECM achieves superior performance in terms of the energy-efficiency, effectiveness of hole healing, energy consumption balance and lifetime compared to previous schemes.

Index Terms—Wireless sensor networks, full coverage, energy efficiency, Delaunay triangulation.

1 Introduction

In wireless sensor networks (WSNs), sensor nodes may die due to battery drain or environmental causes. Also, nodes may deviate from their assigned positions due to the effects of uncontrollable factors (e.g., ocean waves). Coverage holes hamper the ability of WSNs to detect events and reduce network reliability. Therefore, it is crucial to equip the sensor nodes with efficient hole detection and healing capabilities in order to ensure full coverage of the target field.

Numerous schemes have been proposed for hole detection and hole healing. For hole detection, the current approaches can be categorized into two types: sensing border based schemes [1, 2] and Voronoi diagram (VOR) based schemes [3–5]. In sensing border based schemes, each node verifies the coverage of its vicinity by checking if its border is completely covered by other nodes. The target field is covered iff the sensing border of every internal node is covered. While in VOR based schemes, the whole target region is partitioned into Voronoi cells, each of which has one sensor called a *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 a generating node finds some points in a Voronoi cell that are not covered by itself, it moves directly to the farthest point to heal the hole. Both sensing border based schemes and (VOR) based schemes require accurate node location information.

For hole healing, many of previous works [3, 6–15] use sensor movement to improve network coverage. Since mechanical movement is much more energy-expensive than electronic communications [16], the node moving distance should be minimized [17]. However, none of the previous hole healing works aim to find the shortest paths of node movement, which could greatly enhance the energy-efficiency. Also, all of these hole healing schemes also require accurate knowledge of node locations. However, simple localization solutions, such as equipping each node with a GPS receiver or manual configuration using coordinates [8, 18–20], are either energy-expensive or impractical for WSNs in some cases [21].

To overcome the drawbacks, we propose a DElaunay-based Coordinate-free Mechanism (DECM) for full coverage. Here, "coordinate-free" means that each node does not need to know the exact locations but the distances and angles (i.e., relative locations) of all nearby nodes. In this paper, we first present a mathematical model that provides a sufficient and necessary condition for the full coverage of a triangle that has no other nodes inside its circumcircle. Accordingly, DECM first conducts triangulation on a WSN to form such triangles and then conducts node movements to meet the condition in order to minimize the number of node movements for full coverage. The time complexity for building triangulation is $O(bn)$, where b denotes the number of sensors in the vicinity of each node and n denotes the total number of nodes in the network. DECM is a distributed scheme in which every node checks for holes and makes movements to heal holes. As shown in Fig. 1, each node first conducts Delaunay triangulation that divides the target field into triangles that have no other nodes inside. Then, based on the sufficient and necessary condition, each node calculates its safe area where the node can be located while still keeping full coverage of its triangles. Based on the calculated safe areas of each node, DECM can detect coverage holes and find the shortest paths for node movement to heal the holes. DECM utilizes the r-map coordinate system [2] to enable nodes to know the relative locations of nearby nodes for hole detection and healing. DECM is similar to a VOR based method since both construct a diagram for hole healing. However, VOR may generate numerous iterations of node movements because (1) node movement to cover one point rather than the cell area may generate holes in other points in the cell, and (2) only one node is in charge of the coverage of a cell. Unlike in VOR, a node in DECM moves to a point in order

- * *Corresponding Author. Email: shenh@clemson.edu; Phone: (864) 656 5931; Fax: (864) 656 5910.*
- *Chenxi Qiu and Haiying Shen are with the Department of Electrical and Computer Engineering, Clemson University, Clemson, SC, 29634. E-mail: {chenxiq, shenh}@clemson.edu*

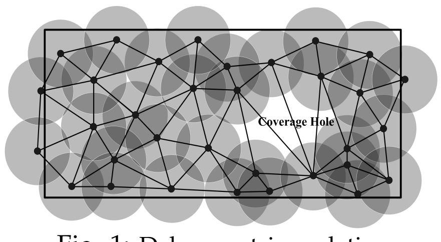


Fig. 1: Delaunay triangulation.

to cover the entire area of a Delaunay cell. Also, three nodes are in charge of the coverage of a cell. Thus, DECM achieves full coverage more quickly and energy-efficiently. Our experiment results show that to heal the coverage holes of a target region, compared to previous methods, DECM saves at least 26% total moving distance of all the nodes, at least 32% total number of moves. Furthermore, DECM enhance the lifetime of the network at least 2.5 times.

This paper is organized as follows. Section 4 presents a review of related works on movement-assisted schemes for full WSN coverage deployment. Section 2 introduces mathematical models for analyzing WSN coverage problems and presents the DECM movement-assisted scheme for detecting and healing coverage holes based on this model. Then, Section 3 presents a performance evaluation of DECM in comparison with several previous schemes. The final section concludes with a summary of contributions and a discussion on further research work.

2 The Design of DECM

We consider a WSN comprised of a set of nodes $\mathcal{S} = \{s_1, s_2, s_3, ..., s_n\}$ that are uniformly distributed over a large *target field* and are designed to detect specified events. Each node, say s_i, can sense specified events in its *sensing range*, denoted by R_{s}. We use $\mathcal{P} = \{P_1, P_2, P_3, ..., P_n\}$ to represent the locations of all the nodes in $\mathcal{S}$. Like many previous works in [8, 9, 11, 17], we model the sensing range of each node as a disk. Extending DECM to complex sensing range models leaves as our future work [22, 23]. In addition, we consider a time slotted system where each node can successfully receive a packet within a time slot [24, 25].

We first find the condition for full coverage of a triangle formed by three sensors with no other nodes inside the triangle's circumcircle (Section 2.1). Since a Delaunay triangle has no other nodes inside the triangle's circumcircle, nodes conduct Delaunay triangulation [26] to divide the field into Delaunay triangles (Section 2.2). Then, each node uses our observed full coverage condition to find a safe area where it can be located while still keeping full coverage of its triangles (Section 2.3). Finally, each node can detect holes and discover the shortest path for its movement to heal holes (Section 2.4). Nodes in DECM periodically conduct Delaunay triangulation, hole detection and node movement, and are synchronized by the diffusion-based synchronization algorithm in [27], which is a distributed method. DECM is proposed for WSNs where the node density is high enough for Delaunay triangulation and that the number of node movement iterations for hole healing is limited.

2.1 Condition for A Triangle's Full Coverage

Consider three nodes in a plane that construct a triangle. The sufficient and necessary condition for the triangle's full coverage is described in Theorem 2.1.

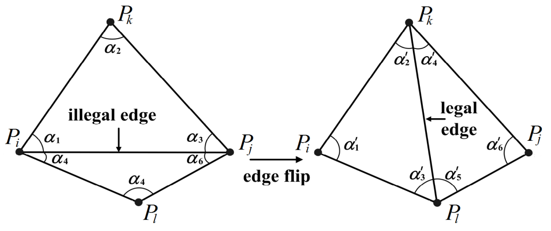


Fig. 2: Illegal edge and edge flip.

Theorem 2.1: Consider a triangle formed by three nodes s_i, s_j and $s_k \in \mathcal{S}$ (s_i, s_j and s_k are located at point P_i, P_j and P_k respectively) with no other nodes placed inside the triangle's circumcircle. Using d_{ij} to denote the Euclidean distance between s_i and s_j, and with the same convention for the other distances, we derive: 1) when the triangle is an acute triangle, the triangle is fully covered iff the following condition is satisfied:

$$R_s \geq \frac{d_{ij}d_{jk}d_{ik}}{\sqrt{\left(d_{ij}^2 + d_{ik}^2 + d_{jk}^2\right)^2 - 2\left(d_{ij}^4 + d_{ik}^4 + d_{jk}^4\right)}} \tag{1}$$

2) when the triangle is an obtuse triangle, the triangle is fully covered iff the following condition is satisfied:

$$R_s \geq \max\left\{\frac{d_{ij}^2 d_{jk}}{d_{ij}^2 + d_{jk}^2 - d_{ik}^2}, \frac{d_{ik}^2 d_{jk}}{d_{ik}^2 + d_{jk}^2 - d_{ij}^2}\right\} \tag{2}$$

The proof of Theorem 2.1 is provided in the supplemental file (Section 6.1). Based on Theorem 2.1, two objectives should be achieved for the full coverage of a WSN's target area while minimizing the number of node movements: (1) the area is partitioned into triangles with no other nodes inside each triangle's circumcircle, and (2) each triangle meets the condition in Theorem 2.1 by node movements. Note that objective (1) is only used to minimize the number of node movements for full coverage. For the triangles having other nodes inside their circumcircles, the condition in Theorem 2.1 is sufficient for their full coverage and the other nodes may already cover the triangle. For objective (1), we can leverage the Delaunay triangulation [26], which is widely used in mathematics and computational geometry.

2.2 Coordinate-free Delaunay Triangulation

The Delaunay triangulation [26] is used in mathematics and computational geometry.

Definition 1 (*Delaunay triangulation*) [26] A triangulation for a set $\mathcal{P}$ of points in a plane is a Delaunay triangulation if no point in $\mathcal{P}$ is inside the circumcircle of any triangle.

Based on Theorem 2.1, we first conduct Delaunay triangulation so that there are no other nodes placed inside each triangle's circumcircle. Before we present how to conduct Delaunay triangulation on a WSN, we first introduce some definitions and theorems [26].

Definition 2 (*edge flip*) [26]: As Fig. 2 shows, consider an edge $e_{ij} = P_iP_j$ of a triangulation. If e_{ij} is not an edge of the unbounded face, it is incident to two triangles $\triangle P_iP_jP_k$ and $\triangle P_iP_jP_l$. If the two triangles form a convex quadrilateral, we can obtain a new triangulation $\mathcal{T}'$ by removing e_{ij} and inserting e_{kl} (P_kP_l) in triangulation $\mathcal{T}$. We call this operation an *edge flip*.

Definition 3 (*illegal edge*) [26]: As Fig. 2 shows, after an edge flip, the only difference between $\mathcal{T}$ and $\mathcal{T}'$ is that the six angles $\alpha_1, ..., \alpha_6$ are replaced by $\alpha'_1, ..., \alpha'_6$. We call the edge e_{ij} an *illegal edge* if

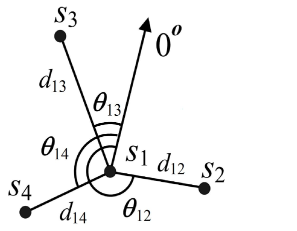


Fig. 3: Cooperative movement.

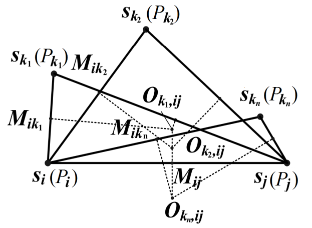


Fig. 4: An example of preventing new holes.

$$\min(\alpha_i) < \min(\alpha_i') \quad (1 \le i \le 6) \tag{3}$$

Definition 4 (*legal triangulation/triangle*) [26]: A triangulation/triangle without any illegal edge.

Theorem 2.2: A triangulation for a set $\mathcal{P}$ of points in a plane is a legal triangulation iff the triangulation is a Delaunay triangulation [26].

The Delaunay triangulation and hole healing require each node to obtain the distances and directions of nearby nodes. For this purpose, DECM uses the r-map system, in which each node measures the locations of its neighbors using its own arbitrary polar-axis.

Definition 5 (r*-map* [2]): The r-map of node s_i is a variant of polar coordinates that specifies the relative location of its r-vicinity, denoted by $N_{s_i}(r)$. The location of any node $P_j \in N_{s_i}(r)$ is presented as (d_{ij}, θ_{ij}), where d_{ij} is the *radial coordinate* that indicates the Euclidian distance between nodes s_i and s_j, and θ_{ij} is the *angular coordinate* of node s_j that denotes the direction of s_j relative to an *arbitrary polar-axis* of s_i.

$P_j = (d_{ij}, \theta_{ij})$ in node s_i's r-map is called the relative location of s_j to s_i. For example, in Fig. 3, the relative locations of s_2, s_3, and s_4 to s_1 are (d_{12}, θ_{12}), (d_{13}, θ_{13}) and (d_{14}, θ_{14}). The concept of coordinate-free r-map was firstly introduced in [2]. All a node needs to know are its distances among its nearby nodes (e.g., through signal strengths), from which it can calculate the angles using cosine/sine law. Building r-map is much less expensive than obtaining accurate node locations that needs GPS receivers or a manual configuration of each node with its coordinates.

Each node periodically exchanges its measured r-map with their neighbors by broadcasting a packet to their neighbors. Each packet will be retransmitted if it cannot be correctly received by its receiver. When s_i receives s_j's r-map containing the location of s_k measured by s_j, s_i can transform s_k's location to the location measured by s_i's polar-axis using the methods in [2, 28]. By multi-hop transmission nodes around a hole can get the r-map information from the remote nodes which are on "the other side" of the hole.

Definition 6 (*shared nodes*): Shared nodes of s_i and s_j are the nodes of which locations exist in both s_i's r-map and s_j's r-map.

Definition 7 (*potential edge*): A potential edge is an edge that has at least one side with no constructed triangle and with at least one shared node.

We consider a Delaunay triangulation as a time-slotted process where time spot $t = 1, 2, 3, ...$. We use s_{seed} to denote the seed node. When a Delaunay triangulation starts at $t = 1$, DECM broadcasts a triangulation starting notification to all nodes and randomly selects a node as s_{seed}. The s_{seed} finds the nearest node in its r-map, say s_j, and builds an edge called potential edge to s_j, denoted by e_{ij}. s_i stores e_{ij} into its Delaunay triangulation table (DT-table, denoted by DT_i) and also asks s_j to store e_{ij} into s_j's DT-table, denoted by DT_j. Then, the two nodes s_i and s_j (we call the two nodes *edged nodes*) choose a nearby node, say s_k, to form a triangle so that the triangles circumcircle is minimized in both sides. We call s_k the nearest triangle neighbor (NTN) of the edged nodes or the edge. As shown in Fig. 2, an edge $e_{ij} = s_is_j$ can be used for building one triangle on each of its two sides. Suppose the formed triangle with the minimum circumcircle is $\triangle s_is_js_k$ ($\triangle P_iP_jP_k$). After two nodes are connected by a new edge (e.g., s_is_k and s_js_k), they check if the new edge is a potential edge. If so, the edged nodes conduct the same operation by choosing their NTN to construct a new triangle with the minimum circumcircle. This process continues until a newly added edge intersects with an existing edge, which means the triangulation is completed, or t reaches TTL_1 (Time To Live), which is an appropriate time period set by DECM for each node to conduct the triangulation process. The requirement of the minimum circumcircles for constructed triangles aims to build triangles with no other nodes in their circumcircles.

To find the NTN on one side of edge e_{ij}, two edged nodes, s_i and s_j, communicate with each other to determine their shared nodes' location, denoted by $N_{s_j(r)} \cap N_{s_i(r)}$. Suppose $P_{k_1}, P_{k_2}, ..., P_{k_m} \in N_{s_j(r)} \cap N_{s_i(r)}$. As shown in Fig. 4, for one side of e_{ij}, all nodes $s_{k_1}, s_{k_2}, ..., s_{k_m}$ are connected with s_i (or s_j), thus generating edges $e_{k_1i}, e_{k_2i}, ..., e_{k_mi}$ (or $e_{k_1j}, e_{k_2j}, ..., e_{k_mj}$). These edges' perpendicular bisectors intersect with the perpendicular bisector of e_{ij} at point $O_{k_1,ij}, O_{k_2,ij}, ..., O_{k_m,ij}$, respectively. We use variable $h_{k,ij}$ to denote the distance between $O_{k,ij}$ and e_{ij}, i.e., $h_{k,ij} = |M_{ij}O_{k,ij}|$, where M_{ij} is the middle point of e_{ij}. If $O_{k,ij}$ and s_k (P_k) are located at the same side of e_{ij}, then $h_{k,ij} > 0$; and if $O_{k,ij}$ and s_k (P_k) are located at different sides, then $h_{k,ij} < 0$. Suppose $h_{k_n,ij}$ has the smallest value among $h_{k,ij}$; then s_{k_n} is the NTN of e_{ij}. Algorithm 1 in the supplementary file (in Section 7) shows the pseudo code for searching NTN, where $\text{InterOfBisector}(e_{ij}, e_{ik_l})$ returns the intersection of e_{ij} and e_{ik_l}'s perpendicular bisectors.

As shown in Fig. 2, an edge $e_{ij} = P_iP_j$ can be used for building one triangle on each of its two sides. Suppose the formed triangle is $\triangle P_iP_jP_k$. When two nodes are connected by a new edge (e.g., e_{ik} and e_{jk}), if the new edge is a potential edge, the edged nodes conduct the same operation by choosing their NTN to construct a new triangle with the minimum circumcircle.

Each node s_i stores the triangles it belongs to in its *Delaunay triangulation table* (DT-table), denoted as DT_i. This process continues until a newly added edge intersects with an existing edge, which means the triangulation is completed. However, it is difficult to globally notify all nodes of completion. Also, a node isolated from the others may keep looking for nodes to build triangle edges. Thus, DECM sets an appropriate time period TTL_1 for each node to conduct the triangulation process. After TTL_1, each node checks to see if an *illegal edge* exists in the triangles in its DT-table, and conducts an *edge flip* if an *illegal edge* exists. This step is to ensure that Delaunay triangulation is formed according to Theorem 2.2. There should not be many *illegal edges* because:

Theorem 2.3: When two edged nodes connect to their NTN to construct a triangle, there are no other nodes inside the triangle's circumcircle on the same side as the NTN.

The proof of Theorem 2.3 is provided in the supplemental file (section 6.2). Then, after conducting triangulation for the specified time period, each node checks if an illegal edge exists in the triangles in its DT-table, and conducts an edge flip if an illegal edge exists. This step aims to realize Delaunay triangulation according to Theorem 2.3. Note that two edged nodes are not necessarily neighbors. Nodes can communicate with each other by multi-hop routing [5]. Though two edged nodes calculate their NTN individually, they will find the same NTN result when their r-maps include their nearby nodes. Also, because only one NTN exists on one side of a potential edge of two edged nodes, it is impossible that a new edge will intersect with an existing edge of the two nodes. Note that given a potential edge, there may exist "multiple" NTNs, which are all on the same circle with the two vertices of the potential edge. In this case, we only select one node as the NTN from this node, as shown in Algorithm 1.

The Delaunay triangulation can take $O\left(n^2\right)$ edge flips even if all node locations are globally known [26]. A local edge flip might generate new illegal triangles, and then the computation might be endless, though the probability of such an event is low according to Theorem 2.3. To avoid endless edge flipping, we set TTL_2 for each sensor node. Once the TTL_2 expires, a node stops flipping edges even though it finds illegal triangles. The TTL_2 strategy does not prevent DECM from achieving full coverage but might generate unnecessary node movements. The strategy might reduce the accuracy of Delaunay triangulation since there could exist some illegal triangles. That is, some other node besides the triangle's three nodes could cover the circumcircle's center. Then, the coverage holes that the three sensor nodes have detected might be covered by some other nodes, leading to unnecessary node movements. Let b denote the number of sensors in the vicinity for each node, let T denote the length of TTL_2. Because each potential edge needs to find its NTN, which use $O(b)$ time, and there are totally $O(n)$ potential edges, the complexity of triangulation is $O(nb)$. In the process of edge flips, each node checks whether the triangles it forms are illegal in each time slot. There are totally $O(n)$ triangles and t time slots, hence the complexity of edge flips is $O(nT)$. Because we always set T as a constant, the complexity of our distributed algorithm is $O(nb)$. Specifically, in conducting Delaunay triangulation on a WSN, each node s_i executes Algorithm 2 in the supplementary file (in Section 7). The details of EdgeFlip operation is introduced in [26].

2.3 Safe Area Detection

Definition 8 (*safe area*): Suppose edge e_{ij} is an edge in s_k's DT-table, the *safe area of* s_k *for* e_{ij}, denoted by A_{ij}^k, is defined as the area where s_k can be located without breaking the full coverage of $\triangle P_iP_jP_k$ ($\triangle s_is_js_k$). The *safe area of* s_k, denoted by A^k, is defined as the intersection of the safe areas of s_i for $\forall e_{ij} \in \text{DT}_k$:

$$A^k = \bigcap_{\forall e_{ij} \in \text{DT}_k} A_{ij}^k \tag{4}$$

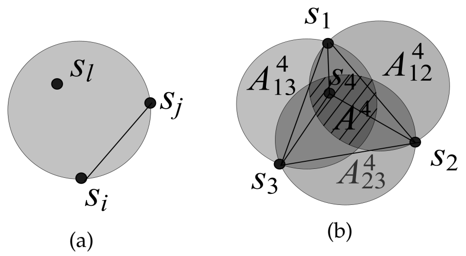


Fig. 5: Examples of a safe area.

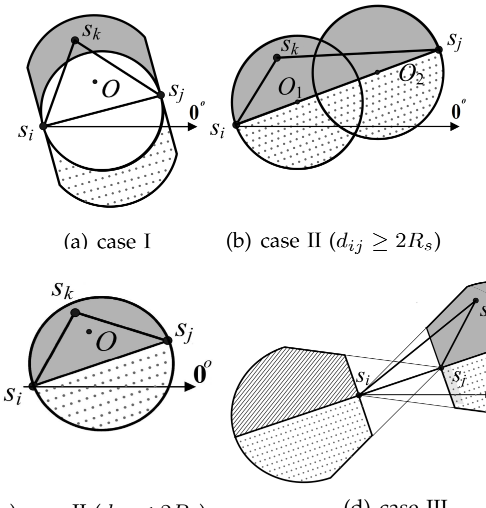


Fig. 6: Node s_k's *safe area* for e_{ij}.

Fig. 5 gives an example for the definition of a safe area. In Fig. 5 (a), the shadow area is the safe area of s_l to e_{ij}, within which s_l stays to ensure the coverage of $\triangle s_is_js_l$. In Fig 5 (b), the shadow area in the middle is the safe area of s_4, which is the intersection of the safe areas of s_4 to e_{12}, e_{23}, e_{13}. If s_4 is out of this area, there will be a coverage hole in $\triangle s_1s_2s_4$, $\triangle s_2s_3s_4$, or $\triangle s_1s_3s_4$. Note that the safe area of a node may be an empty set, which implies that there must be a hole unable to be fixed no matter how this node moves. In this case, the node uses cooperative movement mechanism, in which it requests its nearby nodes to move to fix the holes. The details of this mechanism will be presented in Section 2.5.

In the following, we first calculate A_{ij}^k, and then calculate A^k by Equ. (4). We assume that s_i's r-map contains s_j and s_k and use s_i's r-map for the calculation. Using the r-maps of s_j or s_k can retrieve the same results. First, we assume that $\theta_{ik} \in [\theta_{ij},\ \theta_{ij}+\pi)$. We will discuss the situation when $\theta_{ik} \in [0,\ \theta_{ij}) \cup [\theta_{ij}+\pi,\ 2\pi)$ later on. There are three different kinds of triangle shapes for $\triangle s_is_js_k$:
Case I: an acute triangle (Fig. 6 (a));
Case II: an obtuse triangle with $d_{ij} > \max\{d_{ik},\ d_{jk}\}$ (Fig. 6 (b) and (c));
Case III: an obtuse triangle with $d_{ij} < \max\{d_{ik},\ d_{jk}\}$ (Fig. 6 (d)).

Case I When $\triangle s_is_js_k$ is an acute triangle: As Fig. 6 (a) shows, d_{ij}, d_{jk}, and d_{ik} must satisfy $d_{ij}^2 + d_{ik}^2 > d_{jk}^2$, $d_{ij}^2 + d_{jk}^2 > d_{ik}^2$, and $d_{ik}^2 + d_{jk}^2 > d_{ij}^2$, which can be induced to:

$$d_{ij} - d_{ik}\cos(\theta_{ik} - \theta_{ij}) > 0 \quad \text{and} \quad \theta_{ij} < \theta_{ik} < \theta_{ij} + \frac{\pi}{2} \tag{5}$$

To guarantee the full coverage of the triangle, Equ. (1) in Theorem 2.1 must be satisfied, which can be simplified to:

$$\left(\frac{d_{ik}\cos\theta'_{ik}\,(d_{ij}-d_{ik}\cos\theta'_{ik})}{d_{ik}\sin\theta'_{ik}}-d_{ik}\sin\theta'_{ik}\right)^2+d_{ij}^2\le 4R_s^2$$

where $\theta'_{ik}=\theta_{ik}-\theta_{ij}$. From Equ. (6), we can derive $d_{ij}<2R_s$ and

$$\left(d_{ik}\cos\theta'_{ik}-\frac{d_{ij}}{2}\right)^2+\left(d_{ik}\sin\theta'_{ik}+\frac{\sqrt{4R_s^2-d_{ij}^2}}{2}\right)^2\ge R_s^2$$

$$\left(d_{ik}\cos\theta'_{ik}-\frac{d_{ij}}{2}\right)^2+\left(d_{ik}\sin\theta'_{ik}-\frac{\sqrt{4R_s^2-d_{ij}^2}}{2}\right)^2\le R_s^2 \qquad (6)$$

where $\theta'_{ik}=\theta_{ik}-\theta_{ij}$. Therefore, (7)

Lemma 2.1: $d_{ij}<2R_s$ is a necessary condition for the full coverage of an acute triangle (Case I). The area formed by Equ. (5), Equ. (6), and Equ. (7) is the *safe area* of s_k for e_{ij}, which is the gray area in Fig. 6 (a).

Case II When $\triangle s_is_js_k$ is an obtuse triangle with $d_{ij}>max\{d_{ik},\ d_{jk}\}$: As Fig. 6 (b) shows, d_{ij}, d_{jk}, and d_{ik} must satisfy $d_{ij}^2\ge d_{jk}^2+d_{ik}^2$, which can be simplified to:

$$d_{ik}\le d_{ij}\cos(\theta_{ik}-\theta_{ij}) \qquad (8)$$

To guarantee full coverage of the triangle, Equ. (2) in Theorem 2.1 must be satisfied. We discuss the problem in two cases:

Case II.1 When $d_{ik}\ge d_{jk}$, Equ. (2) can be induced to:

$$\frac{d_{ik}}{\cos(\theta_{ik}-\theta_{ij})}\le 2R_s. \qquad (9)$$

The area formed by Eqs. (8) and (9) is the safe area *of s_k for e_{ij}, which is the gray area in Fig. 6 (b).*

We notice that when $d_{ij}<2R_s$, if Equ. (8) is satisfied, Equ. (9) is automatically satisfied, indicating that the triangle is fully covered. This is because:

$$\frac{d_{ik}}{\cos(\theta_{ik}-\theta_{ij})}\le d_{ij}<2R_s\ \rightarrow\ \frac{d_{ik}}{\cos(\theta_{ik}-\theta_{ij})}\le 2R_s \qquad (10)$$

Case II.2 When $d_{ik}<d_{jk}$, Equ. (2) can be reduced to:

$$\frac{d_{ik}^2-2d_{ik}d_{ij}\cos(\theta_{ik}-\theta_{ij})+d_{ij}^2}{d_{ij}-d_{ik}\cos(\theta_{ik}-\theta_{ij})}\le 2R_s \qquad (11)$$

Similarly, we notice that when $d_{ij}<2R_s$, if Equ. (8) is satisfied, Equ. (11) is automatically satisfied, indicating that the triangle is fully covered. This is because Equ. (8) can be reduced to

$$d_{ik}^2-2d_{ik}\cos\theta'_{ik}d_{ij}+d_{ij}^2\le d_{ij}^2-d_{ik}\cos\theta'_{ik}d_{ij}, \qquad (12)$$

where $\theta'_{ik}=\theta_{ik}-\theta_{ij}$. Because $d_{ij}-d_{ik}\cos(\theta_{ik}-\theta_{ij})>0$ in Equ. (8), Equ. (12) can be reduced to

$$\frac{d_{ik}^2-2d_{ik}\cos(\theta_{ik}-\theta_{ij})d_{ij}+d_{ij}^2}{d_{ij}-d_{ik}\cos(\theta_{ik}-\theta_{ij})}<d_{ij}<2R_s \qquad (13)$$

Thus, Equ. (11) is satisfied. Accordingly,

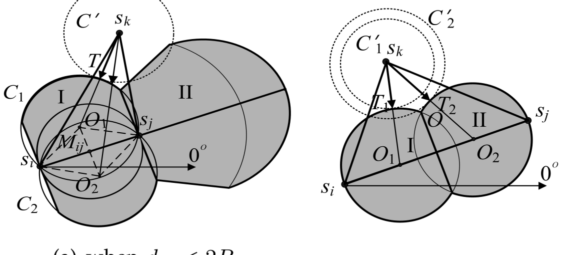
Lemma 2.2: For an obtuse triangle with $d_{ij}>max\{d_{ik},\ d_{jk}\}$ (Case II), when $d_{ij}<2R_s$, Equ. (8) is a sufficient condition for the full coverage of the triangle, and the area formed by Equ. (8) is the *safe area* of s_k for e_{ij}, which is the gray area in Fig. 6 (c); when $d_{ij}\ge 2R_s$, the area formed by Equ. (8), Equ. (9), and Equ. (11) is the *safe area* of s_k for e_{ij}, which is the gray area in Fig. 6 (b).

Case III When $\triangle s_is_js_k$ is an obtuse triangle with $d_{ij}<\max\{d_{ik},\ d_{jk}\}$: This case is further discussed in the following two cases:

Case III.1 When $d_{ik}>\max\{d_{ij},\ d_{jk}\}$: As Fig. 6 (d) shows, d_{ij}, d_{jk}, and d_{ik} must satisfy $d_{ik}^2>d_{ij}^2+d_{jk}^2$, which can be reduced to:

$$d_{ik}\cos(\theta_{ik}-\theta_{ij})>d_{ij} \qquad (14)$$

where $\cos(\theta_{ik}-\theta_{ij})>0$ because $d_{ik}>d_{jk}$. To guarantee full coverage of the triangle, Equ. (2) in Theorem 2.1 should be satisfied. We discuss the problem in two cases:



(a) when $d_{ij}<2R_s$ (b) when $d_{ij}\ge 2R_s$

Fig. 7: Shortest path in node movement for hole healing.

$$\frac{d_{ij}}{2\cos(\theta_{ik}-\theta_{ij})}\le R_s \text{ and } \frac{d_{ik}}{2\cos(\theta_{ik}-\theta_{ij})}<d_{ij} \qquad (15)$$

From Eqs. (14) and (15), we can infer that:

$$\frac{d_{ij}}{2R_s}\le\cos(\theta_{ik}-\theta_{ij})<1\rightarrow d_{ij}<2R_s \qquad (16)$$

Case III.1.2 When $d_{ik}>d_{jk}>d_{ij}$: From Equ. (2), the *safe area* of s_k for e_{ij} can be calculated as:

$$\frac{d_{ik}^2-2d_{ik}d_{ij}\cos(\theta_{ik}-\theta_{ij})+d_{ij}^2}{2(d_{ik}-d_{ij}\cos(\theta_{ik}-\theta_{ij}))}\le R_s \qquad (17)$$

$$\frac{d_{ik}}{2\cos(\theta_{ik}-\theta_{ij})}>d_{ij} \qquad (18)$$

Recall that $d_{ij}<2R_s$ when $d_{ij}>d_{jk}$. Similarly, we can derive that $d_{jk}<2R_s$ when $d_{jk}>d_{ij}$. Then,

$$d_{ij}<d_{jk}<2R_s\rightarrow d_{ij}<2R_s \qquad (19)$$

From Eqs. (16) and (19), we know that

Lemma 2.3: $d_{ij}<2R_s$ is a necessary condition for the full coverage of an obtuse triangle with $d_{ik}>\max\{d_{ij},d_{jk}\}$ (Case III). The area formed by Equ. (14), Equ. (15), and Equ. (17) is the *safe area* of s_k for e_{ij}, which is the gray area in Fig. 6 (c).

Case III.2 When $d_{jk}\ge\max\{d_{ij},\ d_{ik}\}$: If we use s_j's r-map to calculate s_k's *safe area* for e_{ij} in the obtuse triangle, because this is symmetrical to Case III.1, the calculated s_k's *safe area* for e_{ij} is similar to that of case III.1 shown in the slashed area of Fig. 6 (d). In other cases, using s_j's r-map to calculate s_k's *safe area* for e_{ij} leads to exactly the same result. When $\theta_{ik}\in[0,\ \theta_{ij})\cup[\theta_{ij}+\pi,\ 2\pi)$, it only results in a safe area that is symmetrical to the area when $\theta_{ik}\in[\theta_{ij},\ \theta_{ij}+\pi)$ as shown in the dotted and gray areas in Fig. 6 (a)-(d).

From Lemmas 2.1, 2.2, and 2.3, we know that when s_i looks for the safe area of s_k for e_{ij} for full coverage of $\triangle s_is_js_k$, given d_{ij}, if $d_{ij}\ge 2R_s$, s_k's safe area for e_{ij} is only Fig. 6 (b) in Case II, which is re-drawn in Fig. 7 (b). If $d_{ij}<2R_s$, s_k's safe area for e_{iij} can be Fig. 6 (a), (c), and (d) in Case I, II, and III, the combination of which is shown in Fig. 7 (b).

Theorem 2.4: When s_i checks the full coverage of $\triangle P_iP_jP_k$, given d_{ij} when $d_{ij}\ge 2R_s$, the gray area in Fig. 7 (b) is the safe area of s_k for e_{ij}; otherwise, the gray area in Fig. 7 (a) is the safe area of s_k for e_{ij}.

If every node only has the relative location information of other nodes, it would be difficult for the system to detect holes on the edge of the target region [2]. There are several ways to solve this problem, such as configuring nodes before placing them. In our system, we deploy a small number of static nodes on the edge of the region as *anchor nodes* to help detect holes.

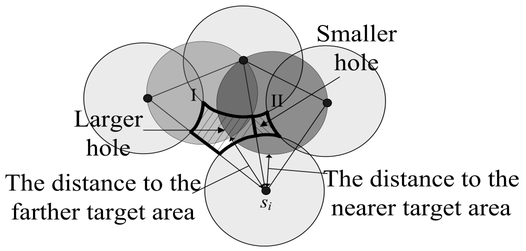

Smaller hole
Larger hole
I
II
The distance to the farther target area
The distance to the nearer target area
s_i

Fig. 8: A farther target area means a larger hole.

2.4 Shortest-path Movement

The hole coverage in DECM is composed of three steps: 1) each node detects coverage holes and notifies the nodes who are responsible for healing the holes; 2) each node selects a target area (i.e., safe area) it needs to move to when receiving multiple notifications; 3) each notification receiver identifies the shortest path to its selected target area. In the following, we describe each step with the focus on the third step.

Step 1. Recall that after the Delaunay triangulation, each node stores the nodes that are in the same triangle as itself in its *DT-table*. We call these nodes *triangle neighbors (TN)* of the node. Each node calculates the safe areas of its triangle neighbors, checks to see if they are in their own safe areas using its r-map based on Theorem 2.4, and then notifies those not in their own safe areas.

Step 2. After hole detection, a node may receive multiple notifications, which provide different target areas for this node to move to. As indicated in [3], a hole father (with a longer distance) to the node means a larger hole around the node. The "distance" here means the shortest path a node needs to move to reach the target area of the hole. As shown in Fig. 8, s_i has two target areas I and II and area I is farther to s_i than area II. We see s_i can fix a larger hole if it moves to area I. Therefore, as VOR [3], , DECM also let each node move to the farthest target area in order to achieve full coverage more rapidly.

Step 3. Suppose that s_k receives the notification from s_i along with d_{ij} and s_k's *safe area* for e_{ij}. s_k then calculates its shortest path to the target area (i.e., *safe area*) for e_{ij}. Similar to hole detection, s_k needs to consider two cases according to d_{ij}.

Case I When $d_{ij} < 2R_s$: As Fig. 7 (a) shows, the *safe area* for s_k is composed of two parts: area I and area II. From Equ. (7) we can get the polar coordinates of C_1's center O_1 and C_2's center O_2: $(R_s,\ \arccos\left(\frac{s_{ij}}{2R_s}\right)+\theta_{ij})$ and $(R_s,\ 2\pi-\arccos\left(\frac{s_{ij}}{2R_s}\right)-\theta_{ij})$. s_k has two choices: (1) constructing an acute triangle (moving towards area I), and (2) constructing an obtuse triangle (moving towards area II). We only let nodes contruct acute triangles because this makes the node positions more similar to the vertices of the static hexagon based permutation, which is considered the best for node distribution [29]. To find the shortest path to region I, using itself as the circle center, s_k draws a circle C', which generates a point of tangent between C' and region I, denoted T; then s_kT is the shortest path. T is on the line of s_kO_1 if $\theta_{ik} \in [\theta_{ij},\ \theta_{ij}+\pi)$ and is on the line of s_kO_2 if $\theta_{ik} \in [0,\ \theta_{ij}) \cup [\theta_{ij}+\pi,\ 2\pi)$. Thus, to cover the hole, s_k moves towards point O_1 or O_2, whichever is nearer, and stops once it enters its safe area. As a result, it moves along the shortest path, the length of which is:

$$s_kT = \sqrt{\left(x_{ik}-\frac{d_{ij}}{2}\right)^2+\left(y_{ik}-\frac{\sqrt{4R_s^2-d_{ij}^2}}{2}\right)^2} - R_s$$

where $x_{ik} = d_{ik}\cos(\theta_{ik}-\theta_{ij})$, $y_{ik} = d_{ik}\sin(\theta_{ik}-\theta_{ij})$.

Node s_k can independently calculate the coordinates of O_1 and O_2 in its own r-map. We notice that the polar coordinates of point O_1 and O_2 must be positioned at s_i and s_j's perpendicular bisector and that $O_1s_i = O_2s_i = R_s$. Accordingly, the coordinates of O_1 in s_k's r-map, $(d_{kO_1}, \theta_{kO_1})$, can be calculated by:

$$d_{kO_1} = \sqrt{(A+B)^2+(C+D)^2} \tag{20}$$

$$\theta_{kO_1} = \begin{cases} \arctan\left(\frac{C+D}{A+B}\right) & \text{if } C+D \geq 0 \\ \arctan\left(\frac{C+D}{A+B}\right)+\pi & \text{if } C+D < 0 \end{cases} \tag{21}$$

The coordinates of O_2 in s_k's r-map, $(d_{kO_2}, \theta_{kO_2})$, can be calculated by:

$$d_{kO_2} = \sqrt{(A-B)^2+(C-D)^2} \tag{22}$$

$$\theta_{kO_1} = \begin{cases} \arctan\left(\frac{C-D}{A-B}\right) & \text{if } C-D \geq 0 \\ \arctan\left(\frac{C-D}{A-B}\right)+\pi & \text{if } C-D < 0 \end{cases} \tag{23}$$

where

$$A = \frac{d_{ki}\cos\theta_{ki}+d_{kj}\cos\theta_{kj}}{2},\ C = \frac{d_{ki}\sin\theta_{ki}+d_{kj}\sin\theta_{kj}}{2}$$

$$B = \frac{\sqrt{R_s^2-\left(\frac{d}{2}\right)^2}\,(d_{kj}\sin\theta_{kj}-d_{ki}\sin\theta_{ki})}{d}$$

$$D = \frac{\sqrt{R_s^2-\left(\frac{d}{2}\right)^2}\,(d_{ki}\cos\theta_{ki}-d_{kj}\cos\theta_{kj})}{d}$$

$$\text{and } d = \sqrt{d_{ki}^2-2\cos(\theta_{ki}-\theta_{kj})d_{ki}d_{kj}+d_{kj}^2}$$

Based on the coordinates of O_1 and O_2, s_k chooses the one closer to itself as the target point, which leads to the shortest moving path.

Case II When $d_{ij} \geq 2R_s$: As Fig. 7 (b) shows, the polar coordinate of O_1 and O_2 in s_k's r-map are (R_s, θ_{ij}) and $(d_{ij}-R_s, \theta_{ij})$, respectively. s_k also has two choices: (1) moving towards region I, and (2) moving towards region II. Node s_k chooses the option that leads to the shorter moving path. Specifically, s_k produces the circle C_1', which has s_k as its center and has a tangent with region I at point T_1. s_k also produces the circle C_2', which has s_k as its center and has a tangent with region II at point T_2.

If $T_1s_k < T_2s_k$, the direction of s_k's movement is towards O_1 and the length of the shortest path T_1s_k is:

$$T_1s_k = \sqrt{d_{ik}^2-d_{ik}d_{ij}\cos l\,(\theta_{ik}-\theta_{ij})+\frac{d_{ij}^2}{4}} - R_s \tag{24}$$

If $T_1s_k \geq T_2s_k$, the direction of s_k's movement is towards O_2 and the length of the shortest path T_2s_k is:

$$T_2s_k = \sqrt{d_{ik}^2-2d_{ik}\cos\theta_{ik}'\left(R_s-\frac{d_{ij}}{2}\right)+\left(R_s-\frac{d_{ij}}{2}\right)^2} - R_s \tag{25}$$

where $\theta_{ik}' = \theta_{ik}-\theta_{ij}$. Similar to Case I, in Case II s_k can figure out the coordinates of O_1 and O_2 through its own r-map's information. Both O_1 and O_2 are located at s_is_j, and $O_1s_i = O_2s_j = R_s$. The coordinates of O_1 in s_k's r-map, $(d_{kO_1}, \theta_{kO_1})$, can be calculated by:

$$d_{kO_1} = \sqrt{A^2+B^2} \tag{26}$$

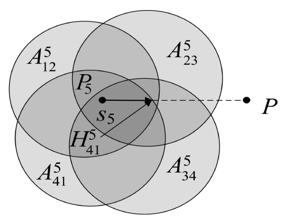


Fig. 9: An example of preventing new holes.

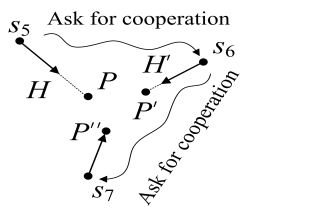


Fig. 10: Cooperative movement.

$$\theta_{kO_1} = \begin{cases} \arctan\left(\frac{B}{A}\right) & \text{if } B \geq 0 \\ \arctan\left(\frac{B}{A}\right) + \pi & \text{if } B < 0 \end{cases} \tag{27}$$

The coordinate of O_2 in s_i's r-map, $(d_{kO_2}, \theta_{kO_2})$, can be calculated by:

$$d_{kO_2} = \sqrt{C^2 + D^2} \tag{28}$$

$$\theta_{kO_2} = \begin{cases} \arctan\left(\frac{D}{C}\right) & \text{if } D \geq 0 \\ \arctan\left(\frac{D}{C}\right) + \pi & \text{if } D < 0 \end{cases} \tag{29}$$

where

$$A = d_{ki}\cos\theta_{ki} + \frac{R_s\left(d_{kj}\cos\theta_{kj} - d_{ki}\cos\theta_{ki}\right)}{d}$$
$$B = d_{ki}\sin\theta_{ki} + \frac{R_s\left(d_{kj}\sin\theta_{kj} - d_{ki}\sin\theta_{ki}\right)}{d}$$
$$C = d_{kj}\cos\theta_{kj} - \frac{R_s\left(d_{kj}\cos\theta_{kj} - d_{kj}\cos\theta_{kj}\right)}{d}$$
$$D = d_{kj}\sin\theta_{kj} - \frac{R_s\left(d_{kj}\sin\theta_{kj} - d_{ki}\sin\theta_{ki}\right)}{d}$$
$$\text{where } d = \sqrt{d_{ki}^2 - 2\cos\left(\theta_{ki} - \theta_{kj}\right)d_{ki}d_{kj} + d_{kj}^2}.$$

Based on the coordinates of O_1 and O_2, s_k moves towards the point closest to it, which leads to the shortest moving path.

2.5 Cooperative Movement

Though the above scheme can find the shortest path for a node to heal a hole, it would generate new holes if the node moves out of its safe area. To handle this problem, we propose a cooperative movement mechanism that can prevent generating new holes during node movements in healing holes. Basically, a node s_k does not move out of its safe area when moving to its destination and asks its triangle neighbors for cooperation by moving to cover s_k's uncovered area to heal the hole. We introduce the details of this mechanism as follows.

For example, in Fig. 9, node s_5's destination point P is outside of its safe area A^5; the intersection of A_{12}^5, A_{23}^5, A_{34}^5 and A_{41}^5. Node s_5's new destination is the closest point to P in A^5 on the line P_5P (s_5 is located at P_5), which is the intersection point between A^5 and line P_5P. We represent the new destination by P_{new} = intersection(A^k, P_5P). From the figure, we can see that H_{12}^5 is the new destination for s_5.

After node s_k moves to its newly calculated destination, it randomly chooses one from its triangle neighbors, calculates the destination of the neighbor and asks it to move in order to cover the hole. As Fig. 10 shows, s_5, s_6 and s_7 construct a triangle. Initially, s_5 receives a notification from s_6 (or s_7) to move to the destination point P. However, due to the constraint of its safe area, s_5 can only move to the point H. In order to cover its uncovered area in the hole, for each edge in its DT-table, s_5 randomly selects one from the two triangle neighbors of the edge. Assume s_6 is selected. s_5 then calculates the cooperation destination point (P_{cor}) for s_6 and sends a notification to s_6. s_6 checks whether this destination is inside its safe area. If yes, it directly moves to the destination; otherwise, it recalculates its new destination by calculating intersection(A^k, P_kP), moves to the new destination, calculates s_7's destination, and asks s_7 for cooperation. s_7 repeats s_6's operation steps, and asks s_5 for cooperation if it cannot move to its destination due to its own safe area. When s_5 receives moving notification for healing the same hole for the second time, it means that all three nodes cannot move enough distances to recover the hole. In this case, s_5 is forced to move out of its safe area to its destination, and stays at the point for a period of time that is predetermined. During this time period, other nodes nearby move to the newly generated holes. This process will continue until no new hole is generated. Because each node stays for a period when it moves out of its safe area, movement oscillation (nodes move back and forth) is prevented. Algorithm 3 in the supplementary file (in Section 7) describes the detailed progress of hole healing conducted by s_i, where TTL_3 is a predefined time period for each node to move to heal holes. Consider that when node s_k moves towards one hole, the area size of this hole is decreased; whereas if s_k 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 with the DECM's hole healing algorithm.

DECM significantly enhances the previous movement-based hole healing methods. Previous methods only let nodes move to cover a hold regardless if a new hole will be generated by the movement, thus leading to movement oscillation. With the cooperative movement scheme, DECM prevents generating new holes during node movements. Even if a node has to generate a new hole in order to recover another hole, after the movement, this node does not move for a period of time for other nodes to move to heal this new hole in order to prevent movement oscillation.

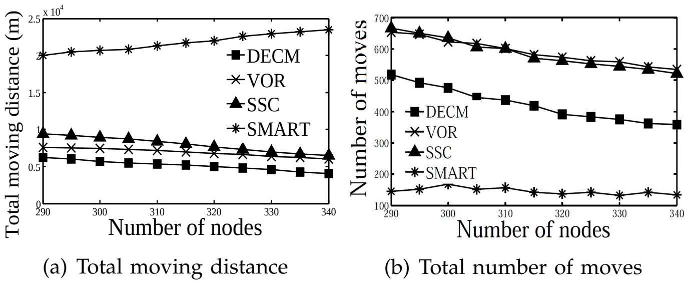


Fig. 11: Energy-efficiency of different schemes with different number of nodes (simulation).

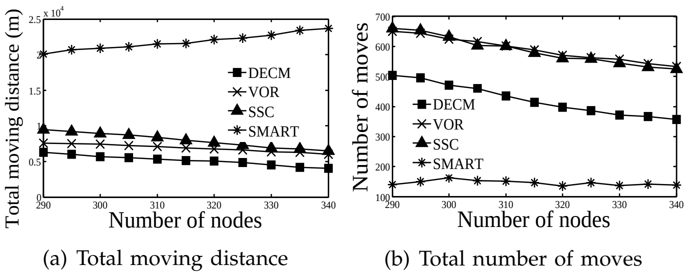
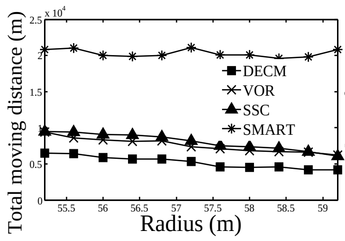


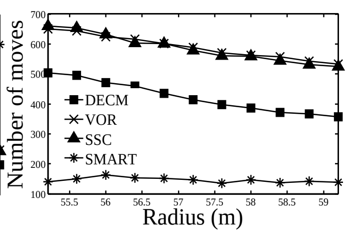
Fig. 12: Energy-efficiency of different schemes with different number of nodes (Orbit Testbed).

3 Performance Evaluation

In this section, we present the experimental results from the simulation and the experiments on GENI Orbit

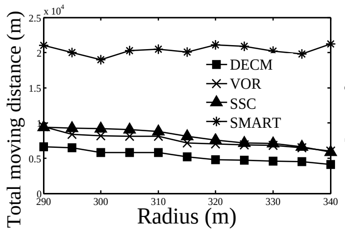


(a) Total moving distance

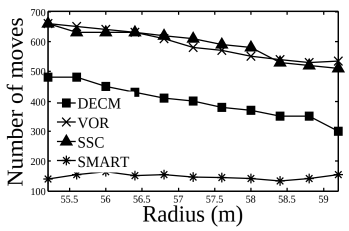


(b) Total number of moves

Fig. 13: Energy-efficiency of different schemes with different radii (simulation).

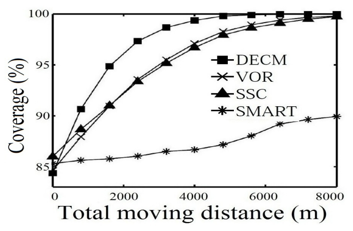


(a) Total moving distance

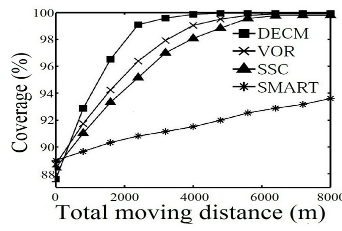


(b) Total number of moves

Fig. 14: Energy-efficiency of different schemes with different radii (Orbit Testbed).

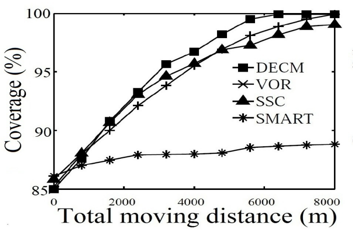


(a) The number of nodes is 300

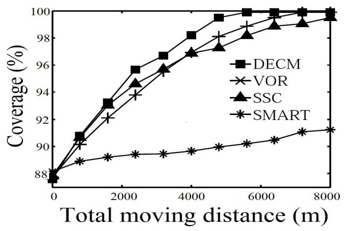


(b) The number of nodes is 330

Fig. 15: Efficiency of healing holes in different schemes (simulation).



(a) The number of nodes is 300



(b) The number of nodes is 330

Fig. 16: Efficiency of healing holes in different schemes (Orbit Testbed).

testbed [30]. The testbed uses a large two-dimensional grid of 400 802.11 radio nodes, which can be dynamically interconnected into specified topologies. We compared DECM with other three movement schemes for WSN full coverage: VORonoi-based algorithm (VOR) [3], Scan-based Movement-Assisted Sensor Deployment (SMART) [9] and Sea Surface Coverage (SSC) [17]. In SSC, nearby nodes can only move in four directions to inherit others' lost "interest points". Since the target region is not fully covered at the beginning in our simulation environment, we assume that every "interest point" in SSC is assigned to its nearest node. We compare DECM with DECM-S, in which each node selects the nearest target area, and DECM-R, in which each node always randomly selects a target area. We simulated the node movement by letting nodes exchange the information of their virtual locations.

The target field is a $1200\text{m} \times 1200\text{m}$ area. The number of sensors was varied from 290 to 340. The radius of the sensing range was varied from 55.2m to 59.2m and the transmission range was set to 120m. Initially, we randomly distributed all sensors in the target field. Then we used a scheme to heal the holes until the coverage reaches 99.9%. We measured the following metrics.

(1) *Total moving distance.* This is the sum of the moving distances of all nodes for hole healing. It reflects the delay and energy cost of node movement in hole healing.
(2) *Total number of moves.* This is the sum of the number of moves of all nodes for hole healing. Since node moving startup consumes more energy than moving, this metric also reflects the energy consumption.
(3) *Coverage.* We distribute 10,000 points uniformly throughout the entire field. The coverage equals the percent points covered. This metric represents the effectiveness of full coverage schemes.
(4) *Standard deviation of energy consumption* (*standard deviation* for simplicity). We record the energy consumption of each node and calculate the standard deviation of these consumptions. This metric reflects the *energy consumption balance* of the network.
(5) *Number of nodes alive.* When parts of nodes in a WSN exhaust, the network cannot continue to work due to disconnections. This metric reflects the lifetime of a WSN with a hole healing scheme.

3.1 Energy Cost of Healing Holes

Fig. 11 and Fig. 12 show the *total moving distance* and *total number of moves* versus the number of nodes in different schemes in simulation and Orbit testbed, respectively. From these figures, we find that DECM has the best performance in *total moving distance*. The *total moving distance* follows SMART$\gg$ SSC $>$ VOR $>$ DECM, and the *total number of moves* follows SSC$>$ VOR $>$ DECM $\gg$ SMART in both simulation and Orbit. Both SSC and VOR have higher costs in both metrics than DECM because they cannot find the shortest paths for node movement to heal the holes. Long moving paths may produce new holes because a node may not be able to cover its original area after moving, thus resulting in more movements. In DECM, each triangle is managed by three nodes. A node's movement aims to fully cover its triangle. Even when a node selects its farthest target area when receiving multiple notifications, a potential new hole in the triangle can be healed by the other two sensor nodes. Furthermore, DECM uses the cooperative mechanism which can prevent new holes generated during the nodes' movement. In VOR, each Voronoi cell is managed by only one node. A node's movement aims to cover one point in its cell, which makes it very likely to generate new holes within its Voronoi cell that cannot be managed by any other nodes. As a result, this node may move back, or other nodes may need to move to cover the hole, resulting in more iterations of node movement. Thus, VOR produces a longer total moving distance and a greater total number of moves than DECM. In addition, SMART generates a significantly higher total moving distance than the others while producing the lowest total number of moves. Recall that the main task of SMART is to balance the distribution of sensor nodes throughout the entire target region in order to achieve full coverage.

Fig. 13 and Fig. 14 show the *total moving distance* and *total number of moves* versus the node sensing radius in different schemes in simulation and Orbit testbed, respectively. We see that the *total moving distance* follows

SMART $\gg$ VOR $\approx$ SSC $>$ DECM, and the *total number of moves* follows VOR $\approx$ SSC>DECM$\gg$SMART in both simulation and Orbit. This is for the same reasons as in Fig. 13. We also observe that for DECM, VOR, and SSC, the two metric results decrease as the radius of sensing range increases. This is because a larger sensing range can reduce the number and size of coverage holes.

3.2 Effectiveness of Healing Holes

Fig. 15 and Fig. 16 show that the *coverage* versus the *total moving distance* in different schemes in simulation and Orbit testbed, respectively. Fig. 15 (a) and Fig. 16 (a) show that to achieve 99.9% coverage, the total moving distance is 5600m in DECM, but is over 8000m in other schemes. Fig. 15 (b) and Fig. 16 (b) show that to achieve 99.9% coverage, the total moving distance is 4000m in DECM, is 5600m in VOR, and is over 8000m in SSC and SMART. DECM always finds the shortest paths for healing holes. A shorter movement distance has a lower probability of generating new holes. Also, three nodes managing a triangle area rather than one node makes it easier to heal newly generated holes caused by a node's movement. More importantly, a node's movement covers the entire triangle rather than a point. Thus, DECM avoids excessive iterations of node movement and achieves full coverage more rapidly than other schemes. The figures show that SMART achieves full coverage very slowly. This is because the objective of SMART is to balance the node distribution and thus hole sizes are not decreased rapidly.

3.3 Summary

From the experimental results, we find that there is no big difference between the simulation results and Orbit real-world testbed results. It is because in the Orbit testbed, we used virtual location exchanges between static nodes to simulate node movement and there is no packet loss during the testing process. These assumptions make the experiment of real testbed almost same as the simulation. So the results of both experiments look similar. The experimental results verify that selecting the farthest target area when finding several uncovered triangles is the optimal method in healing holes. In summary, the simulation and real testbed results show: 1) DECM is more energy-efficient for full coverage than other schemes, even when some sensor nodes die. 2) DECM can heal coverage holes more quickly than other schemes. 3) Selecting the farthest target area when finding several triangles that are not fully covered is an optimal method in DECM. 4) DECM achieves a balanced energy consumption balance among nodes in a WSN. 5) The lifetime of a WSN with DECM is longer than those with other schemes.

4 Related Work

Hole detection: Sensor coverage problem has received significant attention over the last few years [31, 32], and Voronoi diagram has been [3] a particularly important mechanism used for coverage hole detection. A Voronoi diagram is composed of numerous Voronoi cells, each of which has one sensor called *generating node* residing in it. A Voronoi cell is fully covered if a generating node covers all of its Voronoi cells' vertices. Thus, based on Voronoi diagram, each node only needs to detect the vertices of the Voronoi cells that are associated with itself. However, directly building a Voronoi diagram in a centralized manner requires many computations and transmissions. Some previous works have introduced distributed algorithms for the construction of Voronoi diagram in WSNs [4, 5]. Sharifzadeh and Shahabi [4] proposed a method, in which a node uses its collected location information of some nodes to build a 1-order Voronoi diagram. Since a node may not collect some nodes' location information that is important for diagram construction, it cannot guarantee the high accuracy of a Voronoi diagram. Bash and Desnoyers [5] proposed a method to improve the accuracy. This method begins with an initial approximation of Voronoi cell at each node based on its neighboring nodes and then leverages geographic routing primitives (e.g., GPSR [33]) 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 proposes an algorithm for building Voronoi diagram without location information. Simple localization solutions, such as equipping each node with a GPS receiver or manual configuration using coordinates [8, 18–20], are either energy-expensive or impractical for WSNs in some cases [21]. Previous works [22, 34] also used Delaunay triangulation, the dual graph of Voronoi diagram, to handle the sensor coverage problem in WSNs. Based on Delaunay triangulation, Wu *et al.* [22] proposed using a contour-based deployment to eliminate the coverage holes near the boundary of a sensing area with obstacles, and placing a new sensor to the position with the most coverage gains. Vu and Li [34] considered an area coverage problem for a variable sensing radii WSN. However, both works use a centralized algorithm for the Delaunay triangulation. Compared to these previous hole detection schemes, DECM does not need the specific location information of each node. Each node only requires relative location information of its nearby nodes.

Movement for full coverage: Node movement strategies for full area coverage have gained considerable attention during recent years. In virtual force methods [10–13], sensors are likened to electromagnetic particles and have repulsive and attractive forces between them. When the distance between two sensors is too great, the attractive force makes them pull each other closer; when the distance is too small, the repulsive force makes them push each other further away. Consequently, sensor nodes are exploded from dense regions to sparse regions or holes. However, these methods require sensors to move over a series of iterations to balance "virtual forces" between themselves, which may take a long time to converge and is not practical for real applications due to the high energy cost of node movement. Wang *et al.* [3] used the Voronoi diagram for healing coverage holes. As explained in the previous section, in these methods, a node's movement cannot completely heal the coverage holes in its Voronoi cell because one hole healing may produce other holes in its Voronoi cell, subsequently generating many iterative node movements. Many grid quorum-based movement schemes [6–9] view the movement-assisted network redeployment problem as a load balancing problem under the virtual grid model [35]. These schemes partition an entire target region into small grid cells, and consider the number of nodes in each cell as the cell's load. The schemes schedule sensor movement in order to achieve a balanced load distribution among the grid cells. A node within a grid cell can directly communicate with other nodes in its four adjacent cells and makes movement decisions according to information from adjacent cells. Some of these schemes also try to minimize the sensor movement distances. For example, SMART [9] arranges communication between cell heads to identify overloaded and underloaded cells and direct nodes from overloaded cells to move to underloaded cells. Knowing the loads of other cells, each cell tries to avoid any unnecessary movement; thus, both the total moving distance and the total number of moves can be minimized. However, since the target a node moves to is a cell rather than a specific point, the schemes still cannot find the shortest moving paths.

5 Conclusion

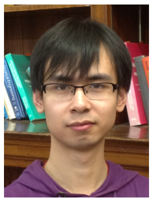
In this paper, we propose a DELaunay-based Coordinate-free Mechanism for full coverage (DECM) in WSNs. The scheme conducts Delaunay triangulation to divide a target field into triangles. Based on a rigorous mathematical model, nodes find the areas they can be located in while still keeping full coverage of their triangles, and discover the shortest paths to move when holes are detected. The scheme is advantageous over previous schemes in three aspects: (1) it builds a mathematical model for detecting and healing coverage holes; (2) it finds the shortest path for node movements to heal coverage holes, which reduces the number of movement iterations; (3) it detects coverage holes without the requirement of accurate location information; (4) it prevents new holes generated during the movement of nodes. As a result, our scheme has superior performance over previous schemes in terms of energy-efficiency, effectiveness in achieving full coverage, load balancing and lifetime. In the future, we will study the following problems. First, as the locally shortest path for node movements may not be the optimal globally shortest path, we will attempt to find the optimal globally shortest path. Second, we will extend DECM to more complex sensing range models. Third, we will extend DECM to achieve full coverage in a target region with obstacles. Fourth, we will use Delaunay triangulation to solve the k-coverage problem, where every point in the target region is covered by at least k sensor nodes.

Acknowledgements

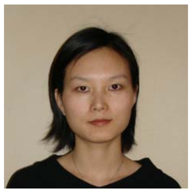
This research was supported in part by U.S. NSF grants CNS-1254006, CNS-1249603, OCI-1064230, CNS-1049947, CNS-1156875, CNS-0917056 and CNS-1057530, CNS-1025652, CNS-0938189, CSR-2008826, CSR-2008827, Microsoft Research Faculty Fellowship 8300751, and U.S. Department of Energy's Oak Ridge National Laboratory including the Extreme Scale Systems Center located at ORNL and DoD 4000111689. An early version of this work was presented in the Proceedings of ICPP' 12 [36].

References

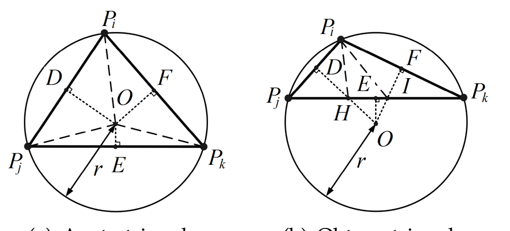
[1] C. Huang and Y. Tseng. The coverage problem in a wireless sensor network. In *Proc. of WSNA*, 2003.
[2] Y. Bejerano. Simple and efficient k-coverage verification without location information. In *Proc. of INFOCOM*, 2008.
[3] G. Wang, G. Cao, and T. F. La Porta. Movement-assisted sensor deployment. In *Proc. of INFOCOM*, 2004.
[4] M. Sharifzadeh and C. Shahabi. Supporting spatial aggregation in sensor network databases. In *International Symposium of ACM GIS*, 2004.
[5] B. A Bash and P. J Desnoyers. Exact distributed voronoi cell computation in sensor networks. In *SIAM Journal on Computing*, 2007.
[6] S. Chellappan, W. Gu, X. Bai, D. Xuan, and K. Zhang. Deploying wireless sensor networks under limited mobility constraints. In *IEEE TMC*, 2007.
[7] S. Chellappan, X. Bai, B. Ma, D. Xuan, and C. Xu. Mobility limited flip-based sensor networks deployment. In *IEEE TPDS*, 2007.
[8] G. Wang, G. Cao, T. Porta, and W. Zhang. Sensor relocation in mobile sensor networks. In *Proc. of INFOCOM*, 2005.
[9] S. Yang, M. Li, and J. Wu. Scan-based movement-assisted sensor deployment methods in wireless sensor networks. In *IEEE TPDS*, 2007.
[10] Y. Zou and K. Chakrabarty. Energy-aware target localization in wireless sensor networks. In *Proc. of PerCom*, 2003.
[11] Y. Zou and K. Chakrabarty. Sensor deployment and target localization based on virtual forces. In *Proc. of INFOCOM*, 2003.
[12] K. Akkaya and S. Janapala. Maximizing connected coverage via controlled actor relocation in wireless sensor and actor networks. In *Computer Networks*, 2008.
[13] N. Heo and P.K. Varshney. Energy-efficient deployment of intelligent mobile sensor networks. In *ITSMC*, 2005.
[14] Z. Butler and D. Rus. Event-based motion control for mobile sensor networks. In *Pervasive Computing*, 2003.
[15] N. Heo and P. K. Varshney. An intelligent deployment and clustering algorithm for a distributed mobile sensor network. In *ITSMC*, 2003.
[16] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energyefficient communication protocol for wireless microsensor networks. In *Proc. of HICSS*, 2000.
[17] J. Luo, D. Wang, and Q. Zhang. Double mobility: Coverage of the sea surface with mobile sensor networks. In *Proc. of INFOCOM*, 2009.
[18] L. Xu and D. Evans. Localization for mobile sensor networks. In *Proc. of MobiCom*, 2004.
[19] D. Rus D. Moore, J. Leonard and S. Tell. Robust distributed network localization with nosiy range measurements. In *Proc. of SenSys*, 2004.
[20] S. Lee, Z. Zhang, S. Sahu, and D. Saha. On suitability of euclidean embedding of internet hosts. In *SIGMETRICS*, 2006.
[21] D. Niculescu. Positioning in ad hoc sensor networks. In *IEEE Network Volume*, 2004.
[22] C. Wu, K. Lee, and Y. Chung. A delaunay triangulation based method for wireless sensor network deployment. In *Proc. of ICPADS*, 2006.
[23] X. Cao, E. Lloyd, and C. Shen. Deploying directional sensor networks with guaranteed connectivity and coverage. In *Proc. of SECON*, 2008.
[24] C. Qiu, H. Shen, S. Soltani, K. Sapra, H. Jiang, and J. Hallstrom. CEDAR: An optimal and distributed strategy for packet recovery in wireless network. In *Proc. of INFOCOM*, 2013.
[25] H. S. Lichte, H. Frey, and H. Karl. Fading-resistant low-latency broadcasts in wireless multihop networks: The probabilistic cooperation diversity approach. In *Proc. of MobiHoc*, 2010.
[26] M. d. Berg, O. Cheong, M. V. Kreveld, and M. Overmars. Computational geometry: Algorithm and applications. *Springer*, 2008.
[27] Q. Li and D. Rus. Global clock synchronization in sensor networks. In *Proc. of INFOCOM*, 2004.
[28] G. S. Kasbekar, Y. Bejerano, and S. Sarkar. Lifetime and coverage guarantees through distributed coordinate-free sensor activation. In *Proc. of MobiCom*, 2009.
[29] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *Proc. of MobiCom*, 2001.
[30] Orbit. http://www.orbit-lab.org/.
[31] J. Carle, D. Simplot-Ryl, and I. Stojmenovic. Localized sensor area coverage with low communication overhead. 2008.
[32] A. Gallais, J. Carle, D. Simplot-Ryl, and I. Stojmenovic. Ensuring k-coverage in wireless sensor networks under realistic physical layer assumptions. In *Proc. of 5th IEEE International Conference on Sensors*, 2009.
[33] B. Kar and H. T. Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. In *Proc. of MobiCom*, 2000.
[34] C. T. Vu and Y. Li. Delaunay-triangulation based complete coverage in wireless sensor networks. In *Proc. of PerCom*, 2009.
[35] Y. Xu, J. Heidemann, and D. Estrin. Geographyinformed energy conservation for ad hoc routing. In *Proc. of MobiCom*, 2001.
[36] C. Qiu and H. Shen. A delaunay-based coordinate-free mechanism for full coverage in wireless sensor networks. In *Proc. of ICPP*, 2012.



Chenxi Qiu Chenxi Qiu received the BS degree in Telecommunication Engineering from Xidian University, China, in 2009. He currently is a Ph.D student in the Department of Electrical and Computer Engineering at Clemson University, SC, United States. His research interests include sensor networks and wireless networks.

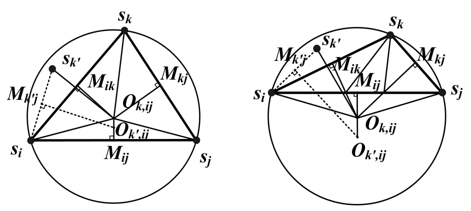


Haiying Shen Haiying Shen received the BS degree in Computer Science and Engineering from Tongji University, China in 2000, and the MS and Ph.D. degrees in Computer Engineering from Wayne State University in 2004 and 2006, respectively. She is currently an Assistant Professor in the Department of Electrical and Computer Engineering at Clemson University. 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 Microsoft Faculty Fellow of 2010 and a member of the IEEE and ACM.



(a) Acute triangle (b) Obtuse triangle

Fig. 17: The condition of a triangle's full coverage.



(a) Acute triangle (b) Obtuse triangle

Fig. 18: Proof of Theorem 2.3.

6 Proofs

6.1 Proof of Theorem 2.1

Proof: Fig. 17 (a) shows an acute triangle formed by three sensor nodes. Point O is the circumcenter of $\triangle P_iP_jP_k$ and r is the radius of the triangle ($r = OP_i = OP_j = OP_k$). Obviously, if $R_s < r$, where

$$r = \frac{d_{ij}d_{jk}d_{ik}}{\sqrt{\left(d_{ij}^2 + d_{ik}^2 + d_{jk}^2\right)^2 - 2\left(d_{ij}^4 + d_{ik}^4 + d_{jk}^4\right)}}, \tag{30}$$

point O cannot be covered by any sensor node; otherwise, every point within $\triangle P_iP_jP_k$ can be covered by at least one of the three nodes.

Fig. 17 (b) shows an obtuse triangle formed by three sensor nodes. Because the circumcenter of the obtuse triangle is outside of the triangle, $R_s \geq r$ is not the necessary condition for triangle's full coverage. In the figure, DH, FI, and EO are the perpendicular bisectors of P_iP_j, P_iP_k and P_jP_k. If $R_s > HP_i = HP_j$, then $\triangle DHP_i$ and $\triangle DHP_j$ are fully covered. Similarly, if $R_s > IP_i = IP_k$, then $\triangle IFP_i$ and $\triangle IFP_k$ are fully covered. In $\triangle HIP_i$, either H or I must be the farthest point from P_i. Therefore, $\triangle DHP_i$, $\triangle DHP_j$, $\triangle IFP_i$, $\triangle IFP_k$ and $\triangle HIP_i$ are fully covered iff

$$\begin{aligned} R_s &\geq \max\{HP_i, IP_i\} \\ &= \max\left\{\frac{d_{ij}^2 d_{jk}}{d_{ij}^2 + d_{jk}^2 - d_{ik}^2}, \frac{d_{ik}^2 d_{jk}}{d_{ik}^2 + d_{jk}^2 - d_{ij}^2}\right\}. \end{aligned}$$

□

6.2 Proof of Theorem 2.3

Proof: As Fig. 18 (a) and (b) show, s_k is the NTN of e_{ij}, and s_i, s_j, and s_k construct a triangle (either acute or obtuse), in which $O_{k,ij}$ is the circumcenter of the triangle. Suppose there is one sensor node $s_{k'}$ located inside the triangle on the same side of e_{ij} as s_k (i.e., $P_{k'}O_{k,ij} < P_jO_{k',ij}$). The perpendicular bisector of $P_iP_{k'}$ definitely intersects $P_iO_{k,ij}$ and $M_{ij}O_{k,ij}$ at point $O_{k',ij}$. Therefore, $h_{k',ij} = h_{k,ij} - O_{k,ij}O_{k',ij}$, which indicates that $h_{k,ij} \leq h_{k',ij}$. This contradicts the definition of the NTN. □

7 Pseudocode of Three Algorithms

Algorithm 1 Finding the NTN of e_{ij}.

```
flagDistance ← ∞, flagIndex ← 0;
for $l$ ← 1 to $m$ do
    $O_{k_l,ij}$ = InterOfBisector($e_{ij}$, $e_{ik_l}$);
    $h_{k_l,ij} = |M_{ij}O_{k_l,ij}|$;
    if $h_{k_l,ij}$ < flagDistance then
        flagDistance ← $h_{k_l,ij}$;
        flagIndex ← $k_l$;
    end if
end for
return flagIndex
```

Algorithm 2 Finding the NTN of e_{ij}.

```
if receive a triangulation notification then
    $t \leftarrow 1$, DT$_i \leftarrow \phi$;
    flagDistance ← ∞, flagEdge ← $\phi$;
    while $t \leq TTL_1 + TTL_2$ do
        if $t = 1$ and $s_i = s_{\text{seed}}$ then
            for each $s_j \in N_{s_i}(r)$ do
                if $d_{ij}$ < flagDistance then
                    flagDistance ← $d_{ij}$;
                    flagEdge ← $e_{ij}$;
                end if
            end for
            DT$_i$ ← DT$_i$ ∪ flagEdge;
            Notify $s_j$ to conduct DT$_j$ ← DT$_j$ ∪ flagEdge;
        else
            if $1 < t \leq$ TTL$_1$ then
                if DT$_i \neq \phi$ then
                    for each $e_{ij} \in$ DT$_i$ and $e_{ij}$ is potential edge do
                        $s_k$ ← FindNTN($e_{ij}$);
                        Notify $s_k$ to conduct DT$_k$ ← $e_{ik} \cup e_{ij} \cup$ DT$_k$;
                    end for
                end if
            end if
        end if
        if $TTL_1 < t \leq$ TTL$_1$ + TTL$_2$ then
            DT$_i$ ← EdgeFlip(DT$_i$);
        end if
        $t \leftarrow t + 1$;
    end while
end if
```

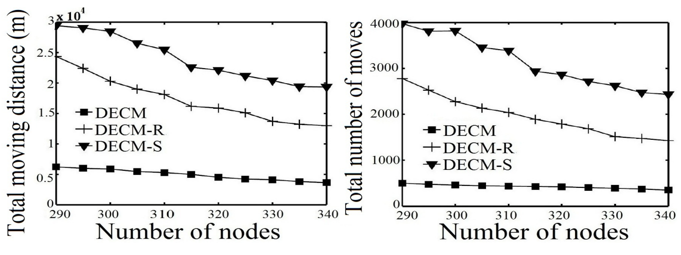
Algorithm 3 Hole healing conducted by node s_i.

```
flagDistance ← ∞, flagIndex ← 0;
while $t$ < TTL$_3$ do
    Listening;
    if Receive a notification to move to $P$ (or $P_{\text{cor}}$) from node $s_j$
    then
        if $P$ (or $P_{\text{cor}}$)$\notin A^i$ then
            $P_{\text{new}}$ = intersection($A^k$, $P_kP(orP_{\text{cor}})$);
            Move to $P_{\text{new}}$;
            for each $e_{jk} \in$ DT$_i$ do
                Randomly choose one node from $s_j$ and $s_k$ (assume $s_k$
                is selected);
                $P_{\text{cor}}$ ← DestinationPoint($s_k$);
                $s_i$ notifies $s_k$ to move to $P_{\text{cor}}$;
            end for
        else
            Move to $P$ (or $P_{\text{cor}}$);
        end if
    end if
end while
$t \leftarrow t + 1$
```

8 Additional Experimental Results

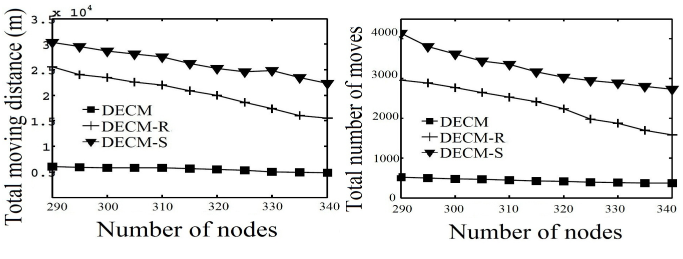
8.1 Different Target Area Selections

Fig. 19 and Fig. 20 show the *total moving distance* and *total number of moves* of DECM, DECM-R and DECM-S in different schemes in simulation and Orbit testbed,



(a) Total moving distance (b) Total number of moves

Fig. 19: Energy-efficiency of DECM with different target area selections (simulation).

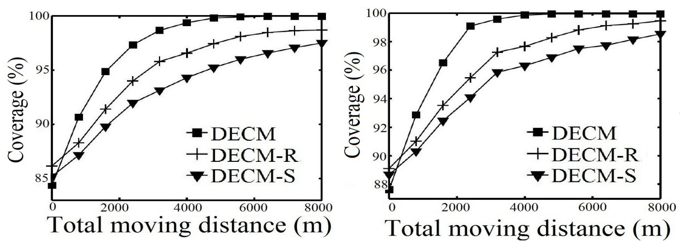


(a) Total moving distance (b) Total number of moves

Fig. 20: Energy-efficiency of DECM with different target area selections (Orbit Testbed).

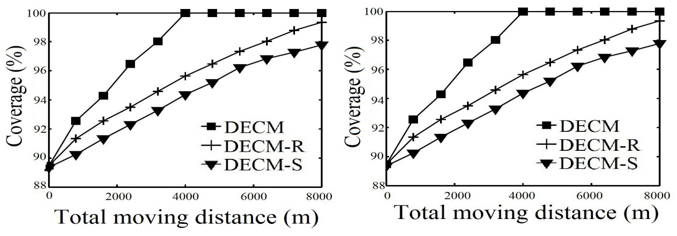
respectively. We see that DECM generates significantly lower results in both metrics than DECM-R and DECM-S. The *overall moving distances* of DECM-R and DECM-S are respectively 3.5 times and 4.9 times as long as that of DECM on average. The *total number of moves* of DECM-R and DECM-S are respectively 4.5 and 7.3 times as large as that of DECM on average. The results verify the effectiveness of DECM in choosing the farthest target area. It can quickly reduce the sizes of large holes, and hence reduce the number of moves. Also, moving towards large holes can balance the distribution of nodes over the entire target region more quickly. From our observations, when nodes move to heal small-size holes, new holes may arise in the original position. Then, the node will move back to heal the newly generated small holes, leading to more iterations.

Fig. 21 and Fig. 22 show the *coverage* versus the *total moving distance* in DECM, DECM-R, and DECM-S when the number of sensors is set to 300 and 330, in simulation and Orbit testbed, respectively. In both figures, DECM achieves full coverage more rapidly than DECM-R and DECM-S. In Fig. 21 (a) and Fig. 22 (a), when the *total moving distance* reaches 5600m, DECM achieves 99.9% coverage, while DECM-R and DECM-S achieve 98.12% and 96.01% coverage, respectively. In Fig. 21 (b) and Fig. 22 (b), when the *total moving distance* reaches 4800m, DECM achieves 99.9% coverage, while DECM-R and DECM-S only achieve 98.27% and 96.89% coverage, respectively. Because DECM always fixes the largest hole first since it selects the farthest target area, it reduces the size of holes more and achieves the full coverage faster than DECM-R and DECM-S. From the experimental results, we find that there is no big difference between the simulation results and Orbit real-world testbed results. It is because in the Orbit testbed, we used virtual location exchanges between static nodes to simulate node movement and there is no packet loss during the testing process. The experimental results verify that selecting the farthest target area when finding several uncovered triangles is the optimal method in healing holes.



(a) The number of nodes is 300 (b) The number of nodes is 330

Fig. 21: Effectiveness of healing holes in DECM with different target area selections (simulation).



(a) The number of nodes is 300 (b) The number of nodes is 330

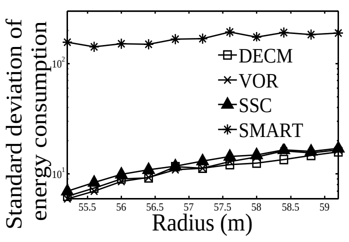
Fig. 22: Effectiveness of healing holes in DECM with different target area selections (Orbit Testbed).

8.2 Energy Consumption Balance Among Nodes

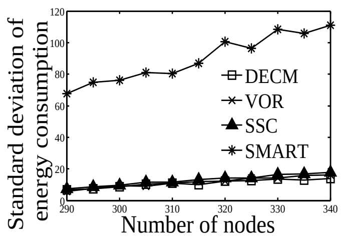
Fig. 23 and Fig. 24 show the *energy consumption balance* of DECM, VOR, SSC and SSC in simulation and Orbit testbed, respectively. Specifically, Fig. 23 (a) and Fig. 24 (a) show the standard deviation of energy consumption versus the radius of sensing range, and Fig. 23 (b) and Fig. 24 (b) show the standard deviation of energy consumption versus the number of nodes.

All these four figures demonstrate that the standard deviation of energy consumption follows SMART $\gg$ VOR $\approx$ SSC $\approx$ DECM. Recall that SMART has longer total moving distance but smaller number of moves compared to the other three schemes, which indicates that each movement in SMART is longer and some nodes do not need to move. Thus, SMART is more likely to make some nodes move very long distances while make other nodes just move zero or a little distance, leading to unbalanced energy consumption among nodes. We also observe that the standard deviation of energy consumption of DECM is slightly lower than that of VOR and SSC. This is because DECM prevents movement oscillation, thus avoiding making some nodes move for a total long distance due to the oscillation movements. Also, DECM's cooperative movement mechanism distributes the movement task originally for one node among multiple nodes. As a result, DECM achieves more balanced energy consumption among nodes.

Fig. 23 (a) and Fig. 24 (a) show that the standard deviations of energy consumption of DECM, VOR and SSC increase with the radius of sensing range increases, while that of SMART maintains nearly constant. Fig. 23 (b) and Fig. 24 (b) show that the standard deviations of energy consumption of all schemes increase as the number of nodes increases. The target region requires fewer moves as the radius increases or as the number of nodes increases. Thus, some nodes do not need to move or only move very short distances, leading to more unbalanced energy consumption. SMART is not affected significantly by the change of sensing radius because it aims to balance the sensor distribution in the target area

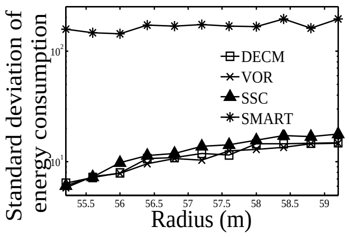


(a) Different radii

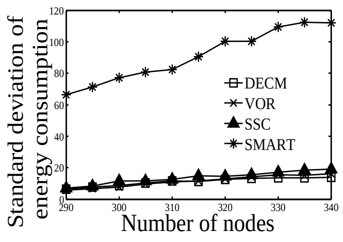


(b) Different number of nodes

Fig. 23: Energy consumption balance in different schemes (simulation).

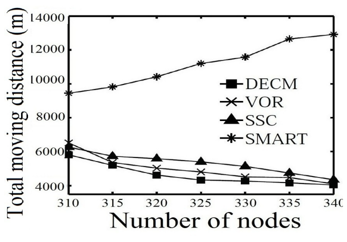


(a) Different radii

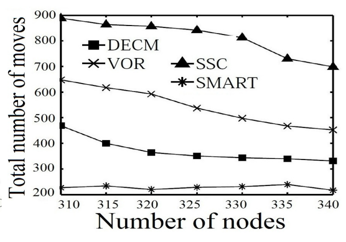


(b) Different number of nodes

Fig. 24: Energy consumption balance in different schemes (Orbit Testbed).

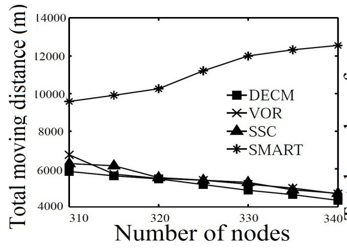


(a) Total moving distance

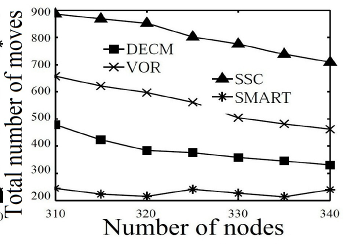


(b) Total number of moves

Fig. 25: Energy-efficiency of different schemes in handling dead nodes (simulation).



(a) Total moving distance



(b) Total number of moves

Fig. 26: Energy-efficiency of different schemes in handling dead nodes (Orbit Testbed).

regardless of the sensing radius.

8.3 Healing Holes Due to Dead Nodes

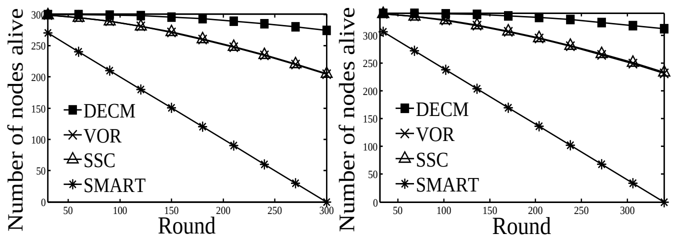
In this experiment, 50 nodes died immediately after all holes were healed from the initial deployment. Fig. 25 (a) and (b), and Fig. 26 (a) and (b) show the *total moving distance* and *number of moves* of the schemes in healing the holes caused by the dead nodes in simulation and Orbit testbed, respectively. We see that the *total moving distance* follows SMART≫SSC>VOR>DECM, and that the *total number of moves* follows SSC>VOR>DECM>SMART in both simulation and Orbit. This is for the same reasons as in Fig. 11. The results show that DECM still exhibits superior performance over others even with dead nodes.

8.4 Lifetime of WSNs

We use *round* to denote the sequence of each time period in which the destinations of moving nodes are calculated. Fig. 27 and Fig. 28 show *the number of nodes alive* over rounds of DECM, VOR, SSC and SMART in simulation and Orbit testbed, respectively. From all these four figures, we see that the lifetime follows DECM $>$ VOR $\approx$ SSC $>$ SMART. Compared to DECM, VOR and SSC, the number of nodes alive decreases rapidly in SMART because (1) its average energy consumption is much higher than the other three schemes (according to Fig. 11 - Fig. 14); (2) the standard deviation of energy consumption for all the nodes in SMART is higher (according to Fig. 23 and Fig. 24), which indicates that the energy stored in some nodes in SMART decreases more rapidly, leading to short WSN lifetime. We define the lifetime of a WSN as the time period in rounds the WSN lasts when 10% of nodes in the WSN exhaust. DECM has the longest lifetime (about 3.5 times as long as VOR and SSC and about 9.1 times as long as SMART) due to its better performance in *total moving distance* and *energy consumption standard deviation* (according to Fig. 11 - Fig. 14, Fig. 23 and Fig. 24). DECM can locally find the shortest path for hole healing. Also, it has the cooperative mechanism that prevents generating new holes during node movements. Since the nodes in DECM have less energy consumption and the energy consumption among nodes is well balanced, DECM always has the longest lifetime comparing to other schemes.

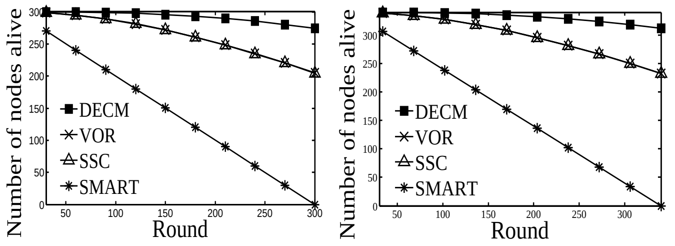
8.5 Effect of TTL_1 and TTL_2

Recall that TTL_1 and TTL_2 are the time constraint for triangulation and edge flip, respectively. Fig. 29 (a) and Fig. 29 (b) show how TTL_1 and TTL_2 affect the coverage performance of DECM in the simulation and Orbit testbed. Fig. 29 (a) shows that the coverage of DECM increases from about 84% to 100% when TTL_1 changes from 1 to 8. Because of the transmission delay among nodes, a smaller TTL_1 produces more nodes that are not included in the triangulation, and hence more undetected coverage holes. Fig. 29 (b) shows that the coverage of DECM increases from about 99% to 100% when TTL_2 changes from 1 to 4. Due to transmission delay, a smaller TTL_2 leads to more illegal triangulations, and hence more unnecessary long movements, which generate more new holes.



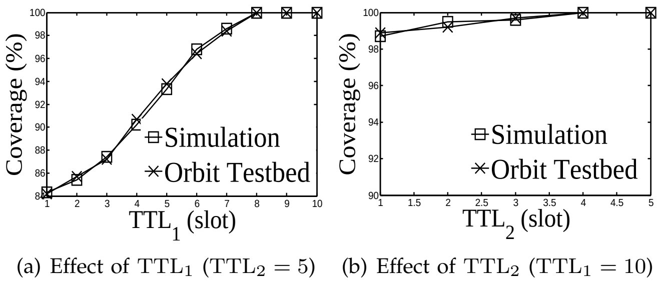
(a) The number of nodes is 300 (b) The number of nodes is 330

Fig. 27: Lifetime of WSNs with different schemes (Simulation).



(a) The number of nodes is 300 (b) The number of nodes is 330

Fig. 28: Lifetime of WSNs with different schemes (Orbit Testbed).



(a) Effect of TTL_1 ($\mathrm{TTL}_2 = 5$) (b) Effect of TTL_2 ($\mathrm{TTL}_1 = 10$)

Fig. 29: How TTL_1 and TTL_2 affect the coverage of DECM.