Localized Sensor Self-deployment for Guaranteed Coverage Radius Maximization

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—Focused coverage is defined as the coverage of a wireless sensor network surrounding a point of interest (POI), and is measured by coverage radius, i.e., minimum distance from POI to uncovered areas. Sensor self-deployment algorithm GRG [4] is designed for autonomous focused coverage formation. It however does not always produce optimal (i.e., maximized) coverage radius. In this paper, we propose optimized GRG, referred to as OGRG, for guaranteed coverage radius maximization, and evaluate its performance in comparison with GRG.

I. INTRODUCTION

Sensor self-deployment traditionally deals with autonomous coverage formation in a mobile sensor network (MSN) over a region of interest (ROI) without particular coverage focus. Sensors are simply required to scatter/gather to construct a maximized hole-free coverage in any part of ROI. Recently, Li et al. [4] introduced a new sensor self-deployment problem, focused coverage formation, where self-governing sensors surround a point of interest (POI), while satisfying a priority requirement: an area close to POI is covered with higher priority than a relatively distant one.

Because of this unique requirement, it is no longer sufficient to merely measure area and sensing holes in the case of focused coverage. An additional metric, coverage radius [4], ought to be used. It is defined as the radius of the maximal hole-free disc centered at POI in the coverage region, i.e., the minimum distance from POI to uncovered areas. Coverage radius is expected to be maximized. Li et al. [4] introduced a localized sensor self-deployment algorithm Greedy-Rotation-Greedy (GRG). This algorithm has many nice properties; but it provides no optimality guarantee in coverage radius [5].

In this paper, we propose optimized GRG, referred to as OGRG, for guaranteed coverage radius maximization. As GRG, OGRG drives in a localized manner sensors to move from vertex to vertex over an equilateral triangle tessellation (TT) to surround POI in a greedy-rotation alternate fashion. That is, sensors greedily proceed as close to POI as possible; when blocked, they rotate around POI to a vertex where greedy advance can resume. Unlike GRG where sensors rotate along hexagons, it guides sensors to rotate along deployment polygons that best approximate circles, and therefore always lead to optimal focused coverage of maximized radius.

II. GREEDY-ROTATION-GREEDY IN A NUTSHELL

In this section, we introduce algorithm GRG [4]. We would like to indicate that, as focused coverage is a new problem with unique requirement, other previous sensor self-deployment algorithms, e.g., [2], [3], [6], [7], are not applicable, or applicable with poor performance, to it. GRG [4] requires sensors to know their own location as well as the location of POI (denoted by Π). Sensors bear the same communication radius $r_c$ and sensing radius $r_s$ for $r_c \geq 3r_s$. They have the information such as locations and moving status of 1-hop neighbors by lower-layer protocols.

In GRG, a TT graph $G_{TT}$ (Fig. 1) is built in the deployment plane with II as vertex and with edge length $l_e = \sqrt{3}r_s$. $G_{TT}$ is locally computable by each sensor using a common orientation. In $G_{TT}$, there are 6$i$ many vertices with equal graph distance $i$ to II. These vertices constitute a distance-$i$ hexagon, denoted by $\mathcal{H}_i$, centered at II. The total number $\nu(i)$ of vertices enclosed inclusively by $\mathcal{H}_i$ is exactly

$$\nu(i) = 3i(i + 1) + 1.$$  

(1)

GRG translates area coverage problem to vertex coverage problem on $G_{TT}$. In GRG, each sensor first by an alignment rule moves to the closest TT vertex and then acts in the following way: greedily proceed from vertex to vertex toward II; when blocked, i.e., when greedy next hop is occupied by others, rotate around II counterclockwise along the residence hexagon to a vertex where greedy advance can resume; if both greedy advance and rotation are blocked, stay put. Greedy advance rules and rotation rules are carefully designed to guarantee progress and termination.

GRG has two variants: GRG-CW and GRG-CV. The former allows greedy-rotation collisions and solves them after, while the latter prevents this particular type of node collision by using additional collision avoidance rules. However, notice that GRG (even the -CV version) is not collision free in general, due to initial stochastic node distribution. To solve node collision and ensure coverage maximization, GRG employs a retreat rule, which allows only one node to stay by pushing the others onto the next outer hexagon after a node collision.

GRG does not require fixed network size and allows dynamic node addition and removal. It works regardless of network asynchrony and disconnectivity. Using merely one-hop neighborhood information, it produces a connected network of TT layout without sensing hole and consequently a maximized coverage according to [8]. It is the first localized sensor self-deployment algorithm that provides such coverage guarantee. GRG ensures a hexagon coverage shape centered at POI. It
is proven that the radius of the resultant focused coverage approximates the maximum value with a factor in [0.88, 1] [5].

III. DEPLOYMENT POLYGON

The radius $\gamma^C$ of a focused coverage produced by GRG is considered equivalent to the radius of the inscribed circle of the outermost fully-occupied hexagon $H_k$ [5]. However, $H_k$ might not approximate its inscribed circle best, and nodes making no contribution to $\gamma^C$ may exist in the corner areas of $H_k$. For this, GRG does not guarantee optimal (maximized) coverage radius. Below, we define deployment polygons that best approximate circles over $TT$ and present their localized computation. Later, in Section IV, we show how to enable GRG to guarantee coverage radius maximization without jeopardizing its any other property using these polygons.

A. Definition

Denote by $C_i$ the inscribed circle of an arbitrary hexagon $H_i$, and by $R(C_i)$ the radius of $C_i$. In discrete graph $TT$, $C_i$ is best approximated without radius reduction by a minimum polygon that encloses $C_i$ and consists of successive vertices. Apparently, this polygon must be completely contained in, or overlapped by, $H_i$. We refer to it as deployment polygon associated with $C_i$ and denote it by $P_i$. Formally, we define:

DEFINITION 1 (DEPLOYMENT POLYGON): In $TT$, a deployment polygon $P_i$ is the perimeter of the polygonal area composed of the TT triangles inside $C_i$ and the TT triangles across the border of $C_i$.

Observe Fig. 1 that shows $P_i$ and $C_i$ for $i = 14$ and 15. $P_i$ is constituted exactly by the vertices that are located on the border of $C_i$ as well as those that reside outside $C_i$ and comprise the TT triangles crossing the border of $C_i$. Hence, by definition, we have the following lemma:

LEMMA 1: In $TT$, a vertex is located on $P_i$ if and only if it itself does not reside inside $C_i$ and at least one of its neighboring vertex lies inside $C_i$.

According to [5], the total number $\nu'(i)$ of vertices enclosed inclusively by $P_i$ is

$$\nu'(i) = \begin{cases} 
\nu(i) & \text{for } i \leq 7 \\
\nu(i) - \rho(i) & \text{for } i > 7 
\end{cases}$$

where $\rho(i) = 6 \sum_{t=0}^{\infty} (2\sqrt{3}t^2 + 1)$ for integer $i \in [1, \infty)$. Hence, $\nu'(i)$ is the number of vertices that exist in/on $H_i$ but fall outside $P_i$.

B. Properties

The six vertex neighbors of any vertex $v$ defines a circle of $l_v$ that is centered at $v$. In order for $v$ to be located on $P_i$, this circle must intersect $C_i$. Clearly, $R(C_i) = i \epsilon$ where $h = \frac{3}{2}r_s$ is the height of a TT triangle. Denote by $|v|$ the Euclidean distance from $v$ to $P_i$. Hence, $v$ can be located only on $P_i$ for $|v| \leq i \epsilon + l_v$. It can be trivially derived that a satisfactory $i$ is $\left\lceil \frac{|v|}{\epsilon} \right\rceil$, and that another possibly satisfactory $i$ is $\left\lfloor \frac{|v|}{\epsilon} \right\rfloor - 1$. Recall $l_v = \sqrt{3}r_s$. Summarizing, we have the following theorem:

THEOREM 1 (RESIDENCE): In $TT$, a vertex $v$ must reside on $P_i$ where $i = \left\lfloor \frac{|v|}{\epsilon} \right\rfloor$; it will also reside on $P_{i+1}$ if it has a neighboring vertex $w$ such that $|v| < \frac{3}{2}(i - 1)r_s$.

LEMMA 2: In $TT$, no edge is shared by two different deployment polygons.

LEMMA 3: On a deployment polygon, the two vertex neighbors of any vertex are not adjacent to each other.

By Lemmas 2 and 3 (whose proofs are omitted due to space limitation) and by exhaustive enumeration, we have:

THEOREM 2 (PATTERN): In $TT$, a non-POI vertex $v$ has four and only four possible neighborhood patterns, with respect to its residence polygon(s):

1) edge (−): $v$ has one residence polygon; it is located on an polygon edge;
2) convex corner (∧): $v$ has one residence polygon; it is located at a convex polygon corner;
3) concave corner (∨): $v$ has one residence polygon; it is located at a concave polygon corner;
4) joint corner (×): $v$ has two residence polygons; examples are the dotted vertices in Fig. 1.

Figure 2 illustrates the four neighborhood patterns, which are obviously locally detectable. From this figure, the vertex neighbors of $v$ may be divided into four disjoint sets: left-hand neighbor set $N_l(v)$, right-hand neighbor set $N_r(v)$, inward neighbor set $N_i(v)$, and outward neighbor set $N_o(v)$. The first two sets respectively contain the vertex neighbors of $v$ on its residence polygons in the clockwise direction and in the counter-clockwise direction around $P_i$; the last two sets respectively consist of those on an inner deployment polygon and on an outer deployment polygon.

By Theorem 1, vertex $v$ must reside on $P_i$ where $i = \left\lfloor \frac{|v|}{\epsilon} \right\rfloor$. Define $i' = i - 1$ if $P_{i-1}$ is also a residence polygon of $v$, or $i' = i$ otherwise. With no difficulty, we may conclude that the vertices in $N_i(v)$ are all located on $P_{i-1}$ and that the vertices in $N_{i'}(v)$ are all located on $P_{i+1}$. Let $Inn(N_i(v))$ and $Out(N_i(v))$ be the vertex neighbors of $v$ on $P_{i-1}$ and $P_{i+1}$, respectively. $Inn(N_i(v)) = Out(N_i(v))$ when $|N_i(v)| = 1$.

To ease later writing, we denote by $Pat(v)$ the neighborhood pattern of $v$, and we further denote by $Ltm(N_i(v))$, $Mid(N_i(v))$ and $Rtm(N_i(v))$ the leftmost, the middle, and the rightmost vertex neighbors of $v$ in $N_i(v)$, respectively. If
properties of GRG. In OGRG, a node located at a vertex without radius degradation, OGRG naturally provides guar-
rotation. As deployment polygons best approximate circles
by using deployment polygons instead of hexagons for node
N
greedily advances to an eligible vertex in
v
Mid
| e
they may collide at
v
is true only when
nodes collide at a non-POI vertex
A. Greedy advance
ensures nodal progress toward II. It hap-
pends to a node when the node proceeds to a vertex v from
an outward vertex neighbor of v. It is possible, only under
the condition |N^o(v)| > 1, that multiple greedily advancing
nodes collide at a non-POI vertex v. By Fig. 3, this condition
is true only when Pat(v) = “−”\^ “ + “.
Examine the corresponding graphs in Fig. 2. If two nodes
are greedily moving to v from vertices e and d in parallel,
they may collide at v. But this situation can be avoided by
the following priority rule as e and d are adjacent to each other.
RULE 1 (PRIORITY RULE): For two nodes aiming at a non-
POI vertex v from two different vertices Ltm(N^o(v)) and
Mid(N^o(v)), the one from Mid(N^o(v)) has higher priority.

Note that, if Pat(v) = “∧” and the two nodes are instead
from e and c, they could also collide at v. To prevent this
collision, the following forbiddance rule can be used.
RULE 2 (FORBIDDANCE RULE): A node at Rtm(N^o(v))
does not take v as greedy next hop when Pat(v) = “∧”.
II is a special vertex with 6 outward vertex neighbors. Its
occupancy is ensured by the following innermost-layer rule.
RULE 3 (INNERMOST-LAYER RULE): A node at a vertex
on P1 moves to II as long as II is to its knowledge unoccupied.
The innermost-layer rule may induce node collision at II,
but fortunately no more than once according to [4]. As in
GRG, this potential greedy-greedy collision can be avoided
by a gateway rule to be introduced later, in Sec. IV-C.

B. Rotation
Rotation resumes blocked greedy advance by guiding nodes
around occupied vertices. It happens to a node when the node
moves to a vertex from the vertex’s left-hand vertex neighbor.
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 the competition rule:
RULE 4 (COMPETITION RULE): When two nodes are com-
peting for vertex v from two different vertices Out(N^l(v))
and Ltm(N^o(v)) (or, Inn(N^l(v)) and Out(N^l(v))), the one
from Out(N^l(v)) (resp., Inn(N^l(v))) wins.
If the two nodes are instead from f and an outward vertex
neighbor different than e of v, they could also collide at v. But
this collision is no longer avoidable by the above rule. We will
elaborate and resolve this situation later, in Sec. IV-C.

Figure 2 shows that Rtm(N^o(v)) is always adjacent to
Out(N^l(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 Out(N^v(v)) if it itself does not chose Out(N^v(v))
as rotation next hop. This is the design consideration of
the following suspension rule, which prevents rotation loop and
ensures nodes’ greedy advance [5].
RULE 5 (SUSPENSION RULE): A node located at vertex v
does not rotate to Out(N^v(v)) if any of its neighbors is
currently rotating to Rtm(N^o(v)).
Note: when the collision avoidance rules to be defined in
Sec. IV-C are applied, the suspension rule ought to be ignored
if greedy advance at Rtm(N^o(v)) is forbidden by those rules.

C. Collision avoidance
The greedy rules preclude non-POI-based greedy-greedy
collision but leave POI-based still possible. The rotation rules
We start with the easiest part, i.e., enabling OGRG to generate no greedy-rotation collision at all. This feature can be accomplished simply by the following gateway rule of GRG, which serves as a replacement of the inner-most layer rule.

**Rule 6 (Gateway rule):** A vertex on \( \mathcal{P}_1 \) (i.e., \( H_1 \)) is pre-defined as the gateway to \( \Pi \). For a node located on \( \mathcal{P}_1 \), it performs only greedy advance if its home vertex is the gateway, or only rotation otherwise.

Now let us focus on greedy-rotation collision avoidance. For ease of description, we say a node’s greedy advance is “safe” if and only if it does not cause any greedy-rotation collision. In order not to risk greedy-rotation collision, a node must not greedily advance unless it knows the movement is definitely safe. From local perspective, a node is able to make such an assurance only when all the left-hand vertex neighbors of its greedy next hop are neighbored by its home vertex. This serves as the design principle of the rules to be presented below.

Consider a vertex \( v \) on \( \mathcal{P}_i \) for \( i > 1 \). Suppose that a node is located at \( v \) and that it has decided by the greedy rules to move to \( \text{Ltm}(N_i(v)) \). Figure 3, where \( a = \text{Ltm}(N_i(v)) \) and \( f = \text{Inn}(N_i(v)) \), enumerates all the possible scenarios with respect to the neighborhood pattern of \( a \). Examining this figure, we define the following collision avoidance rules (whose detailed derivation is omitted due to space limitation):

**Rule 7 (Edge rule):** A node located at vertex \( v \) with \( \text{Pat}(v) = "\wedge" \) does not take \( \text{Ltm}(N_i(v)) \) (or \( \text{Rtm}(N_i(v)) \)) as greedy next hop if \( \text{Pat}(\text{Ltm}(N_i(v))) = "\wedge" \) (resp., \( \text{Pat}(\text{Rtm}(N_i(v))) = "\times" \)).

**Rule 8 (Concave-Corner rule):** A node located at vertex \( v \) with \( \text{Pat}(v) = "\vee" \) does not move to \( \text{Ltm}(N_i(v)) \) (or \( \text{Mid}(N_i(v)) \)) if \( \text{Pat}(\text{Ltm}(N_i(v))) = "\wedge" \) (resp., \( "\times" \)).

**Rule 9 (Convex-Corner rule):** A node located at vertex \( v \) with \( \text{Pat}(v) = "\wedge" \) is not allowed to move to \( \text{Ltm}(N_i(v)) \) if \( \text{Pat}(\text{Ltm}(N_i(v))) = "\wedge" \times \wedge" \). Here and in the following, additional requirements \( \text{Out}(N_i(\text{Ltm}(N_i(v)))) \neq \text{Inn}(N_i(v)) \).

**V. Performance Evaluation**

In this section, we will present our comparative simulation study of GRG [4] and OGRG. As our simulation results are consistent with those detailedly presented in [5], here we only study their performance difference for achieving the same coverage radius (referred to as cR) with proper number of nodes (referred to as nN). It is important to notice from Eqn. (1) and (2) that, OGRG uses increasingly less nodes than GRG when the target cR goes up. We chose the following evaluation metrics that are also adopted in [5]:

- Convergence time (cT): the number of time units that it takes the network to stabilize (have no floating node);
- Number of collisions (nC): the number of times that node collision occurs during the course of self-deployment;
- Mileage (Mg): the average distance that a node travels during the course of its self-deployment;
- Number of moves (mM): the average number of times that a node changes its status from still to moving.

**A. Simulation setup**

We implemented GRG and OGRG within a custom network simulator, and simulated their execution over a MSN randomly and uniformly dropped in a two-dimensional free plane. The geographic center of dropping area is taken as POI. Nodes have 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 fix the size of dropping area to \( 500^2 \) and target at different cRs. By Eqn. (2), OGRG is equivalent to GRG for a cR \( \leq 7 \). We consider the cases of cR = 8, ..., 18 by varying nN accordingly. Using fixed dropping area and varied nN, we are able to study the impact of node density on algorithm performance. To minimize data noises, for each simulation setting, we executed GRG and OGRG in 50 randomly generated network scenarios and computed average results.

**B. Experimental results**

1) Convergence and collision: We first examine the convergence time (cT) and the number of collisions (nC) of GRG and OGRG. As we will see, OGRG has better performance than GRG in these two aspects, especially in dense networks.

```plaintext
Fig. 3: The leftmost inward vertex neighbor a of v
```
distance to \( \Pi \). Then \( \Pi g \) is equal to \( dist_{ini} - dist_{fin} \).

We know that \( dist_{ini} \) is approximately constant as nodes are randomly and uniformly placed at initialization. Observe Fig. 4(c) and notice the difference between GRG and OGRG: the curve of OGRG is always above that of GRG. It is because of the larger \( dist_{fin} \) of GRG, which is in turn due to the coverage redundancy of GRG.

Figure 4(d) shows the difference between GRG and OGRG in Mg. Observe that Mg reaches the lowest value later in OGRG than in GRG. It is because OGRG uses less nodes than GRG for achieving the same coverage radius, delaying the behavioral change of the network. We can also observe that the Mg of OGRG is slightly larger (much smaller) than that of GRG in concentrating (resp., expanding) networks.

In both GRG and OGRG, nodes move along curly path to their final position; thus their Mg is usually larger than their Pg. The ratio mileage over progress (MoP) gives an idea about how costly zigzag node movement is. Note in MoP we use absolute value of progress. From Fig. 4(e) MoP is most of time below 5; it increases dramatically only when cR is around the value leading to lowest Pg (see Fig. 4(c)), although Mg approaches the lowest value at the same time (see Fig. 4(d)). Again, as shown in the figure, the economic node usage of OGRG delays the appearance of the peak value in MoP; GRG and OGRG have nearly same MoP performance in other cases.

In OGRG, nodes are regulated by more hop selection rules and thus may have to stop moving more often than in GRG. On the other hand, GRG requires more nodes than OGRG for achieving the same coverage radius and therefore possibly causes relatively frequent node blocking. As confirmed by Fig. 4(f), 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 algorithm performance on nM, making GRG generate less nM than OGRG; but it is slowly overwhelmed by the latter as cR becomes increasingly large, rendering OGRG eventually overtaking GRG.

REFERENCES