Discrete Mathematics and Theoretical Computer Science DMTCS vol. **17:3**, 2016, 105–130

Robust Wireless Sensor Network Deployment

Milan Erdelj[1] Nathalie Mitton[2] Tahiry Razafindralambo[2*]

[1] *Université de Technologie de Compiègne, France*
[2] *Inria Lille - Nord Europe, France*

received 4th Feb. 2015, revised 15th Oct. 2015, 10th Mar. 2016, accepted 25th Mar. 2016.

In this work we present a decentralized deployment algorithm for wireless mobile sensor networks focused on deployment Efficiency, connectivity Maintenance and network Reparation (EMR). We assume that a group of mobile sensors is placed in the area of interest to be covered, without any prior knowledge of the environment. The goal of the algorithm is to maximize the covered area and cope with sudden sensor failures. By relying on the locally available information regarding the environment and neighborhood, and without the need for any kind of synchronization in the network, each sensor iteratively chooses the next-step movement location so as to form a hexagonal lattice grid. Relying on the graph of wireless mobile sensors, we are able to provide the properties regarding the quality of coverage, the connectivity of the graph and the termination of the algorithm. We run extensive simulations to provide compactness properties of the deployment and evaluate the robustness against sensor failures. We show through the analysis and the simulations that EMR algorithm is robust to node failures and can restore the lattice grid. We also show that even after a failure, EMR algorithm call still provide a compact deployment in a reasonable time.

Keywords: wireless sensor network, deployment

1 Introduction

Environmental monitoring and surveillance is one of the typical applications of wireless sensor networks (WSN). Recently, the advance of technology allows us to extend the possibilities of wireless sensors, above all in terms of data storage capacity, computational power and mobility capabilities. Therefore, these technological advances extend the possibilities and applications of wireless sensor networks in practice.

The ”monitoring” term in the context of WSN applications usually implies the coverage of geographical points, barrier or area of interest. Numerous existing methods for area coverage are based on different techniques, such as virtual forces, coverage pattern and grid quorum. An intelligently conceived deployment algorithm design can provide the user with a sensor network able to self-reconfigure in order to increase the quality of service and dynamically self-redeploy in the case of sudden sensor failures.

Focusing on the grid-based techniques, two approaches stand out in the current state of the art of the grid-based sensor deployment. Wang et al. (2006) propose three distributed algorithms for sensor self-deployment, where they focus on coverage maximization while minimizing the movement distance

*Email: Tahiry.Razafindralambo@inria.fr

and communication complexity. Bartolini et al. (2010) develop an algorithm for sensor deployment in hexagonal formation by repulsing sensors from high density areas and attracting them into coverage holes.

In our work, we propose a novel distributed algorithm for hexagonal lattice mobile sensor deployment that is focused on deployment *Efficiency*, connectivity *Maintenance* and network *Reparation* (EMR). We assume that a set of mobile sensors is located in the field of interest and that the grid-based area coverage should be done in an autonomous manner. EMR deployment algorithm is:

- **distributed**, which means that each sensor runs the algorithm and brings its own movement decisions without the influence from the central unit;
- based on **local knowledge of the network**, i.e., information obtained from the one-hop sensor's neighborhood;
- **asynchronous**, which means that there is no need for any type of synchronization among sensors in the network;
- based on **absolute** or **relative localization** among sensors. It can, thus, operate without absolute localization techniques;
- designed to achieve constant network **connectivity maintenance**;
- the first algorithm that **considers node/link failure** and integrates a **network reparation feature** in the deployment strategy.

Provided simulation campaigns regarding the covered area and the deployment compactness show that this type of local knowledge of the neighborhood is sufficient for achieving area coverage in the field of interest. Furthermore, we show that our algorithm successfully achieves self-reconfiguration and connectivity restoration in the case of multiple sensor failures during the deployment.

The remainder of the paper is organized as follows: the set of assumptions and the notations used in the paper are provided in Sections 2 and 4. Section 3 is devoted to related works. The EMR deployment algorithm and its properties are presented in Section 5. The results of the simulation campaign are provided in Section 6. In Section 7 we discuss the strengths and weaknesses of our algorithm. We conclude the paper in Section 8.

2 Background and motivation

Environmental monitoring has become one of the most important issues in the domain of security. Agents with sensory capabilities are deployed in a certain manner in order to achieve the area, barrier or point of interest monitoring. The full area coverage – meaning the coverage of the maximized surface over the area of interest – is accomplished by the use of several different deployment techniques, generally divided into virtual force, grid quorum and coverage pattern based techniques (see Razafindralambo and Simplot-Ryl (2011) for specific state of the art). The latter approach is inherently used and assumed in numerous works and different applications such as hexagonal cellular backbone network or the pre-deployed static hexagonal wireless sensor network.

The problem of sensor deployment in the area of interest is generally neglected and has not received much attention. The deployment of a sensor network in a regular pattern is assumed to be easily achievable

and, thus, the majority of works in the field of sensor networks or environmental monitoring assume a pre-deployed network and omit the details regarding the deployment itself.

However, in the real application domain, a different paradigm must be taken into consideration. The first problem is the lack of knowledge regarding the deployment field. In the majority of exploration and security monitoring applications, the knowledge of the deployment field is not available, which makes the initial deployment assumptions unrealistic. The second problem, which is more important from our point of view, represents the case of unwanted events such as sudden sensor disappearance or failure, that could ruin the predefined deployment scheme. Therefore, the deployment algorithm should be resilient to unexpected sensor node failures and packet losses. Moreover, since wireless sensor communications depend on propagation model and deployment may rely on sensor communications, the deployment algorithm should be immune to message loss.

Taking all the aforementioned problems into consideration, in the context of sensor networks composed of independent mobile wireless sensing devices, the deployment algorithm should achieve the following:

1) maximal covered area following a grid pattern to optimize coverage,

2) maximal deployment speed,

3) maximal deployment compactness to have an homogeneous deployment,

4) constant network connectivity to transport data from sensors to the data sink,

5) the ability to cope with unexpected sensor failures,

6) localized approach to avoid centralized decision that can lead to complete network failure.

In this work, we present an algorithm that unifies all these demands into one deployment scheme.

3 Related work

Most of the literature work tackles the problem of efficient sensor deployment in the monitoring field, notably by achieving fast and maximized network deployment. One of the pioneering works in the field of autonomous distributed sensor deployment is the work of Howard et al. (2002). In this work, authors implement the node spreading approach in the unknown environment. The proposed algorithm deploys nodes one by one into an unknown environment, with each node making use of information gathered by previously deployed nodes to determine its target location. The algorithm is designed to maximize network coverage whilst simultaneously ensuring network connectivity. In our paper, we pursue the same goal but with no local awareness.

Wang et al. (2006) proposes three distributed self-deployment protocols in order to maximize the coverage area while minimizing the sensor movement and communication complexity. Their approach is based on Voronoi diagrams for discovering the coverage holes, followed by the approaches on moving the sensors from densely populated areas towards coverage holes. Ferrante et al. (2013) introduced an elasticity-based mechanism that swarm particles use in order to self-organize and reach a collectively rotating or translating state. One of the movement patterns that this mechanism achieve is called a hexagonal active crystal, that can be used as a deployment mechanism for a hexagonal grid. While the goal of our work is to achieve an efficient deployment based on the characteristics of the environment, their work focuses on maintaining the swarm formation and is thus based on the characteristics of the swarm. The

stability of swarm motion is studied by Gazi and Passino (2004), where authors provide conditions for collective convergence of a multi-agent swarm. The approach is based on the virtual attractive and repellent forces that make individual agents move to more favorable regions of the field. Similarly, depending on the deployment goal, Liu (2015) proposes different sensor deployment strategies in order to achieve different deployment tasks and extend the network lifetime.

A virtual-force based approach to distributed control of large collections of mobile physical agents is proposed by Spears et al. (2004). Authors introduce a framework that represents a basis for self-organization, fault-tolerance and self-repair, and that is focused on minimality, ease of implementation and run-time efficiency. It is shown how the approach can be used to construct different sensing grids, together with the analysis of potential energy and system phase transitions. The approach is implemented on seven mobile robots to show its practical feasibility.

The problem of sensor placement and sensor dispatch in an arbitrary area of interest with the coverage and connectivity constraints is analyzed by Wang and Hu (2008). Authors present both centralized and distributed approaches to sensor deployment by resolving maximum-weight maximum-matching problem in order to minimize the total movement energy consumption of the sensors. They provide the analysis of the achieved area coverage with the set of sensors in different formations which is of great importance for the distributed sensor deployment calculation. Tan et al. (2009) tackles the problem of sensor self-organization in an arbitrary deployed network with the task of coverage maximization while minimizing the overall sensor movements. They present deployment schemes that are adaptable to deployment scenarios including sensors with arbitrary communication/sensing ranges and without any prior knowledge of the deployment field. The approach presented in their work is based on virtual forces.

The problem of efficient deployment, in terms of coverage speed and coverage area maximization, is discussed by Park et al. (2010). They propose a self-deployment algorithm based on a grid-quorum approach, where the sensing area is divided into cells and sensors are deployed towards grid centers. Li et al. (2010) present a sensor deployment algorithm based on restricted Delaunay triangulation. Their algorithm works by choosing 6 neighbors of a node in the network, and by implementing the virtual force principle to exert the movement. Further developing the deployment algorithms, Li et al. (2011) propose a localized sensor deployment algorithm that relies on locally computable hexagonal tesselation for optimal coverage formation. Two algorithms are proposed, the one where a node greedily advances towards the Point of Interest, and the other where a node greedily advances and rotates around the PoI if the greedy advance is blocked. In this way, the deployment achieves maximized hole-free area coverage.

Xu et al. (2006) puts the accent on a grid-deployment robustness by analyzing the impact of misalignment and random errors that influence coverage. Authors design grid-based deployment schemes with coverage guarantees, and quantify the discrepancy between the ideal and the achieved deployment. Lee and Chong (2008) discuss the problems and practical issues of deploying mobile robots for building mobile robotic sensor networks. They focus on the network's capability of dynamic self-reconfiguration, where neighboring robots interact among each other, move according to their mutual distances and thus form a network of unilateral hexagons over a two-dimensional plane. They prove the convergence of the algorithm, network self-reconfiguration and robustness. While Poe and Schmitt (2009) analyzes the coverage, energy consumption and message delay in the case of random, square grid and tri-hexagon tiling sensor deployments, Yun et al. (2010) studies deployment patterns to achieve full coverage and $k = 6$ connectivity under different ratios of the sensor communication (r_C) and sensing (r_S) ranges.

A hexagonal grid-based deployment algorithm is proposed by Xiao et al. (2010). This approach relies on ant colony algorithm and achieves a full area coverage while trying to minimize the movement

distances. However, authors provide only scarce algorithm evaluation results notably regarding the computational cost and the deployment speed. Another decentralized algorithm for hexagonal topology formation for an arbitrary and sufficiently dense sensor network is presented in Prabh et al. (2009), where the hexagonal backbone network is created based on the existing pre-deployed topology.

By combining the virtual forces and grid-based approach for sensor deployment, Mahfoudh et al. (2014) created a hybrid algorithm that achieves area coverage and network connectivity by employing virtual forces method, and eliminates node oscillations by relying on grid-based approach. The proposed algorithm divides the deployment area into virtual cells, where each cell center determines a desired location of a sensor node. Authors conclude that the elimination of node oscillation in this technique reduces the overall energy consumption by the network.

Although achieving efficient area coverage, cited works do not point out the problem of sensor failures and preserving constant connectivity of the network in these cases. The problem of sensor deployment and re-deployment in case of sensor failures is studied by Lin et al. (2015). Authors consider the problem from two perspectives: global deployment of sensors and local sensor network repair. The approach relies on the first algorithm that achieves sensor deployment, while the second algorithm is launched when particular sensors consume all the energy and thus create a void in the network. The deployment is based on virtual forces among nodes, with the addition of boundary repulsive force, and it is implemented on a proof-of-concept robotic testbed. Similarly, network disconnection issues that occur in the case of sensor failures and fast reconnection techniques are proposed by Cheng and Huang (2015). Authors focus on reducing the response time in the case of node failures, reducing the overall moving distances and prolonging overall network lifetime in that way.

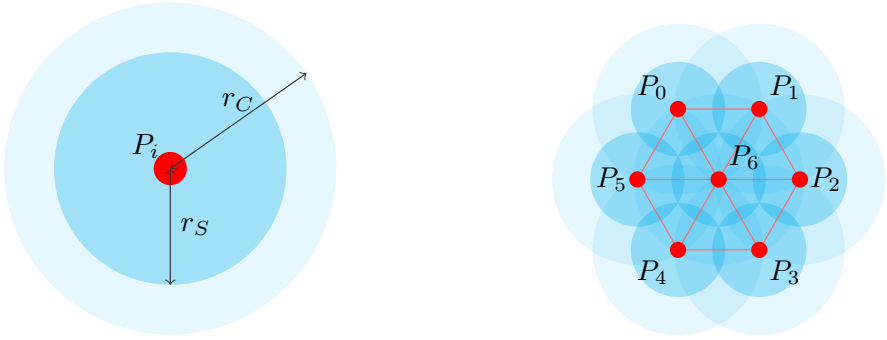
The work that approaches the sensor deployment in the most similar fashion to ours is presented by Bartolini et al. (2010), where authors propose an algorithm for grid deployment that does not require any prior knowledge of the area of interest, and that achieves complete uniform coverage even in the case of irregular shaped target areas. Their algorithm achieves a hexagonal tiling by spreading the sensors out of the higher density areas and pushing them into the detected coverage holes. All the decisions on movement are brought on locally available information. The main difference in comparison to our work is the question of constant network connectivity and algorithm's ability to cope with sudden sensor failures. However, the obstacle avoidance scheme proposed by Bartolini et al. (2010) is very efficient and we will inspire from this part to enhance our own approach in future works.

All these works provide an analysis of deployment strategy or algorithm which aim at providing a grid deployment and/or maximal area coverage. While most of the related works address only some of the issues, none of the works addresses all the deployment issues mentioned above.

4 Assumptions and definitions

In this section we provide the reader with the set of assumptions and definitions that will later ease the explanation of proposed deployment algorithm.

We assume that a set of mobile wireless sensors is deployed in the field of interest with the goal of **maximizing the covered area** and **minimizing the time of complete coverage**. Absolute geographical positioning mechanism is neither needed nor assumed, however, sensors should be able to deduce their relative positions among each other, similarly to the assumption of Bartolini et al. (2010). Dedicated sectored distance sensors implemented in each of the mobile sensors can be assumed and easily implemented. In this manner, a sensor would be able to deduce its relative position with regards to its neighbors. For the



(a) (b)

Fig. 1: (a) Sensor P_i with its sensing r_S and communication r_C ranges. (b) The set of neighboring sensors of the sensor P_6.

sake of simplicity, we do not discuss the sensing aspect of the sensor and its hardware implementation. We refer the interested reader to Erdelj et al. (2013) for more details. In order to ease the understanding of the deployment algorithm, we introduce the following notations for the sensor and the sensor network.

We assume a set $\mathcal{S}$ of $|\mathcal{S}| = n$ sensors. Each sensor $s_i \in \mathcal{S}$ is characterized by its unique identifier $i, i \in \{0, .., n\}$, sensing (r_S) and communication ranges (r_C) as shown on Figure 1(a) and it is assumed that the sensors are able to periodically broadcast and receive $Hello$ messages. Hello messages are emitted at a frequency $1/T_{Hello}$ and contain the information about the sensor's identifier, position and routing gradient g_i (see Definition 8).

The deployment algorithm in this work focuses on a hexagonal lattice grid, where each sensor is aimed at covering one grid point and has maximum of 6 neighbor locations. In this paper, we assume that $r_C = \sqrt{3} r_S$ in order to ensure the constraint of only one-hop direct communication. However, this assumption can be easily relaxed to any ratio between r_C and r_S such as studied by Yun et al. (2010). The set of sensors that share the same grid point with sensor s_i is referred to as $\mathcal{N}$, while the one-hop neighboring sensors of sensor the s_i is referred to as N^i.

For the sake of simplicity, we name each specific point of the deployment following Figure 1(b). In this figure, the actual position of the sensor is always P_6 and its set of possible destinations is $P_{j \in \{0,..,5\}}$. We assume that a node/sensor has a unique origin before it gets to its actual position P_6. For example, before reaching its actual position, sensor s_i may come from any of the $P_{j \in \{0,..,5\}}$ locations.

We will discuss later how we choose and modify this origin. Let us assume that the origin position of sensor s_i is $P_x, x = \{0, .., 5\}$.

Definition 1 *The set of regressive positions N_r^i of node s_i is: $N_r^i = \{P_j \mid j = (x) \pmod 6 \ \vee \ j = (x+1) \pmod 6 \ \vee \ j = (x+5) \pmod 6\}$, where x is the number of origin position. For example, if the sensor's s_i origin is $x = 5$, $N_r^i = \{P_4, P_5, P_0\}$. The number of elements in N_r^i is at most 3. It is important to notice that the origin position is used to restore connectivity in case of failure.*

Definition 2 *The set of progressive positions N_p^i of node s_i is: $N_p^i = \{P_j \mid j = (x+2) \pmod 6 \ \vee \ j = (x+3) \pmod 6 \ \vee \ j = (x+4) \pmod 6\}$, where x is the number of origin position. It represents the*

set of positions where the sensor should move to extend the coverage. For example, if the origin of the sensor s_i is $x = 5$, then $N_p^i = \{P_1, P_2, P_3\}$. The number of elements in N_r^i is at most 3.

Definition 3 *The set of neighbors of node s_i denoted by $\mathcal{N}^i$: $\mathcal{N}^i = \{s_k \mid s_k \in \mathcal{S} \wedge s_k$ is located on $P_i, \ i = 0,..,6 \ \wedge \ s_k \neq s_i\}$. The node s_k is considered to be a neighbor of s_i if it is located on any spot $P_j, i = \{0,..,6\}$ relatively to s_i.*

Definition 4 *We define the occupancy number $\mathcal{O}_j^i$ of position P_j, $j = \{0,..,6\}$ relatively to the sensor s_i as the number of sensors located at point P_j.*

Definition 5 *The set of non-occupied (or void) positions N_v^i of node s_i is: $N_v^i = \{P_j \mid \mathcal{O}^i_{j_{j=0..5}} = 0\}$. Note here that $P_6 \notin N_v^i$ i.e. $\mathcal{O}_6^i > 0$ since sensor s_i itself is located at P_6.*

Definition 6 *We define $s^i_{6min} = \{s_j \mid \min_{s_j \in P_6} j\}$. s^i_{6min} is the sensor located at P_6 with the smallest unique identifier. s^i_{6min} can be s_i or any other node located at the same position as s_i depending on the unique identifier.*

Definition 7 *Two sensors s_i and s_k are connected even if $s_k \notin \mathcal{N}^i$ if there exists at least one sequence called SEQ of connected nodes that are also connected to s_i and s_k. $SEQ = \{s_i...s_j s_{j+1}...s_k\} \ \forall j, s_j \in N^{j-1}$, $s_0 = s_i$ and $s_m = s_k$. The sensor network is said to be connected if every pair of sensors is connected. The number of sensors in $m = |SEQ|$ is called the number of hops.*

Definition 8 *The gradient number g_i of a sensor s_i is a natural positive number that represents the minimum hop number between s_i and s_0, where s_0 is a specific static sensor called the sink.*

Assumption 1 *Each node initializes its gradient number equal to infinite, except node s_0 for which $g_0 = 0$, g_0 never changes. In case of a failure and recovery of a node or a new arriving node in the network, this value is again set to infinity.*

Definition 9 *A node s_i keeps track, in a list $\mathcal{L} = L_1, ..., L_\infty$, of all its movements in order to be able to come back to its origin position in case of a failure. This value is different from the P_x value, and represents the actual real path a node has taken. See Remark 2 for the size of $\mathcal{L}$ (Page 118).*

Assumption 2 *We assume that all the sensors are located at the same location at the beginning of the deployment. This location is also the locations of the sensor s_0.*

Assumption 3 *We assume that the sensors have magnetometer to be able to achieve a common orientation.*

5 EMR deployment algorithm: description and properties

In this section, we present our algorithm dedicated to sensor deployment Efficiency, connectivity Maintenance and network Reparation (EMR), we describe its steps and discuss its properties.

5.1 Deployment Algorithm

The pseudo code of the algorithm is given in Alg. 1. The algorithm is divided into three overall parts:

1. movement *decision* (start at line 1 of Alg. 1),
2. movement *direction* (start at line 7 of Alg. 1),

Data: Each sensor s_i runs the algorithm
Result: Sensor destination and movement

1 /* Movement decision */
2 **if** ($\mathcal{O}_x^i = 0$ || ($\mathcal{O}_6^i \geq 2 \wedge s_i \neq s_{6min}^i$)) **then**
3 | $CanMove \leftarrow TRUE$
4 **else**
5 | $CanMove \leftarrow FALSE$
6 **end**

7 /* Movement direction */
8 **if** ($CanMove = TRUE$) **then**
9 | **if** ($\mathcal{O}_x^i = 0$) **then**
10 | | $NextPosition \leftarrow P_x$
11 | **else**
12 | | **if** ($N_r^i \cap N_v^i \neq \emptyset$) **then**
13 | | | $NextPosition \leftarrow$ rand($\{P_j \in N_r^i \cap N_v^i\}$)
14 | | **else**
15 | | | $NextPosition \leftarrow$ rand($\{P_j \in N_p^i | j = \min(O_j^i)\}$)
16 | | **end**
17 | **end**
18 **end**

19 /* Movement realization */
20 **if** ($CanMove = TRUE \wedge NextPosition \neq P_6$) **then**
21 | Move to $NextPosition$
22 **end**

ALGORITHM 1: EMR deployment algorithm.

3. movement *realization* (start at line 19 of Alg. 1).

To ease the reading, we will use the following notation: "Alg.1:7" stands for "the line 7 of the algorithm Alg. 1" and "Alg.1:7–18" stands for "lines 7 to 18 of algorithm Alg. 1". The algorithm is executed at each sensor except at the sink s_0, and does not require any kind of synchronization. In the following, s_i describes the actual sensor that runs the algorithm and stores all its related local variables.

Movement decision

The node s_i may decide to move or not based on the outcomes of this step, described in Alg.1:1–6. It will decide to move (variable $CanMove$ set to $TRUE$) only in two cases: *(i)* its point of origin P_x is void i.e. $\mathcal{O}_x^i = 0$ (Alg.1:2). This means that there is a *hole* in the coverage. *(ii)* there are more than one node located at the same point as node s_i, that is, $\mathcal{O}_6^i \geq 2$ (Alg.1:2). There is a redundancy and thus the coverage can be extended. Only the node with the lowest identifier will remain at this position. All other sensors will move toward a progressive position. It is important to notice here that metrics such as battery level or any other combination of metrics that could uniquely identify a sensor can be used. For example, the level of battery can also be considered (the node with highest level of battery can move) and in case of

tie, the node with the highest identifier will move. In any other case, sensor s_i set its $CanMove$ variable to $FALSE$ as stated in Alg.1:5 and will not move.

Movement Direction

If the output of the previous step is that sensor s_i should move, in this step it has to decide where to move (Alg.1:7–18). This choice is made based on the condition described in Alg.1:9 and Alg.1:12. If $\mathcal{O}_x^i = 0$, the $NextPosition$ of s_i is set to P_x. This condition appears in Alg.1:9. If this condition (Alg.1:9) is not met, s_i tests if there is a non-occupied or free spot in its set of regressive positions. This condition is tested in Alg.1:12. In this case, the free spot is chosen as the $NextPosition$. If the node has more than one possibility, a random choice is made among them (Alg.1:13). If all regressive positions are filled with a sensor, the $NextPosition$ is chosen among the progressive positions with the minimum number of occupancy. In case of tie, a random choice is drawn. These conditions and selection are done in Alg.1:15.

Movement Realization

The last step of the algorithm is the movement itself (Alg.1:20–22). Sensor s_i moves to the computed position. It is important to notice that we assume a sensor s_i knows if it has reached its destination based on possible localization information or based on its engines (odometry). This information is used to send updates to other sensors as detailed in Alg. 2. We use the measured maximum speed of a WifiBot robotic plateform for its movements which is randomly drawn between [0.8;1.2] m/s,

5.2 Local information update

In this section, we describe a part of the algorithm regarding the update of a sensor s_i local variables. The update procedure is described in Alg. 2. The procedure is divided into three sub-procedures, each of which has its own execution way. It is important to highlight that all these update procedures are run by a sensor only after a movement. The first sub-procedure is executed exactly once after the sensor reaches its destination. The second sub-procedure is executed periodically with a given frequency but still after the sensor has reached its destination. Note that this procedure is permanently executed when the sensors has reach its destination and is not moving. Similarly, the last sub-procedure is executed in the background based on external events but only when the sensor reaches its destination.

First, when a node s_i reaches its destination it increases its gradient value by one as stated in Alg.2:4. This value is always incremented even if the node is going back to its origin. Indeed this value will be updated when the node receives a messages from its neighbors to determine the real gradient and the origin point which is the position of the neighbor that give the node the smallest gradient number. It also updates its list of movements $\mathcal{L}$ (Alg.2:5). This movement list is a way to keep track of all the steps undergone by the sensor. It allows the sensor to come back to its initial position in case of failure. It is important to notice that the last entry of $\mathcal{L}$ is not necessarily its point of origin P_x. The sensor also sets a local variable g_{min} to its own gradient value g_i (Alg.2:6). As stated in Remark 2 at Page 118, this list has a finite size. It is also important to notice that if an external positioning system is available, $\mathcal{L}$ can be reduced to the position of the sink. It is also worth noting that if a sensor looses this list, it can request one from its neighbors.

Once these first updates are done, the sensor locally broadcasts a message containing its identifier, gradient value, and any useful information such as global position (if available) to its one-hop neighbors. This procedure of sending messages is repeated periodically (Alg.2:11–12).

Data: Each Sensor s_i runs the algorithm
Result: Update local information

```
1  /* When s_i reaches its destination (executed only once)                    */
2  if ReachDest = TRUE then
3  |   g_i ← g_i + 1
4  |   Update path in L
5  |   g_min ← g_i
6  end
7  /* Send information (executed periodically)                                 */
8  while (ReachDest = TRUE) do
9  |   /* Send information to neighbors                                       */
10 |   /* Wait T_Hello                                                        */
11 end
12 /* Receive information from neighbors (executed in background)             */
13 while (ReachDest = TRUE) do
14 |   /* Wait for next packet                                                */
15 |   /* Received from neighbor s_j at position P_k                          */
16 |   O_k^i ← O_k^i + 1
17 |   if (g_i > g_j) then
18 |   |   g_i ← g_j + 1
19 |   |   P_x ← P_k
20 |   end
21 |   if (g_min > g_j ) then
22 |   |   g_min ← g_j
23 |   end
24 end
```

ALGORITHM 2: Updating local information.

Since each sensor executes the same algorithm, sensor s_i may receive message from its neighbors containing the same information as it sends. Let us assume that sensor s_i receives a message from sensor s_j at position P_k (here the position P_k depends on s_i). This message is processed only if s_i has reached its destination. In this case, sensor s_i updates its local occupancy variable $\mathcal{O}_k^i = \mathcal{O}_k^i + 1$ as stated in Alg.2:16. Sensor s_i updates its gradient if the received gradient $g_i > g_j$. In this case, g_i is set to $g_j + 1$ (Alg.2:18) and P_x is set to P_k (Alg.2:19). Finally, if $g_{min} > g_j$ then g_{min} is set to g_j (Alg.2:22).

5.3 Compactness and robustness

In this section, we describe an extension of our algorithm in order to increase its robustness and the resulting deployment compactness (further details in Section 6.4). Alg. 3 describes this procedure, which is used only in case of failure or to subsequently modify the outcome of the random selection of $NextPosition$. Note that this procedure is only used when the $CanMove$ variable is set to $FALSE$. Moreover, this procedure is executed directly after Alg. 1.

Data: Each Sensor s_i runs the algorithm
Result: Network healing

```
if (CanMove = FALSE) then
    /* Broken network                                    */
    if g_i ≤ g_min then
        NextPosition ← Last entry in L
        Remove last entry from L
    end
end
```

ALGORITHM 3: Compactness and robustness procedure.

When the network encounters a failure, it should be repaired. For example if a set of sensors disappears or a given sensor fails, the network should be able to automatically reconfigure itself. One way to detect a failure is already given in Alg. 1, when occupancy at position P_x is tested. Another way is to test the value of g_i and g_{min}. If g_i is lower of equal to g_{min} that either means that some sensors have disappeared or that s_i is not able to receive messages anymore. In both cases, s_i will set its $NextPosition$ to the last entry of its $\mathcal{L}$ list, remove this last entry and move to $NextPosition$ in order to come back to its previous position.

5.4 Properties

In this section, we assume that Alg. 3 is never executed, we also assume that Alg. 2 is executed correctly without failure. We assume no message loss at transmission and reception. For this, we rely on duplication sending of a message. Assumptions that exclude failures are made in order to ease the proof. This section provides properties of our EMR algorithm such as connectivity and termination under assumed conditions. It is important to notice that in the next section, EMR algorithm deals with message reception failures and sensor failures.

Assumption 4 *In this section, and only in this section, we assume that no failures occur.*

Lemma 1 *Under the assumptions presented in Sections 4 and 5.4 , the occupancy of the origin of node i is never equal to 0 ($\mathcal{O}_x^i \neq 0$). Here, i represents the sensor of interest s_i and x is its origin position (here x is an index for the spots as defined in Section 4).*

Proof: If $\mathcal{O}_x^i \leq 0$, that means that the algorithm does not update correctly its local information. That is, the sensor s_i did not meet the condition Alg.2:20. Since if the node has entered this part, it should have updated its P_x and its gradient number based on a message received from P_x. A value of $\mathcal{O}_x^i \leq 0$ means that a sensor has updated its P_x value based on its real movement and that this value is the latest position recorded in $\mathcal{L}$ (Alg.2:3-7). Let us now assume that $\mathcal{O}_x^i \leq 0$ at time t_k for node s_i. If we analyze recursively what happens to node s_i, we have the following cases:

- at t_{k-1}, s_i did node receive any message from P_x based on $\mathcal{L}$ value of P_x. This can be due to a failure of a node at $\mathcal{O}_6^i$ at time t_{k-1}. Since we assume that failures can not occur, this situation is impossible.

- at t_{k-1}, s_i has decided to move from its position where its occupancy $\mathcal{O}_6^i \leq 1$. This movement is only possible if Alg.1:2 is the elected choice of s_i at time t_{k-2}. Applying this reasoning recursively means that at t_0, $\mathcal{O}_6^i \leq 1$. This is impossible since at t_0 all the nodes are located at the same spot as the sink and failures can not occur (Assumption 4). □

Theorem 2 *Connectivity. Under the above assumptions, if the network is connected at time t_k, it is still connected at time t_{k+1}.*

Proof:

1) Let us assume that at t_k, the network is connected, that every pair of nodes in the network is connected. Let us first assume that between time instants t_k and t_{k+1} only one node runs the algorithm described in the previous sections. The algorithm ensures that if a node moves, its actual position is within the communication range of its previous position (definition of the grid construction and the communication range which is occupied based on Lemma 1).

2) Let us assume that two nodes s_a and s_b are located on the same spot. The algorithm ensures that only one of the two nodes will be allowed to move (Alg.1:2).

3) If more than one node can move between t_0 and t_1, the algorithm ensures that each moving node is leaving a spot with at least one other sensor (Alg.1:2), since node s_i will not enter Alg.1:3 based on assumptions and Lemma 1. Therefore, if the network is connected at time instant t_k, then any possible node movement based on our algorithm will leave the network connected at t_{k+1}. □

Theorem 3 *Gradient consistency. Under above assumptions, the gradient construction is consistent. In other words, nodes closer to the sink will have lower gradient than nodes farther away from the sink.*

Proof: Let us assume that, at time instant t_k, nodes closer to the sink have higher or equal gradient than nodes farther away from the sink, in other words, let us assume that the gradient construction is inconsistent. In our algorithm, we assume that there is no failure and that based on Lemma 1 and Theorem 2 the network is connected. Since the sink has the lowest gradient $g_0 = 1$, the nodes around the sink will have a gradient equal to 2. Applying the reasoning iteratively and based on the update mechanisms described in Alg. 2, the gradient construction is consistent. □

Theorem 4 *No oscillations. Under the above assumption, a node s_i will never go back to any of its previous positions.*

Proof: We know that a node will select one of the two following choices:

1) The node will move to a free spot or

2) the node will move to an occupied spot.

In the first case, a node chooses a free spot, which means that it has never passed through this spot before (since when a node leaves a spot there is at least one node left on this spot (Theorem 2). In the second case, a node will choose a spot that is in its progressive set (Alg.1:15). Since the gradient construction is consistent (Theorem 3), which means that the construction of the progressive set of a node is consistent, a

node will never go back to the spot that it has already visited. If a node goes back to the spot it has already visited, then it means that at a given time instant one visited spot has changed its gradient number to be a part of the node's progressive set, which is self-contradictory to Theorem 3. □

Theorem 5 *Termination. Under the above assumption, and with a finite set of nodes, the algorithm terminates.*

Proof: In our algorithm, a node will stop moving if it is the only node in a spot or if there is no progressive set due to the field boundary or obstacle in the field. In a case when a node reaches the boundary of the field, it cannot move back (Theorem 4). These two conditions mean that if we assume infinite deployment duration, then each node will be in one of the above described states and, therefore, the algorithm terminates. □

Theorem 6 *The covered area is increasing. Under the above assumption, the area covered by the set of sensors is stable or increasing.*

Proof: Based on above theorems (Theorem 2, Theorem 4 and Lemma 1). Indeed, once a spot is covered by a node, it will stay covered (Lemma 1). Moreover, since a node is never coming back to one of tis previous visited spot (Theorem 4 and Theorem 2). This means that the covered area (the number of covered spots by at least one sensor) is stable or increasing all along the deployment procedure. □

Theorem 7 *One sensor per spot. Let us assume that both the number of grid points is infinite while the number of sensors is finite. Under these assumptions, each grid point will be covered by exactly one sensor at the end of the deployment.*

Proof: Based on Theorem 5, we know that the algorithm terminates. Moreover, based on Theorem 6, we know that the covered area is expanding. Therefore, if the deployment field does not have any boundaries, there will be exactly one sensor per spot at the end of the deployment. □

Theorem 8 *Full coverage. If we assume that the number of sensors is infinite and the number of grid points is finite, then we can also assume that, at each time instant, exactly one sensor will make a decision to move (in a sequential way). Under these assumptions, the area will be fully covered at the end of the deployment.*

Proof: Based on previous theorems, we know that the covered surface is expanding or stable (Theorem 6) and that the algorithm terminates (Theorem 5). The combination of Theorem 6, Theorem 4 and Theorem 5 shows that, under the above assumptions, if the algorithm does not reach termination, the coverage is expanding since a sensor will always move to a free spot unless it is the unique sensor on its actual spot. Let us assume that the nodes are ordered by their unique identifier and that each node makes a decision to move in a sequential way. We suppose that at time instant t_k, node $s_{k \pmod N}$ can make a decision to move or not. Taking this condition into account and supposing the sufficient number of nodes, Alg.1:15 will lead to an even distribution of nodes among the spots, which means that the area will be fully covered at the end of the deployment. □

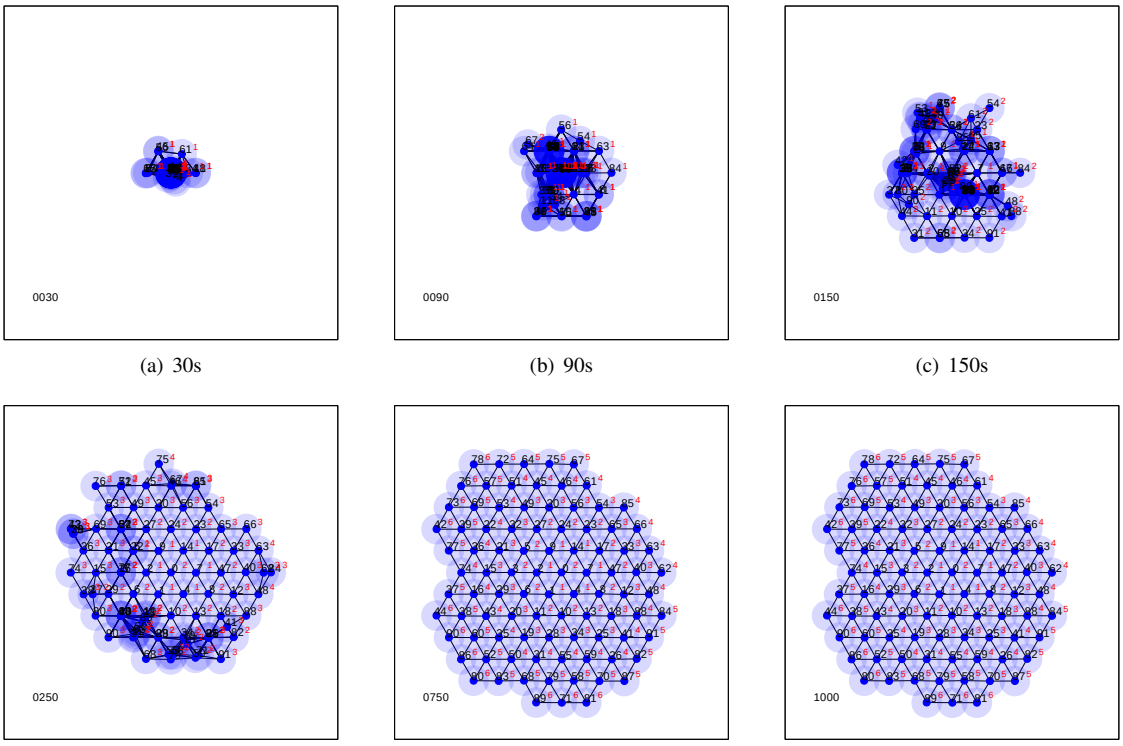
Remark 1 *It is important to notice that in a confined space and with a finite number of sensors the full coverage may not be possible. Please refer to Section 7 for a discussion about confined space.*

Remark 2 *It is important to notice that since the algorithm eventually terminate and that their is no oscillation, the size of $\mathcal{L}$ is bounded.*

6 Evaluation results

In this section we provide a set of simulation results to evaluate the robustness of EMR algorithm. This section is intended to complete the algorithm properties section by relaxing the failure assumptions. Note that in all the following simulations, messages can be lost due to transmission collisions. Simulation results that show the self-healing ability and connectivity restoration are presented in Section 6.5.

The simulation source code, simulation results, and sensor deployment videos are publicly available and can be freely downloaded[(i)].



(a) 30s (b) 90s (c) 150s

(d) 250s (e) 750s (f) 1000s

Fig. 2: Deployment snapshots of the EMR algorithm. These figures show the evolution of the deployment and the formation of the hexagonal grid.

[(i)] http://supdmtcs.gforge.inria.fr/

6.1 Simulation setup

We implement our algorithm in a wireless sensor network simulator WSNet[(ii)]. We use a common wireless sensor network protocol stack for the communication and set our choice to IEEE 802.15.4 standard at the medium access control level which uses a collision avoidance scheme. However, collisions can still occur and messages can be lost. Then, we implement an IP based routing layer, although the routing itself is not used in our algorithm. The physical layer is modelled using a free space propagation with a bit error rate increasing with the distance.

The following parameters are used in the simulation – the communication range is set to $r_C = 18$ m, and the sensing range, used to evaluate the coverage, is set to $r_S = 9$ m. These values are chosen by measurements performed on real robotic platform Wifibot[(iii)] where the video recording represents data sensing. Each $Hello$ message is broadcast every 1 second. The sensors have to wait at least 3 seconds in their position to gather messages from their neighbors before taking the decision to move. We added a random value of [1:10] s to this waiting time to break the synchronization. It is important to notice that this random waiting time slows down the deployment. The speed of sensors is randomly chosen from the interval between 0.8 and 1.2 m/s.

6.2 Deployment

Figure 2 shows an example of a deployment snapshot during the deployment procedure. The lines between the sensor nodes represent the communication links among them. The blue circles represent the sensing ranges. We can see from this example the grid formation and the connectivity preservation all along the deployment procedure. This figure also shows the compactness of the deployment. Indeed, sensors are spread regularly around their starting point and not moving all in the same direction. It is also important to notice that the algorithm terminates as stated in Theorem 5. Results regarding deployment compactness are presented in Section 6.4.

It is hard to compare our algorithm to other algorithms from the literature, such as the one by Bartolini et al. (2010), since the assumptions are not the same. While all the sensors are randomly located at the beginning of the deployment in the paper of Bartolini et al. (2010), in EMR algorithm all the sensors are located close to the sink. Moreover, the algorithm developed by Bartolini et al. (2010) is executed in confined space whereas our is executed in open large field. Anyway, we have tried to compare both algorithm since the objectives are very close.

6.3 Coverage

In Figure 3 we show the coverage evolution depending on time. In this figure, the coverage is given as a percentage of maximal achievable coverage, and the 100% coverage is achieved when each sensor is located on a different spot. The figure is plotted with different number of sensors and it shows two things:

1. the coverage is strictly increasing (as stated by Theorem 6),
2. each sensor is located at its unique spot (as stated by Theorem 7).

We can see from Figure 3 that when the number of sensors is low the deployment reaches the 100% coverage faster since sensors travel less. However, since sensors are not synchronized and move autonomously, the time needed for 128 sensor to reach 80% coverage is much less than the double of time

(ii) http://wsnet.gforge.inria.fr
(iii) http://www.wifibot.com

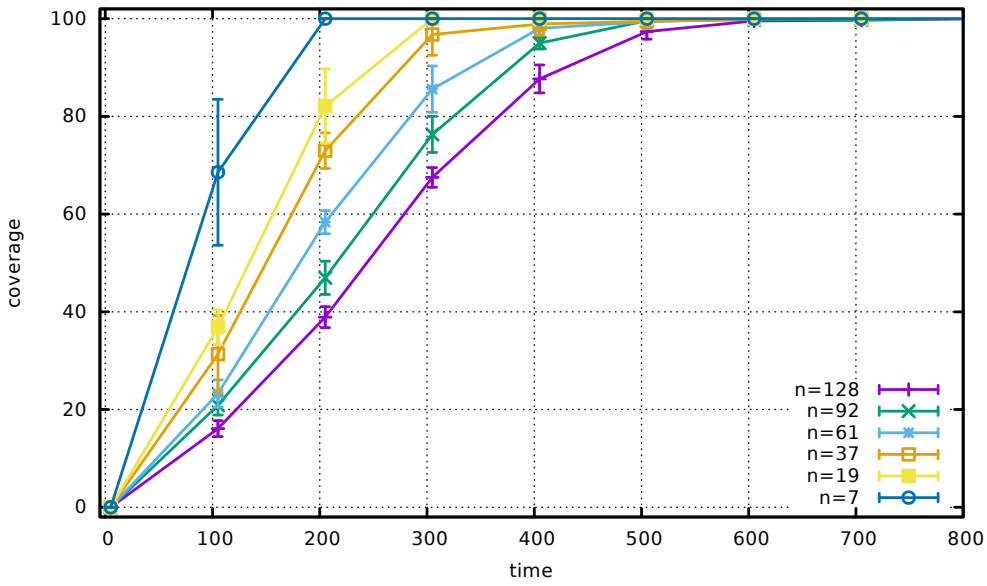


Fig. 3: The percentage of covered area along time for different number of nodes.

needed for 61 sensors to reach 80% coverage. Indeed, one can expect that the time to reach 80% coverage for 128 sensors shoul be the double of the time needed for 61 sensors to reach 80% coverage.

6.4 Compactness

We know that for each gradient number, there is a maximum number of available positions. For example, for gradient 0, there is 1 position, for gradient 1, there are six positions, for gradient 2, there are 12 positions, etc. Therefore, for gradient g there exists $p_g^{max} = g \times 6$ maximum positions for $g \geq 1$ and $p_0^{max} = 1$.

During the deployment, since the movement decision is random in nature and since the covered area is expanding, at the gradient g, there can be a number of sensors greater or lower than the maximum number of available positions. That is: $p_g^{node} \geq p_g^{max}$ or $p_g^{node} \leq p_g^{max}$. Let us assume that we have the number of sensors, n. Based on this value, we can get the perfect compactness regarding deployment. For each gradient number g, we have p_g^{max}, we thus know how many sensors should be at the gradient g. If all the positions at gradient $g < G$ are filled : $p_g^{node} = p_g^{max}$ for $g < G$ and that we have $p_G^{node} < p_G^{max}$, we assume that gradient G is compact since we cannot have more nodes at positions in the gradient G.

We define the compactness of gradient g in percentage as $c_g = \frac{(\min\{p_g^{max}, p_g^{node}\}) \times 100}{p_g^{max}}$. This definition tells us that if at gradient g, $p_g^{node} > p_g^{max}$ then gradient g is considered to be compact. From this definition, we define the average compactness as $C_{ave} = \frac{1}{G_{max}} \sum_g c_g$ where G_{max} is the maximum reachable gradient depending of the number of nodes n. We also define the minimum compactness $C_{min} = \min_g c_g$.

Figure 4 and 5 plots the evolution of C_{ave} and C_{min} depending on time for a simulation with different number of sensors. This simulation result shows that the average compactness reached by our algorithm

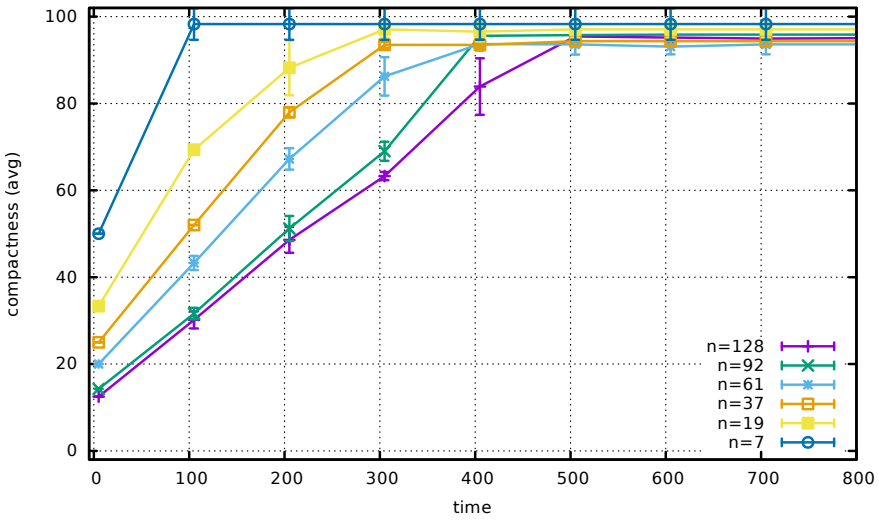


Fig. 4: Average compactness along time for different number of nodes.

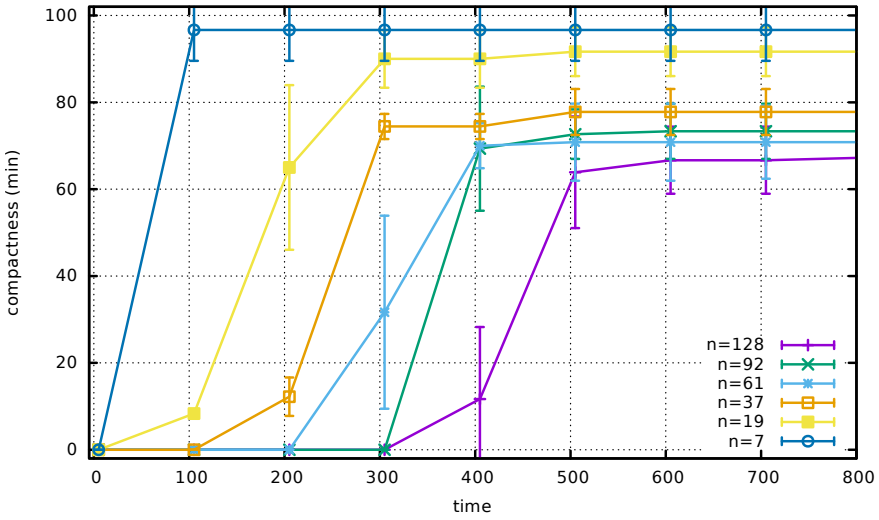


Fig. 5: Minimum compactness along time for different number of nodes.

is above to 95% for each number of sensors after 800s which shows that the effect of the random position selection and Alg. 3. We can also see from Figure 5 that the minimum compactness is 0% until the first sensor reaches the highest possible gradient depending on the number of sensors.

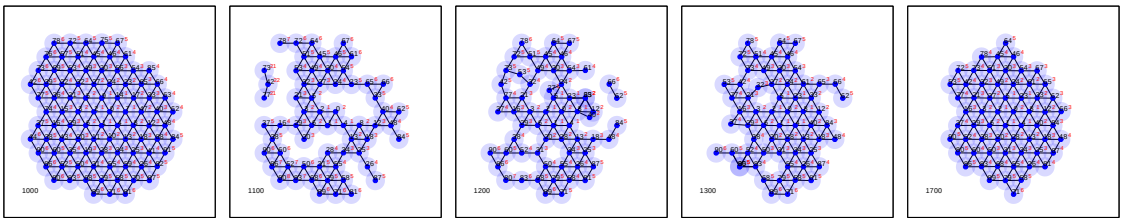
It is worth noting that for 7 sensor in the network, the compactness is 100%. This is due t the fact

that sensors are not synchronized. Each sensor starts to move at different time and thus the movement direction run in Alg.1:7-17 allows this perfect compactness.

6.5 Self-healing

In this section, we provide the simulation results that show the self-healing ability and connectivity restoration. We simulate a set of scenarios where there are 30, 50, 70 and 90% of sensors that disappear at time 1000s.

Figure 6 shows the network evolution when 30% of nodes disappear starting at 1000s. This figure shows the results of the self-healing mechanisms based only on local view as described in the algorithm and especially in Alg. 3.



(a) 1000s (b) 1100s (c) 1200s (d) 1300s (e) 1700s

Fig. 6: Deployment snapshots with 30% of node disappearing at 1000s and the self-healing mechanisms.

We first show the ability of our algorithm to restore network connectivity. To evaluate the connectivity of the graph, we use the reachability metrics as described by Razafindralambo and Simplot-Ryl (2011). The reachability of a network composed of N nodes is: $\rho = \frac{nb.\ of\ connected\ pairs}{\binom{N}{2}}$. Note here that when $\rho = 1$ the network is connected and when $\rho = 0$ all the nodes are isolated. We compute reachability based on the neighborhood table of each sensor which is updated upon reception of $Hello$ messages. It is important to notice here that some messages can be lost due to wireless collisions or physical layer imperfections.

Figure 7 shows the evolution of reachability depending on time for a network composed of 92 sensors at the beginning of the simulation. This figure shows that the higher the number of disappearing nodes, the lower the reachability drops. This figure also shows that reachability is increased faster when the percentage of disappearing nodes is lower. We can also see from this figure that reachability can decrease. This is mainly due to the fact that a node may move and break connectivity based on Alg.3 since sensors use a local approach to evaluate global connectivity.

Figure 8 shows the evolution of the average compactness depending on time for different percentage of node failures. This figure shows that the value of compactness is not necessarily negatively nor positively influenced by the percentage of disappearing sensors. Since the disappearing sensors are chosen randomly, they may affect the compactness in different ways. We can notice from the simulation that in any cases, the compactness is restored to a high value, close to the one before the failure.

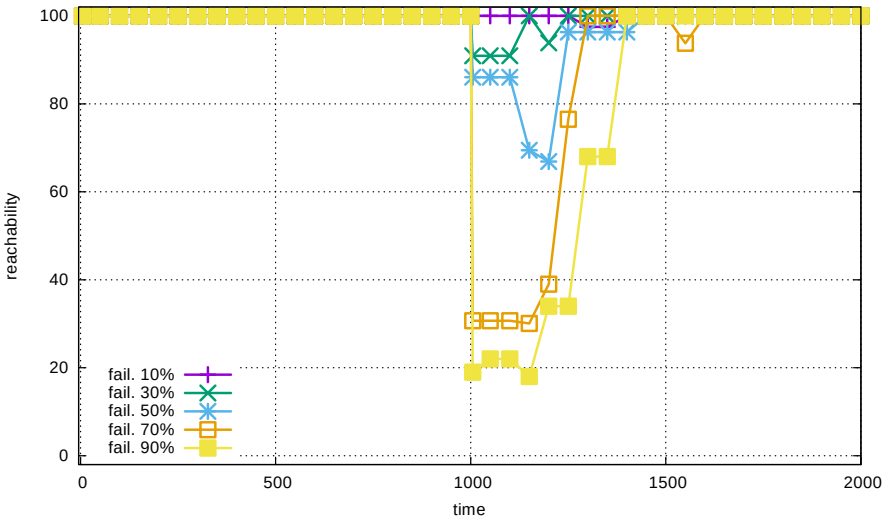


Fig. 7: Deployment reachability during the self-healing process.

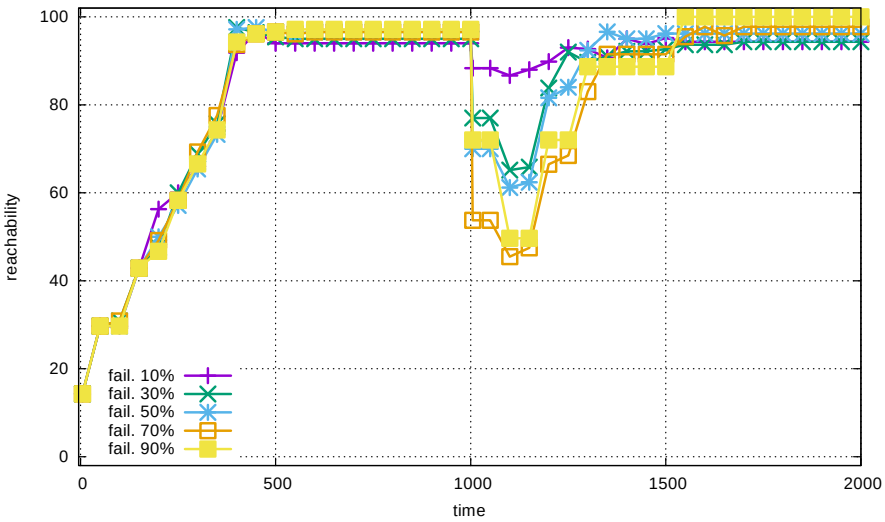
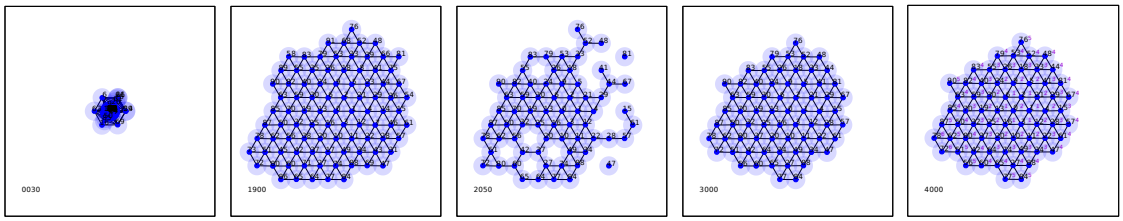


Fig. 8: Deployment compactness during the self-healing process.

6.6 Comparisons

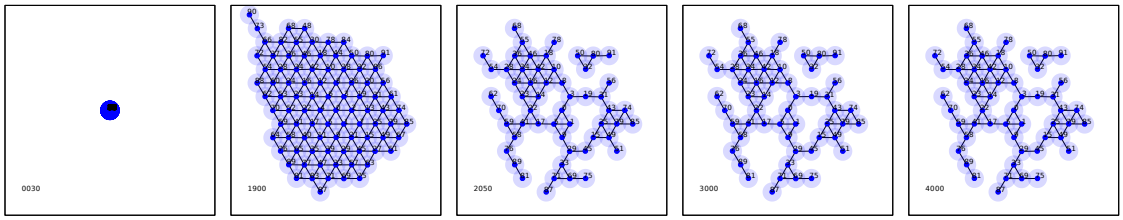
In this section, we provide a comparison with the closest work from the literature. We have implemented in the simulator the work of Bartolini et al. (2010) called hereafter BAR which shows a grid coverage process and the work described by Razafindralambo and Simplot-Ryl (2011) called hereafter RSR that uses virtual forces. In both implementations we have set the frequency of message sending to be equal as the one of our algorithm to have a fair comparison. We have modified the algorithm described by Bartolini et al. (2010) to prevent sensor from different locations to snap slave sensors to the same spot.

Figures 9, 10 and 11 present several snapshots of each deployment. In these figures, 30% of the nodes die at 2000 s. We can see from this figure (Fig. 9) as previously stated that EMR algorithm is robust an can restore connectivity after the nodes' crash. BAR algorithm does not implement such mechanisms and therefore after the node crash, the network remain as it is (Fig. 10). It is important to notice that the BAR algorithm can restore connectivity if there exists a non-snapped node available to restore it. This would be, for example, the case in a confined space where each spot is occupied by more than one sensor. The RSR algorithm relies on virtual forces and has two main drawbacks. First, coverage holes can be created due to repulsive forces (See Fig. 11 at 1900s). Second, if two nodes are within communication range but some communication messages are lost, then a network can disconnect. Disappearing connections can be noticed in Fig. 11 at 3000s and 4000s.



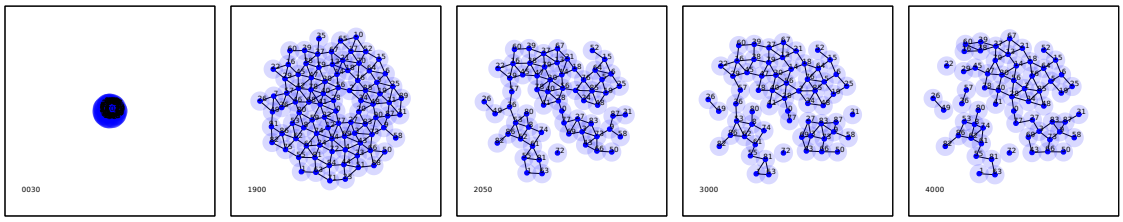
(a) 0030s (b) 1900s (c) 2050s (d) 3000s (e) 4000s

Fig. 9: EMR algorithm presented in this paper. Deployment snapshots with 30% of node disappearing at 2000s and the self-healing mechanisms.



(a) 0030s (b) 1900s (c) 2050s (d) 3000s (e) 4000s

Fig. 10: BAR algorithm developed by Bartolini et al. (2010). Deployment snapshots with 30% of node disappearing at 2000s and the self-healing mechanisms.



(a) 0030s (b) 1900s (c) 2050s (d) 3000s (e) 4000s

Fig. 11: RSR algorithm developed by Razafindralambo and Simplot-Ryl (2011). Deployment snapshots with 30% of node disappearing at 2000s and the self-healing mechanisms.

Coverage

In this subsection, we compare the coverage performance of the three algorithms. We can see from Fig. 12 that the three algorithms reach 100% coverage. However, we can see that RSR algorithm fluctuates around 100%. This is due to the fact that when using virtual forces, sensors are attracted by each other and thus the coverage percentage may not be stable at 100%. We can see also from this figure that BAR algorithm reaches 100% coverage and is stable. However, compared to EMR algorithm, both RSR and BAR reach 100% coverage slower. For BAR, this slow behaviour is due to the fact that unsnapped sensors cannot take the decision to move. Only installed sensors can give un-installed sensors their new position, and these mechanisms increase latency. For RSR this slow behaviour is due to attractive and repulsive force used in the algorithm. Since virtual forces uses also attractive forces, these forces slow down the deployment.

Compactness

In this subsection, we compare the compactness of the algorithms before and after the self-healing process. We can see from Fig 13 that algorithms BAR and RSR do not restore compactness after the failure at 2000s. This is because the two algorithms do not have self-healing process. We can notice that before the failure occurs, the three algorithms have the same compactness around $\sim 90\%$. The results in Fig. 13 represent a single simulation run.

Reachability

In this subsection, we present the reachability of each algorithm. We can see from the single run simulation presented in Fig. 14 that algorithms RSR and BAR are not able to restore reachability after the failure at 2000s. This is due to the fact that these algorithms do not implement self-healing process and that there are not enough sensors to restore the connectivity. As stated earlier, the reachability of RSR fluctuates since the connectivity can be broken due to message loss.

Table 1 summarizes the comparison between the three algorithms. In this table, we added a column regarding the use of these algorithms in confined space. In their papers, Razafindralambo and Simplot-Ryl (2011) and Bartolini et al. (2010) have shown that in a confined space, their algorithm can evenly distribute sensors over the field. This is not the case for EMR algorithm. Moreover, we assume that in case of a confined space, algorithms RSR and BAR can restore reachability, coverage and compactness. This evaluation and comparison in confined space is left for future work. The use of our algorithm in confined space is discussed in the next section.

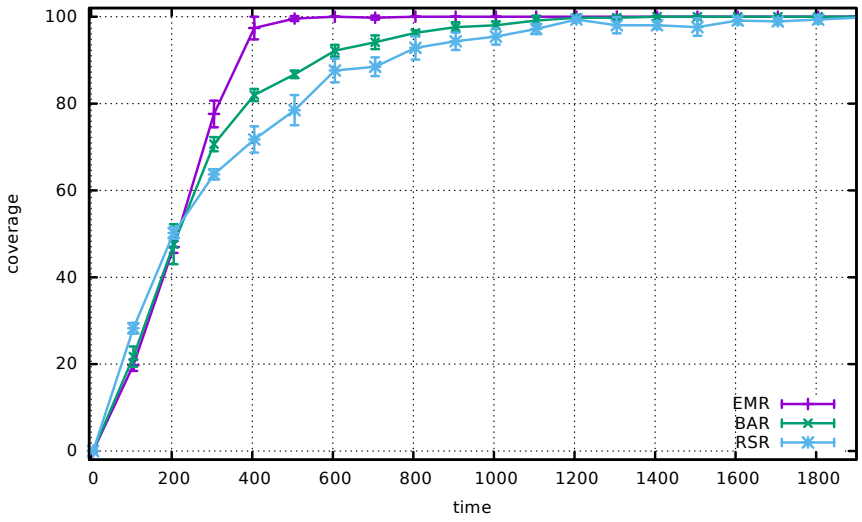


Fig. 12: Comparison of coverage efficiency for the three algorithms before the self-healing process (in average).

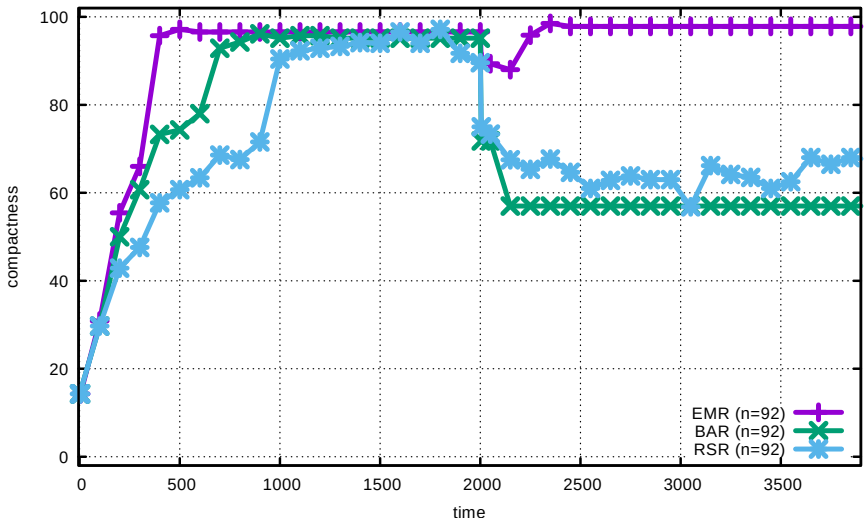


Fig. 13: Comparison of compactness before and after the self-healing process (a single run).

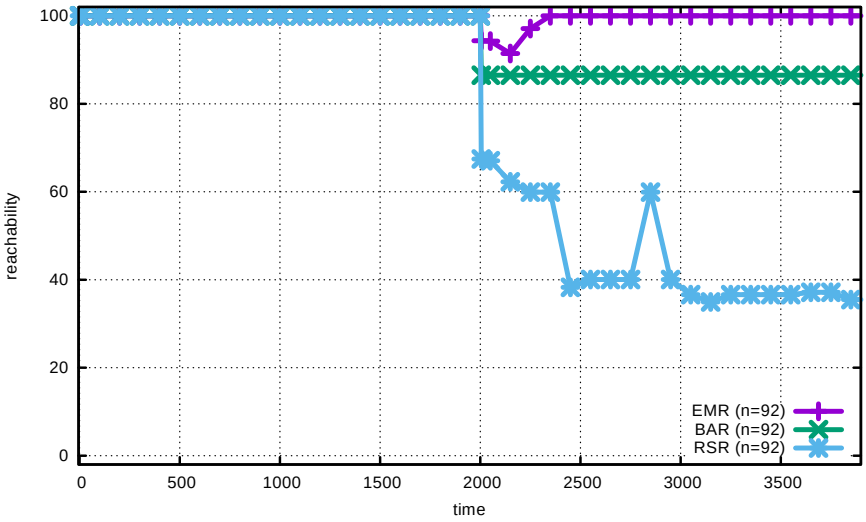


Fig. 14: Comparison of reachability before and after the self-healing process (a single run).

Tab. 1: Comparison of the algorithms – a summary.

Deployment algorithm	Coverage		Compactness		Reachability		Self-healing	Tolerance to failure	Confined space (distribution)
	before failure	after failure	before failure	after failure	before failure	after failure			
EMR	optimal	optimal	~90%	~90%	1	1	automatic	yes	uneven
BAR	optimal	optimal	~90%	~60%	1	not restored	not managed	no	even
RSR	holes	holes	~90%	~60%	1	not restored	automatic	no	even

7 Discussion and extensions of the work

Obstacles or Points of Interest. In our simulations, we consider a field without obstacles or specific points of interest. The obstacle assumption can be removed by considering that some spots around a given sensor are not accessible. In order to provide a repulsive effects of an obstacle, we can consider that the occupancy of a specific spot is infinite, which will force sensors to get into that spot. If, on the contrary, we want some specific spots to attract as many sensors as possible (or a given number of sensors), it is possible to modify its occupancy by avoiding sensors to send messages (or allow only a limited number of sensor to send messages) when they are above theses specific spots.

Multiple origins of sensors. In case of multiple sensor origins with or without the same orientation, a merging process can be triggered. Some techniques such as the one described by Bartolini et al. (2010) can be implemented. Bartolini et al. (2010) suggest to add a common timestamp in the message sent by the sensors to identify their origin and the beginning of the deployment. When two sensors with different timestamps are within communication range the sensor with the newest timestamp changes its grid placement and orientation according to the sensor with the oldest timestamps. In our case, instead of timestamps, gradient number can also be used.

Sensors locations. In our simulation, we added a random error of [-2:2] m to each sensor position to mimic the GPS position approximation. The simulation shows that this randomness has no effect on the algorithm.

Initial sensor position. In our simulation, we consider that the sensors have the same initial location (with 2m of position errors). However, this assumption is not practically possible to implement especially as the number of sensors increases. To overcome this issue, we have implemented a virtual localization in each sensor. This virtual localization is the position in which a sensor should be. Sensors try to minimize the difference between their virtual localization and their real localization by moving towards their expected position whenever possible. This mechanism may provoke communication errors since the difference in real and virtual localisation can cause the loss of connectivity. However, EMR algorithm does support message loss.

Confined space. In the current version of the algorithm, we assume that the field is large enough so that it cannot be fully covered by the sensors. We do not consider the fact in which the number of sensor can cover the whole field. In this case, the sensors will stay at the boundary of the field. However, it is easy for a sensor to identify itself as a sensor with a full overlapping coverage since for this sensor, all the spots in the ”regressive positions” set are occupied and all the spots in the ”progressive positions” set are or not accessible. Evenly distributing sensors over a confined field is left for future work.

8 Conclusions and future work

In this work, we presented a distributed EMR algorithm for the robust deployment of mobile wireless sensors. The algorithm asynchronously runs on all the sensors in the network. The sensors only need locally available information that is periodically emitted via $Hello$ messages, in order to achieve the successful deployment and robustness of the network. By step-wise movements, the algorithm is proven to drive a group of mobile sensors to achieve maximized coverage while maintaining global network connectivity. Furthermore, the algorithm is also proven to terminate without oscillations. A complementary simulation campaign shows different aspects of the deployment algorithm regarding the covered area, robustness and compactness of the deployment.

Future work in this field will be focused on designing the deployment algorithm that achieves both area coverage and energy efficient sensor deployment. Furthermore, we will consider different scenarios with energy providers present in the deployment field, where the goal will be to design an algorithm that will be able to maximize the covered area while relying on energy providers to sustain the sensors and thus prolong the network lifetime.

Acknowledgements

This work was partially supported by a grant from CPER Nord-Pas-de-Calais/FEDER Campus Intelligence Ambiante and by the Region Picardie, France, through the European Regional Development Fund (ERDF).

References

N. Bartolini, T. Calamoneri, E. G. Fusco, A. Massini, and S. Silvestri. Push & pull: autonomous deployment of mobile sensors for a complete coverage. *Wirel. Netw.*, 16(3):607–625, Apr. 2010. ISSN 1022-0038. doi: 10.1007/s11276-008-0157-7.

C.-F. Cheng and C.-W. Huang. An energy-balanced and timely self-relocation algorithm for grid-based mobile wsns. *Sensors Journal, IEEE*, 15(8):4184–4193, Aug 2015. ISSN 1530-437X. doi: 10.1109/JSEN.2015.2413367.

M. Erdelj, T. Razafindralambo, and D. Simplot-Ryl. Covering points of interest with mobile sensors. *Parallel and Distributed Systems, IEEE Transactions on*, 24(1):32–43, 2013. ISSN 1045-9219. doi: 10.1109/TPDS.2012.46.

E. Ferrante, A. E. Turgut, M. Dorigo, and C. Huepe. Elasticity-based mechanism for the collective motion of self-propelled particles with springlike interactions: A model system for natural and artificial swarms. *Phys. Rev. Lett.*, 111:268302, Dec 2013. doi: 10.1103/PhysRevLett.111.268302. URL `http://link.aps.org/doi/10.1103/PhysRevLett.111.268302`.

V. Gazi and K. M. Passino. Stability analysis of social foraging swarms. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(1):539–557, Feb 2004. ISSN 1083-4419. doi: 10.1109/TSMCB.2003.817077.

A. Howard, M. J. Matarić, and G. S. Sukhatme. An incremental self-deployment algorithm for mobile sensor networks. *Auton. Robots*, 13(2):113–126, 2002. ISSN 0929-5593.

G. Lee and N. Chong. A geometric approach to deploying robot swarms. *Annals of Mathematics and Artificial Intelligence*, 52(2-4):257–280, 2008. ISSN 1012-2443. doi: 10.1007/s10472-009-9125-x.

J. Li, B. Zhang, and L. Cui. A restricted delaunay triangulation graph based algorithm for self-deployment in mobile sensor networks. In *IEEE 6th International Conference on Mobile Adhoc and Sensor Systems (MASS '09)*, pages 3155–3162, 2010.

X. Li, H. Frey, N. Santoro, and I. Stojmenovic. Strictly localized sensor self-deployment for optimal focused coverage. *Mobile Computing, IEEE Transactions on*, 10(11):1520–1533, Nov 2011. ISSN 1536-1233. doi: 10.1109/TMC.2010.261.

T.-Y. Lin, H. Santoso, and K.-R. Wu. Global sensor deployment and local coverage-aware recovery schemes for smart environments. *Mobile Computing, IEEE Transactions on*, 14(7):1382–1396, July 2015. ISSN 1536-1233. doi: 10.1109/TMC.2014.2353613.

X. Liu. A deployment strategy for multiple types of requirements in wireless sensor networks. *Cybernetics, IEEE Transactions on*, 45(10):2364–2376, Oct 2015. ISSN 2168-2267. doi: 10.1109/TCYB.2015.2443062.

S. Mahfoudh, I. Khoufi, P. Minet, and A. Laouiti. Gdvfa: A distributed algorithm based on grid and virtual forces for the redeployment of wsns. In *Wireless Communications and Networking Conference (WCNC), 2014 IEEE*, pages 3040–3045, April 2014. doi: 10.1109/WCNC.2014.6952969.

P. Park, S.-G. Min, and Y.-H. Han. A grid-based self-deployment schemes in mobile sensor networks. In *Ubiquitous Information Technologies and Applications (CUTE), 2010 Proceedings of the 5th International Conference on*, pages 1–5, 2010. doi: 10.1109/ICUT.2010.5677834.

W. Y. Poe and J. B. Schmitt. Node deployment in large wireless sensor networks: coverage, energy consumption, and worst-case delay. In *Asian Internet Engineering Conference*, AINTEC '09, pages 77–84, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-614-4. doi: 10.1145/1711113.1711127.

K. S. Prabh, C. Deshmukh, and S. Sachan. A distributed algorithm for hexagonal topology formation in wireless sensor networks. In *Proceedings of the 14th IEEE international conference on Emerging technologies & factory automation*, ETFA'09, pages 675–681, Piscataway, NJ, USA, 2009. IEEE Press. ISBN 978-1-4244-2727-7.

T. Razafindralambo and D. Simplot-Ryl. Connectivity preservation and coverage schemes for wireless sensor networks. *Automatic Control, IEEE Transactions on*, 56(10):2418–2428, 2011. ISSN 0018-9286. doi: 10.1109/TAC.2011.2163885.

W. Spears, D. Spears, J. Hamann, and R. Heil. Distributed, physics-based control of swarms of vehicles. *Autonomous Robots*, 17(2-3):137–162, 2004. ISSN 0929-5593. doi: 10.1023/B:AURO.0000033970.96785.f2.

G. Tan, S. A. Jarvis, and A.-M. Kermarrec. Connectivity-guaranteed and obstacle-adaptive deployment schemes for mobile sensor networks. *IEEE Transactions on Mobile Computing*, 8:836–848, 2009. ISSN 1536-1233.

G. Wang, G. Cao, and T. L. Porta. Movement-assisted sensor deployment. *Mobile Computing, IEEE Transactions on*, 5(6):640–652, 2006. ISSN 1536-1233.

Y.-C. Wang and C.-C. Hu. Efficient placement and dispatch of sensors in a wireless sensor network. *IEEE Transactions on Mobile Computing*, 7(2):262–274, 2008. ISSN 1536-1233.

J. Xiao, S. Han, Y. Zhang, and G. Xu. Hexagonal grid-based sensor deployment algorithm. In *Control and Decision Conference (CCDC), 2010 Chinese*, pages 4342–4346, 2010. doi: 10.1109/CCDC.2010.5498352.

K. Xu, G. Takahara, and H. Hassanein. On the robustness of grid-based deployment in wireless sensor networks. In *Proceedings of the 2006 international conference on Wireless communications and mobile computing*, IWCMC '06, pages 1183–1188, New York, NY, USA, 2006. ACM. ISBN 1-59593-306-9. doi: 10.1145/1143549.1143786.

Z. Yun, X. Bai, D. Xuan, T. H. Lai, and W. Jia. Optimal deployment patterns for full coverage and k-connectivity ($k \leq 6$) wireless sensor networks. *IEEE/ACM Trans. Netw.*, 18(3):934–947, June 2010. ISSN 1063-6692. doi: 10.1109/TNET.2010.2040191.