Fast Localized Sensor Self-Deployment for Focused Coverage*

László Blázovics[1] and Tamás Lukovszki[2]

[1]Department of Automation and Applied Informatics, Budapest University of Technology and Economics, Budapest, Hungary
laszlo.blazovics@aut.bme.hu
[2] Faculty of Informatics, Eötvös Lóránd University, Budapest, Hungary
lukovszki@inf.elte.hu

Abstract. We consider the focused coverage self-deployment problem in mobile sensor networks, where an area with maximum radius around a Point of Interest (POI) must be covered without sensing holes. Li et al. [9][10] described several algorithms solving this problem. They showed that their algorithms terminate in finite time. We present a modified version of the Greedy-Rotation-Greedy (GRG) algorithm by Li et al., which drive sensors along the equilateral triangle tessellation (TT) graph to surround a POI. We prove that our modified GRG (mGRG) algorithm is collision free and always ends up in a hole-free network around the POI with maximum radius in $O(D)$ steps, where D is the sum of the initial distances of the sensors from the POI. This significantly improves the previous bound on the coverage time. The theoretical results are also validated by simulations.

Keywords: self-deployment, mobile sensor network, localized algorithms

1 Introduction

Mobile sensor networks (MSN) are distributed collections of nodes, where each node has sensing, computation, communication and locomotion capabilities.

By assuming a large scale sensor network with unpredictable sensor failure, limited sensing and communication range, decentralized or localized sensor self-deployment methods are more beneficial and scale invariant than centralized

* Partially supported by the EU EIT project SmartUC and the grant EITKIC-12-1-2012-0001 of the National Development Agency. Performed in cooperation with the EIT ICT Labs Budapest Associate Partner Group. Partially supported by the European Union and the European Social Fund through project FuturICT.hu (grant no.: TAMOP-4.2.2.C-11/1/KONV-2012-0013) organized by VIKING Zrt. Balatonfüred. Partially supported by the Hungarian Government, managed by the National Development Agency, and financed by the Research and Technology Innovation Fund (grant no.: KMR_12-1-2012-0441). This work is connected to the scientific program of the "Development of quality-oriented and cooperative R+D+I strategy and functional model at BUTE" project.

solutions. In this context *localized* means that each sensor makes independent decisions using neighborhood information only.

There are situations where sensors should cover a dedicated area around a Point of Interest (POI). These scenarios are typical in such cases like area discovery for survivors around the epicenter of disaster. In these cases the area close to the POI has higher priority and it is more important to be covered than the distant one. This type of coverage is called focused coverage or F-coverage [10].

In this article we present a localized, synchronous algorithm for the sensor self-deployment problem with optimal F-coverage.

1.1 Focused coverage (F-coverage)

We follow the terminology of Li et al. [10]. The *coverage region* of a sensor network is the region which is enclosed by the outer boundary of the network. If the coverage is not complete there will be still *sensing* (or coverage) *holes*. Sensing holes are closed areas inside the coverage region which are not not covered by the sensing range of the sensors.

The *coverage radius* (or *radius of an F-coverage*) is the radius of the maximal hole-free disc enclosed by sensors and centered as POI. The optimal F-coverage has maximized coverage radius. If the number of sensors is unlimited and the sensing radius of the sensors approaching zero then the maximum hole-free disc has a circular shape. Since the sensing radius of the sensors is finite, we consider a discrete variant of coverage radius measured by *layer distance*. Layer distance, also called convex layers in computational geometry represents the number of successive complete convex polygons adjacently surrounding POI. More precisely, we consider a discrete set of convex polygons $P_i, (i = 1, 2, ...)$ composed of sensors, centered at POI, and having a diameter of $i \cdot d$ for some constant d. Then the coverage radius is the maximum value k, such that P_k is completely in the coverage region.

1.2 The equiliteral triangle tessellation

The equiliteral triangle tessellation is a tiling of the plane in equiliteral triangles with no overlaps and no gaps. The equalateral triangle tessellation (TT) maximizes the coverage area of a given number of sensors without coverage gap when sensor separation is equal to $r_s\sqrt{3}$, where r_s is the sensing radius of the sensors [1], [11]. If the communication radius r_c of the sensors is at least $r_s\sqrt{3}$, the deployment of the sensors corresponding to a TT layout guarantees the connectivity of the network. The convex polygons defining the layers and the layer distance of the F-coverage are hexagons centered at the POI.

1.3 Problem Statement

We are given n mobile sensors with communication radius r_c and sensing radius is r_s of each, $r_c \geq r_s\sqrt{3}$. We assume that the n mobile sensors are initially placed at the vertices of the TT, such that each sensor is placed in a different vertex. This is an unrealistic assumption if the sensors are dropped from a plane. In that case the sensors can perform the Snap and Spread algorithm by Bartolini et al. [2] to achieve the above condition. The sensors may be disconnected at the beginning. All sensors have a common coordinate system and they all know the location of the POI. Without loss of generality, the POI at the origin of the coordinate system. Furthermore, the sensors only have information about their 1- and 2-hop neighbors. The sensors are able to move only on the edges of the TT graph (see Fig. 2). They all move synchronously with uniform speed, s.t. they travel an edge of the TT in one time unit.

The sensors operate corresponding to the *Look-Compute-Move* model. In one cycle, a sensor takes a snapshot of the current configuration (Look), makes a decision to stay idle or to move to one of its adjacent nodes (Compute), and in the latter case makes an instantaneous move to this neighbor (Move).

The motion ends when the sensors uniformly surround the POI by forming hole-free network with maximized coverage radius. From now on we will use the terms node and sensor interchangeably.

1.4 Our Contribution

We present a modified version of the GRG/CV algorithm of Li et al. [10]. We prove that our modified GRG (mGRG) algorithm guarantees, that after $O(D)$ steps each node reaches its final layer, where D is the sum of initial hop distances of the nodes from the POI in the TT. We validate our theoretical results also with simulations.

An important difference between the requirements of the GRG of Li et al. [10] and our mGRG algorithm is that the GRG in [10] uses the knowledge about the 1-hop neighborhood of the sensors, while our mGRG algorithm needs the knowledge of the 2-hop neighborhood. We give examples, that show that the knowledge about the 2-hop neighborhood of the sensors is necessary to avoid collision situations and make the deployment process faster.

This paper is organized as follows. Section 2 gives an overview of related work. In Section 3 we introduce our mGRG algorithm and mathematical notations. We prove the convergence of the mGRG algorithm in Section 4 and present an $O(D)$ upper bound on the surrounding time. Section 5 presents our experimental results. Finally, Section 6 summarizes the work.

2 Related Work

In the field of mobile sensor networks sensor self-deployment problem has been an important research topic that deals with autonomous coverage formation.

In the article of Gage et al.[6] three type of formation was introduced. However it was a military oriented article, from the perspective of the F-coverage only the *blanket* formation is relevant. In this formation the nodes form static connected group in order to maximizes the detection rate of targets appearing within the coverage area.

The most common sensor self-deployment method is the vector or virtual-force-based approach. The algorithms which rely on this approach use potential fields, generated around the sensors which moves the neighbors by attract or repulse them (depending on the distance). The first work which used this approach was published by Howard et al. [7].

Large amount of research deals with sensor deployment algorithms for coverage formation over a Region of Interest (ROI). An excellent summary can be found in the works of Nayak et al. [12] and Brass et al. [3].

Cortes et al. [5] proposed Voronoi diagram based sensor self-deployment method for the coverage of the ROI. The main idea of self-deployment with Voronoi diagrams is to move sensors to minimize their local uncovered areas (equivalently speaking, to maximize their sensing-effective areas) by aligning their sensing range with their Voronoi regions as much as possible.

Li at al. [10], [8], [9] introduced the F-coverage problem. They solved the problem in a discrete case on an equilateral triangle tessellation. Collision of sensors during the deployment was allowed, i.e. more than one sensors can occupy the same triangle vertex at the same time. They presented a proof of the convergence of their solution within finite time. The convergence time, energy consumption and number of collisions has been evaluated by simulations.

In the work of Yang et al. [13] a distributed load-balancing sensor self-deployment algorithm was presented which partitions the plane into a 2D mesh, and treats nodes as load. By this algorithm, nodes in each cell form a cluster covering the cell and are managed by an elected cluster head. This approach also requires dense network coverage and inter-agent communication.

Bartolini et al. [2] have presented a localized algorithm on a hexagonal grid map in which the entities simultaneously use the *snap* and the *spread* activities in order to cover the given area. The nodes are dispersing from their initial position while occupying the free hexagons. On each occupied hexagon only the occupier allowed to stay, which forwards the others towards the borderline of the covered area.

Cord-Landwehr et al. [4] studied the problem of gathering mobile robots with an extent at a fixed position as dense as possible to form a disk of minimum radius around the gathering point. The authors present an algorithm for the continuous case and the discrete case, where the robots are moving on a grid. They prove an $O(nR)$ upper bound for the gatheringg time, where n is the number of robots and R is the distance of the farthest robot from the gathering point. They empirically studied the continuous case, where in they report a few deadlock situations in the simulations.

3 The modified GRG

Before the introduction of our modified GRG algorithm, we briefly review the collision avoidance version GRG/CV [10] of the GRG algorithm.

3.1 The GRG/CV algorithm

The Greedy-Roatation-Greedy (GRG) algorithm and its version with collision avoidance (GRG/CV) are designed for the asynchronous model. The self-deployment decision of the sensors only uses the information about the 1-hop neighborhood. In order to keep the description simple, we declare the POI unoccupyable. Therefore, the POI related rules were not used. The sensors try to move toward the POI along the TT edges and decrease the hop distance to the POI step by step. This movement is called greedy advance movement. If the greedy advance movement is blocked, the sensors use another type of movement, called rotation movement, i.e. they try to move on the same layer. The rotation is restricted to a particular, say the counterclockwise, direction so as to avoid unnecessary collision among rotating nodes. The key is that a sensor should not move away from the POI once it moves closer to it. A sensor stops rotating when it reaches a vertex where greedy advance can resume, or when it returns to the vertex where it started rotating or the rotational movement is blocked. In the case that a greedy advance movement and a rotation movement target the same vertex, a competition rule is applied, which gives higher priority to the greedy advance movement.

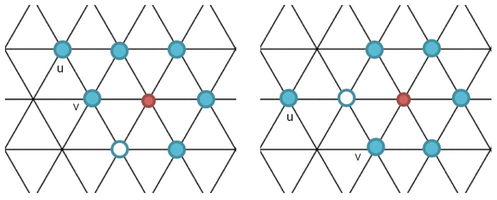
Although the GRG/CV is an asynchronous algorithm, in most cases it works also in a synchronous environment. However, as illustrated on Fig. 1(a), there are situations where a sensor is unable occupy an empty vertex. The sensor u on the second layer is unable to move to the empty vertex, because it only knows its 1-hop neighborhood. Thus, u does not see the empty vertex. Therefore, u rotates counterclockwise. In the same time step v moves to the empty place, and the previous position of v becomes empty. After one step (See Fig. 1(b)) the u is in front of the empty place, however due to the *Safety Rule* in [10] it is not allowed to occupy the empty place[1]. The empty space moves in the first layer in clockwise direction and u on the second layer in counterclockwise direction. After three steps the same situation appears as in Fig. 1(a), just rotated around the POI by an angle of $2\pi/3$ counterclockwise. After making a full circle around the target, u will stop without occupying the empty place.

It is not solved in [10].

3.2 The mGRG algorithm

The main ideas of our concept to make the surrounding of the POI faster are the following. First, each hexagonal layer is assigned a heading direction, such that any two neighboring layers have opposite heading direction. For example, odd layers have counterclockwise and even layers clockwise heading direction. When a sensor on a certain layer performs a rotation step, it moves around the POI in the given heading direction. Second, if a greedy advance movement and a

[1] The Safety Rule in [10] describes that a node must not greedily advance unless it knows the movement is definitely safe. It says that a node u does not choose inward vertex neighbor x as greedy next hop if the neighbor of x on the layer of x in clockwise direction is not a one-hop neighbor of u.



(a) Before the step (b) After the step

Fig. 1. Endless loop with the modified GRG/CV

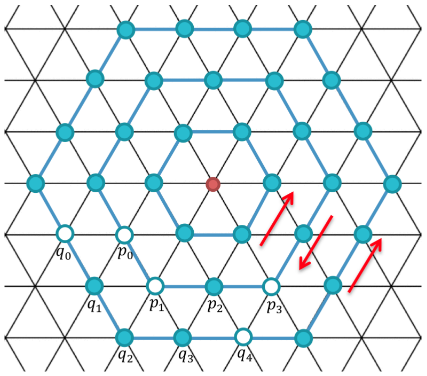


Fig. 2. The hexagonal layers (trajectories) and their heading direction in the equilateral triangle tessellation

rotation movement target the same vertex, the rotational movement gets higher priority. This principle will ensure that each sensor can keep moving in each time step, since rotation will be always possible.

The sensors move straight towards the POI until they reach the innermost hexagonal layer. This is the *primary trajectory* (T_1).

Similarly to the base GRG, if a node is unable to the get closer to the POI – because of another node is in front of it, or it has reached the innermost layer – it should rotate on the current layer. If the node is able to move to an inner layer it should check whether an other node is trying to get to the same place. An example can be seen on Fig. 2.

Now we define the priority rule more precisely. Consider a vertex x on the layer T_i The vertex x has at most four neighboring vertices from which it can be occupied in the next step if x is a corner vertex, and at most three, otherwise. One such neighboring vertex is on the same layer T_i and at most three vertices on the next higher layer T_{i+1}. Regarding x the highest priority is assigned to the neighbor vertex on T_i. The heading direction T_{i+1} defines an order on the neighbor vertices on T_{i+1}. The first one in this order gets the second highest priority, the second the next highest, etc... For example, in Fig. 2 vertex p_1 can be occupied from vertices q_1, q_2, q_3, p_2. The priority order from highest to lowest is p_2, q_3, q_2, q_1. A sensor u obtains the same priority than the vertex currently occupied by u. A sensor u can occupy a vertex x, if no other sensor resides on a vertex with higher priority regarding x. Note that each sensor is aware of the sensors that can occupy the same vertex, since they are in the 2-hop neighborhood of each other. Thus, each sensor can decide locally, whether it has the highest priority among them.

If a sensor u is equally far from two vertices closer to the POI than u (like q_1 from p_0 and p_1 in Fig. 2 and the heading direction of u is counterclockwise (clockwise), then the vertex left (right) from the direction of the POI is prefered. If there is another sensor with higher priority regarding this vertex, then u choose the other. If this vertex also can be occupied by a higher priority sensor, then u must rotate.

These rules imply that that each sensor either moving towards the target or rotating around it, it never stays on same place in the next time step. They also imply that a node must know the 2-hop neighborhood in order to avoid collision and to detect an occupyable vertex in the next inner layer.

4 Analysis

In this section we prove that by using the mGRG algorithm a group of mobile sensors will always enclose a given POI with maximum coverage radius.

We assume that at the beginning each sensor resides on different vertices of the TT and tries to move on the edges towards of the POI.

We say that two sensor u and v are in *conflict* if they may target the same vertex x of the TT in one step.

We prove that the sensors always can move, never stuck in deadlock situation and we give a convergence guarantee of the surrounding process.

4.1 Upper bound on the coverage time

First we show that each sensors can move either into the direction of the POI or rotate on the same layer around the POI. Therefore, the sum of the distances between the nodes and the POI never increases during the process.

Lemma 1. *Each sensor v, which is not on the innermost layer T_1, can move towards the POI if v is not in conflict with another sensor u with higher priority. Otherwise, v can move on its current layer around the POI in the corresponding heading direction. The distance between the POI and the sensor never increases during the coverage process.*

Proof. First we consider a sensor v which is already on the innermost layer T_1. We show that v can move on that layer and its distance never increases. Since all sensors on T_1 (if any) move in the same direction around the POI synchronously, their distance to each other remains the same, and thus, they do not cause a collision. Another sensor u is only allowed to move to T_1, if it does not cause a collision.

Now we consider a sensor v which not in T_1. If v is not in conflict with any another sensor u regarding a neighboring vertex on the next inner layer, then it moves towards the POI and its distance strictly decreases. Otherwise, by similar argument than above, v can rotate on its current layer and its distance from the POI does not change. □

Now we are able to prove a guarantee of the convergence of the coverage process.

Theorem 1. *Until the inner layers have not been occupied with sensors (i.e. an inner orbit T_{in} contains an unoccupied vertex), the sum of hop-distances of the sensors from the POI decreases by at least 1 within $3(i+1)+1$ steps, where i is the index of the innermost layer with an unoccupied vertex T_i.*

Proof. Our rules guarantee that for each sensor the distance from the POI never increases. If a sensor is not prohibited by other sensors, it is moving towards the POI, until it reaches the innermost layer. If a sensor does not decrease its hop-distance to the POI by one unit in a time step, then it is either on the innermost layer or it is in conflict with another sensor.

Let T_i be the innermost layer which contains an unoccupied vertex. Consider a sensor v on the layer T_j, such that $j > i$ and j is smallest among them. If no such sensor exists, then we are done, all inner layers are filled. Otherwise, if v is not at a corner vertex of the hexagonal layer T_j, then v can only be prohibited to move in the direction of the POI by sensors on T_i. If v is at a corner vertex of T_j, it also can be prohibited to move in the direction of the POI by another node v' on a neighboring vertex of the same layer. Then we substitute v by v'. The sensor v decreases its distance until it reaches layer T_{i+1} where it starts the rotation. The sensor v and the unoccupied vertex on T_i rotate in opposite direction. Within $3(i+1)+1$ steps either v can move into the unoccupied vertex on T_i or another sensor filled it before v. Thus, within $3(i+1)+1$ steps at least one sensor decreased its hop-distance to the POI at least by one. □

Theorem 2. *After $O(D)$ time steps all inner layers are filled, where D is the sum of initial hop-distances of the sensors to the POI.*

Proof. For $i \geq 1$, the number of vertices of layer T_i is $6i$. Let i^* be the smallest index, such that the number of sensors n is less than or equal to $\sum_{i=1}^{i^*} 6i$. For $1 \leq i < i^*$, let $m_i = 6i$ and let $m_{i^*} = n - \sum_{1 \leq i < i^*} 6i$. We show that after $O(D)$ time steps all vertices of layers T_i, $1 \leq i < i^*$, become occupied and layer T_{i^*} contains m_{i^*} sensors.

For $1 \leq i \leq i^*$ and $1 \leq j \leq m_i$, let $t_{i,j}$ be the time, when all layers T_ℓ, $\ell < i$, are already filled and the number of sensors on layer T_i increases from $j-1$ to j. Let $v_{i,j}$ be the sensor moving to layer T_i at time $t_{i,j}$. T_i will be the final layer of $v_{i,j}$, since all layers closer to the POI are already filled. Let $V_{i,j}$ be the set of sensors that have already reached their final layer at time $t_{i,j}$. To simplify the description let $t_{1,0}$ be the starting time and $t_{i+1,0} := t_{i,m_i}$, for $1 \leq i < i^*$. Consider the time $dt_{i,j} := t_{i,j} - t_{i,j-1}$, $1 \leq i \leq i^*, 1 \leq j \leq m_i$. Let

$u_{i,j} \in V \setminus V_{i,j-1}$ be a sensor at time $t_{i,j-1}$ which is closest to the POI and not prohibited to move in the direction of the POI by nodes in $V \setminus V_{i,j-1}$. The sensor $u_{i,j}$ can move in the direction of the POI in each time step, until it reaches the layer T_{i+1}. (In case $u_{i,j}$ could not decrease its hop-distance to the POI, then it is in conflict with another sensor w on the same layer. Then we simply replace $u_{i,j}$ by w. Note that at time $t_{i,j-1}$, w was also on the same layer as $u_{i,j}$, since $u_{i,j}$ was closest to the POI.) Then after $O(i)$ steps of rotation steps on layer T_{i+1} it can occupy an unoccupied vertex in T_i, if T_i has not been filled before this time step. Thus, we obtain that $dt_{i,j} \leq d_{i,j-1}(u_{i,j}, o) + O(i)$, where $d_{i,j-1}(u_{i,j}, o)$ is the distance of $u_{i,j}$ to the POI at time $t_{i,j-1}$, which is not greater than the initial distance $d(u_{i,j}, o)$ of $u_{i,j}$ and the POI. Furthermore, $d(u_{i,j}, o) \geq i$. Therefore, $dt_{i,j} \leq c \cdot d(u_{i,j}, o)$, for some constant $c > 1$. Thus we can conclude that the time t until all the sensors reach their final layer is:

$$t = \sum_{i=1}^{i^*} \sum_{j=1}^{m_i} dt_{i,j} \leq \sum_{u \in V} c \cdot d(u, o) = O(D).$$

$\square$

5 Evaluation

In order to evaluate our solution we have implemented both the GRG/CV and the mGRG algorithms in our custom synchronous simulation environment. We performed simulations where the nodes were placed uniformly at random on the vertices of a TT graph. In all scenarios the POI was placed on one of the centre vertex of the TT graph. The sensors know their 2-hop neighborhood in the TT. Due to the synchronous environment, the speed and the taken distance were the same for each sensor in each time step. We made two group of simulations during the evaluation process.

In the first group we measured the performance of the GRG/CV and the mGRG when the *dropping area* was a fixed 30×30 square area and network size was varying from 30 to 315 nodes. With each parameter we performed 20-20 simulations. In the second group we kept the number of the nodes fixed (90), while we varied the size of the dropping area from 25×25 to 45×45. We performed 40-40 simulations with each parameter.

5.1 GRG/CV vs. mGRG

Below we will introduce the results of the two simulation groups. The results are visualised on Fig 3. Fig. 3(a) and 3(b) show, that the mGRG always required less time steps than GRG/CV for finalising the coverage.

5.2 Fixed sized dropping area and varied network size

Fig.3(a) shows the coverage time of the algorithms. As it was noticed in simulations of Li et al.[10] for the GRG/CV algoritm the the curves are tendentially

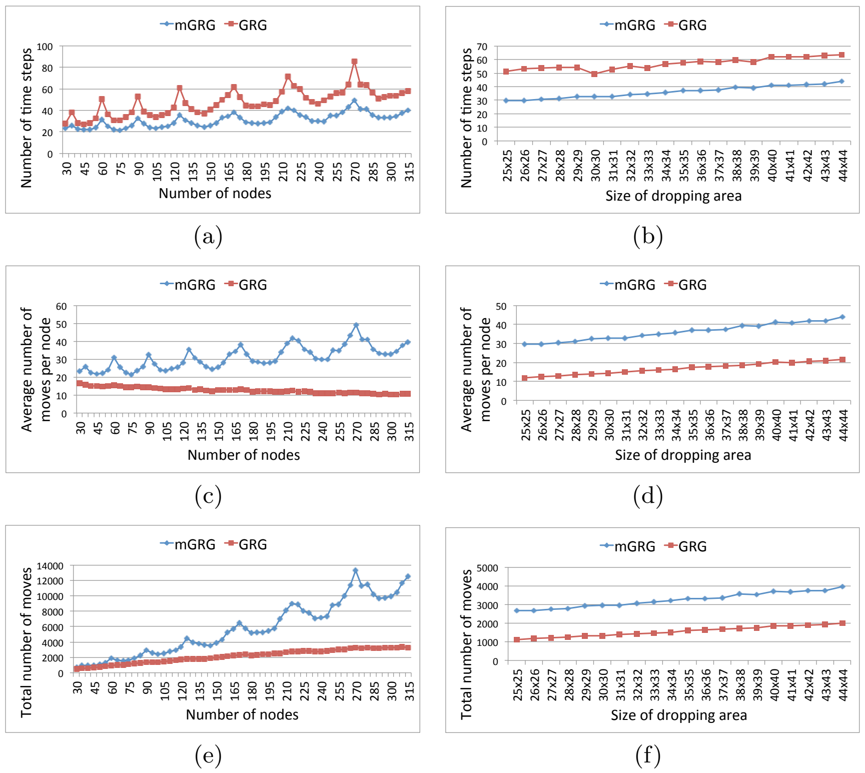


Fig. 3. Simulation results for mGRG (blue) GRG/CV (red), (a)-(c) var. size network fixed size dropping area, (d)-(f) fixed size network, var. size dropping area, (a)(d) Coverage time (b)(e) average moves per node (c)(f) overall moves of nodes

increasing however they contain similar intervals in which they descend. Our mGRG algoritm shows a similar behavior. This is because both algorithms do not converge until there is no node which are able to move closer to the POI. However if the outermost layer has more free vertices than moving nodes, the occupation of these vertices required less steps than those situations when all vertices must be occupied. The latter case can be observed in the peaks of Fig.3(a) where the number of nodes enables to fully fill all the layers. In both situations mGRG always performs better than GRG/CV. It can be observed that difference will be more significant with increasing number of sensor nodes.

Fig.3(c) shows the average number of steps taken by the sensor nodes. Because the nodes in mGRG never stop moving, each node make the same amount of steps which is equal to the convergence time. In contrast, the nodes in GRG/CV often stop moving in order to allow neighbors with higher priority to continue their motion and to avoid collisions. The moving of the nodes is prohibited more frequently as the density of the nodes increases. That is why the average number of steps in GRG/CV decreases and that is the main reason why the GRG/CV is slower than mGRG.

Fig.3(e) shows the total number of moves taken during the coverage process. GRG/CV performs less moves than the mGRG it is less sensitive to number of nodes. However we should note this is because that nodes in GRG/CV often stop and then start moving again. The nodes in mGRG keep moving even they reached their final layer.

5.3 Fixed sized network and varied sized dropping area

Fig.3(b). shows that the mGRG performs the converage faster than the GRG/CV. In addition it is less dependend on the startup constellation. It can be also observed that these curves are monotonically increasing. This is because the number of the entities is fixed and the coverage time only depends on the size of the dropping area.

Fig.3(d) shows, that both the GRG/CV and the mGRG require more steps as the dropping area getting larger. It can be also observed that in that case nodes in the GRG/CV take less steps. This is mainly caused by the stopped nodes as we have already described.

Similar to the results of the first group the overall number of time steps of GRG/CV was less in various field sizes too, as it can be seen of Fig.3(f). As it was already described this is because the nodes in mGRG are always moving even when they have reached their final layer.

5.4 Simulation Summary

In the simulations our mGRG algorithm were always faster but it required more moving steps than the GRG/CV. We note in certain situations like aerial application, the difference between standing or hovering and moving is not significant from the perspective of energy consumption. In those scenarios the coverage time is more relevant.

6 Conclusions

We have presented a new algorithm mGRG to solve the focused coverage problem in self-deploying mobil sensor networks. Our algorithm is a modified version of the GRG/CV algorithm by Li et al. [10]. We have proved that our algorithm always guarantees that the sensor nodes enclose the POI without sensing holes in $O(D)$ time step, where D is the sum of distances of the nodes from the POI in the initial configuration. This significantly improves the previous bound on the coverage time. The theoretical results are also validated by simulations. The simulations show that our mGRG algorithms results in a faster coverage than the GRG/CV.

References

1. Bai, X., Kumar, S., Xuan, D., Yun, Z., Lai, T.H.: Deploying wireless sensors to achieve both coverage and connectivity. In: MobiHoc. pp. 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. 4th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS). pp. 451–456. Springer-Verlag (2008)
3. Brass, P.: Bounds on coverage and target detection capabilities for models of networks of mobile sensors. ACM Trans. Sen. Netw. 3(2) (2007)
4. Cord-Landwehr, A., Degener, B., Fischer, M., Hüllmann, M., Kempkes, B., Klaas, A., Kling, P., Kurras, S., Märtens, M., Der Heide, F.M.A., Raupach, C., Swierkot, K., Warner, D., Weddemann, C., Wonisch, D.: Collisionless gathering of robots with an extent. In: Proceedings of the 37th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2011). pp. 178–189. Springer Verlag, LNCS (2011)
5. Cortes, J., Martinez, S., Karatas, T., Bullo, F.: Coverage control for mobile sensing networks. Robotics and Automation, IEEE Transactions on 20(2), 243–255 (2004)
6. Gage, D.W.: Command control for many-robot systems. Naval Command Control and Ocean Surveilance Center RDT and E Div San Diego CA (1992)
7. Howard, A., Matarić, M.J., Sukhatme, G.S.: An incremental self-deployment algorithm for mobile sensor networks. Auton. Robots 13(2), 113–126 (2002)
8. Li, X., Frey, H., Santoro, N., Stojmenovic, I.: Localized sensor self-deployment for guaranteed coverage radius maximization. In: Communications, 2009. ICC '09. IEEE International Conference on. pp. 1 –5 (2009)
9. Li, X., Frey, H., Santoro, N., Stojmenovic, I.: Focused-coverage by mobile sensor networks. In: 6th IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS). pp. 466–475 (2009)
10. Li, X., Frey, H., Santoro, N., Stojmenovic, I.: Strictly localized sensor self-deployment for optimal focused coverage. IEEE Trans. Mob. Comput. 10(11), 1520–1533 (2011)
11. Ma, M., Yang, Y.: Adaptive triangular deployment algorithm for unattended mobile sensor networks. IEEE Trans. Computers 56(7) (2007)
12. Nayak, A., Stojmenovic, I.: Wireless Sensor and Actuator Networks: Algorithms and Protocols for Scalable Coordination and Data Communication. Wiley-Interscience, New York, NY, USA (2010)
13. Yang, S., Li, M., Wu, J.: Scan-based movement-assisted sensor deployment methods in wireless sensor networks. IEEE Transactions on Parallel and Distributed Systems 18(8), 1108–1121 (2007)