Focused-Coverage by Mobile Sensor Networks

Xu Li*, Hannes Frey†, Nicola Santoro‡, and Ivan Stojmenovic*

*SITE, University of Ottawa, Canada – {xuli, ivan}@site.uottawa.ca
†CS, University of Paderborn, Germany – hannes.frey@uni-paderborn.de
‡SCS, Carleton University, Canada – santoro@scs.carleton.ca

***Abstract*—We pinpoint a new sensor self-deployment problem, *constructing focused coverage around a Point of Interest (POI)*, and introduce an evaluation metric, *coverage radius*. We propose two solutions, *Greedy Advance (GA)* and *Greedy-Rotation-Greedy (GRG)*, which are to our knowledge the first sensor self-deployment algorithms that operate in a purely localized manner and yet provide coverage guarantee. The two algorithms drive sensors to move along a locally-computed equilateral triangle tessellation (TT) 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 to a TT vertex where greedy advance can resume. They both yield a connected network of TT layout with hole-free coverage; GRG furthermore assures a hexagon coverage shape centered at POI. We prove their correctness and analyze their coverage radius property. Our study shows that GRG guarantees optimal hexagonal coverage radius and near optimal circular coverage radius. Through extensive simulation we as well evaluate their performance on convergence time, energy consumption, and node collision.**

I. Introduction

Sensor self-deployment is an important research issue that deals with autonomous coverage formation in mobile sensor networks (MSN). Considering network scalability, unpredictable node failure, dynamic topological change, and narrow network bandwidth, a solution algorithm should be carried out in a localized manner. Term "localized" means that each node makes its self-deployment decision independently, using its k-hop neighborhood information for a constant k. When $k = 1$, 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, called Point of Interest (POI). One example are sensors scattered around a chemical plant to monitor its distance-dependent pollutional impact on the soil/air in the vicinity. They 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*. In this paper, we address how to achieve optimal focused-coverage by sensor self-deployment.

A. Focused 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 holes are two key evaluation metrics for traditional area coverage. They reflect the sensitivity of a sensor network over a Region of Interest (ROI). An ideal area coverage has maximized area and no sensing hole. In the focused coverage problem, measuring area and hole existence is no longer sufficient, because distance from POI to uncovered areas also makes significant sense and must be taken into consideration. In this case, we introduce an additional metric, *coverage radius*.

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

Optimal focused coverage has maximized radius. In continuous domain, there exists a sensor node at every point in the coverage region, and the maximum hole-free disc therefore has a circular shape. In this case, coverage radius is called *circular radius* and measured by Euclidean distance. In discrete domain, the shape of the disc is however not circular but polygonal, and coverage radius is thus referred to as *polygonal radius* and alternatively measured by *layer distance*. Layer distance, also called convex 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 P_i $(i = 1, 2, \cdots)$ 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 sensor network's coverage region.

B. Problem statement

We consider an asynchronous MSN of unknown size randomly dropped in a 2D free field (e.g., an area on ocean surface in practice) and may possibly be disconnected at initiation. Sensors bear the same communication radius r_c and the same sensing radius r_s. They move asynchronously possibly at different speeds. Sensors know the location of POI, denoted by $\mathcal{P}$. Because the global coordinate system can be easily (through trivial local processing) converted to one with $\mathcal{P}$ as origin, we use $(0, 0)$ as $\mathcal{P}$ without loss of generality.

The goal is to develop a strictly localized sensor self-deployment algorithm that yields a network surrounding $\mathcal{P}$ with an equilateral triangle tessellation (TT) layout. The reasons why this TT layout is required are that it maximizes the coverage area of a given number of nodes without coverage

Part of this work was done when the first author was with SCS, Carleton University, Canada.

gap when nodal 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 $\mathcal{P}$.

We consider this 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 GPS devices or any effective localization algorithm; (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).

C. Our contributions

In this paper, we identify a new sensor self-deployment problem, *achieving focused coverage around a Point of Interest (POI)*, and introduce an evaluation metric, *coverage radius*, that reflects the significance of distance from POI to uncovered areas. We propose two strictly localized solution protocols, *Greedy Advance (GA)* and *Greedy-Rotation-Greedy (GRG)*, which convert the area coverage problem to a vertex coverage problem over a locally-computable equilateral triangle tessellation (TT). In the two algorithms, self-governing sensors relocate themselves on the TT grid and move from vertex to vertex to surround POI, according to their one-hop neighborhood information only. Specifically, in GA, nodes greedily proceed as close to POI as they can; in GRG, when their greedy advance is blocked, nodes rotate around POI to a TT vertex where greedy advance can resume. In both algorithms, when sensors are compactly placed or collide, they may move away from POI. Both GA and GRG yield a connected network of TT layout with hole-free coverage; GRG furthermore assures a hexagon coverage shape centered at POI. Thanks to their purely localized nature, the two algorithms are resilient to node addition and removal (failure) and work regardless of network disconnectivity. We prove their correctness and analyze their coverage radius property. Our study shows that GRG guarantees optimal hexagonal coverage radius, and near optimal circular coverage radius. We finally evaluate their performance on convergence time, energy consumption, and node collision through extensive simulation.

II. Related work

To our knowledge there is no previous work addressing the focused coverage formation problem. Sensor self-deployment algorithms for area coverage 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 article [10].

The most known sensor self-deployment approach is vector-based approach. Algorithms that belong to this category include [5], [8], [11], [12], just to name a few. The basic idea is: each node computes movement vectors for its neighbors in rounds using their relative position and then moves according to the vector summation.

Howard et al. [7] proposed an incremental sensor self-deployment algorithm. In this algorithm sensors are deployed

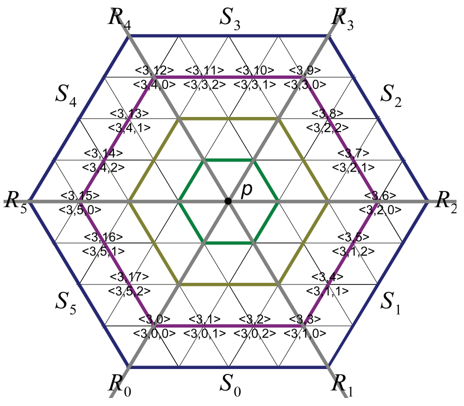


Fig. 1. Triangle Tessellation

one at a time using an occupancy/configuration grid, based on the information collected from previously deployed sensors. This is a centralized algorithm.

Heo and Varshney [6] 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 include [4], [12].

Chellappan et al. [3] presented a centralized algorithm for mobility-limited sensors. They divide the target field into weighted regions and model the sensor self-deployment problem as a minimum-cost maximum-flow problem.

Yang et al. [13] presented a scan-based sensor deployment scheme (SMART). By this scheme, the target field is partitioned into a 2D mesh, and nodes are treated as load. The goal is then converted to load balancing among mesh cells through multi-rounds of scan.

Bartolini et al. [2] presented a snap and spread self-deployment scheme. Sensors simultaneously construct a hexagonal tiling portion for ROI by pushing and pulling sensors to hexagon centers. Tilling portions of different sensors merge when they meet.

These existing algorithms, when used for focused coverage formation, have no guarantee on coverage radius (in worst case, the resulting coverage radius can be as bad as 0). 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.

III. Equilateral Triangle Tessellation

An equilateral triangle tessellation (TT) is a planar graph composed of congruent equilateral triangles. Given an orientation, say north, and edge length l_e, each sensor is able to locally compute a unique TT containing $\mathcal{P}$ as vertex. Denote the TT graph by G_{TT}. In our work, l_e is set to $\sqrt{3}r_s$. It is because we want to finally locate sensors on vertices of G_{TT}, and this particular edge length ensures connectivity and minimizes sensing range overlapping [1], [11], [14].

Let $SP(u,v)$ represent the shortest path connecting two vertices u and v in G_{TT}. Then the *TT distance* between u and v is defined as the number of edges in $SP(u,v)$ and referred to as $|SP(u,v)|$. There are $6i$ vertices with equal TT distance i to $\mathcal{P}$ in G_{TT}. They constitute a distance-i hexagon, denoted by $\mathcal{H}_i$. These $\mathcal{H}$ hexagons are concentric to $\mathcal{P}$. The total number $\nu(i)$ of vertices enclosed by $\mathcal{H}_i$ (inclusive) is

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

We call the vertices located at hexagon corners *corner vertices*, the others *edge vertices*. Corner vertices form 6 rays $\mathcal{R}_0, \ldots, \mathcal{R}_5$, in counterclockwise order, jointing at $\mathcal{P}$ with mutual angle of $\frac{\pi}{3}$ and divide the entire plane evenly into 6 sectors. The sector toward south is named "Sector 0" and denoted by $\mathcal{S}_0$; the other sectors are named after their sequence number after $\mathcal{S}_0$ in counterclockwise direction. For $0 \le j \le 6$, $\mathcal{S}_j$ is defined by two rays $\mathcal{R}_j$ and $\mathcal{R}_{(j+1)\%6}$, where $\%$ stands for modulus operation; its clockwise next sector and counterclockwise next sector are $\mathcal{S}_{(j+5)\%6}$ and $\mathcal{S}_{(j+1)\%6}$, respectively. G_{TT} is drawn in Fig. 1, where sectors and rays are labeled, and $\mathcal{H}$ hexagons are highlighted.

We assign every vertex v on $\mathcal{H}_i$ an in-hexagon index and an in-sector index. The former, denoted by k' ($0 \le k' < 6i$), is equal to the TT distance from v to $\mathcal{R}_0$ along $\mathcal{H}_i$ in clockwise direction; the latter, referred to as k ($0 \le k < i$), is equal to the TT distance from v to $\mathcal{R}_j$ along $\mathcal{H}_i$ in clockwise direction within its residing sector $\mathcal{S}_j$. Notice that $k'\%i = 0$ and $k = 0$ if v is corner vertex. We can uniquely address v using either a pair $\langle i, k' \rangle$ or a triple $\langle i, j, k \rangle$, which are mutually convertible by $k' = ij + k$, $j = \lfloor \frac{k'}{i} \rfloor$, and $k = k'\%i$. Figure 1, displays the addresses of vertices on $\mathcal{H}_3$ in both formats.

In the sequel, a vertex's address will be expressed in the format of $\langle i, j, k \rangle$. We define the address of $\mathcal{P}$ as $\langle 0, \star, 0 \rangle$, where $\star$ can be any non-negative integer less than 6. The geographic coordinate of any $\langle i, j, k \rangle$ can be easily computed.

IV. Localized Self-deployment Algorithms

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

Based on the above intuition, we propose two strictly localized sensor self-deployment algorithms, Greedy Advance (GA) and Greedy-Rotation-Greedy (GRG), which are both resilient to node failure and able to operate regardless of network partition. The two algorithms are composed of a set of simple hop selection rules. By these rules, nodes make their self-deployment decision using merely 1-hop neighborhood information and move asynchronously toward $\mathcal{P}$ step by step. They stop when no next hop is available.

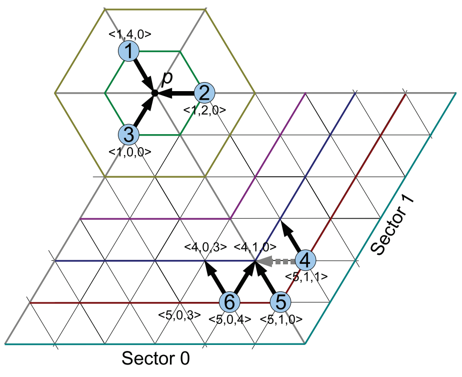


Fig. 2. Hop selection in GA

For simplicity, a TT vertex v is said to be *occupied* by a node if the node is not moving and is located in close proximity to v, or if the node is moving toward v; $\mathcal{P}$ is also considered occupied in the case that it is not physically occupiable. We assume for the time being that sensors are all initially located at distinct vertices of G_{TT}. This temporary assumption will be relaxed immediately after, in Sec. IV-D.

A. Greedy Advance (GA)

In GA, a node moves greedily along TT edges as close to $\mathcal{P}$ in terms of TT distance as it can. It has one and only one possible next hop $\langle i-1, j, k \rangle$ if its residing vertex $\langle i, j, k \rangle$ is a corner vertex (i.e., $k = 0$), or two possible next hops $\langle i-1, j, k-1 \rangle$ and $\langle i-1, j, k \rangle$ if, otherwise, it is an edge vertex (i.e., $0 < k < i$). Figure 2 shows six nodes and their possible next hops, which are marked by thick arrowed lines. Each node chooses from its next hop candidates one that will not cause node collision according to its best local knowledge. If no such a next hop is available, it stays still. A still node resumes greedy advance whenever possible.

Consider an arbitrary edge vertex, say $\langle 4, 0, 3 \rangle$. It has two possible previous hops $\langle 5, 0, 3 \rangle$ and $\langle 5, 0, 4 \rangle$. Examine the example scenario given in Fig. 2. If $\langle 4, 0, 3 \rangle$ is chosen as next hop by node 6 and another node (which is not shown in the figure) at $\langle 5, 0, 3 \rangle$ at the same time, node collision will likely occur. However, since the two nodes are neighboring each other, they know about the potential collision and thus can prevent it from actual happening by the following rule:

***Rule** IV-A.1 (Priority Rule):* If two nodes are greedily moving to $\langle i, j, k \rangle$ from $\langle i+1, j, k \rangle$ and $\langle i+1, j, k+1 \rangle$ (or $\langle i+1, (j+5)\%6, i \rangle$ if $k = 0$), the one from $\langle i+1, j, k+1 \rangle$ (resp., $\langle i+1, j, k \rangle$) has higher priority to proceed.

Special attention should be paid to any corner vertex $\langle i, j, 0 \rangle$ that has totally three possible previous hops $\langle i+1, (j+5)\%6, i \rangle$, $\langle i+1, j, 0 \rangle$, and $\langle i+1, j, 1 \rangle$. Let us again examine the scenario in Fig. 2. By the priority rule, neither nodes 4 and 5 nor nodes 5 and 6 may collide at corner vertex $\langle 4, 1, 0 \rangle$. But nodes 4 and 6 may, if they simultaneously move to $\langle 4, 1, 0 \rangle$, and the collision is not locally avoidable since the two nodes are not aware of each other. To eliminate this undesired situation, we can simply force node 4 not to take $\langle 4, 1, 0 \rangle$ as next hop by the following rule:

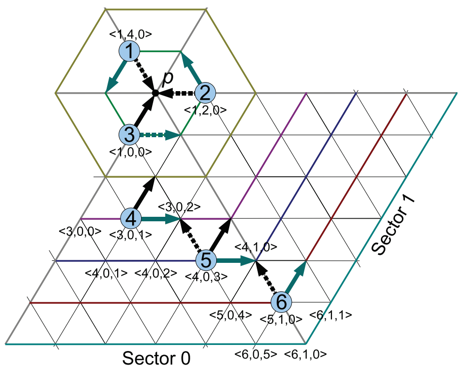


Fig. 3. Hop selection in GRG

***Rule** IV-A.2 (Forbiddance Rule):* A node located at vertex $\langle i+1, j, 1\rangle$ does not chose vertex $\langle i, j, 0\rangle$ as greedy next hop.

In Fig. 2, hop selection that is forbidden by the forbiddance rule is shown by dashed arrowed lines.

$\mathcal{P}$ has six possible previous hops, i.e., the six vertices on $\mathcal{H}_1$, in total. Consider a scenario where $\mathcal{H}_1$ is fully occupied, and $\mathcal{P}$ is not occupied. In this particular case, a deadlock occurs due to the priority rule, and none of the six occupant nodes on $\mathcal{H}_1$ will attempt to move to $\mathcal{P}$. To avoid this deadlock, we define an additional rule as follows:

***Rule** IV-A.3 (Innermost-Layer Rule):* A node located at $\langle 1, j, 0\rangle$ moves to $\mathcal{P}$ as long as $\mathcal{P}$ is not occupied.

The innermost-layer rule may cause node collision at $\mathcal{P}$. However, such a collision takes place at most once, because a node will stay at $\mathcal{P}$ after it reaches $\mathcal{P}$ and no node will try to move to $\mathcal{P}$ once $\mathcal{P}$ is occupied.

B. Greedy-Rotation-Greedy (GRG)

GRG involves not only greedy advance but also a new type of node movement - *rotation*, which forms the final network in a shape of hexagon centered at $\mathcal{P}$. An arbitrary node, when its greedy advance is blocked, tries rotation around $\mathcal{P}$ along its residing hexagon without increasing its TT distance to $\mathcal{P}$. Note that rotation should be restricted to a particular, say counterclockwise, direction so as to avoid collision among rotating nodes. Formally speaking, a node at $\langle i, j, k\rangle$ chooses only $\langle i, j, k+1\rangle$ (or $\langle i, (j+1)\%6, 0\rangle$ if $k = i-1$) as rotation next hop. Figure 3 shows six nodes and their possible next hops, among which rotation next hops and greedy next hops are differentiated using different colors. A node stops rotating when it reaches a vertex where greedy advance can resume, or when it returns to the vertex where it starts rotating. To properly react to its neighborhood change (due to sensor deployment or node failure), a return node resets its rotation starting point to null whenever it finds that its rotation next hop becomes occupied.

In an asynchronous environment, a rotating node on $\mathcal{H}_i$ may never be able to move onto $\mathcal{H}_{i-1}$ despite the vacancies on $\mathcal{H}_{i-1}$, if its neighboring nodes on $\mathcal{H}_{i-1}$ rotate together with it and keep blocking its greedy advance. To prevent this problematic situation, we define the following rule:

***Rule** IV-B.1 (Suspension Rule):* A node located on $\mathcal{H}_{i-1}$, before starting next rotation step, checks if there is any neighbor rotating on $\mathcal{H}_i$. If yes, it gives up its rotation plan.

Consider node 4 in Fig. 3. Suppose that $\langle 3, 0, 0\rangle$ and $\langle 4, 0, 1\rangle$ are both occupied, and that $\langle 3, 0, 2\rangle$ and $\langle 4, 0, 2\rangle$ are both empty. In this case, the greedy advance of node 4 is blocked. According to the suspension rule, node 4 does not rotate to $\langle 3, 0, 2\rangle$. The intuition is that the node knows that, if it itself stays put, the node at $\langle 4, 0, 1\rangle$ will rotate to $\langle 4, 0, 2\rangle$ and then greedily advance to $\langle 3, 0, 2\rangle$. By the suspension rule, a rotating $\mathcal{H}_i$ node will either meet an empty vertex on $\mathcal{H}_{i-1}$, surpassing some $\mathcal{H}_{i-1}$ nodes in between, or find no vacancy on $\mathcal{H}_{i-1}$ and stops at its rotation starting point.

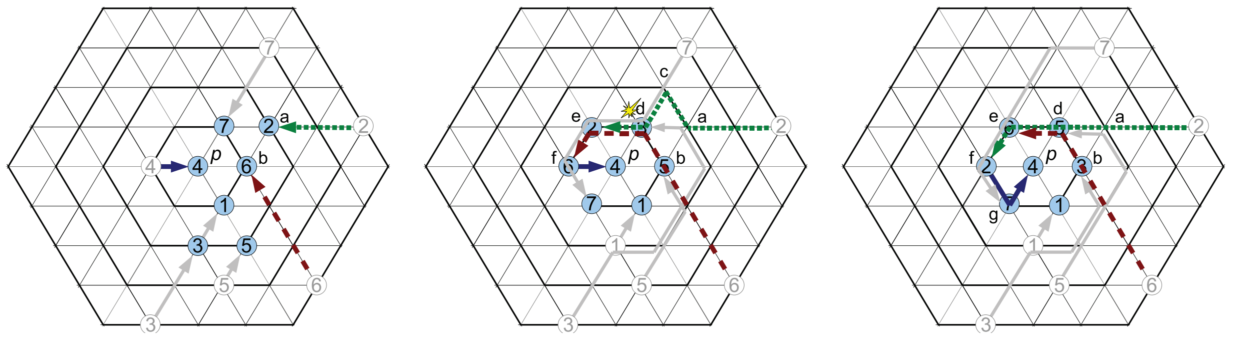
***Rule** IV-B.2 (Competition Rule):* In the case that a greedily advancing node and a rotating node are targeting at the same vertex, the former proceeds as usual, while the latter changes its deployment decision accordingly.

In GRG, each non-POI vertex $\langle i, j, k\rangle$ has two possible previous hops $\langle i+1, j, k\rangle$ and $\langle i+1, j, k+1\rangle$ (or, $\langle i+1, (j+5)\%6, i\rangle$ if $k = 0$) for greedy advance and one previous hop $\langle i, j, k-1\rangle$ (resp., $\langle i, (j+5)\%6, i-1\rangle$) for rotation. As we discussed in Sec. IV-A, greedy-greedy collision does not happen at $\langle i, j, k\rangle$, because the two previous greedy hops are neighboring each other. Observe that vertices $\langle i+1, j, k\rangle$ and $\langle i, j, k-1\rangle$ (or, $\langle i+1, (j+5)\%6, i\rangle$ and $\langle i, (j+5)\%6, i-1\rangle$ if $k = 0$) are also each other's neighbor. Thus the greedy-rotation collision caused by nodes from these two vertices can be locally avoided as well, by the competition rule. Let us examine the greedy-rotation collision due to nodes from $\langle i+1, j, k+1\rangle$ and $\langle i, j, k-1\rangle$ (or, $\langle i+1, j, k\rangle$ and $\langle i, (j+5)\%6, i-1\rangle$ if $k = 0$). Because the two nodes are out of each other's communication range, the collision can not be inferred by them from their local knowledge. Depending on the way of handling this situation, GRG has two variants: **Collision alloWance (CW)** and **Collision aVoidance (CV)**.

1) GRG-CW: In this variant, no additional restriction is applied; greedy-rotation collision is allowed. During a greedy-rotation collision, the rotating node is required to make its next deployment decision first, which immediately affects the other's motion plan. The reason why the rotating node is given priority is to prevent collision loop caused by endless rotating-retreating role switch. We will come back to this in Sec. IV-D, when introducing retreat movement for collision resolution.

Through ordered decision making, greedy-rotation collision could appear as a transient phenomenon. For example, in the scenario given in Fig. 4(b), collision between nodes 2 and 6 takes place at d and is then automatically resolved. However, there is no assurance that collision does not remain permanent (this drawback will be resolved later, in Sec. IV-D).

2) GRG-CV: In this variant, greedy-rotation collision becomes impossible due to the edge rule and the corner rule to be introduced below. The purpose of the two rules is to restrict greedy advance to rotation direction, i.e., counterclockwise direction. The intuition stems from the observation that locally-unknown greedy-rotation collision occurs only when greedy advance and rotation are opposite to each other. For example,



(a) GA (b) GRG-CW (c) GRG-CV

Fig. 4. Final node distribution after sensor self-deployment

in Fig. 3, node 4 and 5 have a potential collision at $\langle 3, 0, 2\rangle$; node 5 and 6 have a potential collision at $\langle 4, 1, 0\rangle$.

***Rule** IV-B.3 (Edge Rule):* A node located at edge vertex $\langle i, j, k\rangle$ only takes $\langle i-1, j, k\rangle$ (or $\langle i-1, (j+1)\%6, 0\rangle$ if $k = i-1$) as the next hop of its greedy advance.

***Rule** IV-B.4 (Corner Rule):* A node located at corner vertex $\langle i, j, 0\rangle$ performs no greedy advance.

Special attention should be paid to the nodes located on $\mathcal{H}_1$. By the corner rule, none of these nodes will move to $\mathcal{P}$, generating a sensing hole at $\mathcal{P}$. Under this circumstance, we appoint a particular vertex, denoted by $Gate(\mathcal{P})$, on $\mathcal{H}_1$ the *gateway* to $\mathcal{P}$ and allow only a gateway node to move to $\mathcal{P}$.

***Rule** IV-B.5 (Gateway Rule):* A node located on $\mathcal{H}_1$ performs only greedy advance if its residing vertex is $Gate(\mathcal{P})$, or only rotation otherwise.

By the corner rule and the gateway rule, any $\mathcal{H}_1$ node not located at $Gate(\mathcal{P})$ has to first rotate to $Gate(\mathcal{P})$ in order to occupy $\mathcal{P}$. A gateway node's greedy advance is safe as no other $\mathcal{H}_1$ node is moving to $Gate(\mathcal{P})$ in the mean time.

Figure 3, where $Gate(\mathcal{P}) = \langle 1, 0, 0\rangle$, shows the possible next hops of 6 nodes in GRG-CV with solid arrowed lines. In the figure, dashed arrowed lines imply the hop selection allowed in GRG-CW but forbidden in GRG-CV. Thanks to the strict hop selection rules of GRG-CV, nodes are aware of, thus able to avoid, any potential collision.

C. Execution examples

In the following, we will comparatively show how GA and the two variants of GRG, i.e., GRG-CW and GRG-CV, work through examples. Although they operate regardless of network size and asynchrony, we consider for ease of understanding a simple fully synchronized scenario, where 7 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 case, a sensor is not able to know where its neighbors are moving and sometimes has to make conservative decision (by assuming those neighbors are staying put). Figure 4 shows the final node distribution obtained respectively by GA, GRG-CW, and GRG-CV. In the figure, node trajectories are marked by thick arrowed lines, pointing from initial position to final position. Note that the initial position of node 1 is the final position of node 3 in Fig. 4(a), and that the initial position of node 4 is the final position of node 6 in Fig. 4(b) and of node 2 in Fig. 4(c). To have a clear view of node movement, we focus only on nodes 2, 4 and 6.

Figure 4(a) indicates that, when GA is applied, the three nodes 2, 4 and 6 move toward $\mathcal{P}$ and stop respectively at a, $\mathcal{P}$ and b according to the forbiddance rule and the innermost-layer rule. As shown in Fig. 4(b), when GRG-CW is employed, node 4 proceeds in the same way as in GA; whereas, nodes 2 and nodes 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 forbidden by the forbiddance rule. Under this circumstance, it has to rotate around $\mathcal{P}$ along its residing hexagon. When node 2 rotates to c, node 6 arrives at b. At that moment, d becomes empty due to node 7's departure, and POI 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. 4(c) for GRG-CV. Because node 4 is not allowed to move to $\mathcal{P}$ by the corner rule, it has to first rotate to a particular gateway vertex (which is set to be g in this example) and then greedily proceed to $\mathcal{P}$ by the gateway rule. Node 7 can not start with greedy advance but has to perform rotation first according to the corner rule, ending up with a completely different trajectory, which directly affects node 2's deployment process: now, after reaching a, node 2 is able to continue its greedy advance and immediately proceed 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 $\mathcal{P}$. Then node 6 has to wait because its can perform neither greedy advance or 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 node 2 and 6 in GRG-CW does not occur in GRG-CV.

D. Resolving node collision

We previously assumed that nodes are initially located at distinct TT vertices, which however rarely happens in practice because of randomized node distribution. This temporary assumption can be readily relaxed by the following rule:

***Rule** IV-D.1 (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.

The alignment rule is very likely to cause various node collision. In the following, we shall introduce a new type of movement - *retreat* - for collision resolution. Retreat is opposite to greedy advance. It happens from a vertex on $\mathcal{H}_i$ ($i \geq 0$) to a vertex on $\mathcal{H}_{i+1}$. With nodal retreat, permanent collision no longer remains, and both GA and GRG gain the ability to spread out compactly-placed sensors.

After some nodes collide at a TT vertex, they enter a local ranking process. These colliding nodes are able to do the ranking locally and independently because they are neighboring each other. During this process, each of them is assigned a rank based on either a random selection or certain criterion (if available) such as residual energy or node ID or the combination thereof. Then, the node with the highest rank makes its next deployment decision first, and the others follow in accordance with the decreasing order of their ranks. If the t-th node decides to stay, every node with rank lower than t retreats by the following rule:

***Rule** IV-D.2 (Retreat Rule):* When a node located at a $\mathcal{H}_i$ vertex decides to retreat, it retreats to one of its neighboring $\mathcal{H}_{i+1}$ vertices that is occupied by the least number of nodes. In case of tie, a random choice is made.

In GRG-CW, retreat movement might cause greedy-rotation collision loop and endless movement in some rare scenario as shown in Fig. 5. Figure 5(a) shows that $\mathcal{P}$ and entire $\mathcal{H}_1$ have been occupied, and that there is only one empty vertex b on $\mathcal{H}_2$. Under this situation, node 1 (located at vertex a on $\mathcal{H}_3$) decides to greedily move to b, and node 2 decides to rotate to b. Because the two nodes are not neighboring each other, they will collide at b as shown in Figure 5(b). Assume that, after the collision, node 1 is assigned a higher rank than node 2. In this case, node 1 decides to stay at b, while node 2 has to retreat onto $\mathcal{H}_{k+1}$ by the retreat rule.

Suppose that node 2 happens to retreat to a and that node 1 meanwhile finishes one rotation step. We end up with a scenario (see Fig. 5(c)) that is exactly the same as the one given in Fig. 5(a). Since we are in an asynchronous environment and we do not assume any specific ranking method, it is possible that similar situation occurs when node 2 makes its greedy advance later on. If continuing that way, each node on $\mathcal{H}_2$ will make full rotation and then retreat onto $\mathcal{H}_3$ rather than stop at its rotation starting point; in the next step the node returns to $\mathcal{H}_2$ again and starts the next rotation. As a consequence, a greedy-rotation collision loop appears, and all nodes are rotating along $\mathcal{H}_2$ infinitively often.

This collision loop is due to the problematic rotating-retreating role switch, which refreshes the rotating node's rotation record. It will not take place if we prevent the rotating node from being retreated outwards, 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**. Notice that, whether the ranking policy is applied or not, collision loop never occurs in GRG-CV where no greedy-rotation collision is possible.

V. Analysis

In this section, we will analyze the correctness and the coverage radius properties of GA and GRG. Due to space limitation, some proofs are sketched, and some are omitted.

***Lemma** 1:* Both GA an GRG ensure that $\mathcal{P}$ be occupied by a single node within finite time.

Proof: Because, after the initial node dropping, the distance from each node to its closest TT vertex is fixed, the node alignment process will terminate within finite time. By the alignment rule, $\mathcal{P}$ could be occupied by multiple nodes during the initial node alignment. If $\mathcal{P}$ is still empty after the alignment process terminates, it will be eventually occupied by at least one node through greedy advance, because the algorithms ensures there be a winer in every competition for greedy advance. In any case, $\mathcal{P}$ becomes occupied within finite time. Once $\mathcal{P}$ is occupied, no node will move to it. If multiple nodes exist at $\mathcal{P}$ at some moment, one and only one of them will stay, while the others will retreat to $\mathcal{H}_1$ according to the retreat rule. Hence, the lemma holds. ∎

***Theorem** 1:* GA terminates within finite time.

Proof: Because, after the initial node dropping, the distance from each node to its closest TT vertex is fixed, the node alignment process will terminate within finite time. By Lemma 1, $\mathcal{P}$ will be occupied by a single node within finite time. Henceforth, we safely assume that the deployment step already passed the alignment process and that $\mathcal{P}$ has been occupied by a single node.

When a node is leaving a TT vertex due to the retreat rule, the TT vertex is occupied by another node. The priority rule and the forbiddance rule prevent two nodes located at different TT vertices from greedily moving toward the same TT vertex. In summary, the number of occupied TT vertices never decreases.

Assume for the sake of contradiction that GA never terminates. Since the number of occupied TT vertices never decreases, it follows that there exists an $m \leq n$ where n is the network size such that the algorithm runs infinitively long on m occupied TT vertices.

Consider that after a finite number of deployment steps the TT vertices $T = \{t_1, \ldots, t_m\}$ are occupied. In subsequent steps, the set T may only change due to a greedy rule. Assume for the sake of contradiction that T changes due to the retreat rule. Whenever an unoccupied TT vertex is visited by the retreat rule, the number of occupied TT vertices increases by

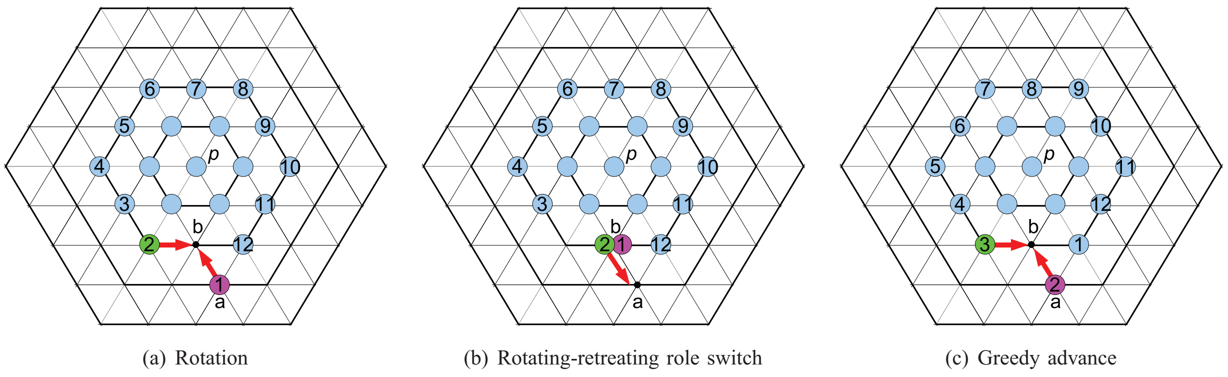


Fig. 5. An example of collision loop in GRG-CW

one, contradicting the assumption that GA runs infinitively long on m occupied TT vertices.

Define by $\sum(T)$ the sum of the TT distance from $\mathcal{P}$ to the TT vertices in T, i.e., $\sum(T) = \sum_{t_i \in T} |SP(t_i, \mathcal{P})|$. Whenever T changes to T' due to a greedy rule, a node moves from a hexagon $\mathcal{H}_{i+1}$ to a hexagon $\mathcal{H}_i$. It follows, $\sum(T') = \sum(T) - 1$. Since $\sum(T) \geq 0$, it follows that the set of occupied TT vertices can only change a finite number of times.

Let $F = \{f_1, \ldots, f_m\}$ be the final set of TT vertices visited by GA, i.e., F no longer changes in subsequent steps. Subsequent deployment steps are only due to the retreat rule since a greedy rule will always visit a non-occupied TT vertex and would thus change the final set F.

Define d as the maximum distance between $\mathcal{P}$ and the finally occupied TT vertices, i.e., $d = \max\{|SP(f_i, \mathcal{P})|\}$, $f_i \in F$. Let $Q = \{q_1, \ldots, q_n\}$ be the multi set of the TT vertices occupied by the sensor nodes, i.e. q_i is the TT vertex occupied by sensor node n_i. Nodes moving due to the retreat rule always move from a hexagon $\mathcal{H}_i$ to a hexagon $\mathcal{H}_{i+1}$. Thus, a change from Q to Q' always satisfies $\sum(Q') = \sum(Q) + 1$.

It follows that after a finite number of deployment steps the multi set Q of occupied TT vertices satisfies $\sum(Q) > nd$, i.e., there exists an occupied TT vertex t which satisfies $|SP(t, \mathcal{P})| > d$. Thus, the visited TT vertex t is not an element of F which finally contradicts the assumption that F is the final set of visited TT vertices. ■

***Lemma** 2:* Let $\mathcal{H}_0, \cdots, \mathcal{H}_{i-1}$ be fully occupied without co-located nodes. Let $n \geq \nu(i)$. In GRG, $\mathcal{H}_i$ will be fully occupied without co-located nodes within finite time.

Proof: When $\mathcal{H}_0, \cdots, \mathcal{H}_{i-1}$ are all fully occupied, nodes that have decided to stay on $\mathcal{H}_i$ never leave $\mathcal{H}_i$ but counterclockwise rotate along $\mathcal{H}_i$ because they are assigned highest rank in any local ranking process triggered by node collision, making unoccupied $\mathcal{H}_i$ vertices "rotating" in the opposite direction. In worst case, they make a full rotation and then stop moving forever, rendering unoccupied vertices fixed. In any case, some $\mathcal{H}_{i+1}$ nodes are guaranteed to meet the empty vertices on $\mathcal{H}_i$ (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 co-located nodes on the $i - 1$ inner hexagons, $\mathcal{H}_i$ will be fully occupied at the end as nodes keep moving toward it and eventually stop on it. Nodal retreat guarantees that no $\mathcal{H}_i$ vertex be occupied by multiple nodes. Hence, the lemma holds. ■

***Lemma** 3:* Let $\mathcal{H}_0, \mathcal{H}_1, \cdots, \mathcal{H}_{i-1}$ be fully occupied without co-located nodes. Let $\nu(i-1) < n < \nu(i)$. In GRG, nodes located on $\mathcal{H}_i$ will stop moving within finite time.

Proof: As inner hexagons $\mathcal{H}_0, \mathcal{H}_1, \cdots, \mathcal{H}_{i-1}$ are all fully occupied, nodes from outer hexagons will rotate along $\mathcal{H}_i$, after arriving at $\mathcal{H}_i$. 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 (refer to Sec. IV-D). Because $\nu(i-1) < n < \nu(i)$, at least one of them will make a full rotation. By protocol definition, this node will then stop moving forever, which will in turn block the rotation of any following node. Eventually, the nodes on $\mathcal{H}_i$ will become fixed. After the nodes on $\mathcal{H}_i$ stop moving, the nodes on $\mathcal{H}_{i+1}$ (if any) will get onto $\mathcal{H}_i$ and possibly rotate along $\mathcal{H}_i$ as well. These newly arriving nodes will stop moving and become fixed within finite time because of the blocking from previously stopped nodes on $\mathcal{H}_i$. ■

***Theorem** 2:* GRG terminates within finite time.

Proof: It follows immediately from Lemmas 1 – 3. ■

***Theorem** 3:* Both GA and GRG yields a connected network with hole-free coverage.

Proof: We prove this theorem by contradiction. By Theorem 1 and 2, 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 from $\mathcal{P}$ on the border of the hole and by $\langle i, j, k \rangle$ the address of v. There must exist a node at $\langle i+1, j, k \rangle$ (or $\langle i+1, j, i \rangle$ if $k = 0$), 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 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 $r_c \geq \sqrt{3} r_s$. ■

In GA, the final coverage of a MSN has an unpredictable shape, depending very much on the initial sensor placement.

As shown in Fig. 4(a), it is possible that $\mathcal{P}$ 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 γ^H_{opt} be the optimal hexagonal coverage radius (measured in layer distance) that the network can provide. Then we have $\gamma^H_{opt} = \lfloor \nu^{-1}(n) \rfloor$ where $\nu^{-1}(n) = \frac{\sqrt{12n-3}-3}{6}$ is the inverse function of $\nu(n)$ (Eqn. 1).

Let $\mathcal{F}$ be the focused coverage constructed by GRG using this network. Denote by γ^H the hexagonal radius of $\mathcal{F}$. From Lemmas 1 – 3, the following optimality result follows:

***Theorem** 4:* In GRG, $\gamma^H = \gamma^H_{opt}$.

We will now study the circular radius γ^C (measured in Euclidean distance) of $\mathcal{F}$. Denote by γ^C_{opt} the optimal circular coverage radius that the network can provide. Further, let S be the size (area) of $\mathcal{F}$, and $\mathcal{H}_\kappa$ the outmost hexagon of $\mathcal{F}$, where $\kappa = \lceil \nu^{-1}(n) \rceil$. We first derive bounds on γ^C_{opt}.

***Theorem** 5:* $\frac{3}{2}(\kappa - 1)r_s \leq \gamma^C_{opt} \leq 3\sqrt{\frac{\sqrt{3}}{2\pi}}\kappa r_s$.

Proof: The lower bound, which is the radius of the inscribed circle of $\mathcal{H}_{\kappa-1}$ is obvious. Recall the definition of coverage. It is provable that the hexagonal node placement produces maximized coverage over the TT. In this case, γ^C_{opt} must not be larger than the radius of the circle whose area is equal to the area of $\mathcal{H}_\kappa$, that is, $\gamma^C_{opt} \leq 3\sqrt{\frac{\sqrt{3}}{2\pi}}\kappa r_s$. ■

Then we shall show GRG yields optimal or near optimal circular coverage radius, depending on the network size n. We have to examine two cases: (1) $n = \nu(\kappa)$, meaning that, $\mathcal{H}_\kappa$ is fully occupied; (2) $n \neq \nu(\kappa)$ (precisely $\nu(\kappa-1) < n < \nu(\kappa)$), meaning that, $\mathcal{H}_\kappa$ is partially occupied.

***Lemma** 4:* In GRG, $0.95\gamma^C_{opt} \leq \gamma^C \leq \gamma^C_{opt}$ for $n = \nu(\kappa)$.

Proof: In the case of $n = \nu(\kappa)$, γ^C is equal to the radius of the inscribed circle of $\mathcal{H}_\kappa$, namely, $\gamma^C = \frac{3}{2}\kappa r_s$. By Theorem 5, $\frac{\gamma^C}{\gamma^C_{opt}} \geq \frac{\frac{3}{2}\kappa r_s}{3\sqrt{\frac{\sqrt{3}}{2\pi}}\kappa r_s} = \sqrt{\frac{\pi}{2\sqrt{3}}} = 0.95$. And obviously, $\gamma^C \leq \gamma^C_{opt}$. Hence, the lemma holds. ■

***Lemma** 5:* In GRG, $0.95\frac{\kappa-1}{\kappa}\gamma^C_{opt} \leq \gamma^C \leq \gamma^C_{opt}$ for $n \neq \nu(\kappa)$.

Proof: When $n \neq \nu(\kappa)$, γ^C must not be less than the radius of the inscribed circle of $\mathcal{H}_{\kappa-1}$, i.e., $\gamma^C \geq \frac{3}{2}(\kappa-1)r_s$. By Theorem 5, $\frac{\gamma^C}{\gamma^C_{opt}} \geq \frac{\frac{3}{2}(\kappa-1)r_s}{3\sqrt{\frac{\sqrt{3}}{2\pi}}\kappa r_s} = \frac{\kappa-1}{\kappa}\sqrt{\frac{\pi}{2\sqrt{3}}} = 0.95\frac{\kappa-1}{\kappa}$. Obviously, $\gamma^C \leq \gamma^C_{opt}$. Hence, the lemma holds. ■

Summarizing Lemmas 4 and 5, we have the theorem below:

***Theorem** 6:* Let $\delta = \begin{cases} 1, & n = \nu(\kappa); \\ 1 - \frac{1}{\kappa}, & n \neq \nu(\kappa). \end{cases}$ Then in GRG, $0.95\delta\gamma^C_{opt} \leq \gamma^C \leq \gamma^C_{opt}$

By Theorem 6, it would appear that the resultant circular radius of GRG was far from optimal in small-size network. For instance, if $\kappa = 2$, the lower bound will be $0.475\gamma^C_{opt}$. This is misleading because the lower bound are too coarse in the case of $k \leq 6$, as indicated by the following complementary theorem whose proof is omitted due to space limit.

***Theorem** 7:* In GRG, $\gamma^C > 0.86\gamma^C_{opt}$ for $n \neq \nu(\kappa) \wedge \kappa \leq 6$.

VI. Performance Evaluation

Although sensor self-deployment is not a new research issue, sensor self-deployment for focused coverage formation is to our best knowledge a new problem addressed for the first time in this paper. Existing sensor self-deployment algorithms may possibly yield a network with coverage radius as bad as 0. Because we emphasize on optimizing coverage radius, they are not comparable to GRG which guarantees optimal or near optimal coverage radius. Thus in the sequel, we are going to comparatively evaluate GA and GRG only.

A. 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, and they themselves do not generate any message during the course of self-deployment, communication cost is not our concern here.

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

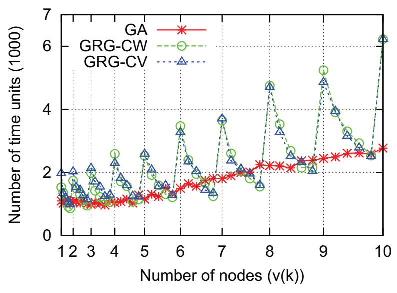
2) Energy consumption: It is measured by *number of moves (V)*, *mileage (M)*, and *mileage over progress ratio (R)*. V and M are respectively defined as the number of times a node started its motor and the total distance it traveled for its self-deployment. Let D_{ini} and D_{fin} respectively be its initial and its final Euclidean distance to POI. Then $R = \frac{M}{P}$, where $P = |D_{ini} - D_{fin}|$ is progress.

3) Node collision (C): We consider that two nodes collide as long as they are located sufficiently close to each other. Collision is due to randomized initial node placement and algorithmic design. Although 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, causing various communication failure.

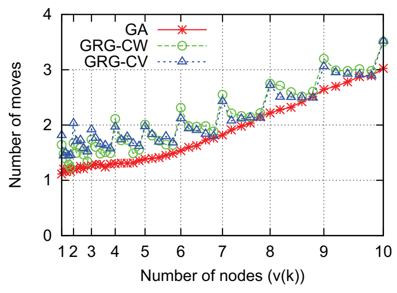
B. Simulation setup

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. Nodes 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. Through simulation we study the performance of the two algorithms under different node density by fixing the size of dropping area to 200^2 and varying network size n from $\nu(1) = 7$ to $\nu(10) = 331$. For each simulation setting, we run GA and GRG over 50 randomly generated network scenarios in order to get average results.

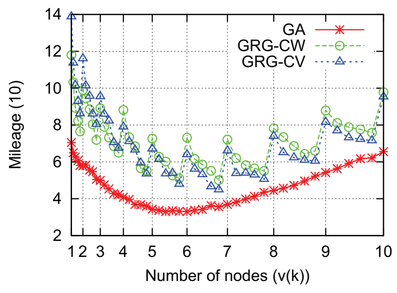
In fact, we also conducted another set of experiments to evaluate the two algorithms with different average initial node



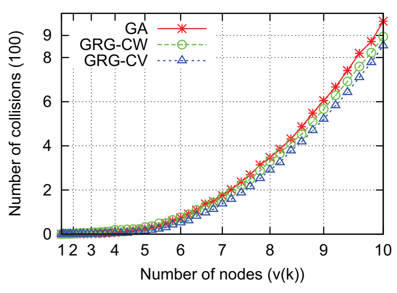
(a) Convergence time



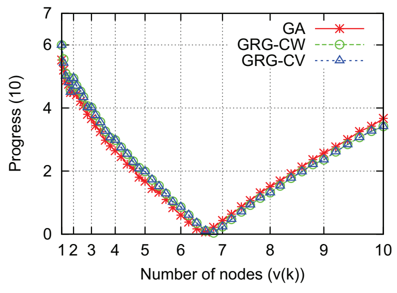
(b) Number of moves per node



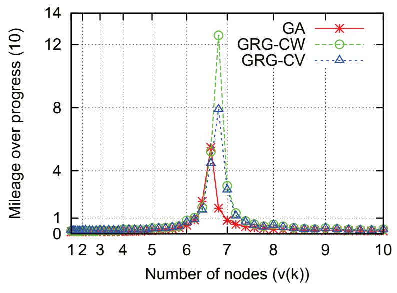
(c) Mileage per node



(d) Number of collisions



(e) Progress per node



(f) Ratio of mileage over progress

Fig. 6. Experimental results

distance. Due to space limitation, these experimental results are not presented here. They can be found in [9]

C. Simulation results

In the following, we are going to elaborate on our simulation results displayed in Fig. 6. As we will see, GA outperforms GRG in the aspects of convergence time and energy consumption; GRG-CV is more suitable for dense networks when compared with GRG-CW.

Examine Fig. 6(a) and 6(b), which respectively indicate T and V as a function of n and contain curves of similar trend. We first investigate the monotonically increasing curves of GA. When $n = \nu(1)$, 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 T and V. As n increases, the frequency of blocking and nodal retreat rises, and waiting and resuming happen more and more often. As a result, both T and V increase.

Now, let us look at the curves for GRG-CW and GRG-CV in the two figures. If we link the points with $n = \nu(\kappa)$, 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 GA. It is because GRG involves extra rotation movement, which complexes the self-deployment process. Observe any interval between $\nu(\kappa-1)$ and $\nu(\kappa)$ for an integer κ, and we find that the curve of either variant of GRG descends in this interval, which is reasonable. In the case of $n = \nu(\kappa)$, GRG does not converge until the outmost hexagon is fully occupied; in any other case, it, as we mentioned in Sec. VI-A1, converges as soon as all the inner $\kappa - 1$ hexagons are fully filled, making a dramatic decrease of both T and V. In fact, when n is very close to $\nu(\kappa)$, GRG performs even better than GA, as shown in the two figures, since the latter converges only when nodes all stop moving.

Figure 6(c) illustrates how M varies as n changes. It is observed that the curves for GRG-CW and GRG-CV have a declining trend when n lies in the range between $\nu(\kappa-1)$ and $\nu(\kappa)$ for an integer κ. This phenomenon is due to exactly the same reason as the similar phenomena observed in Fig. 6(a) and 6(b). If we link points on GRG curves with $n = \nu(\kappa)$ together, we also get two closely-located monotonically-increasing curves. The two new curves surpass the curve for GA for every value of n because GA does not generate rotation movement. Besides, they also have the same trend as GA: firstly declining and then climbing. It is because, as n goes up, the network becomes increasingly dense, and D_{fin} thereby drops 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 $\nu(6)$), the network shows an expanding behavior, namely, that nodes move outwards for coverage maximization, leading to the monotonic increase of M with increased n.

Closely examine the three figures 6(a) - 6(c) again. We can find that GRG-CW performs better in sparse networks, but worse in dense networks, than 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 T, V and M, happens rarely. In this case, aggressive GRG-CW beats conservative GRG-CV, as the latter often unnecessarily forces nodes to travel increased distance. As n mounts up, the network shows more and more a rotating or expanding behavior, and node collision occurs increasingly often, as confirmed by Fig. 6(d) and discussed in next paragraph. The positive impact of the strict hop selection rules of GRG-CV keeps growing, while their negative effect constantly decrease, finally rendering it outperform GRG-CW.

Figure 6(d) shows C in relation with n. Observe that C keeps ascending as n increases because the probability of node collision climbs as node density, which is proportional to network size in the case of fixed-sized dropping area, increases. Also observe that GRG-CV always yields smaller C than GRG-CW. Recall that 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, GA and GRG has nearly the same performance in a small-sized network, and they deviate from each other as n goes up. GRG-CV is below GA in all cases because rotation helps to reduce retreat-related collision. 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 GA (after $n = \nu(6)$) because the proportion of greedy-rotation collision diminishes, and that of retreat-related collision avoided by rotation contrarily emerges.

Figure 6(e) illustrates P as a result of n. The curves corresponding to GA and the two versions of GRG are all in a "V" shape with the lowest point rooted around $n = \nu(7)$. 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 $\nu(7)$, the difference between D_{ini} and D_{fin} becomes bigger and bigger, resulting in the rise of C. With no difficulty, we can see that the network shows a concentrating behavior when $n < \nu(7)$ and an expending behavior when $n > \nu(7)$.

Figure 6(f) exhibits R versus n. It is observed that R is lower than 10 and very close to 1 for both GA and GRG almost for all the values of n. In the figure, R reaches its peak value around $n = \nu(7)$. In fact, R can go to infinity in the case of $P = 0$. Although this extreme situation may not come into reality, but 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 R decreases and approaches 1 closer and closer as n increases or decreases toward the two end values. Through a comparative study on the two figures 6(c) and Fig. 6(e), the reason for this phenomenon becomes fairly obvious: P has a way smaller value (nearly equal to 0) than M round $n = \nu(7)$ and it climbs at a much faster speed than M with increased/decreased n.

VII. Conclusions and Future work

Research on sensor self-deployment is still on its initial stage, where defining the problem and finding basic self-deployment techniques extendable to more complex protocols are the main tasks. In this paper, we pinpointed a new sensor self-deployment problem, *focused coverage formation* and introduced an evaluation metric, *coverage radius*. By converting area coverage problem to vertex coverage problem on a virtual equilateral triangulation (TT), we proposed the first localized solutions, Greedy Advance (GA) and Greedy-Rotation-Greedy (GRG) with desired coverage guarantee. We proved their correctness, and studied their properties and performance by throughout analysis and extensive simulation.

The two proposed algorithms GA and GRG were described in the context of a single point of interest (POI). However, there are complex scenarios where a series of POIs form a Line of Interest (LOI), representing an object like the trace of certain event or the border of a landmark. The combined greedy-rotation technique (GRG) presented here is extendable to LOI case as follows: each sensor takes a common point on LOI as reference and builds a TT graph, and independently finds the smallest set of successive vertices that best represent LOI; these vertices form a *coverage backbone*, and sensors self-deploy around the coverage backbone following the same philosophy as GRG. Detailed algorithmic design of this extension is not trivial. We leave it for future work.

VIII. Acknowledgements

This work was partially supported by NSERC Strategic Grant STPSC356913-2007B and NSERC Discovery grants.

References

[1] X. Bai, S. Kumary, D. Xuan, Z. Yun and T. H. Lai, "Deploying Wireless Sensors to Achieve Both Coverage and Connectivity". In *Proc. of 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. of IEEE DCOSS*, pp. 451-456, 2008.

[3] S. Chellappan, X. Bai, B. Ma and D. Xuan. "Sensor networks deployment using flip-based sensors". *Proc. of IEEE MASS*, pp. 291-298, 2005.

[4] J. Cortes, S. Martinez, T. Karatas, and F. Bullo. "Coverage control for mobile sensing networks". *IEEE Tran. on Robotics and Automation*, 20(2): 243-255, 2004.

[5] M. Garetto, M. Gribaudo, C.-F. Chiasserini and E. Leonardi. "A Distributed Sensor Relocation Scheme for Environmental Control". In *Proc. of IEEE MASS*, 2007

[6] N. Heo and P. K. Varshney. "Energy-Efficient Deployment of Intelligent Mobile Sensor Networks". IEEE Tran. on Systems, Man, and Cybernetics - Part A: Systems and Humans, 35(1): 78-92, 2005.

[7] A. Howard, M. J. Mataric, and G. S. Sukhatme. "An Incremental Self-Deployment Algorithm for Mobile Sensor Networks". Autonomous Robots, 13(2):113-126, 2002.

[8] A. Howard, M. J. Mataric, and G. S. Sukhatme. "Mobile Sensor Network Deployment using Potential Fields: A Distributed, Scalable Solution to the Area Coverage Problem". In *Proc. of DARS*, pp. 299-308, 2002.

[9] X. Li, H. Frey, N. Santoro, and I. Stojmenovic. "Localized Self-Deployment of Mobile Sensors for Optimal Focused-Coverage Formation." TR-2007-13, SITE, Univ. of Ottawa, Dec. 2007.

[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, 2009. To appear.

[11] M. Ma and Y. Yang. "Adaptive Triangular Deployment Algorithm for Unattended Mobile Sensor Networks". IEEE Tran. on Computers, 56(7): 946-958, 2007.

[12] G. Wang, G. Cao, and T. L. Porta. "Movement-Assisted Sensor Deployment". IEEE Tran. on Mobi. Comp., 5(6): 640-652, 2006.

[13] S. Yang, M. Li, and J. Wu. "Scan-Based Movement-Assisted Sensor Deployment Methods in Wireless Sensor Networks". IEEE Tran. on Para. and Dist. Syst., 18(8): 1108-1121, 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.