

# Strictly Localized Sensor Self-Deployment for Optimal Focused Coverage

Xu Li, Hannes Frey, Nicola Santoro, and Ivan Stojmenovic, *Fellow, IEEE*

**Abstract**—We consider sensor self-deployment problem, *constructing FOCUSED coverage (F-coverage) around a Point of Interest (POI)*, with novel evaluation metric, *coverage radius*. We propose to deploy sensors in polygon layers over a locally computable equilateral triangle tessellation (TT) for optimal F-coverage formation, and introduce two types of deployment polygon,  $\mathcal{H}$ -polygon and  $\mathcal{C}$ -polygon. We propose two strictly localized solution algorithms, *Greedy Advance (GA)*, and *Greedy-Rotation-Greedy (GRG)*. The two algorithms drive sensors to move along the TT graph to surround POI. In GA, nodes greedily proceed as close to POI as they can; in GRG, when their greedy advance is blocked, nodes rotate around POI along locally computed  $\mathcal{H}$ - or  $\mathcal{C}$ -polygon to a vertex where greedy advance can resume. We prove that they both yield a connected network with maximized hole-free area coverage. To our knowledge, they are the first localized sensor self-deployment algorithms that provide such coverage guarantee. We further analyze their coverage radius property. Our study shows that GRG guarantees optimal or near optimal coverage radius. Through extensive simulation we as well evaluate their performance on convergence time, energy consumption, and node collision.

**Index Terms**—Coverage, self-deployment, localized algorithms, mobile sensor networks.



## 1 INTRODUCTION

SENSOR self-deployment is an important research issue that deals with autonomous coverage formation in mobile sensor networks (MSN). Relevant research is still on its initial stage, with emerging new problem statements and development of basic self-deployment techniques extendable to future more complex protocols. Considering potentially large network scale, unpredictable sensor failure, dynamic topological change, and limited network bandwidth, a sensor self-deployment algorithm should be carried out in a localized manner. Term “localized” means that each sensor makes self-deployment decision independently, using  $k$ -hop neighborhood information for a constant  $k$ . In the case of  $k = 1$ , we call the algorithm *strictly localized*.

There exist a class of sensor network applications, where sensors are designated to monitor concerned events or environmental changes around a given strategic site or coverage focus, called *Point of Interest (POI)*. For instance, in a battle field scenario, sensors are deployed around a battalion headquarter to detect intrusion events, whose distance to the headquarter reflects their degree of danger.

Another example is sensors scattered around a chemical plant to monitor its distance-dependent pollutional impact on the soil/air in the vicinity. These applications uniquely require that an area close to POI have higher priority to be covered than a distant one. We call the coverage of such a surrounding network *FOCUSED coverage* or *F-coverage*. In this paper, we address how to achieve optimal F-coverage through sensor self-deployment approach.

### 1.1 F-Coverage Evaluation

The *coverage region* of a sensor network is the region enclosed by the outer boundary of the network. A *sensing hole* is a closed uncovered area inside the coverage region. The *coverage* of a sensor network is measured by area. It is defined as the subtraction of the total area of sensing holes from the area of the coverage region. Area and sensing hole are two key evaluation metrics for traditional area coverage problem. They reflect the sensitivity of a sensor network over a *Region of Interest (ROI)*. An ideal area coverage has maximized area and no sensing holes. In the F-coverage problem, measuring area and hole existence are no longer sufficient, because distance from POI to uncovered areas is also important and must be taken into consideration. In this case, we introduce an additional metric, *coverage radius*.

**Definition 1 (Coverage Radius).** *The radius of an F-coverage is the radius of the maximal hole-free disc enclosed by sensors and centered at POI.*

Optimal F-coverage has maximized coverage radius. If number of sensors is unlimited and with sensing ranges approaching zero, sensors can be deployed densely and achieve close to circular coverage. The maximal hole-free disc therefore has near circular shape. In this case, coverage radius is called *circular radius*. Since the sensing radius is finite, we consider here instead a discrete variant of coverage radius, referred to as *polygonal radius*. It is alternatively measured by layer distance. *Layer distance*, also called convex

- X. Li is with the Department of Electrical and Computer Engineering, University of Waterloo, 200 University Avenue West, Waterloo, ON N2L 3G1, Canada. E-mail: x279li@bbcr.uwaterloo.ca.
- H. Frey is with the Faculty of Computer Science, Electrical Engineering and Mathematics Department, University of Paderborn, Warburger Str. 100, 33098 Paderborn, Germany. E-mail: hannes.frey@uni-paderborn.de.
- N. Santoro is with the School of Computer Science, Carleton University, 1125 Colonel By Drive, Ottawa, ON K1S 5B6, Canada. E-mail: santoro@scs.carleton.ca.
- I. Stojmenovic is with the School of Information Technology and Engineering, University of Ottawa, 800 King Edward Avenue, Ottawa, ON K1N 6N5, Canada, and the Department of Electronic, Energetics and Telecommunications, FTN, University of Novi Sad, Serbia. E-mail: ivan@site.uottawa.ca.

Manuscript received 4 Feb. 2010; revised 25 Oct. 2010; accepted 12 Nov. 2010; published online 21 Dec. 2010.

For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2010-02-0054. Digital Object Identifier no. 10.1109/TMC.2010.261.

layers in computational geometry or Tukey's depth in statistics, represents the number of successive complete convex polygons adjacently surrounding POI. More precisely, we consider a discrete set of convex polygons  $\mathcal{P}_i (i = 1, 2, \dots)$  composed of sensors, centered at POI, and having a diameter of  $i * d$  for some constant  $d$ . We count the total number of such polygons lying completely in the coverage region.

## 1.2 Problem Statement

We consider an asynchronous MSN of unknown size  $n$ . The network is randomly deployed in a 2D free field (e.g., an area on ocean surface in practice) and possibly initially disconnected. Sensors bear the same communication radius  $r_c$  and the same sensing radius  $r_s$ . They move asynchronously possibly at different speeds. Sensors know about the location of POI, denoted by  $F$ . We place  $F$  at origin  $(0, 0)$  without loss of generality.

The goal is to develop strictly localized sensor self-deployment algorithms that yield a network surrounding  $F$  with an equilateral triangle tessellation (TT) layout. This TT layout is desirable because it maximizes the coverage area of a given number of sensors without coverage gap when sensor separation is equal to  $\sqrt{3}r_s$  [1], [11], [14], and that it automatically maintains network connectivity when  $r_c \geq \sqrt{3}r_s$ . As an additional requirement, the final network should have maximized coverage radius with respect to  $F$ .

We consider this new sensor self-deployment problem under the following common assumptions: 1)  $r_c \geq \sqrt{3}r_s$ , 2) sensors know their own spatial coordinates by attached GPS devices or any effective localization algorithm, and 3) through lower-layer protocols (minor modification may apply), sensors have the information about their 1-hop neighbors, i.e., location, moving status, and movement destination (if moving). In the sequel, we will use terms "sensor" and "node" interchangeably.

## 1.3 Our Contributions

We introduce an F-coverage evaluation metric, *coverage radius*, which reflects the need to maximize the distance from  $F$  to uncovered areas. This leads to a novel sensor self-deployment problem, *F-coverage formation around a given coverage focus*. We convert this problem to vertex coverage problem over a locally computable equilateral triangle tessellation (TT) and propose, by the coverage radius definition, to locate sensors in polygon layers concentric to  $F$  in TT. We introduce two types of deployment polygon,  $\mathcal{H}$ -polygon and  $\mathcal{C}$ -polygon. The former are hexagons. The latter are polygons best approximating inscribed circles of, and thus requires less nodes than, the former for achieving the same circular coverage radius.

We propose two strictly localized algorithms, *Greedy Advance (GA)* and *Greedy-Rotation-Greedy (GRG)*. In the two algorithms, self-governing sensors align themselves with the TT grid and locally compute virtual  $\mathcal{H}$ - or  $\mathcal{C}$ -polygons. In GA, sensors greedily proceed, from polygon to polygon, as close to  $F$  as they can; in GRG, when their greedy advance is blocked, sensors rotate around  $F$  along the polygon that they are traversing, to a vertex where greedy advance can resume. In both algorithms, when sensors are compactly placed or collide, they may temporarily move away from  $F$ . Both GA and GRG are resilient to dynamic node addition

and removal (failure) and work regardless of network disconnectivity.

We formally prove that the two algorithms both yield a connected network of TT layout with hole-free coverage. We also analyze their coverage radius property. Our study indicates that GRG with  $\mathcal{H}$ -polygon (i.e., Hex-GRG) generates optimal hexagonal F-coverage and near optimal circular F-coverage, and that GRG with  $\mathcal{C}$ -polygon (i.e., Cir-GRG) generates, compared with Hex-GRG, circular F-coverage yet closer to the optimal using less nodes. We evaluate the performance of GA and GRG on convergence time, energy consumption, and node collision through extensive simulation.

This paper is an integration and generalization of our previous work [8], [9], along with detailed and extended analysis. It corrects a few misclaims about coverage radius maximization made in [9]. We briefly review related work in Section 2. We introduce  $\mathcal{H}$ -polygon and  $\mathcal{C}$ -polygon in Section 2. Then we propose algorithms GA and GRG in Section 4, and present their analytical and simulation study in Sections 5 and 6. We discuss possible extensions and practice issues in Section 7, followed by the closing remarks presented in Section 8.

## 2 RELATED WORK

To our knowledge, there is no previous work addressing the F-coverage problem as identified here in this paper. Sensor self-deployment algorithms for coverage formation over ROI with no particular coverage focus exist in the literature. Below we will review some of these related work at very short length. An extensive survey can be found in our recent paper [10].

The most known sensor self-deployment approach is vector-based (or virtual-force-based) approach. Algorithms that belong to this category include [4], [6], [11], [12], to name a few. The basic idea is that each node computes movement vectors in rounds using the relative position of its neighbors and moves according to the vector summation. Although proposed solutions appear efficient in communication, they leave coverage holes and could be highly inefficient in terms of coverage radius.

Heo and Varshney [5] presented a Voronoi diagram-based algorithm. This algorithm enables sensors to identify local sensing holes using Voronoi diagram and align their sensing range along its Voronoi polygon for minimizing uncovered area. Similar algorithms are VOR [12] and the one in [3]. In [3], robots find centroids of their Voronoi polygons based on a utility function, e.g., Gaussian density function representing reduced monitoring ability for a sensor with squared distance from it. Sensors then move toward that centroid, exchange messages and repeat the step until they stabilize around POI. Voronoi diagram-based solutions are not localized since some of links could be very long and thus not between communication neighbors. Further, coverage radius is again not considered.

Yang et al. [13] presented a scan-based sensor deployment scheme. The target field is partitioned into a mesh, and nodes are treated as load. The goal is converted to load balancing among mesh cells through multirounds of scan. This approach requires the network to be dense enough so that load balancing can be proceeded in the entire field. As

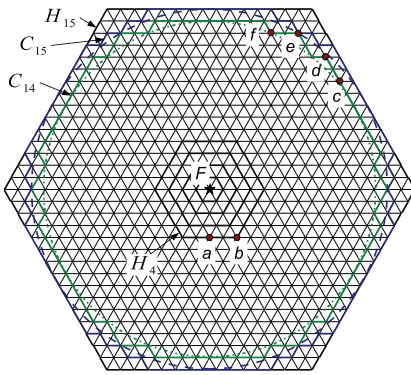


Fig. 1. Equilateral triangle tessellation ( $G_{TT}$ ).

the authors admitted, it may generate huge message overhead when the network is very dense due to the increased number of rounds of scans.

Bartolini et al. [2] presented a snap and spread self-deployment scheme. Sensors simultaneously construct a hexagonal tiling portion by pushing and pulling sensors to hexagon centers. Tiling portions of different sensors merge when they meet. The algorithm is not purely localized because, according to the implementation presented in [2], in a pull process for filling adjacent empty hexagons, a snapped sensor has to visit (by sending a message) every other hexagon in the worst case before finding an un-snapped sensor.

Existing algorithms, when used for focused coverage formation problem, do not provide (and even do not study) guarantee on coverage radius. In worst case, the resulting coverage radius can be as bad as 0 (meaning that POI is located outside the network or on the network border). Besides, they have major weaknesses such as unrealistic assumptions (e.g., initial connectivity out of randomized placement or fixed network size), requirement for global computation (e.g., Voronoi diagram construction or clustering), vulnerability to node failure, and so on. The unsuitability and the incompleteness of previous work motivate our research presented here.

### 3 DEPLOYMENT POLYGONS

An equilateral triangle tessellation (TT) is a planar graph composed of congruent equilateral triangles, as shown in

Fig. 1. Given a common orientation, say the North, and edge length  $l_e$ , nodes are able to compute a unique TT containing  $F$  as vertex. Denote this TT by  $G_{TT}$ . Two vertices are neighboring or adjacent to each other if there is an edge between them. With knowledge of its own location, each node can determine whether it is located at a vertex in  $G_{TT}$  and which vertex (if so), and adjacent vertices (in communication range).

If we deploy sensors at vertices around  $F$  in  $G_{TT}$ , we automatically obtain a network with the required TT layout; if we further assure that no empty vertex exist in the coverage region, and that the coverage region have an (approximate) circular shape centered at  $F$ , we as well achieve the desired F-coverage with no sensing hole and with (near) maximized radius. By this means, we actually convert the F-coverage problem to a vertex coverage problem over  $G_{TT}$ .

In our work, TT edge length  $l_e$  is set to  $\sqrt{3}r_s$  because it ensures connectivity and minimizes sensing range overlapping [1], [11], [14]. By coverage radius definition, sensors should be deployed in polygon layers for optimal F-coverage. In the following, we introduce two types of deployment polygon in  $G_{TT}$ , which will be used later in our proposed sensor self-deployment algorithms.

The residence polygon of a vertex is the polygon that the vertex belongs to; the residence vertex of a node is the vertex at which the node is located. We denote by  $|ab|$  the euclidean distance of two points (nodes)  $a$  and  $b$  and by  $|S|$  the size of a set  $S$ . Let  $SP(u, v)$  be the set of edges along the shortest path connecting vertices  $u$  and  $v$  in  $G_{TT}$ . The TT distance of  $u$  and  $v$  is  $|SP(u, v)|$ .

#### 3.1 $\mathcal{H}$ -polygon

**Definition 2 ( $\mathcal{H}$ -polygon).** An  $\mathcal{H}$ -polygon ( $\mathcal{H}_i$ ) of layer distance  $i$  is a polygon whose perimeter is composed of successive vertices that have equal TT distance  $i$  to  $F$ .

Each  $\mathcal{H}$ -polygon is an hexagon. In  $G_{TT}$ , a vertex  $v (\neq F)$  has two neighborhood patterns along  $\mathcal{H}$ -polygon:

1. Edge (-):  $v$  is located on polygon edge (Fig. 2a).
2. Convex corner ( $\wedge$ ):  $v$  is located at convex polygon corner (Fig. 2b).

In Fig. 1, vertices  $a$  and  $b$  both reside on  $\mathcal{H}_4$ , respectively, with edge and convex corner neighborhood patterns.

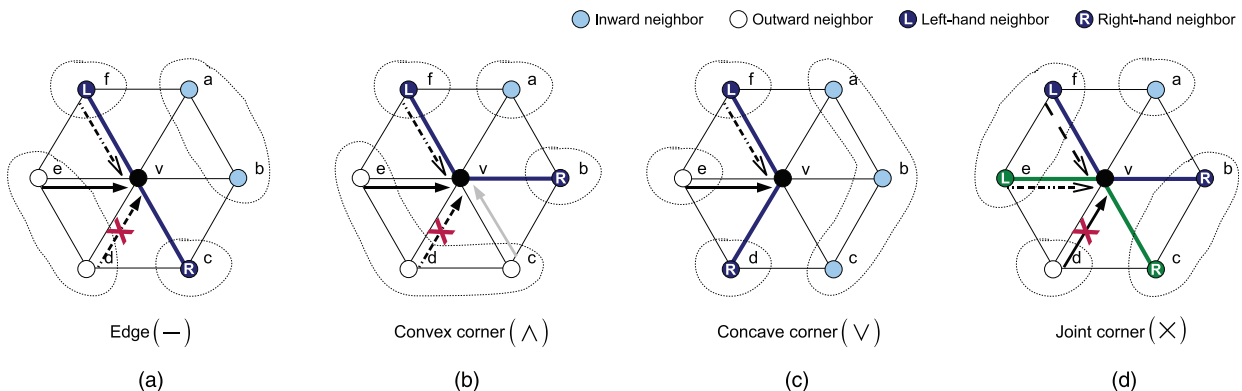


Fig. 2. Neighborhood pattern of vertex  $v$ .

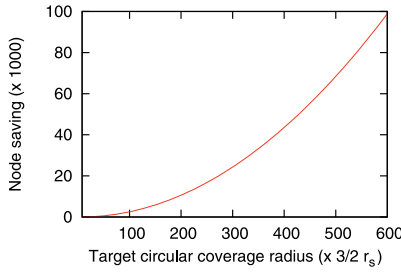


Fig. 3. Node saving  $\rho(i)$  versus circular coverage radius  $i \times \frac{3}{2} r_s$ .

$\mathcal{H}$ -polygons are concentric to  $F$ , as shown in Fig. 1 where  $\mathcal{H}_1, \dots, \mathcal{H}_4$  and  $\mathcal{H}_{15}$  are highlighted by thick black lines. Each  $\mathcal{H}$ -polygon  $\mathcal{H}_i$  ( $i \geq 1$ ) is composed of  $6i$  vertices. The total number  $\nu_H(i)$  of vertices enclosed by  $\mathcal{H}_i$  (inclusive) is the sum of number of composing vertices of all  $\mathcal{H}_j$  ( $j \leq i$ ) plus 1 (counting for  $F$ ). That is,

$$\nu_H(i) = 1 + \sum_{q=1}^i 6q = 3i(i+1) + 1. \quad (1)$$

Then the optimal hexagonal coverage radius  $\gamma_H(n)$  (in layer distance) that  $n$  nodes can provide over  $G_{TT}$  is

$$\gamma_H(n) = \max\{i | \nu_H(i) \leq n\}. \quad (2)$$

### 3.2 C-Polygon

For simplicity, we abuse the definition of layer distance to allow concave polygons and define  $\mathcal{C}$ -polygon below:

**Definition 3 (C-polygon).** A  $\mathcal{C}$ -polygon ( $\mathcal{C}_i$ ) of layer distance  $i$  is a minimum area polygon enclosing the maximal inscribed circle  $C(\mathcal{H}_i)$  of  $\mathcal{H}_i$  and consisting of successive vertices.

$\mathcal{C}_i$  best approximates circle  $C(\mathcal{H}_i)$  without radius reduction (thus, named  $\mathcal{C}$ -polygon), and it must be completely contained in, or overlapped by,  $\mathcal{H}_i$  because, otherwise, it is not the minimum area polygon. In Fig. 1,  $C(\mathcal{H}_{14})$  and  $C(\mathcal{H}_{15})$  are shown as dotted or dashed circles, and  $\mathcal{C}_{14}$  and  $\mathcal{C}_{15}$  are marked and labeled.

**Lemma 1.** A vertex belongs to  $\mathcal{C}_i$  if and only if it itself does not reside inside  $C(\mathcal{H}_i)$  and at least one of its neighboring vertices lies inside  $C(\mathcal{H}_i)$ .

Knowing  $\mathcal{H}_i$ , and  $C(\mathcal{H}_i)$  and according to Lemma 1, one may compute the total number  $\nu_C(i)$  of vertices enclosed by  $\mathcal{C}_i$  (inclusive) as follows:

$$\nu_C(i) = \begin{cases} \nu_H(i), & \text{for } i \leq 7, \\ \nu_H(i) - \rho(i), & \text{for } i > 7, \end{cases} \quad (3)$$

where  $\rho(i) = 6 \sum_{t=1}^{\lfloor \frac{1-\sqrt{3}}{2} i - 1 \rfloor} (2 \lfloor \frac{1}{2} (i-t-\sqrt{6it-3t^2}) \rfloor + 1)$  represents the number of vertices that exist inside  $\mathcal{H}_i$  or on its perimeter, but fall outside  $\mathcal{C}_i$ . The computation is omitted here for space limit and can be found in [7]. The optimal (i.e., maximum) circular coverage radius  $\gamma_C(n)$  that  $n$  nodes can provide over  $G_{TT}$  is

$$\gamma_C(n) = \frac{3}{2} r_s \max\{i | \nu_C(i) \leq n\}. \quad (4)$$

From (3),  $\mathcal{C}$ -polygon is equivalent to  $\mathcal{H}$ -polygon for the first seven layers, and then it requires less nodes than

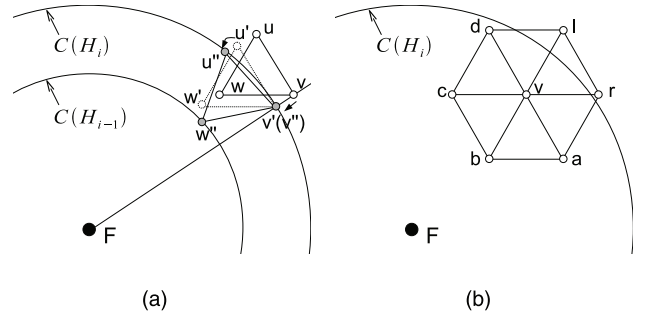


Fig. 4. Pictures for Lemmas 2 and 3. (a) Lemma 2. (b) Lemma 3.

$\mathcal{H}$ -polygon for producing the same circular coverage radius (in euclidean distance) (see Fig. 3).

Below we study localized computation of  $\mathcal{C}$ -polygon and its properties. Six neighboring vertices of a vertex  $v$  define a circle of radius  $l_e = \sqrt{3}r_s$  that is centered at  $v$ . For  $v$  to belong to  $\mathcal{C}_i$ , this circle has to intersect  $C(\mathcal{H}_i)$ . Denote by  $R_i$  the radius of  $C(\mathcal{H}_i)$ .  $R_i = ih$ , where  $h = \frac{3}{2}r_s$  is the height of TT triangle. Hence,  $v$  can belong only to such  $\mathcal{C}_i$  that  $|vF| \leq R_i + l_e$ . It can be derived that a satisfactory  $i$  is either  $\lfloor \frac{|vF|}{h} \rfloor$  or  $\lfloor \frac{|vF|}{h} \rfloor - 1$ . Summarizing,

**Theorem 1.** A vertex  $v$  residing on  $\mathcal{C}_i$  for  $i = \lfloor \frac{2|vF|}{3r_s} \rfloor$  will also reside on  $\mathcal{C}_{i-1}$  if and only if it is adjacent to a vertex  $w$  such that  $|wF| < \frac{3}{2}(i-1)r_s$ .

**Lemma 2.** Any two different  $\mathcal{C}$ -polygons share no common TT edges.

**Proof.** Assume for the sake of contradiction that a TT edge  $uv$  is part of  $\mathcal{C}_i$  and  $\mathcal{C}_j$  ( $i > j$ ). Let  $w$  be the common vertex neighbor of  $u$  and  $v$  that resides on the same side of  $uv$  as  $\mathcal{C}_i$ . Let  $d = |wF|$ . By Lemma 1,  $u$  and  $v$  must not lie inside  $C(\mathcal{H}_i)$ ; whereas,  $w$  must be located inside  $C(\mathcal{H}_j)$  (thus, inside  $C(\mathcal{H}_i)$ ), namely,  $d < R_j$  (radius of  $C(\mathcal{H}_j)$ ). Draw line segment  $vF$ . It intersects  $C(\mathcal{H}_i)$  at  $v'$ . Translate  $\Delta uvw$  for vector  $vv'$  and obtain  $\Delta u'v'w'$ . Then rotate  $\Delta u'v'w'$  around  $v'$  to obtain  $\Delta u''v''w''$  such that  $u'' = u'$ ,  $u''$  is located on  $C(\mathcal{H}_i)$  and  $w''$  inside  $C(\mathcal{H}_i)$ . This translation and rotation process is shown in Fig. 4a. Notice that  $w''$  is located on  $C(\mathcal{H}_{i-1})$ , i.e.,  $|w''F| = R_{i-1} \geq R_j$ , and obviously  $|w''F| \leq d$ . Therefore, we have  $d \geq R_j$ , which contradicts our previous result.  $\square$

**Lemma 3.** On a  $\mathcal{C}$ -polygon, the two vertex neighbors of any vertex are not adjacent to each other.

**Proof.** Consider an arbitrary vertex  $v$  on an arbitrary  $\mathcal{C}$ -polygon  $\mathcal{C}_i$ . By Lemma 1,  $v$  must not lie inside  $C(\mathcal{H}_i)$ . Denote the six vertex neighbors of  $v$  by  $a, b, c, d, l$  and  $r$ . Further, let  $l$  and  $r$  be the two located on  $\mathcal{C}_i$ . Assume for the sake of contradiction that  $l$  and  $r$  are adjacent to each other, as shown in Fig. 4b.

By Lemma 1, at least one of the four vertices  $a, b, c, d$  lies inside  $C(\mathcal{H}_i)$  so that  $v$  is located on  $\mathcal{C}_i$ . Let  $a$  be that vertex. Then  $b$  must be inside  $C(\mathcal{H}_i)$ , because, otherwise,  $b$  will be on  $\mathcal{C}_i$  as well, which is not possible. Since  $b$  is inside  $C(\mathcal{H}_i)$ ,  $c$  must be inside  $C(\mathcal{H}_i)$  for the same reason. This way, every vertex neighbor of  $v$  other than  $l$  and  $r$  is inside  $C(\mathcal{H}_i)$ , giving us a contradictory result:  $v$  itself must be located inside  $C(\mathcal{H}_i)$ .  $\square$

By Lemmas 2 and 3 and through exhaustive enumeration, we obtain the following theorem:

**Theorem 2.** In  $G_{TT}$ , a vertex  $v(\neq F)$  has four and only four possible neighborhood patterns along  $\mathcal{C}$ -polygon:

1. Edge ( $-$ ):  $v$  has one residence  $\mathcal{C}$ -polygon, and is located on a polygon edge (Fig. 2a).
2. Convex corner ( $\wedge$ ):  $v$  has one residence  $\mathcal{C}$ -polygon, and is located at convex polygon corner (Fig. 2b).
3. Concave corner ( $\vee$ ):  $v$  has one residence  $\mathcal{C}$ -polygon, and is located at concave polygon corner (Fig. 2c).
4. Joint corner ( $\times$ ):  $v$  has two residence  $\mathcal{C}$ -polygons (Fig. 2d).

In Fig. 1, vertices  $c, d, f$  all have only one residence  $\mathcal{C}$ -polygon  $\mathcal{C}_{14}$ ; their neighborhood patterns are edge, convex corner, and concave corner, respectively. Vertex  $e$  is a joint corner vertex, shared by  $\mathcal{C}_{14}$  and  $\mathcal{C}_{15}$ .

### 3.3 Neighborhood Division

We generally denote by  $\mathcal{P}_i$  deployment polygon (simply polygon), whether  $\mathcal{H}$ -polygon or  $\mathcal{C}$ -polygon, of layer distance  $i$  ( $i \geq 1$ ) and by  $\nu(i)$  the total number of its enclosed vertices. The neighborhood pattern set in  $\mathcal{C}$ -polygon is a superset of that in  $\mathcal{H}$ -polygon, and thus serves as the neighborhood pattern set for generalized  $\mathcal{P}_i$ .

Nodes are able to compute all polygons  $\mathcal{P}_i$  and thus the neighborhood pattern of any vertex. But, since we are aiming at strictly localized algorithm, they only determine the neighborhood patterns of the vertices within their communication range on the fly.

Each vertex  $v(\neq F)$  has at least one and at most two residence polygons, depending on its neighborhood pattern  $Pat(v)$ . For generalization purpose, suppose  $v$  has two residence polygons. Let the outer one be  $\mathcal{P}_i$  and the inner one  $\mathcal{P}_{i'}$ .  $i' = i - 1$  if  $v$  indeed resides on two different polygons, and  $i' = i$  otherwise. The layer distance  $d(v)$  of  $v$  to  $F$  is equal to the layer distance of its inner residence polygon, i.e.,  $d(v) = i'$ . The left-hand (right-hand) side of  $v$  is the clockwise (respectively, counterclockwise) direction around  $F$ .

As shown in Fig. 2, the vertex neighbors of  $v$  may be divided into four disjoint groups: *left-hand* neighbors, *right-hand* neighbors, *inward* neighbors, and *outward* neighbors. The first two groups share the same residence polygons  $\mathcal{P}_i$  and  $\mathcal{P}_{i'}$  with  $v$ , while the last two groups, respectively, belong to  $\mathcal{P}_{i'-1}$  and  $\mathcal{P}_{i+1}$ . We define four multisets  $N^L(v)$ ,  $N^R(v)$ ,  $N^O(v)$ , and  $N^I(v)$ , with respect to these four groups, to ease future presentation.

$N^L(v)$  contains two elements, denoted by  $Inn(N^L(v))$  and  $Out(N^L(v))$ , respectively, the left-hand vertex neighbors of  $v$  on  $\mathcal{P}_{i'}$  and  $\mathcal{P}_i$ . When  $i' = i$ ,  $Inn(N^L(v)) \equiv Out(N^L(v))$ , refers to the same only left-hand vertex neighbor.  $N^O(v)$  is composed of three elements, denoted as  $Ltm(N^O(v))$ ,  $Mid(N^O(v))$ , and  $Rtm(N^O(v))$ . When  $v$  has three outward vertex neighbors, they, respectively, represent the leftmost, the middle, and the rightmost one. When  $v$  has two such neighbors,  $Ltm(N^I(v))$  refers to the left one, and  $Mid(N^I(v)) \equiv Rtm(N^I(v))$  refers to the right one. When  $v$  has one outward vertex neighbor,  $Ltm(N^O(v)) \equiv Mid(N^I(v)) \equiv Rtm(N^I(v))$  implies this only outward vertex neighbor.  $N^R(v)$  and  $N^I(v)$  are similarly defined as  $N^L(v)$  and  $N^O(v)$ .

## 4 F-COVERAGE BY SELF-DEPLOYMENT

In this section, we propose two strictly localized sensor self-deployment algorithms, *Greedy Advance* and *Greedy-Rotation-Greedy*, which are both resilient to node failure and able to operate regardless of network partition. They are composed of a set of simple hop selection rules. By these rules, nodes make self-deployment decision using merely 1-hop neighborhood information and move asynchronously toward  $F$ . They stop when no eligible next hop is available and resume deployment movement whenever possible.

Both GA and GRG require use of deployment polygon  $\mathcal{P}$ , which is unspecified in algorithm definition for generalization purpose. The previously introduced  $\mathcal{H}$ -polygon and  $\mathcal{C}$ -polygon are two special cases. For hexagonal F-coverage,  $\mathcal{H}$ -polygon should be adopted (version Hex-GA and Hex-GRG); for circular F-coverage,  $\mathcal{C}$ -polygon is engaged (version Cir-GA and Cir-GRG).

For simplicity, a TT vertex  $v$  is considered *occupied* if a node is not moving and is located in close proximity to  $v$ , or if a node is moving toward  $v$ ;  $F$  is considered occupied in the case that it is not physically occupiable.

### 4.1 Greedy Advance

In GA, a node moves greedily along TT edges as close to  $F$  in terms of layer distance as it can. Generally speaking, it moves step by step, each step from current residence vertex  $w$  to an empty (i.e., unoccupied) *inward* vertex neighbor  $v \in N^I(w)$  determined by a number of hop selection rules. When multiple such vertices are available, a random choice is made.

When multiple nodes are present at the same vertex at the same time, collision occurs. Rules are necessary for avoiding node collision which is not desirable since each vertex is expected to be occupied by at most one node for coverage maximization. Below we introduce GA rules including the priority rule, the forbiddance rule, and the innermost-layer rule. We assume for the time being that nodes are initially located at distinct vertices of  $G_{TT}$ . This assumption rarely holds in practice. We relax it immediately after, by a few extra rules.

Notice that it is only when  $N^O(v)$  (a multiset) contains multiple distinct vertices that greedy advance may cause node collision at vertex  $v(\neq F)$ . According to Fig. 2, this is the case when  $Pat(v) = "-" \wedge "-" \wedge "$ . Examine the corresponding graphs Figs. 2a and 2b. If two nodes are greedily moving to  $v$  from vertices  $e$  (i.e.,  $Ltm(N^O(v))$ ) and  $d$  (i.e.,  $Mid(N^O(v))$ ) in parallel, they may collide at  $v$ . But this situation can be avoided by the following priority rule as the two nodes are actually neighboring each other and know the potential collision.

**Rule 1 (Priority Rule).** For two nodes aiming at a vertex  $v(\neq F)$  from two different vertices  $Ltm(N^O(v))$  and  $Mid(N^O(v))$ , the one from  $Ltm(N^O(v))$  has higher priority to proceed.

If  $Pat(v) = "-" \wedge "$  and the two nodes are from vertices  $e$  (i.e.,  $Ltm(N^O(v))$ ) and  $c$  (i.e.,  $Rtm(N^O(v))$ ), they could also collide at  $v$ . In this case, because they are not adjacent to each other, the collision is not locally avoidable. To eliminate this undesirable situation, we introduce the following conservative forbiddance rule.

**Rule 2 (Forbiddance Rule).** *In the case of  $Pat(v) = "\wedge"$ , a node located at  $Rtm(N^O(v))$  does not take vertex  $v (\neq F)$  as greedy next hop.*

In Fig. 2, greedy advances to vertex  $v$  are shown as lines with solid arrows. Among them, the dashed have lower priority (by the priority rule), and the gray is forbidden (by the forbiddance rule). Note: when  $Pat(v) = "\wedge"$  (Fig. 2b), greedy advance to  $b$  from  $c$  is allowed, whichever neighborhood pattern that  $c$  has, as long as  $b$  is not occupied. Thus, the forbiddance rule itself does not block a node's advance toward  $F$ .

Now let us examine  $F$  (see Fig. 1). All six adjacent vertices of  $F$  are located on  $\mathcal{P}_1$  and are outward vertex neighbors of  $F$ . If  $F$  is not occupied, then the final  $F$ -coverage will have the worst radius, equal to 0, by definition. We can ensure the occupancy of  $F$  by the following innermost-layer rule.

**Rule 3 (Innermost-Layer Rule).** *A node located at a vertex on  $\mathcal{P}_1$  moves to  $F$  as long as  $F$  is to its knowledge unoccupied.*

The above aggressive innermost-layer rule may induce greedy-greedy collision (i.e., multiple greedily advancing nodes colliding) at  $F$ . Such a collision takes place at most once, because a node will stay at  $F$  after it reaches  $F$  and no node will try to move to  $F$  once  $F$  is occupied.

We now relax the temporary assumption that nodes are initially located at distinct TT vertices by introducing the alignment rule:

**Rule 4 (Alignment Rule).** *A node located inside or on the border of a TT triangle moves to the triangle vertex that is occupied by the least number of nodes. If more than one such triangle vertex exists, the closest is selected. A random choice is made in case of tie.*

This alignment rule is however very likely to cause node collision and thus deployment redundancy at some vertices. This leads us to develop a new type of node movement, *retreat*, for collision resolution. Retreat is the opposite of greedy advance. It happens from a vertex on  $\mathcal{P}_i$  ( $i \geq 0$ ) to a neighboring vertex on  $\mathcal{P}_{i+1}$  that is occupied by the least number of nodes. In case of tie, a random choice is made. Here,  $\mathcal{P}_0 = F$ . By nodal retreat, permanent collision no longer exists; both GA and GRG gain the ability to spread out compactly placed sensors. The following retreat rule defines when to perform retreat movement.

**Rule 5 (Retreat Rule).** *After some nodes collide at a TT vertex, they enter a local ranking process, during which each of them is assigned a rank. The node with the highest rank makes its next deployment decision first; the others follow in accordance with the decreasing order of their ranks. If the  $t$ th node decides to stay at the vertex, every node with rank lower than  $t$  retreats.*

The retreat rule does not specify how local ranking is conducted. It can be done either at random or according to certain criterion (if available) such as residual energy or node ID or the combination thereof. The colliding nodes are able to do the ranking locally and independently because they are neighboring each other.

## 4.2 Greedy-Rotation-Greedy

GRG involves, in addition to greedy advance, a new type of movement, *rotation*, which guides nodes around blocking peers. Rotation is along nodal residence polygon, and is restricted to a particular, say the counterclockwise, direction so as to avoid unnecessary collision among rotating nodes. Specifically, a node located at vertex  $v$  tries rotation by moving to *inner right-hand* vertex neighbor  $Inn(N^R(v))$  when GA fails.

Notice that rotation is always along inner residence polygon. The logic is that a node should not move away from  $F$  once it moves closer (in terms of layer distance) to it. A node stops rotating when it reaches a vertex where greedy advance can resume, or when it returns to the vertex where it started rotating. To properly react to neighbor failure, a return node resets its rotation starting point to null whenever it finds its rotation next hop becomes occupied.

In asynchronous environment, a node rotating on  $\mathcal{P}_i$  might never be able to move onto  $\mathcal{P}_{i-1}$  despite the vacancies on  $\mathcal{P}_{i-1}$ , if its neighboring nodes on  $\mathcal{P}_{i-1}$  rotate together with it and keep blocking its greedy advance. However, by observing Fig. 2 we can find that  $Rtm(N^O(v))$  is always adjacent to  $Out(N^R(v))$  regardless of  $Pat(v)$ . If a node located at  $v$  discovers that some node is rotating to  $Rtm(N^O(v))$ , then it knows that the node will proceed to  $Inn(N^R(v))$  through  $Rtm(N^O(v))$  (and  $Out(N^R(v))$  if  $Inn(N^R(v)) \neq Out(N^R(v))$ ) if it itself does not chose  $Inn(N^R(v))$  as rotation next hop. Thus, we introduce the following suspension rule:

**Rule 6 (Suspension Rule).** *A node located at vertex  $v$  does not rotate to  $Inn(N^R(v))$  if any of its neighbors is currently rotating to  $Rtm(N^O(v))$ .*

By the suspension rule, a node rotating on  $\mathcal{P}_i$  will either meet an empty vertex on  $\mathcal{P}_{i-1}$ , surpassing some  $\mathcal{P}_{i-1}$  nodes in between, or is blocked by a node located at a joint corner of  $\mathcal{P}_i$  and  $\mathcal{P}_{i-1}$ , or find no vacancy on  $\mathcal{P}_{i-1}$  and stops at its rotation starting point. No rotation loop will take place. Note: when the collision avoidance rules to be defined in Section 4.2.2 are applied, the suspension rule ought to be ignored if greedy advance at  $Rtm(N^O(v))$  is forbidden by those rules.

Nodal rotation brings about greedy-rotation collision that needs to be taken care of. Examine Fig. 2. If a node is moving to  $v$  from  $f$  while another node is moving to  $v$  from  $e$ , they are likely to collide at  $v$ . This collision can be prevented by a competition rule:

**Rule 7 (Competition Rule).** *When two nodes are competing for  $v$  from two different vertices  $Out(N^L(v))$  and  $Ltm(N^O(v))$  ( $Inn(N^L(v))$  and  $Out(N^L(v))$ ), the one from  $Ltm(N^O(v))$  (respectively,  $Out(N^L(v))$ ) wins.*

If the two nodes are instead from  $f$  and an outward vertex neighbor (e.g.,  $d$  in Figs. 2a, 2b, and 2d) different than  $e$ , they could also collide at  $v$ . But this greedy-rotation collision is no longer avoidable by the above rule. Depending on the way of handling this situation, GRG has two variants: **Collision alloWance (CW)** and **Collision aVoidance (CV)**.

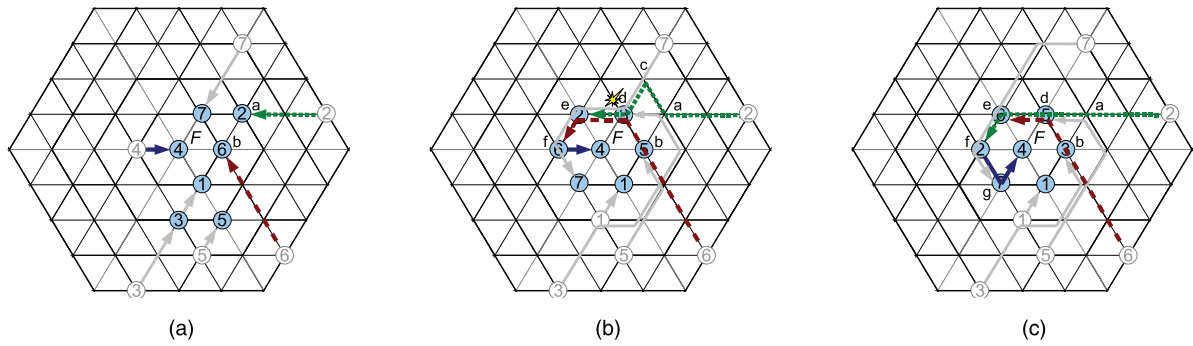


Fig. 5. Final node distribution after sensor self-deployment. (a) GA. (b) GRG/CW. (c) GRG/CV.

#### 4.2.1 GRG/CW

In this variant, no additional restriction is applied. Greedy-rotation collision is allowed and handled by the retreat rule which defines movement order and when to retreat. Through ordered decision making, greedy-rotation collision could appear as a transient phenomenon. For example, in the scenario given in Fig. 5, collision between nodes 2 and 6 takes place at  $d$  and is then automatically resolved. However, there is no assurance that it does not occur infinitely often. As illustrated in [8], retreat movement may cause greedy rotation collision loop and endless movement in some rare scenarios. This is due to the problematic rotation and retreat role switch, which refreshes the rotating node's rotation record. It will not take place if we prevent the rotating node from being retreated outward, which in turn can be achieved by enforcing the following ranking policy: a node that rotates is always assigned the highest rank in a local ranking process for collision resolution.

#### 4.2.2 GRG/CV

The priority rule and the forbiddance rule preclude non-POI-based greedy-greedy collision but leave POI-based (due to the innermost-layer rule) still possible. Unidirectional rotation prohibits rotation-rotation collision. Greedy-rotation collision is eliminated only in part by the competition rule. It is because the rule relies on the adjacency of the greedy prior hop and the rotation prior hop of a vertex, which however does not always remain. In CRG/CV, these collisions are totally precluded through extra collision avoidance rules.

We say a node's greedy advance to vertex  $v \neq F$  is "safe" if and only if it will cause no greedy-rotation collision at  $v$ . In order not to risk greedy-rotation collision, the node must not greedily advance unless it knows the movement is definitely safe. From local perspective, it is able to make such an assurance only when its residence vertex is adjacent to both  $Inn(N^L(v))$  and  $Out(N^L(v))$ . When  $Pat(v) = "-" \vee " \wedge "-" \vee " \vee "$ ,  $Inn(N^L(v)) = Out(N^L(v))$ ; when  $Pat(v) = " \times "$ ,  $v$  has only one unique outward vertex neighbor, which is adjacent to  $Out(N^L(v))$ . We define a safety rule for avoiding greedy-rotation collision.

**Rule 8 (Safety Rule).** A node does not choose inward vertex neighbor  $v$  as greedy next hop if its residence vertex is not neighboring  $Inn(N^L(v))$ .

In Fig. 2, the greedy advances prevented by the above rule are marked by "X" sign. Now we shall see how to

avoid greedy-greedy collision at  $F$ . This can be accomplished simply by replacing the innermost-layer rule with the following gateway rule.

**Rule 9 (Gateway Rule).** A vertex on  $\mathcal{P}_1$  is predefined as the gateway to  $F$ . A node located on  $\mathcal{P}_1$  performs only greedy advance if its residence vertex is the gateway, or only rotation otherwise.

#### 4.3 Execution Examples

In the following, we will comparatively show through examples how GA and the two variants of GRG, i.e., GRG/CW and GRG/CV, work. Although they operate regardless of network size and asynchrony and how nodes are distributed, we consider for ease of understanding a simple fully synchronized scenario, where seven nodes initially placed at distinct TT vertices start the self-deployment algorithms simultaneously, make deployment decision at the same time, and move step by step at the same speed. In this setting, a sensor is not able to know where its neighbors are moving and sometimes has to make conservative decision (by assuming those neighbors are static).

For simplicity, we place these nodes within  $\mathcal{P}_4$ . The execution procedure is the same whether  $\mathcal{H}$ -polygon or  $\mathcal{C}$ -polygon is used as in this case the two types of polygon are equivalent according to (3). Fig. 5 shows the final node distribution, respectively, by GA, GRG/CW, and GRG/CV. Node trajectories are marked by thick arrowed lines, pointing from initial position to final position. The initial position of node 1 is the final position of node 3 in Fig. 5a. The initial position of node 4 is the final position of node 6 in Fig. 5b and of node 2 in Fig. 5c. We focus on nodes 2, 4, and 6.

Fig. 5a indicates that, when GA is applied, nodes 2, 4, and 6 move toward  $F$  and stop, respectively, at  $a$ ,  $F$ , and  $b$  by the forbiddance rule and the innermost-layer rule. As shown in Fig. 5b, when GRG/CW is employed, node 4 proceeds in the same way as in GA; whereas, nodes 2 and 6 travel along an extended path. Specifically, after reaching  $a$ , node 2 finds that vertex  $d$  is occupied by node 7, and that greedy advance to  $b$  is not allowed by the forbiddance rule. Thus, it has to rotate around  $F$  along its residence polygon. When node 2 rotates to  $c$ , node 6 arrives at  $b$ . At that moment,  $d$  becomes empty due to node 7's departure, and  $F$  has been taken by node 4. Then node 2 decides to greedily proceed to  $d$ , and node 6 also decides to rotate to  $d$ . Because the two nodes are not neighboring each other, they do not know each other's motion plan and consequently collide at

*d*. Because a rotating node is given priority to take the next deployment step in such a greedy-rotation collision, node 6 continues its rotation, while node 2 has to wait. Finally, node 6 rotates to its final position *f*, passing by *e*; node 2 rotates to *e* after node 6 leaves *e* for *f*.

Observe Fig. 5c for GRG/CV. By the safety rule, node 4 is not allowed to move to *F* directly. It first rotates to a particular gateway vertex (*g* in this example) and then greedily proceed to *F* from there. Node 7 cannot start with greedy advance but has to perform rotation first according to the safety rule, ending up with a different trajectory, which directly affects node 2's self-deployment process. After reaching *a*, node 2 is able to continue its greedy advance and immediately proceeds to *d* since no one is occupying *d*. When node 6 reaches *b* through greedy advance, node 2 just arrives at *d*, and node 4 already got to *F*. Then, node 6 has to wait because it can perform neither greedy advance nor rotation in this case. The suspension of node 6's movement in turn affects nodes 3 and 5's trajectories, which we will not go through here. Finally, node 2 rotates to *f*, and node 6 rotates to *e*. Notice that the collision between nodes 2 and 6 in GRG/CW does not occur in GRG/CV.

## 5 ANALYSIS

In this section, we first prove the correctness of the two proposed algorithms GA and GRG. We prove that both of them terminate and that they yield a connected network with hole-free coverage. Afterward, we analyze their coverage radius property. We derive that GA has no guarantee on coverage radius, and that GRG guarantees (near) optimal coverage radius.

### 5.1 Correctness

**Lemma 4.** *Both GA and GRG ensure that  $F$  will be occupied by a single node within finite time.*

**Proof.** By the alignment rule,  $F$  could be occupied by multiple nodes during the initial node alignment. If  $F$  is still empty after the alignment process terminates, it will be eventually occupied by at least one node through greedy advance, because the algorithms ensure a winner in every competition for greedy advance. In any case,  $F$  becomes occupied within finite time. Once  $F$  is occupied, no node will move to it. If multiple nodes exist at  $F$  at some moment, one and only one of them will stay, and the others will move onto  $\mathcal{P}_1$  by the retreat rule.  $\square$

**Theorem 3.** *GA terminates within finite time.*

**Proof.** In the initial alignment step, all nodes move toward their closest vertex notwithstanding the movement decisions of any other nodes. Thus, the alignment process terminates obviously within finite time. Moreover, by Lemma 4,  $F$  will be occupied by a single node within finite time. Henceforth, we safely assume that the alignment process already passed and that  $F$  has been occupied by a single node.

The GA rules prevent a node from greedily moving to an already occupied vertex. They also prevent two nodes located at different vertices from greedily moving to the same empty vertex. A node may leave a vertex by retreat only if the vertex is occupied by another node.

Otherwise, the node has no reason to retreat. Hence, the number of occupied vertices never decreases.

Assume for the sake of contradiction that GA never terminates. Since the number of occupied vertices never decreases, there exists  $m \leq n$  where  $n$  is the network size such that the algorithm runs infinitively long on  $m$  occupied vertices. It is important to distinguish between occupied vertices and the nodes actually occupying those vertices. For the rest of the proof, we assume that GA already arrived at that maximum number  $m$  of occupied vertices. This assumption does not mean that this set may not change over time, but that the set of occupied vertices will never again change in size.

Consider the currently occupied  $m$  vertices  $T = \{t_1, \dots, t_m\}$ . Whenever an unoccupied vertex is visited by a retreating node not colliding with a greedily advancing node, the number of occupied vertices increases by one. This would contradict the assumption that GA already arrived at the maximum number  $m$  of occupied vertices.  $T$  may only change due to a greedy advance. Since  $m$  is fixed, the set  $T$  changes only by nodes performing greedy advance. We will show that this greedy advance is however possible finite number of times.

Define by  $\sum(T)$  the sum of the layer distance from  $F$  to the vertices in  $T$ , i.e.,  $\sum(T) = \sum_{t \in T} d(t)$ . Recall that we consider the occupied vertices not the nodes actually occupying these vertices. Whenever  $T$  changes to  $T'$  due to greedy advance, a node moves from a polygon  $\mathcal{P}_{i+1}$  to a polygon  $\mathcal{P}_i$  ( $\mathcal{P}_{i-1}$  if the target vertex is joint corner vertex). It follows,  $\sum(T') \leq \sum(T) - 1$ . Since  $\sum(T) \geq 0$ , it follows that the set of occupied vertices can only change a finite number of times.

If the algorithm does not terminate then there exist infinite number of retreat moves only, which then occur after the last performed greedy move. However, we will show now that the number of consecutive retreat moves is also finite. Retreat only moves which are possible only from a vertex left occupied after moving by another node, to another occupied vertex, as otherwise  $m$  would increase. Retreating nodes always move from a polygon  $\mathcal{P}_i$  to a polygon  $\mathcal{P}_{i+1}$ . Thus, a change from  $T$  to  $T'$  always satisfies  $\sum(T') \geq \sum(T) + 1$ . This sum is limited by the farthest possible distance that each node could move. Each retreat increases distance to  $F$ , and the maximal distance cannot exceed  $n$ . Otherwise, there will be an unoccupied vertex on the shortest path from retreating node to  $F$ , and a node located before that vertex would be able to make greedy advance, which contradicts our previous conclusion about lack of further greedy advances. Therefore, retreat steps will also terminate in a finite number of steps (it can be easily shown that this is only possible when  $m = n$ ). This means that the algorithm itself will terminate, and proof is complete.  $\square$

**Lemma 5.** *Let  $\mathcal{P}_0, \dots, \mathcal{P}_{i-1}$  be fully occupied without collocated nodes. Let  $n \geq \nu(i)$ . In GRG,  $\mathcal{P}_i$  will be fully occupied without collocated nodes within finite time.*

**Proof.** When  $\mathcal{P}_0, \dots, \mathcal{P}_{i-1}$  are all fully occupied, nodes that have decided to stay on  $\mathcal{P}_i$  never leave  $\mathcal{P}_i$  but counter-clockwise rotate along  $\mathcal{P}_i$  because they are assigned highest rank in any local ranking process triggered by

node collision, making unoccupied  $\mathcal{P}_i$  vertices appear “rotating” in the opposite direction. In worst case, they make a full rotation and then stop moving, rendering unoccupied vertices fixed. In any case, some  $\mathcal{P}_{i+1}$  nodes are guaranteed to meet the empty vertices on  $\mathcal{P}_i$  by counterclockwise rotation and move to fill their location by the suspension rule and the competition rule. Because  $n \geq \nu(i)$  and there are no colocated nodes on the  $i - 1$  inner polygons,  $\mathcal{P}_i$  will be fully occupied at the end as nodes keep moving toward it and eventually stop on it. Nodal retreat guarantees that no  $\mathcal{P}_i$  vertex be occupied by multiple nodes.  $\square$

**Lemma 6.** *Let  $\mathcal{P}_0, \dots, \mathcal{P}_{i-1}$  be fully occupied without colocated nodes. Let  $\nu(i - 1) < n < \nu(i)$ . In GRG, nodes located on  $\mathcal{P}_i$  will stop moving within finite time.*

**Proof.** As inner polygons  $\mathcal{P}_0, \dots, \mathcal{P}_{i-1}$  are all fully occupied, nodes from outer polygons will rotate along  $\mathcal{P}_i$ , after arriving at  $\mathcal{P}_i$ . In GRG/CW, these rotating nodes could collide with some greedily advancing nodes; but their rotation is not affected since they are assigned highest rank in the local ranking process. Because  $\nu(i - 1) < n < \nu(i)$ , either (at least) one of them will make a full rotation if  $\mathcal{P}_i$  does not have a joint corner with  $\mathcal{P}_{i-1}$ , or some of them will be blocked by nodes located at those joint corners otherwise. By protocol definition, such a node will stop moving and block the rotation of any following node. Eventually, the nodes on  $\mathcal{P}_i$  will become fixed. After the nodes on  $\mathcal{P}_i$  stop moving, the nodes on  $\mathcal{P}_{i+1}$  (if any exists) will get onto  $\mathcal{P}_i$  and possibly rotate along  $\mathcal{P}_i$  as well. These newly arriving nodes will stop moving within finite time because of the blocking from previously stopped nodes.  $\square$

**Theorem 4.** *GRG terminates within finite time.*

**Proof.** It follows immediately from Lemmas 4-6.  $\square$

**Theorem 5.** *Both GA and GRG yield a connected network with hole-free coverage.*

**Proof.** We prove this theorem by contradiction. By Theorems 3 and 4, we know that both GA and GRG terminate within finite time. Assume that there is a sensing hole in the coverage region at some moment after the algorithm (either GA or GRG) terminates. Denote by  $v$  a vertex farthest (in layer distance) from  $F$  on the border of the hole. There must exist a node at  $Ltm(N^O(v))$ , because, otherwise,  $v$  would not be the farthest border vertex of the hole. In this case, that node will greedily proceed to occupy  $v$  by protocol definition. This actually contradicts our assumption that the algorithm has terminated. Thus, the final coverage constructed by the algorithm (either GA or GRG) contains no sensing hole. Then network connectivity simply follows from the lack of sensing holes and the assumption of  $r_c \geq \sqrt{3}r_s$ .  $\square$

## 5.2 Coverage Radius

In GA, the final coverage of a MSN has an unpredictable shape, depending very much on the initial sensor placement. As shown in Fig. 5a, it is possible that F is located on the border of the network, rendering coverage radius equal to 0. This example implies that GA does not

provide coverage radius guarantee either in layer distance or in euclidean distance. In contrast, as we will see below, GRG generates optimal or near optimal focused coverage in both metrics.

Consider a MSN of size  $n$ . Let  $\mathcal{F}_H(n)$  ( $\mathcal{F}_C(n)$ ) be the F-coverage constructed by Hex-GRG (respectively, by Cir-GRG) using this network, and  $k_H(k_C)$  the index of the outermost deployment polygon of  $\mathcal{F}_H(n)$  (respectively,  $\mathcal{F}_C(n)$ ). For  $\mathcal{F}_H(n)$  ( $\mathcal{F}_C(n)$ ), if  $n = \nu_H(k_H)$  (respectively,  $\nu_C(k_C)$ ), the outermost deployment polygon  $\mathcal{H}_{k_H}$  (respectively,  $\mathcal{C}_{k_C}$ ) is fully occupied, and partially occupied otherwise.

Denote by  $r_H(n)$  ( $r_C(n)$ ) the hexagonal radius of  $\mathcal{F}_H(n)$  (respectively, circular radius of  $\mathcal{F}_C(n)$ ). From (2) and (4) and Lemmas 4-6, we have the following results.

**Theorem 6.**  $r_H(n) = \gamma_H(n)$  for any  $n$ .

**Theorem 7.**  $r_C(n) = \gamma_C(n) - \delta$ , where  $\delta = 0$  if  $n = \nu_C(k_C)$ , or  $0 < \delta < \frac{3}{2}r_s$  otherwise.

Because of the complex neighborhood pattern in  $\mathcal{C}$ -polygon, the execution procedure of Cir-GRG would intuitively be more complicated and costly than that of Hex-GRG. In the following, we will see whether the relatively simple Hex-GRG can be used as replacement of Cir-GRG for (near) optimal circular F-coverage. Later, in Section 6, we will study whether Cir-GRG is indeed more expensive than Hex-GRG through simulation.

Let  $r_H^C(n)$  be the circular coverage radius of  $\mathcal{F}_H(n)$ . We first derive the upper bound of  $\gamma_C(n)$  in terms of  $k_H$  and then show  $r_H^C(n)$  is near optimal.

**Lemma 7.**  $\gamma_C(n) \leq 3\sqrt{\frac{\sqrt{3}}{2\pi}}k_H r_s$ .

**Proof.** It is provable that hexagonal node placement produces maximized coverage over the TT.  $\gamma_C(n)$  must not be larger than the radius of the circle whose area is equal to the area of  $\mathcal{H}_{k_H}$ , that is,

$$\gamma_C(n) \leq 3\sqrt{\frac{\sqrt{3}}{2\pi}}k_H r_s. \quad \square$$

**Lemma 8.**  $r_H^C(n) \geq 0.95\gamma_C(n)$  for  $n = \nu_H(k_H)$ .

**Proof.** In the case of  $n = \nu_H(k_H)$ ,  $r_H^C(n)$  is equal to  $R_{k_H}$ , the radius of the maximal inscribed circle  $C(\mathcal{H}_{k_H})$  of  $\mathcal{H}_{k_H}$ . Namely,  $r_H^C(n) = \frac{3}{2}k_H r_s$ . By Lemma 7,

$$\frac{r_H^C(n)}{\gamma_C(n)} \geq \frac{\frac{3}{2}k_H r_s}{3\sqrt{\frac{\sqrt{3}}{2\pi}}k_H r_s} = \sqrt{\frac{\pi}{2\sqrt{3}}} > 0.95. \quad \square$$

**Lemma 9.**  $r_H^C(n) \geq 0.95\frac{k_H-1}{k_H}\gamma_C(n)$  for  $n \neq \nu_H(k_H)$ .

**Proof.** In the case of  $n \neq \nu_H(k_H)$ ,  $r_H^C(n)$  must not be less than  $R_{k_H-1}$ , i.e.,  $r_H^C(n) \geq \frac{3}{2}(k_H - 1)r_s$ . By Lemma 7,

$$\frac{r_H^C(n)}{\gamma_C(n)} \geq \frac{\frac{3}{2}(k_H - 1)r_s}{3\sqrt{\frac{\sqrt{3}}{2\pi}}k_H r_s} = \frac{k_H - 1}{k_H} \sqrt{\frac{\pi}{2\sqrt{3}}} = 0.95\frac{k_H - 1}{k_H}. \quad \square$$

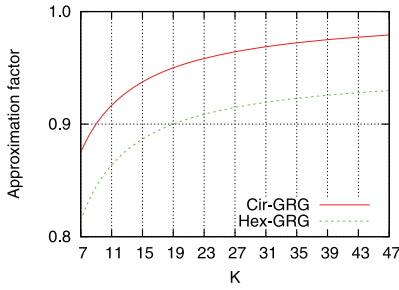


Fig. 6. Approximation factor in relation to  $k$ .

Summarizing Lemmas 8 and 9, we have the following theorem:

**Theorem 8.**  $r_H^C(n) \geq 0.95\delta\gamma_C(n)$ , where  $\delta = 1$  if  $n = \nu_H(k_H)$ , or  $\delta = 1 - \frac{1}{k_H}$  otherwise.

According to Theorems 7 and 8, it would appear that the resultant circular radius of GRG was far from optimal in small-size network. For instance, if  $k = 2$ , the lower bound will be  $0.5\gamma_C(n)$  for Cir-GRG and  $0.475\gamma_C(n)$  for Hex-GRG. This is however not true, as indicated by the following theorem, because the lower bounds are too coarse in the case of  $k \leq 6$ .

**Theorem 9.**  $r_H^C(n) = r_C(n) > 0.86\gamma_C(n)$  for  $n \neq \nu(k) \wedge k \leq 7$ .

**Proof.** By (3),  $\mathcal{H}_k \equiv \mathcal{C}_k$ , i.e.,  $r_H^C(n) = r_C(n)$ , for  $k < 8$ . In this case, when  $n \neq \nu(k)$ ,  $\gamma_C(n)$  is bounded above by  $\frac{3}{2}kr_s$  and the radius  $R'_{k-1} = \sqrt{3}(k-1)r_s$  of the circumcircle of  $\mathcal{H}_{k-1}$ . Enforcing  $R'_{k-1} \leq \frac{3}{2}kr_s$ , we get  $k \leq 7$ . Thus,

$$\frac{r_H^C(n)}{\gamma_C(n)} \geq \frac{\frac{\sqrt{3}}{2}R'_{k-1}}{R'_{k-1}} = \frac{\sqrt{3}}{2} > 0.86$$

for  $k \leq 7$ .  $\square$

Fig. 6 is obtained from Theorems 7 and 8. It shows the lower bound (of the approximation factor) of the circular coverage radius for  $k \geq 7$ . It is observed that GRG produces increasingly near optimal F-coverage as the network size increases.

## 6 PERFORMANCE EVALUATION

Although sensor self-deployment is not a new research issue, maximizing coverage radius for F-coverage is addressed for the first time in this paper. Existing sensor self-deployment algorithms are either nonlocalized or may possibly yield a network with coverage radius as bad as 0. Because we emphasize on optimizing coverage radius by localized solutions, they are not comparable to our proposed main algorithm (GRG) here. Thus, we are going to comparatively evaluate GA and GRG only.

### 6.1 Evaluation Metrics

We evaluate the performance of GA and GRG in three aspects: *convergence time*, *energy consumption*, and *node collision*. Because nodes obtain their neighborhood information from lower layer protocols (e.g., routing protocol), and they themselves do not generate any message during the

course of self-deployment, communication cost is not our concern.

Convergence time, also known as *deployment latency*, is defined as the number of time units ( $nT$ ) that it takes a self-deployment algorithm to yield a stabilized network (with no floating nodes). When  $n \neq \nu(k)$ , we consider from guaranteed coverage viewpoint that GRG converges once the  $k - 1$  inner polygons are fully filled.

Energy consumption is measured by *number of moves* ( $nM$ ), *mileage* ( $Mg$ ), *progress* ( $Pg$ ), and *mileage over progress ratio* ( $MoP$ ).  $nM$  and  $Mg$  are, respectively, defined as the number of times a node started its motor and the total distance it traveled for self-deployment. Let  $D_{ini}$  and  $D_{fin}$ , respectively, be its initial and its final euclidean distance to  $F$ . We define  $Pg = |D_{ini} - D_{fin}|$ . Then,  $MoP = \frac{Mg}{Pg}$ . It gives an idea about how costly zigzag node movement is.

Node collision happens when two nodes are present at the same vertex. It is due to randomized initial node placement and/or algorithmic design. Although node collision appears as transient phenomenon both in GA and in GRG, it matters because it could bring colliding nodes radio signal interference at physical layer in practice, causing various communication failure. We measure number of node collisions ( $nC$ ) in simulation.

We say that a node conducted a wasted greedy step if it performs a retreat step immediately after. Wasted greedy movement increases both convergence time and energy consumption. Since it results only from node collision, measuring  $nC$  during sensor self-deployment can help further clue in algorithm performance.

### 6.2 Simulation Setup

We implemented GA and GRG (including CW and CV variants) within a custom network simulator, and simulated their execution over a MSN randomly dropped in 2D free plane. The geographic center of the dropping area is taken as POI (i.e.,  $F$ ). Sensors are equipped with sensing radius 10 and communication radius  $10 \times \sqrt{3} \approx 18$ . They may move at different speeds, ranging from 0.05 to 0.2 per simulated time unit, for every step. We conducted two streams of simulation experiments.

In the first stream, we study the performance of GA and GRG by using  $\mathcal{H}$ -polygon as deployment polygon (i.e., Hex-GA and Hex-GRG). We first evaluate their performance under different node densities by fixing the size  $Sz$  of the dropping area to  $200^2$  and varying network size  $n$  from  $\nu_H(1) = 7$  to  $\nu_H(10) = 331$ . Then, we evaluate them with different average initial node distance to POI by fixing  $n$  to  $\nu(7) = 169$  and varying  $Sz$  from  $0^2$  to  $500^2$ .  $Sz = 0^2$  implies that nodes are all initially placed at POI at initiation.

According to Section 5, Hex-GRG produces near optimal circular F-coverage. So in the second stream, we wish to study the performance of Hex-GRG and Cir-GRG (where  $\mathcal{C}$ -polygon is used as deployment polygon) for achieving the same target circular coverage radius, referred to as  $cR$ . We fix  $Sz$  to  $500^2$  and vary  $cR$  from 8 to 18 by changing the network size  $n$  according to (1) and (3). In both streams, for each simulation setting we run the tested algorithms over 50 randomly generated network scenarios in order to get average results.

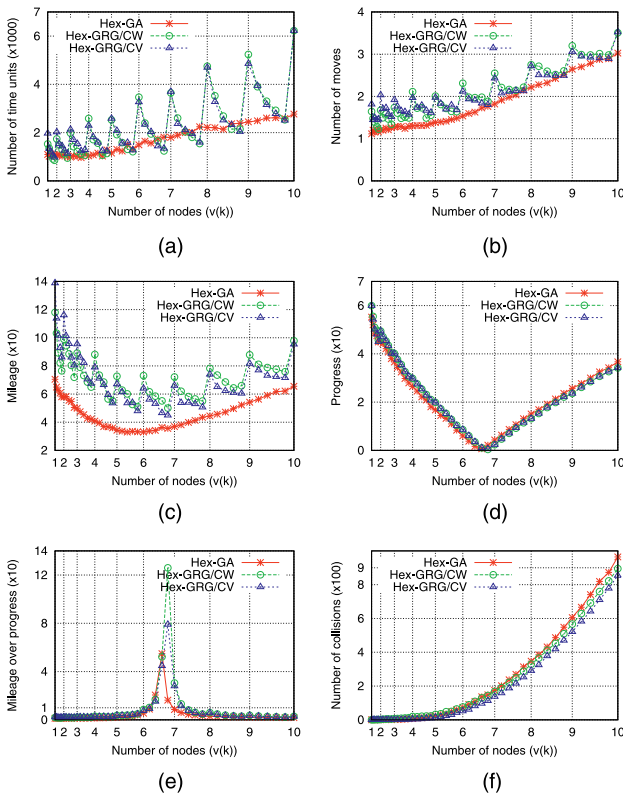


Fig. 7. GA and GRG with  $\mathcal{H}$ -polygon ( $Sz = 200^2$ ). (a) Convergence time ( $cT$ ). (b) # of moves per node ( $nM$ ). (c) Mileage per node ( $Mg$ ). (d) Progress per node ( $Pg$ ). (e) Mileage over Progress ( $MoP$ ). (f) # of node collisions ( $nC$ ).

### 6.3 Hex-GA versus Hex-GRG

Below we will elaborate on our simulation results displayed in Figs. 7 and 8. As we will see, Hex-GA outperforms Hex-GRG in the aspects of convergence time and energy consumption; Hex-GRG/CV is more suitable for dense networks when compared with Hex-GRG/CW.

#### 6.3.1 Varied-Sized Network with Fixed-Sized Field

Examine Figs. 7a and 7b, which, respectively, indicate  $cT$  and  $nM$  as a function of network size  $n$  and contain curves of similar trend. We first investigate the monotonically increasing curves of Hex-GA. When  $n = \nu_H(1)$ , the network is very sparse and has a very small size of 7. In such a network, greedy advance overwhelmingly dominates the self-deployment process, and nodes move most of time without frequently (or even never) being blocked and waiting, resulting in low-valued  $cT$  and  $nM$ . As  $n$  increases, the frequency of blocking and nodal retreat rises, and waiting and resuming happens more often. As a result, both  $cT$  and  $nM$  increase.

Now, let us look at the curves for Hex-GRG/CW and Hex-GRG/CV in the two figures. If we link the points with  $n = \nu_H(k)$ , we get two closely located monotonically increasing curves in both figures. In either figure, the two new curves are both located above the curve for Hex-GA. It is because Hex-GRG involves extra rotation movement, which complexes the self-deployment process. Observe any interval between  $\nu_H(k-1)$  and  $\nu_H(k)$  for an integer  $k$ , and we find that the curve of either variant of Hex-GRG

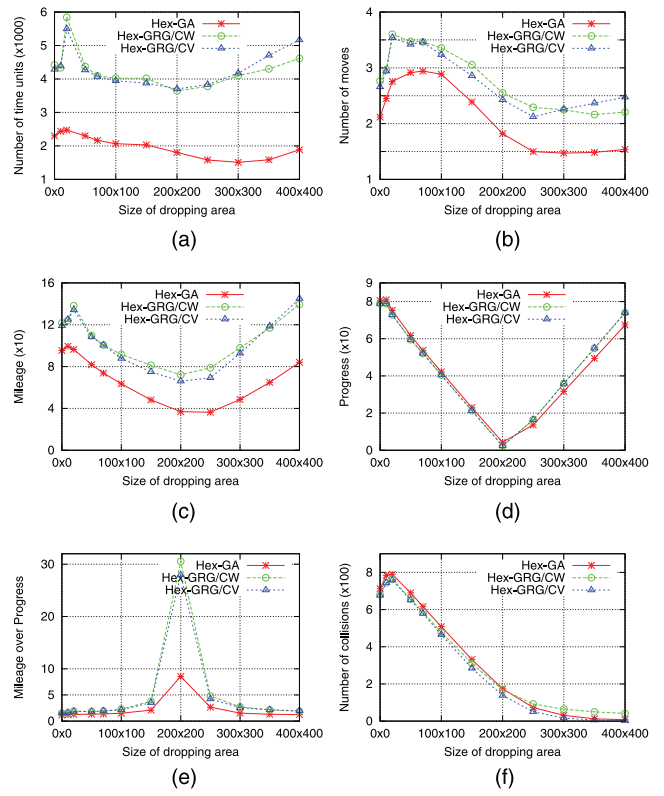


Fig. 8. GA and GRG with  $\mathcal{H}$ -polygon ( $n = \nu(7)$ ). (a) Convergence time ( $cT$ ). (b) # of moves per node ( $nM$ ). (c) Mileage per node ( $Mg$ ). (d) Progress per node ( $Pg$ ). (e) Mileage over Progress ( $MoP$ ). (f) # of node collisions ( $nC$ ).

descends in this interval, which is reasonable. In the case of  $n = \nu_H(k)$ , Hex-GRG does not converge until the outermost hexagon is fully occupied; in any other case, it converges as soon as all the inner  $k-1$  hexagons are fully filled, making a dramatic decrease of both  $cT$  and  $nM$ . In fact, when  $n$  is very close to  $\nu_H(k)$ , Hex-GRG performs even better than Hex-GA, as shown in the two figures, since the latter converges only when all nodes stop moving.

Fig. 7c shows how  $Mg$  varies as  $n$  changes. It is observed that the curves for Hex-GRG/CW and Hex-GRG/CV have a declining trend when  $n$  lies between  $\nu_H(k-1)$  and  $\nu_H(k)$  for an integer  $k$ . This phenomenon is due to exactly the same reason as the similar phenomena observed in Figs. 7a and 7b. If we link points on Hex-GRG curves with  $n = \nu_H(k)$ , we also get two closely located monotonically increasing curves. The two new curves surpass the curve for Hex-GA for every value of  $n$  because Hex-GA does not generate rotation movement. Besides, they also have the same trend as Hex-GA: first declining and then increasing. It is because, as  $n$  increases, the network becomes denser, and  $D_{fin}$  thereby decreases and approaches  $D_{ini}$ , which in turn makes nodes travel a decreased distance. But, after node density is beyond a saturated value (when  $n$  is around  $\nu_H(6)$ ), the network shows an expanding behavior, i.e., that nodes move outward for coverage maximization, leading to the monotonic increase of  $Mg$  with increased  $n$ .

Examine the three Figs. 7a, 7b, and 7c again. We can find that Hex-GRG/CW performs better in sparse networks, but worse in dense networks, than Hex-GRG/CV. This phenomenon is arguable. When  $n$  is small, greedy advance

dominates sensor self-deployment, and node collision, which has obvious negative impact on  $cT$ ,  $nM$ , and  $Mg$ , happens rarely. In this case, aggressive Hex-GRG/CW beats conservative Hex-GRG/CV, as the latter often unnecessarily forces nodes to travel increased distance. As  $n$  increases, the network shows more a rotating or expanding behavior, and node collision occurs increasingly often, as confirmed by Fig. 7f and discussed in next paragraph. The positive impact of the strict hop selection rules of Hex-GRG/CV keeps growing, while their negative effect constantly decrease, finally rendering it outperform Hex-GRG/CW.

Fig. 7d illustrates  $Pg$  as a result of  $n$ . The curves corresponding to Hex-GA and the two versions of Hex-GRG are all in a "V" shape with the lowest point rooted around  $n = \nu_H(7)$ . They imply that this particular value of  $n$  makes the network reach a saturated status, namely,  $D_{ini}$  is roughly equal to  $D_{fin}$  such that nodes make no (large) progress during the course of self-deployment. Such a network shows a rotating behavior in general. When  $n$  deviates more and more from  $\nu_H(7)$ , the difference between  $D_{ini}$  and  $D_{fin}$  becomes larger, resulting in the rise of  $nC$ . With no difficulty, we can see that the network shows a concentrating behavior when  $n < \nu_H(7)$  and an expanding behavior when  $n > \nu_H(7)$ .

Fig. 7e exhibits  $MoP$  versus  $n$ . It is observed that  $MoP$  is lower than 10 and very close to 1 for both Hex-GA and Hex-GRG almost for all the values of  $n$ . In the figure,  $MoP$  reaches its peak value around  $n = \nu_H(7)$ . In fact,  $MoP$  can go to infinity in the case of  $Pg = 0$ . Although this extreme situation appears highly unlikely, it is possible in theory, for example, when all the nodes are by any chance located at the right deployment points at initiation. Additionally, it is observed that  $MoP$  decreases and approaches 1 closer as  $n$  increases or decreases toward the two end values. Through a comparative study on the two Figs. 7c and 7d, the reason for this phenomenon becomes fairly obvious:  $Pg$  has a way smaller value (nearly equal to 0) than  $Mg$  round  $n = \nu(7)$  and it increases at a much faster speed than  $Mg$  with increased/decreased  $n$ .

Fig. 7f shows  $nC$  in relation with  $n$ . Observe that  $nC$  keeps ascending as  $n$  increases because the probability of node collision increases with node density, which is proportional to network size in the case of fixed-sized dropping area, increases. Also observe that Hex-GRG/CV always yields smaller  $nC$  than Hex-GRG/CW. Recall that Hex-GRG/CV itself does not cause node collision. Collision occurs during its execution only for the sake of randomized initial node placement. As shown in the figure, Hex-GA and Hex-GRG has nearly the same performance in a small-sized network, and they deviate from each other as  $n$  increases. Hex-GRG/CV is below Hex-GA in all cases because rotation helps to reduce retreat-related collision. Hex-GRG/CW is first above GA because it generates a large proportion of greedy-rotation collisions in a sparse network with concentrating behavior, and then gets below Hex-GA (after  $n = \nu_H(6)$ ) because the proportion of greedy-rotation collision diminishes, and that of retreat-related collision avoided by rotation contrarily emerges.

### 6.3.2 Fixed-Sized Network with Varied-Sized Field

Figs. 8a, 8b, and 8c, respectively, show  $cT$ ,  $nM$ , and  $Mg$  as a function of  $Sz$ . The curves in the three figures share a

similar trend. When nodes are all located at POI (i.e., when  $Sz = 0^2$ ), they spread out with equal probability in every direction along TT edges. In this simple scenario, neither rotation nor greedy advance often occurs, and nodes are very likely to travel a short distance to their final position through continuous retreat movement. As  $Sz$  increases, rotation and greedy advance take place more and more frequently, complexing the self-deployment process. In such a complicated situation, nodes are prone to move intermittently and perform wasted greedy advance, rendering the increase of  $cT$ ,  $nM$ , and  $Mg$ . As shown in the figures, the curves keep ascending until  $Sz$  reaches certain value about  $20^2$  and then descends thereafter. It is because  $Sz$  gradually becomes large enough for continuous node movement and for reducing the possibility of wasted greedy advance. But, after the network becomes sufficiently sparse the curves rise again because nodes block each other during their concentrating self-deployment, leading to large amounts of nonprogressive rotation, frequent stops and, therefore, increased waiting time.

In Figs. 8a, 8b, and 8c, Hex-GA is always located below Hex-GRG due to its algorithmic simplicity. It is also observed that Hex-GRG/CV stays below Hex-GRG/CW before  $Sz = 300^2$  and surpasses it thereafter as  $Sz$  increases. The reason for this phenomenon is rooted at the semigreedy nature of Hex-GRG/CV. It generates no greedy-rotation collision due to its strict greedy rules and therefore yields less wasted greedy advance compared with Hex-GRG/CW. Under this circumstance, Hex-GRG/CV outperforms Hex-GRG/CW in a dense network ( $Sz < 300^2$ ), where greedy advance does not occur as often as other types of movement; in an adequately sparse network ( $Sz > 350$ ) where greedy advance is prevailing, Hex-GRG/CW however performs better than Hex-GRG/CV because the latter yields more frequent stop and nonprogressive rotation.

Fig. 8f illustrates  $nC$  versus  $Sz$ . When  $Sz = 0^2$ , collision-prone retreat movement overwhelmingly dominates the self-deployment process and gives rise to high-valued  $nC$ . With a slight increase of  $Sz$ , the occurrence frequency of retreat movement does not change apparently, but rotation and greedy happen relatively much often, leading to the rise of  $nC$ . However, as shown in the figure, this rising trend exists only within a small range of  $Sz$ , from  $0^2$  to  $50^2$ . Note that the boosting phenomenon in this  $Sz$  range happens to other metrics like  $cT$ ,  $nM$ , and  $Mg$  as well (refer to Fig. 8a, 8b, and 8c). After  $Sz$  is beyond  $50^2$ , retreat movement gradually loses its dominating role, and retreat-related collision becomes less and less possible, inducing the monotonic drop of  $nC$ . In fact, if the network is very sparse, the nodes close to POI could possibly stop moving before remote ones arrive, leading to  $nC = 0$ . In a dense network ( $Sz < 200^2$ ), rotation helps reduce the probability of retreat-related collision, making Hex-GRG outperforms Hex-GA. In a sparse network ( $Sz > 200^2$ ), Hex-GRG/CW has worse performance than Hex-GA because it can generate extra greedy-rotation collisions, and Hex-GRG/CV stays superior to Hex-GA as it brings no additional collisions (it in fact prevents greedy collision at POI).

Recall that  $Pg = |D_{ini} - D_{fin}|$ . Because  $n$  stays unchanged, the average node final distance  $D_{fin}$  to POI is fixed, and  $Pg$  depends solely on  $D_{ini}$ , which is in turn subject to the size of dropping area. Fig. 8d shows  $Pg$  in relation with  $Sz$ . It is observed that, for both Hex-GA and

Hex-GRG,  $P_g$  is lowest (nearly equal to 0) when  $S_z = 200^2$ , and that it rises as  $S_z$  approaches the two end values, rendering the curves in “V” shape. This phenomenon indicates that the closer  $S_z$  is to  $200^2$ , the closer  $D_{ini}$  is to  $D_{fin}$ . We can see that the two variants of Hex-GRG have the same performance, and that they always yield larger  $P_g$  than Hex-GA. We can also find that the gap between Hex-GRG and Hex-GA becomes larger and larger after  $S_z$  exceeds  $200^2$ . It is because rotation increases the chance of nodes for greedy advance, especially in a sparse network. Note that the curves in this figure do not match those in Fig. 8d but show a sharper change, because nodes do not move straight to their final position but through curly paths in the TT, especially in GRG that involves rotation.

Fig. 8e exhibits  $MoP$  in relation with  $S_z$ . It is observed that  $MoP$  is lower than five (in fact, very close to 1) for both Hex-GA and Hex-GRG almost for all the values of  $S_z$ . The figure shows that  $MoP$  reaches its peak value when  $S_z = 200^2$ . However, there is no maximum value for  $MoP$  since  $P_g$  could be equal to 0 in theory. Additionally, it is displayed that  $MoP$  decreases and approaches 1 closer and closer as  $S_z$  increases or decreases toward the two end values. Through a comparative study on Figs. 8c and 8d, the reason for this phenomenon becomes fairly obvious:  $P_g$  has a way smaller value (nearly equal to 0) than  $M_g$  at  $S_z = 200^2$ , and it climbs at a much faster speed than  $M_g$  with  $S_z$  increased/decreased from  $200^2$ .

#### 6.4 Hex-GRG versus Cir-GRG

We already know that Cir-GRG requires less nodes than Hex-GRG for achieving the same circular coverage radius. Below we will see that it has shorter convergence time and less node collisions, and consume slightly more energy in sparse networks but much less energy in dense networks. So Cir-GRG is generally superior to Hex-GRG.

Since Cir-GRG uses less nodes than Hex-GRG to achieve the same  $cT$  (specifically, when  $cR > 7$ ), it should as well have smaller  $cT$  than Hex-GRG. This is true as confirmed by Fig. 9a, from which we can see the  $cT$  curve of Cir-GRG always stays below that of Hex-GRG. Notice that the gap between these curves gets bigger and bigger after  $cR$  is beyond number 13. Later, we will see that this “magic”  $cR$  value is the one that leads to the lowest nodal progress of Hex-GRG.

Fig. 9b depicts  $nC$  as a result of  $cR$ . Observe that Hex-GRG always generates a larger  $nC$  than Cir-GRG. It is because Hex-GRG requires more nodes than Cir-GRG for the same  $cR$ . Since the dropping area is fixed, the more nodes, the larger node density, and thus the more often node collision occurs. By Fig. 3, the two algorithms’ difference in required  $n$  becomes bigger and bigger as  $cR$  goes up. This change is reflected by the growing gap of the  $nC$  curves for Hex-GRG and Cir-GRG in Fig. 9b.

$D_{ini}$  is approximately constant as nodes are randomly and uniformly placed at initiation. Observe Fig. 9c and notice the difference between Hex-GRG and Cir-GRG: the curve of Cir-GRG is always above that of GRG. It is because of the larger  $D_{fin}$  of Hex-GRG, which is in turn due to the coverage redundancy of Hex-GRG (between  $\mathcal{H}$ -polygon and  $\mathcal{C}$ -polygon).

Fig. 9d shows the difference between Hex-GRG and Cir-GRG in  $M_g$ . Observe that  $M_g$  reaches the lowest value later in Cir-GRG than in Hex-GRG. It is because Cir-GRG uses

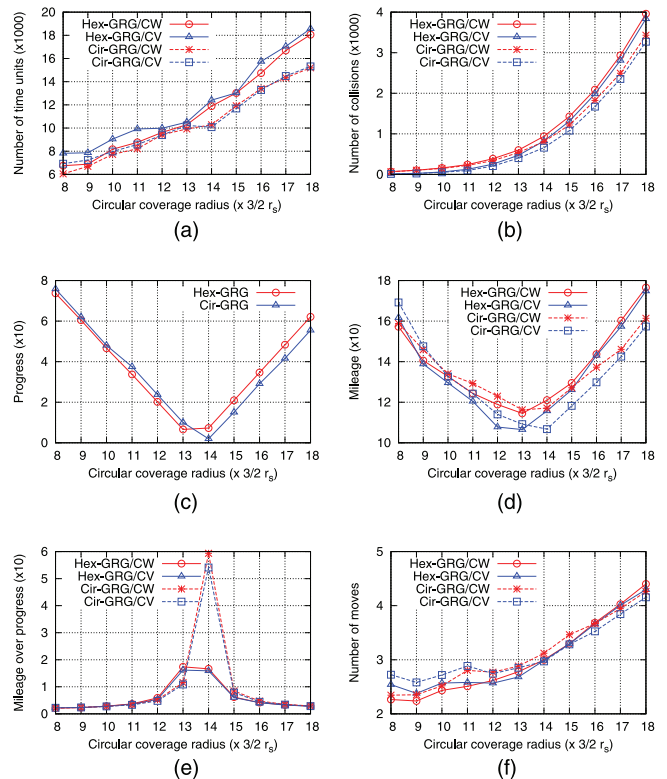


Fig. 9. GRG with  $\mathcal{H}$ -polygon and with  $\mathcal{C}$ -polygon. (a) Convergence time ( $cT$ ). (b) # of collisions ( $nC$ ). (c) Progress per node ( $P_g$ ). (d) Mileage per node ( $M_g$ ). (e) Mileage over progress ( $MoP$ ). (f) # of moves per node ( $nM$ ).

less nodes than Hex-GRG for achieving the same  $cR$ , delaying the behavioral change of the network. We can also observe that the  $M_g$  of Cir-GRG is slightly larger (much smaller) than that of Hex-GRG in concentrating (respectively, expanding) networks.

From Fig. 9e,  $MoP$  is most of time below 5; it increases dramatically only when  $cR$  is around the value leading to lowest  $P_g$  (see Fig. 9c), although  $M_g$  approaches the lowest value at the same time (see Fig. 9d). Again, as shown in the figure, the economic node usage of Cir-GRG delays the appearance of the peak value of  $MoP$ ; Hex-GRG and Cir-GRG have nearly same  $MoP$  performance in other cases.

In Cir-GRG, nodes have to stop moving more often than in Hex-GRG due to the complex neighborhood pattern. On the other hand, Hex-GRG requires more nodes than Cir-GRG for achieving the same  $cR$  and therefore possibly causes relatively frequent node blocking. As confirmed by Fig. 9f, when  $cR$  is not too large compared with the dropping area, in other words, when node density is moderate, the impact of hop selection rules dominates the algorithm performance on  $nM$ , making Hex-GRG generate smaller  $nM$  than Cir-GRG; but it is slowly overwhelmed by the latter as  $cR$  becomes increasingly large, rendering Cir-GRG eventually overtaking Hex-GRG.

## 7 DISCUSSIONS

GA and GRG were presented and evaluated in the scenario of  $\frac{r_c}{r_s} \geq \sqrt{3}$  in previous sections. Their design however does not rely on this ratio. Generally speaking, we first need to find the node arrangement pattern for optimal area coverage. Then

based on this pattern, we identify the successive polygon layers around POI for sensor placement and their localized computation. After that, we do neighborhood division according to these polygons (see Section 3.3). Finally, GA or GRG may be applied directly. For example, when  $\frac{1}{2}3^{\frac{1}{2}} \leq \frac{r_c}{r_s} \leq \sqrt{2}$ , a square grid layout produces optimal area coverage [1], and in this case, we naturally choose squares or minimum area polygons containing the inscribed circles of the squares as deployment polygons. Correctness and optimality analysis will follow the same technique as detailed in Section 5. Polygonal coverage radius maximization remains, regardless of the  $\frac{r_c}{r_s}$  ratio. The optimality of circular coverage radius maximization will be different since the deployment polygons used approximate circles differently. The performance on convergence time and related metrics will be different too. Due to space limitations, here we only addressed the scenario of  $\frac{r_c}{r_s} \geq \sqrt{3}$ .

GA and GRG were described and studied in the context of a single POI. As we have seen, this case is not trivial to resolve. It involves careful design of localized rules that govern the sensors movement and guarantee termination and coverage optimality. The challenge stems from the lack of global knowledge. When there are multiple POIs, we are dealing with not only sensor self-deployment but also POI assignment to sensors. Having been assigned a POI, sensors run GA or GRG directly to construct F-coverage around that POI. A simple solution is random assignment. However, it is not efficient since sensors may be shared between closely located POIs. POI assignment is in essence a multirobot task allocation problem, which has been studied for years in robotics and is not our focus here. GA and GRG will serve as the basic techniques that later can be extended to more complex protocols, for example, for multi-POI scenarios.

In this paper, we did not consider localization errors and transmission errors, which are common in reality. The former may cause sensing holes, and GA and GRG therefore lose their coverage optimality. Transmission errors will cause incomplete neighborhood information, and sensors may therefore make wrong deployment decisions, resulting in node collision. Although GA and GRG are able to resolve node collisions, their performance on deployment latency and energy consumption will decrease. The detailed study on how localization errors and transmission errors affect the algorithms performance is part of future work.

## 8 CONCLUSIONS

Mobile sensor networks have been extensively studied in recent years [15], [16], [17], [18]. In this paper, we pinpointed a new sensor self-deployment problem, F-coverage formation and introduced an evaluation metric, coverage radius. By converting the area coverage problem to a vertex coverage problem on a virtual equilateral triangulation, we proposed the first localized solutions, i.e., Greedy Advance and Greedy-Rotation-Greedy, with desired coverage guarantee. We proved their correctness, and studied their properties and performance by thorough analysis and simulation.

## ACKNOWLEDGMENTS

This research was partially supported by NSERC Strategic grant STPSC 356913 - 07 (maintaining fault-tolerant networks

of robots for supporting wireless sensor networks), NSERC Discovery grants, and by the following grant: "Innovative electronic components and systems based on inorganic and organic technologies embedded in consumer goods and products," TR32016, Serbian Ministry of Science and Education. Part of this work was done while Xu Li was with the School of Computer Science at Carleton University.

## REFERENCES

- [1] X. Bai, S. Kumar, D. Xuan, Z. Yun, and T.H. Lai, "Deploying Wireless Sensors to Achieve Both Coverage and Connectivity," *Proc. ACM MobiHoc*, pp. 131-142, 2006.
- [2] N. Bartolini, T. Calamoneri, E.G. Fusco, A. Massini, and S. Silvestri, "Snap & Spread: A Self-Deployment Algorithm for Mobile Sensor Networks," *Proc. IEEE Fourth Int'l Conf. Distributed Computing in Sensor Systems (DCOSS '08)*, pp. 451-456, 2008.
- [3] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage Control for Mobile Sensing Networks," *IEEE Trans. Robotics and Automation*, vol. 20, no. 2, pp. 243-255, Apr. 2004.
- [4] M. Garetto, M. Gribaudo, C.-F. Chiasserini, and E. Leonardi, "A Distributed Sensor Relocation Scheme for Environmental Control," *Proc. IEEE Int'l Conf. Mobile Adhoc and Sensor Systems*, 2007.
- [5] N. Heo and P.K. Varshney, "Energy-Efficient Deployment of Intelligent Mobile Sensor Networks," *IEEE Trans. Systems, Man, and Cybernetics Part A: Systems and Humans*, vol. 35, no. 1, pp. 78-92, Jan. 2005.
- [6] 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," *Proc. Distributed Autonomous Robotic Systems (DARS '02)*, 2002.
- [7] X. Li, H. Frey, N. Santoro, and I. Stojmenovic, "Localized Self-Deployment of Mobile Sensors for Optimal Focused-Coverage Formation," Technical Report 2007-13, Univ. of Ottawa, Dec. 2007.
- [8] X. Li, H. Frey, N. Santoro, and I. Stojmenovic, "Focused Coverage by Mobile Sensor Networks," *Proc. IEEE Sixth Int'l Conf. Mobile Adhoc and Sensor Systems (MASS '09)*, pp. 466-475, 2009.
- [9] X. Li, H. Frey, N. Santoro, and I. Stojmenovic, "Localized Sensor Selfdeployment for Guaranteed Coverage Radius Maximization," *Proc. IEEE Int'l Conf. Comm. (ICC '09)*, 2009.
- [10] X. Li, A. Nayak, D. Simplot-Ryl, and I. Stojmenovic, "Sensor Placement in Sensor and Actuator Networks," *Wireless Sensor and Actuator Networks: Algorithms and Protocols for Scalable Coordination and Data Communication*, Wiley, 2010.
- [11] M. Ma and Y. Yang, "Adaptive Triangular Deployment Algorithm for Unattended Mobile Sensor Networks," *IEEE Trans. Computers*, vol. 56, no. 7, pp. 946-847, July 2007.
- [12] G. Wang, G. Cao, and T.L. Porta, "Movement-Assisted Sensor Deployment," *IEEE Trans. Mobile Computing*, vol. 5, no. 6, pp. 640-652, June 2006.
- [13] S. Yang, M. Li, and J. Wu, "Scan-Based Movement-Assisted Sensor Deployment Methods in Wireless Sensor Networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 18, no. 8, pp. 1108-1121, Aug. 2007.
- [14] H. Zhang and J.C. Hou, "Maintaining Sensing Coverage and Connectivity in Large Sensor Networks," *Ad Hoc & Sensor Wireless Networks*, vol. 1, pp. 89-124, 2005.
- [15] K. Akkaya, I. Guneydas, and A. Bicak, "Autonomous Actor Positioning in Wireless Sensor and Actor Networks Using Stable-Matching," *Int'l J. Parallel, Emergent and Distributed Systems*, vol. 25, no. 6, 2010.
- [16] M. Esnaashari and M.R. Meybodi, "Dynamic Point Coverage Problem in Wireless Sensor Networks: A Cellular Learning Automata Approach," *Ad Hoc & Sensor Wireless Networks*, vol. 10, nos. 2-3, pp. 193-234, 2010.
- [17] S. He, J. Chen, Y. Sun, D.K.Y. Yau, and N.K. Yip, "On Optimal Information Capture by Energy-Constrained Mobile Sensors," *IEEE Trans. Vehicular Technology*, vol. 59, no. 5, pp. 2472-2484, 2010.
- [18] K. Selvaradjou and C. Siva Ram Murthy, "Maximizing Lifetime of Mobile Actors in Wireless Sensor and Actor Networks," *Ad Hoc & Sensor Wireless Networks*, vol. 9, nos. 3-4, pp. 179-202, 2010.

► For more information on this or any other computing topic, please visit our Digital Library at [www.computer.org/publications/dlib](http://www.computer.org/publications/dlib).