Localized Sensor Self-Deployment with Coverage Guarantee in Complex Environment

Xu Li, Nathalie Mitton, Isabelle Ryl, David Simplot

▶ **To cite this version:**

HAL Id: hal-00544913

https://hal.archives-ouvertes.fr/hal-00544913

Submitted on 9 Dec 2010

Localized Sensor Self-Deployment with Coverage Guarantee in Complex Environment

Xu Li, Nathalie Mitton, Isabelle Ryl, and David Simplot

CNRS/INRIA/Univ. of Lille 1, France
{firstname.lastname}@inria.fr

Abstract. In focused coverage problem, sensors are required to be deployed around a given point of interest (POI) with respect to a priority requirement: an area close to POI has higher priority to be covered than a distant one. A localized sensor self-deployment algorithm, named Greedy-Rotation-Greedy (GRG) [10], has recently been proposed for constructing optimal focused coverage. This previous work assumed obstacle-free environment and focused on theoretical aspects. Here in this paper, we remove this strong assumption, and extend GRG to practical settings. We equip GRG with a novel obstacle penetration technique and give it the important obstacle avoidance capability. The new version of GRG is referred to as GRG/OP. Through simulation, we evaluate its performance in comparison with plain GRG.

1   Introduction

Recently, a new sensor self-deployment problem, *focused coverage formation* [10], was brought into attention for dedicated applications. In this problem, mobile sensors are required to surround a given coverage focus, called *point of interest* (POI), while satisfying a priority requirement: area close to POI is covered with higher priority than distant one. Focused coverage is measured by *coverage radius*, which is defined as the minimum distance from POI to uncovered areas. Optimal focused coverage has maximized radius. Assuming obstacle-free environment, a localized algorithm Greedy-Rotation-Greedy (GRG) was presented and analyzed in [9, 10]. It is proven that GRG generates optimal hexagonal focused coverage. Under the same set of assumptions, GRG is optimized to produce optimal circular focused coverage in [11].

In this paper, we extend GRG to realistic obstacle-prone environment. Inspired by the physical behavior of liquid in a U-tube, we equip GRG with a novel obstacle penetration technique without jeopardizing its correctness and localized nature. The resultant version of GRG is referred to as GRG/OP (GRG with Obstacle Penetration). In GRG/OP, sensors around an obstacle simulate a liquid body in a virtual U-tube. They autonomously "flow" to ensure balanced pressure at the tube bottom, i.e., that the top sensors in the two tube arms are at the same deployment layer. Rules are designed to handle collision caused by nodal flowing behavior. By GRG/OP, sensors are able to pass around obstacles during self-deployment such that coverage optimality is preserved.

The remainder of this paper is organized as follows: Section 2 reviews previous related work; Section 3 briefly describes GRG; Sections 4 and 5 present the localized obstacle penetration technique and its implementation; Section 6 evaluates GRG/OP through simulation; Section 7 concludes the paper.

2 Related work

Previous sensor self-deployment algorithms except GRG [10] were designed for area coverage over a region of interest (ROI) without particular coverage focus. When used for focused coverage formation, they may cause sensors to settle in any arbitrary area in the deployment plan, leading to a coverage radius as bad as 0. In this section, we review some of these relevant work at very short length. A comprehensive and detailed survey can be found in [12].

In [7], sensors are placed in ROI incrementally, i.e., one at a time, to increase coverage based on information gathered from previously deployed sensors. In [4, 6, 8, 13, 18], each node computes movement vectors due to its neighbors using their relative position and move according to the vector summation in rounds to maximize coverage. In [5, 15], sensors align their sensing ranges with their Voronoi regions in rounds to minimize local uncovered area. In [2], sensors are pushed or pulled to hexagon centers of a hexagonal tiling over ROI. In [3], sensors deployment is modeled as a minimum cost maximum flow problem from source regions to hole regions in ROI. In [14], ROI is partitioned into a grid whose vertices are assigned recursively to sensors using a rooted spanning tree. In [16], the network is assumed dense enough, and nodes are treated as load and balanced among the sub-regions of ROI by multi-rounds of scan.

3 Greedy-Rotation-Greedy in a Nutshell

GRG [10] assumes that there is no physical obstacle in the deployment plane, and requires sensors to know their location and the location of POI. Sensors have the same sensing radius r_s and communication radius $r_c \geq \sqrt{s}r_s$. They have information such as location and moving status of 1-hop neighbors by lower-layer protocols. A virtual equilateral triangle tessellation (TT) G_{TT} of edge length $l_e = \sqrt{3}r_s$ is built in the plane with POI as vertex (see Fig. 2). G_{TT} is locally computable to each sensor given a common orientation. Vertices with equal graph distance i to POI form a distance-i hexagon $\mathcal{H}_i$ centered at POI.

GRG translates area coverage problem to vertex coverage problem on G_{TT}. Each sensor first by an alignment rule moves to the closet TT vertex and then acts in the following way: greedily proceed from vertex to vertex toward POI; when blocked, i.e., when greedy next hop is occupied by others, rotate around POI counterclockwise along its 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-and-Rotation (G-R) collision and solves it 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 network disconnectivity. Using merely one-hop neighborhood information, it produces a connected network of TT layout without sensing hole and consequently a maximized area coverage according to [17]. It is the only known localized sensor self-deployment algorithm with such coverage guarantees. GRG yields optimal focused coverage of maximized radius.

4 An obstacle penetration technique

In the following, we present a novel localized obstacle penetration technique for algorithm GRG. We assume that there is no concave obstacle in the plane. We make this assumption because a concave obstacle can entrap sensors into its pockets (opening against POI) and lead a partitioned network.

4.1 Virtual U-tube

A U-tube, as shown in Fig. 1, consists of three parts, left arm, right arm, and bottom. When it is filled with the same type of liquid, the liquid surface in its two arms always remain at the same level so that the pressure (due to gravity) from both sides settles at the bottom. This is simple physics. Our idea of obstacle penetration is inspired by this liquid behavior.

We treat POI as center of Gravitation and sensors as liquid molecules. Sensors are attracted to POI. Imagine a virtual U-tube (thus tube for simplicity) wrapping around an obstacle. It has a very small diameter equal to the diameter of a single liquid molecule. The two tube arms point away from POI across a
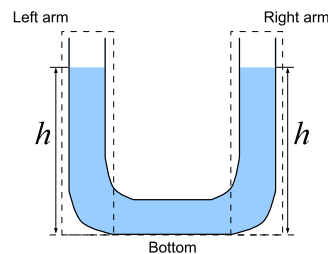
Fig. 1. Water surface remains at the same level in the two arms of a U-tube
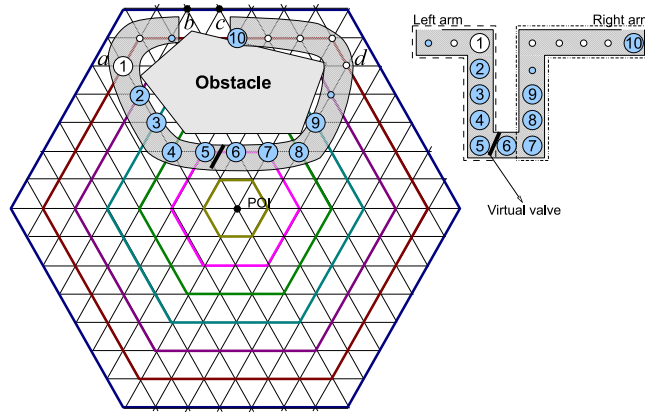
3

Fig. 2. Virtual U-tube

number of hexagon layers, and the height of the tube is measured by the number of those layers. There is a valve between the left arm and the bottom of the tube, which allows liquid (i.e., sensors) to enter the left arm from the bottom and prevents the reverse flow. Figure 2 shows a tube of height 5.

In G_{TT}, a tube is composed of a sequence of vertices, called *tube vertices*. If it contains liquid, then liquid molecules (i.e., sensors) must be located at tube vertices. There are three basic types of tube vertex, i.e., *bottom* vertex, *right-arm* vertex, and *left-arm* vertex, corresponding to the three parts of the tube. A right-arm (or left-arm) vertex is a vertex whose rotation next hop (resp., previous hop) is occupied by the obstacle. A bottom vertex is a vertex whose greedy previous hops and only greedy previous hops are occupied by the obstacle. Note that the rightmost and the leftmost bottom vertices are shared by the two tube arms and thus considered arm vertex as well.

In Fig. 2, these three types of tube vertex are shown by solid dots. It is observed that the subgraph of right-arm (or left-arm) vertices are not connected. To ensure tube connectivity, we introduce another type of tube vertex, *bridge* vertex. The definitions of bridge vertex for the left arm and for the right arm are symmetric. A right-arm (or left-arm) bridge vertex is a vertex whose greedy next hop(s) are occupied by the obstacle, and rotation next (resp., previous) hop leads to a right-arm (resp., left-arm) vertex. In Fig. 2, bridge vertices are marked by hallow dots. With bridge vertices, the left arm and the right arm become a connected component; they are finally linked together via bottom vertices.

By the status of neighboring vertices, a node is able to tell the basic tube tole of its residence vertex but not the bridge role. For example, in Fig. 2, there is no difference between vertices a and b (or, c and d) from local view. With one-hop knowledge, a node can only identify potential bridge vertices; to infer exact information, two-hop knowledge is needed. Specifically, a vertex is a left-arm (or right-arm) bridge vertex if its rotation previous (resp., next) hop is left-arm (resp., right-arm) vertex or left-arm (resp., right-arm) bridge vertex.

4

4.2 Fluid-like behavior

A node is said to be in a tube, thus referred to as *tube node*, if it is located at a tube vertex. Each node exchanges with neighboring nodes its tube role (if applicable and certain) to enable bridge node self-identification. When two tubes join, tube nodes in the joint part behave with respect to the two tubes separately. Henceforth, we concentrate on the scenario with a single independent tube only. We use term "obstructed" to imply that a node is stopped by an obstacle, and term "blocked" to indicate that a node is stopped by another node.

When a right-arm node, e.g., node 10 in Fig. 2, finds that its rotation is obstructed and it has on unobstructed greedy next hop, it tries to rotate backward to a right-arm vertex with unobstructed greedy next hop. During *backward rotation*, it may walk through a sequence of bridge vertices, staying at the same hexagon layer. Because it is aware of its presence in the right-arm (by knowing its rotation was obstructed), such a backward rotating node immediately knows that it is a bridge node at each visited bridge vertex. A right-arm bridge node turns itself to backward_rotation node as soon as it finds that its right tube neighbor is performing backward_rotation.

Backward rotation ensures right-arm nodes to "flow" down to the tube bottom, simulating the behavior (due to gravity) of liquid drops attached on tube side. The tube bottom is considered *open* unless all the bottom nodes' greedy advance is blocked or obstructed. With open bottom, bottom nodes and upcoming right-arm nodes will eventually leak out of the tube, no longer being affected by the obstacle. We only need to handle sealed tube bottom case.

Suppose that the bottom is sealed. As tube nodes keeps flowing into the bottom from the right arm, and bottom nodes rotate to the left, an *effective liquid body* (ELBD) occupying the three parts (i.e., the bottom and the two arms) of the tube will be accumulated. It is a linear sequence of connected tube nodes that are aware of their tube roles without uncertainty. For example, in Fig. 2, ELBD is composed of nodes 2–9; node 1 (which is assumed to arrive by greedy advance) does not belong to ELBD because it is not sure about its tube role. Only nodes that belong to ELBD need to take further special action for obstacle avoidance. We focus on these nodes only.

In ELBD, nodes exert pressure on adjacent low-level nodes. Aggregated pressure from the two arms meet through bottom nodes. If it is balanced, no node moves. If it is toward the right arm, no node moves either because of the prevention of the virtual valve, which is naturally realized by bottom nodes' unidirectional rotation. The two cases imply that ELBD's right end node's virtual rotation next hop (on the other side of the obstacle) is currently occupied, and it has to stay still for now.

If the pressure biases toward the left arm, then a *pressure adjustment process* starts. In this process, all the ELBD nodes shift their position to the "left" by one hop. Here, "left" means the clockwise direction along the obstacle if nodal rotation direction is counter-clockwise, or counter-clockwise direction otherwise. Position shifting renders the entire ELBD flow to the left by one position. It appears that the right-end node of the ELBD conducted rotation or greedy ad-

vance, "penetrating" the obstacle. Pressure adjustment is performed iteratively until the pressure from the right is no longer larger than that from the left.

For each ELBD node, its position shifting is realized by one of the four types of movement: greedy advance, rotation, backward_rotation, and backward_greedy. *Backward_greedy* is different from retreat (which is also opposite to greedy advance). The former happens only in the left-arm; whereas, the later may take place anywhere and for randomly selected vertex.

5 Implementation

In this section, we show how to implement the obstacle penetration technique without affecting the properties of GRG. The implementation includes design of collision avoidance and resolution rules, definition of loop avoidance policies, and development of a pressure adjustment protocol.

5.1 Tube collision handling

When a retreat node hits an obstacle, it moves, remaining as retreat node, along the obstacle to the left to find an empty vertex. In case of collision, it is assigned lowest priority to take next deployment step. As such, retreat movement does not affect nodal decision making; thus without jeopardizing correctness, we ignore it in collision avoidance and resolution. We generally refer to tube nodes' movement in the tube as "flowing" behavior. Because collision that does not involve nodal flowing behavior is avoided or resolved by GRG rules, we focus on flowing-related collision, called *tube collision*.

Tube collision needs special treatment because nodal flowing behavior may be invalid in GRG and can not be handled by GRG rules. Suppose that a tube node n is flowing to tube vertex v. We examine all possible tube collision that may occur at v by exploring Fig. 3–5. These figures exhaustively list the neighborhood scenarios of v with respect to the obstacle (shaded). Therein, solid arrowed lines stand for node n's flowing movement; broken arrowed lines indicate possible colliding movement (dashed ones for greedy advance and dotted ones for rotation). We define a general rule for tube collision avoidance as follows:

Rule 1 (Tube collision avoidance rule) *When two neighboring nodes, a flowing node n and a non-flowing node m (with respect to the same tube/obstacle), are both aiming at the same tube vertex, n proceeds as usual, while m changes its deployment decision accordingly.*

In collision resolution we no longer need to consider nodal movement locally prevented by the above rule. In Fig. 3–5, it is the movement marked by an "X" sign. Observe that, in Fig. 3(a)–3(c), the flowing behavior of node n is in fact greedy advance. In these case, its resulting tube collision is naturally resolved by regular GRG rules and henceforth out of our consideration as well. It is noticed that no tube collision is possible in Fig. 3(f) and 4(f).

6
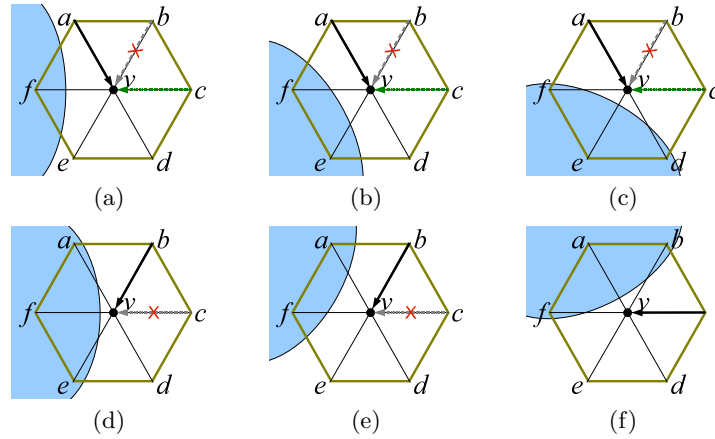
Fig. 3. The neighborhood of right-arm vertex v

Summarizing, we only need to take care of the following tube collisions, which are not locally prehibitable because colliding nodes are be out of each other's transmission rang:
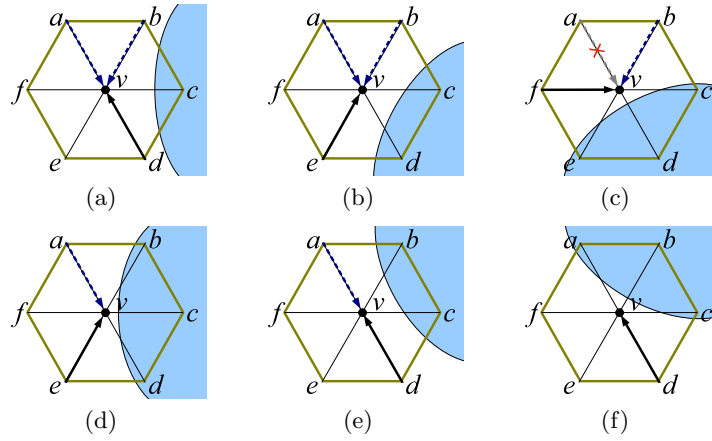
- Backward_Rotation-and-Greedy (BR-G)
    - Figures 4(c), 5(b), 5(c), where node n comes from vertex f by backward_rotation and collides with a greedy node from vertex b.
- Backward_Rotation-and-Rotation (BR-R)
    - Figures 5(b), 5(c), where node n comes from vertex f by backward_rotation and collides with a rotation node from vertex c.
- Backward_Greedy-and-Greedy (BG-G)
    - Figures 4(a), 4(b), 4(d), 4(e), 5(a), where node n comes from vertex e or d by backward_greedy and collides with a greedy node from a or b.
- Backward_Greedy-and-Rotation (BG-R)
    - Figures 5(a), where node n comes from vertex e by backward_greedy and collides with a rotation node from vertex c.
- Backward_Greedy-and-Backward_Rotation (BR-BG)
    - Figure 6, where v is a joint vertex of two tubes, and node n comes from d by backward_greedy and collides with a backward_rotation node from f.

In the following, we resolve these tube collisions by a number of localized rules. We start with BR-G resolution.

Rule 2 (BR-G rule) *In BR-G, backward_rotation node has higher priority to make deployment decision than greedy node. The latter will retreat if the former decides to stay and it itself can not conduct rotation.*

Depending on which arm it occurs in, BR-R is resolved in two different, but symmetric, ways. It is not difficult to see that in right-arm BR-R the previous hop

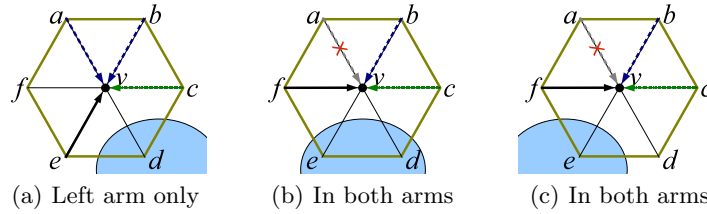**Fig. 4.** The neighborhood of left-arm vertex v

of the backward_rotation node can be taken either by a greedy node or by another backward_rotation node. In this case, if the rotation node continues rotating, probably after colliding with a sequence of intermediate backward_rotation nodes and persisting rotation, it will reach either an empty vertex or a vertex occupied by a greedy node. In the former case, right-arm BR-R is resolved; in the later case, G-R collision happens and is then resolved by GRG rules. Justified by this analysis, we define the following rule:

Rule 3 (Right-arm BR-R rule) *In right-arm BR-R, backward_rotation node has higher priority to make deployment decision than rotation node. If the former decides to stay, the latter will keep rotating whether the rotation next hop is occupied or not.*
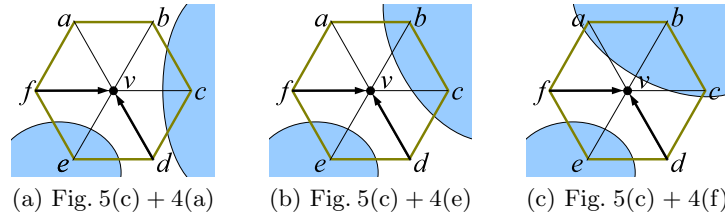
Symmetric to the situation in right-arm BR-R, in left-arm BR-R the rotation node must have left behind a vertex between its current home vertex and the obstacle. That vertex could be occupied either by a greedy node or by another rotation node. If the backward_rotation node persists backward rotating, it will reach a vertex which is either empty or occupied by a greedy node. In the former case, left-arm BR-R is resolved; in the latter case, BR-G collision occurs and is then resolved by the BR-G rule. Therefore, we define the following rule:

Rule 4 (Left-arm BR-R rule) *In left-arm BR-R, rotation node has higher priority to make deployment decision than backward_rotation node. If the former decides to stay, the latter will keep rotating backward whether the backward_rotation next hop is occupied or not.*

We now proceed to the resolution of BG-related collision, which happens only in the left arm. Recall that a backward_greedy node is impersonating a rotation node (i.e., the right end node of ELBD). BG-G, BG-R, and BG-BR are

(a) Left arm only　　(b) In both arms　　(c) In both arms

Fig. 5. The neighborhood of bridge vertex v

(a) Fig. 5(c) + 4(a)　　(b) Fig. 5(c) + 4(e)　　(c) Fig. 5(c) + 4(f)

Fig. 6. BR-BG collision at a joint vertex v of two tubes

equivalent or similar respectively to G-R, left-arm BR-R, and right-arm BR-R collision, and therefore can be resolved similarly. We define the following rules:

Rule 5 (BG-G rule) *In BG-G, backward_greedy node has higher priority to make deployment decision than greedy node. If the former decides to stay, the latter will retreat.*

Rule 6 (BG-R rule) *In BG-R, rotation node has higher priority to make deployment decision than backward_greedy node. If the former decides to stay, the latter rotates backward whether the backward_rotation next hop is occupied or not.*

Rule 7 (BG-BR rule) *In BG-BR, backward_rotation node has higher priority to make deployment decision than backward_greedy node. If the former decides to stay, the latter rotate whether the rotation next hop is occupied or not.*

5.2　Collision loop prevention

Because ELBD is composed of discrete nodes, asynchronous nodal flowing may cause transit empty vertices between ELBD nodes. A *transit empty vertex* is a tube vertex between two consecutive flowing ELBD nodes. It could appear available to adjacent non-tube nodes from their local view. We define *tube intrusion* as the movement (either greedy or rotation) of a non-tube node toward a transit empty vertex. By this definition, tube intrusion will never happen in tube bottom. Now we shall investigate every possible tube intrusion scenario in the two tube arms.

Consider a transit empty vertex v in the left arm. Examine all the possible neighborhood scenarios of v enumerated in Fig. 4. By definition, intrusion is not
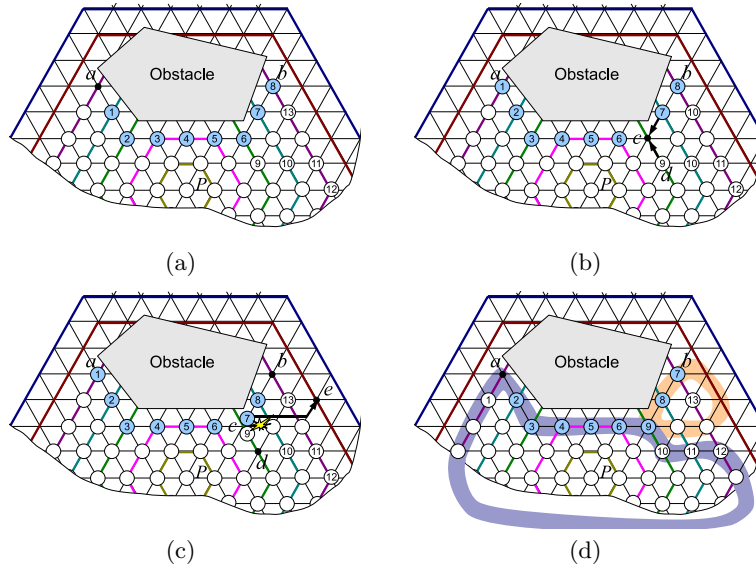
Fig. 7. Tube intrusion and resulting endless movement

possible in Fig. 4(d) – 4(f). In Fig. 4(a) – 4(c), intrusion may be only from vertex a, and it is locally prevented by the tube collision avoidance rule (note: there is a node occupying vertex b because, otherwise, v is not a transit empty vertex). We see that tube intrusion will not actually occur in the left arm.

Examine Fig. 3. Assume that the right-arm vertex v is a transit empty. Intrusion is not possible in Fig. 3(f) by definition; in Fig. 3(c) – 3(e), the only possible intrusion (from either b or c) is prevented by the tube collision avoidance rule. In Fig. 3(a) and 3(b), the intrusion from vertex b is prohibited by the tube collision avoidance rule, while that from c is not locally avoidable since the invading node is not neighboring the flowing node n. As we will see below, this intrusion may lead to tube collision loop.

Figure 7, where solid big dots represent ELBD nodes and node ID is for illustrative purpose only, shows an example of tube intrusion and its resulting collision loops. The network configuration before ELBD flows is given in Fig. 7(a). While ELBD is flowing, node 9 invades ELBD by rotation from vertex d to transit empty vertex c, as shown in Fig. 7(b). This intrusion causes a tube collision with flowing node 7, as displayed in Fig. 7(c). After the tube collision, node 9 stays, and node 7 retreats according to GRG. Meanwhile, nodes $10 - 12$ and nodes on the hexagon containing vertices a and b move according to GRG such that the empty vertex d is filled and that tube vertex a becomes empty again. Suppose that node 7 by any chance retreats to vertex e and that node 13 has not made deployment decision yet before node 7 reaches e. By the suspension rule of GRG, node 13 gives way to node 7, which then rotates and greedily advances to vertex b, leading to a configuration in Fig. 7(d). This is the same as

10

Fig. 7(a). The arrival of node 7 at b triggers an another round of ELBD flowing. Assume it takes place exactly in the same way. Then we have two joint tube collision loops as marked in Fig. 7(d).

Recall that the terminatability of GRG is grounded on the fact that once inner hexagons H_j ($j < i$) are fully occupied, nodes on H_i will never leave H_i (see the proof of Lemma 2 in ref. [9]). This correctness basis is violated in tube intrusion, and for this, collision loop occurs as illustrated in the above example. Since tube intrusion takes place only in the right arm, we require right-arm nodes to flow in a synchronized squirm-like fashion. That is, an ELBD node starts to flow to the left if and only if its right ELBD neighbor has arrived. By this means, no transit empty vertex will appear in the right arm, and collision loop no long exists. Terminatability follows as a result.

5.3 Pressure adjustment

In the following, we will discuss how to timely capture pressure change in the tube and trigger pressure adjustment. Each tube vertex v is associated with a pressure vector $\kappa(v)$. This vector indicates the pressure that a node located at v would contribute to the left. It is defined as

$$\kappa(v) = \begin{cases} +1 & \text{if } H(v) > H(N_L(v)) \\ 0 & \text{if } H(v) = H(N_L(v)) \\ -1 & \text{if } H(v) < H(N_L(v)), \end{cases}$$

where $N_L(v)$ is the left tube vertex neighbor of v, and $H(v)$ the level of the home hexagon of v. By this definition, a right-arm vertex has non-negative pressure vector, while a left-arm one has non-positive pressure vector.

Denote by $N_R(v)$ the right tube vertex neighbor of v. A tube node at v considers itself the left-end node of ELBD if and only if $\kappa(v) \leq 0$, $N_R(v)$ is not empty, and $N_L(v)$ is empty or occupied by a non-tube node or a node not aware of its own tube role. A tube node sends a *notification message* to the right along the tube if one of the following conditions holds: (1) it just becomes left-end node; (2) it just lost its left-end node status; (3) it remains to be left-end node, but with position change. The message is used to notify the right-end node of ELBD (if applicable) of possible pressure change in the left arm. It may be dropped by a receiver node having no right tube neighbor.

A tube node at v considers itself the right-end node of ELBD if and only if $\kappa(v) \geq 0$, $N_L(v)$ is not empty, and $N_R(v)$ is empty or occupied by a non-tube node or a node not aware of its own tube role. A tube node initiates a *pressure adjustment process* whichever of the following conditions is satisfied: (1) it just becomes right-end node; (2) it remains to be right-end node, but with position change; (3) it remains to be right-end node and just received a notification message. Each of these conditions indicates potential pressure difference between the two tube arms. By acting upon them, timely pressure adjustment is ensured.

In a pressure adjustment process, the initiator node sends a *pressure message* to the left along the tube, carrying a vector λ equal to its pressure vector
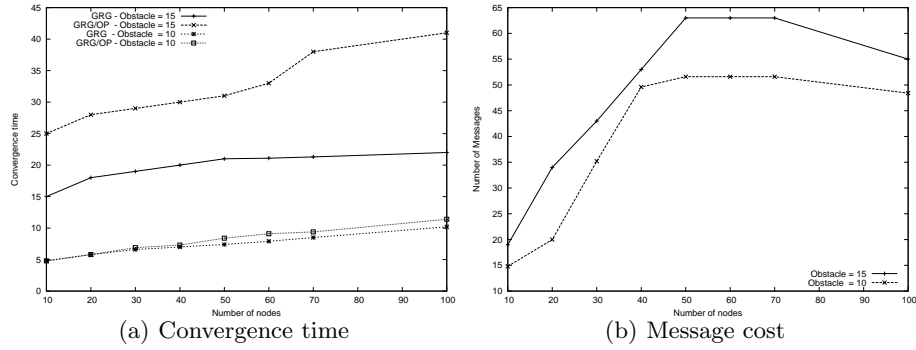
11

(a) Convergence time (b) Message cost

Fig. 8. Simulation results

(in fact, that of its residence vertex). After receiving the pressure message, an intermediate tube node increases λ by its own pressure vector, and forwards the updated message to its left tube neighbor. It drops the message if it has no left tube neighbor. The left-end node of ELBD (if applicable) will receive the message and retrieve λ, which implies aggregated pressure from the right.

Apparently, $\lambda = 1$, if the left-end node is at the same level (hexagon layer) as the right-end node; $\lambda < 1$, if it is at a higher level; $\lambda > 0$, otherwise. In the case of $\lambda > 1$, the left-end node sends an *action* message to the right-end node (which may not be the node that initiates the pressure adjustment process) and then flows (to the left by one hop). An intermediate ELBD node flows as soon as it forwards the action message if it is a left-arm node or a bottom node; otherwise, it flows only after its right ELBD neighbor arrives. The right-end node flows immediately after it receives the action message.

6 Simulation results

We implemented GRG (the -CW variant) and GRG/OP using a custom network simulator with reliable MAC layer; we simulated their execution over a mobile sensor network randomly and uniformly dropped in plane. We comparatively study their performance on convergence time, i.e., the number of time units that it takes the network to stabilize. We also study the message cost of GRG/OP for obstacle avoidance. In our simulation, sensors have sensing radius 0.03 and communication radius $0.05 \approx \sqrt{3} \times 0.03$; they may move at different speeds, ranging from $2m.s^{-1}$ to $10m.s^{-1}$ per simulated time unit, for every step. We fix the dropping area to size 1×1, and take its geographic center as POI.

There is a single obstacle nearby POI, generated in the following way: we first draw a TT graph and choose a vertex as the based of the obstacle; then we grow the obstacle around the base vertex (to ensure a convex shape) till the desired size is reached. We used two different obstacle sizes. One is 10-vertex large, meaning the obstacle occupies 10 vertices; the other is 15-vertex large. We vary the network size from 10 to 100 nodes. By this means, we are

12

able to investigate the impact of node density and obstacle size on algorithm performance. For each simulation setting, we executed GRG and GRG/OP in 50 randomly generated network scenarios and computed average results, which are going to be elaborated below.

In GRG, nodes simply stop when their rotation next hops are obstructed by the obstacle; whereas, in GRG/OP, they pass around the obstacle by simulating fluid behavior in a U-tube. Therefore, we can expect that GRG/OP has larger convergence time then GRG. This expectation is confirmed by Fig. 8(a), where the convergence time curves of GRG are blow their counterparts for GRG/OP. From the figure, we also observe that convergence time is long for scenarios with large-sized obstacle. It is because nodes have to move along a long path to pass around a large obstacle, increasing convergence time.

We ignore communication (used by lower layer protocols) for neighbor information gossips. GRG/OP generates messages only for obstacle penetration, and message transmission takes place only in virtual tube (i.e., along obstacle). The length of a virtual tube depends on the size of the part of the obstacle included in the final network. The larger the part, the longer the tube, and thus the higher the message cost. In our simulation, the tube length increases when the obstacle becomes bigger. Consequently the message cost curve for large obstacle is always above that for small obstacle in Fig. 8(b).

When obstacle size is fixed, the larger the network size, the more the obstacle is included in the final network, and therefore the longer the tube. This indicates, the message costs of GRG/OP goes up as the network size increases, which can be observed in Fig. 8(b). Notice that after the network size is beyond a threshold, the message cost stabilizes. It is because, when the obstacle is completely included in the network, nodes above it are no longer affected (i.e., no extra message transmission). The threshold value clearly depends on obstacle size. In our simulation it is 40 for small obstacle and 50 for large obstacle.

7   Conclusions

Sensor self-deployment for *focused coverage* is an emerging research issue. The only known solution Greedy-Rotation-Greedy (GRG) [10] was designed for ideal environment with no obstacle. It has limited applicability in practice. In this paper, we removed this assumption by adding a novel obstacle penetration ability to GRG. This version of GRG with Obstacle Penetration (GRG/OP) can be used in realistic obstacle-prone environment. It enables mobile sensors to behave like fluid when obstructed by obstacles, and preserves the optimality of final coverage. We simulated GRG/OP and studied its convergence time and message cost in a single-obstacle environment. In the future, we will test its energy cost for nodal movement and in multi-obstacle scenarios. We notice that GRG/OP can be easily extended for solving area coverage problem if we treat the border of the region of interest (ROI) as obstacle. The point of interest (POI) can be the geographic center of ROI or a collectively computed location, e.g., the location of an elected sensor. This extension will be part of our future work.

References

1. Bai, X., Kumary, S., Xuan, D., Yun, Z., Lai, T.H.: Deploying Wireless Sensors to Achieve Both Coverage and Connectivity. In Proc. of ACM MobiHoc. 131-142 (2006)
2. Bartolini, N., Calamoneri, T., Fusco, E.G., Massini, A., Silvestri, S.: Snap and Spread: A Self-deployment Algorithm for Mobile Sensor Networks. In Proc. of IEEE DCOSS. 451-456 (2008)
3. Chellappan, S., Bai, X., Ma, B., Xuan, D.: Sensor networks deployment using flip-based sensors. In Proc. of IEEE MASS. 291-298 (2005)
4. Garetto, M., Gribaudo, M., Chiasserini, C.-F., Leonardi, E.: A Distributed Sensor Relocation Scheme for Environmental Control. In Proc. of IEEE MASS. 1-10 (2007)
5. Heo, N., Varshney, P.K.: 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)
6. Howard, A., Mataric, M.J., Sukhatme, G.S.: Mobile Sensor Network Deployment using Potential Fields: A Distributed, Scalable Solution to the Area Coverage Problem. In Proc. of DARS. 299-308 (2002)
7. Howard, A., Mataric, M.J., Sukhatme, G.S.: An Incremental Self-Deployment Algorithm for Mobile Sensor Networks. Autonomous Robots. 13(2), 113-126 (2002)
8. Poduri, S., Pattern, S., Krishnamachari, B., Sukhatme, G.: Using Local Geometry for Tunable Topology Control in Sensor Networks. IEEE Tran. on Mobile Computing (to appear)
9. Li, X., Frey, H., Santoro, N., Stojmenovic, I.: Localized Self-Deployment of Mobile Sensors for Optimal Focused-Coverage Formation. TR-2007-13. SITE, University of Ottawa, Canada (2007)
10. Li, X., Frey, H., Santoro, N., Stojmenovic, I.: Localized Sensor Self-Deployment with Coverage Guarantee. ACM SIGMOBILE Mobile Computing and Communications Review. 12(2), 50-52 (2008)
11. Li, X., Frey, H., Santoro, N., Stojmenovic, I.: Localized Sensor Self-deployment for Guaranteed Coverage Radius Maximization. In Proc. of IEEE ICC (to appear)
12. Li, X., Nayak, A., Stojmenovic, I.: Sensor Placement in Sensor and Actuator Networks. Wireless Sensor and Actuator Networks: Algorithms and Protocols for Scalable Coordination and Data Communication, Wiley (to appear)
13. Ma, M., Yang, Y.: Adaptive Triangular Deployment Algorithm for Unattended Mobile Sensor Networks. IEEE Tran. on Computers. 56(7), 946-958. (2007)
14. Mousavi, H., Nayyeri, A., Yazdani, N., Lucas, C.: Energy Conserving Movement-Assisted Deployment of Ad hoc Sensor Networks. IEEE Communications Letters. 10(4), 269-271 (2006)
15. Wang, G., Cao G., La Porta, T.: Movement-Assisted Sensor Deployment. IEEE Tran. on Mobile Computing. 5(6), 640-652 (2006)
16. Yang, S., Li, M., Wu, J.: Scan-Based Movement-Assisted Sensor Deployment Methods in Wireless Sensor Networks. IEEE Tran. on Parallel and Distributed Systems. 18(8), 1108-1121 (2007)
17. Zhang, H., Hou, J.C.: Maintaining Sensing Coverage and Connectivity in Large Sensor Networks. Ad Hoc & Sensor Wireless Networks. 1(1-2), 89-124 (2005)
18. Zou, Y., Chakrabarty, K.: Sensor deployment and target localization in distributed sensor networks. ACM Tran. on Embedded Computing Systems. 3(1), 61-91 (2004)