Autonomous or Assisted Deployment by Mobile Robots of Wireless Sensor Networks: Coverage and Connectivity Issues

Ines Khoufi

UNIVERSITY PARIS 6 PIERRE ET MARIE CURIE

PHD THESIS

To obtain the title of

DOCTEUR DE L'UNIVERSITE PIERRE ET MARIE CURIE

Speciality: Computer Science

Autonomous or Assisted Deployment by Mobile Robots of Wireless Sensor Networks: Coverage and Connectivity Issues

Defended by

Ines Khoufi

On September 30, 2015

Thesis Advisor: Mrs. **Pascale Minet**
Thesis Co-Advisor: Mr. **Anis Laouiti**

Prepared at Inria Paris-Rocquencourt Research center
Eva Team

Jury:

Mr.	Nadjib Achir	Reviewer	Associate Professor HDR, University of Paris 13
Mr.	André-Luc Beylot	Reviewer	Professor, ENSEEIHT, France
Mrs.	Pascale Minet	Advisor	Research Scientist HDR, Inria Rocquencourt, France
Mr.	Anis Laouiti	Co-Advisor	Associate Professor HDR, Telecom SudParis, France
Mr.	Marcelo Dias de Amorim	Examiner	Research Director , Pierre & Marie Curie University
Mr.	Azzedine Boukerche	Examiner	Professor, University of Ottawa, Canada
Mrs.	Leila Saidane	Examiner	Professor, ENSI, Manouba University, Tunisia
Mr.	Tuan Dang	Examiner	Doctor, Engineer Research Expert, EDF, France

Acknowledgements

I would like to begin by expressing my gratitude to Mr Nadjib Achir and Mr André-Luc Beylot for kindly accepting to review my thesis. Also, my thanks go to the members of the jury for the interest they showed in my research: Mr Marcelo Dias de Amorim, Mr Azzedine Boukerche, Mrs Leila Saidane and Mr Tuan Dang.

I would never have been able to finish my dissertation without the guidance, help and support of colleagues, friends and family. I thank all those people who made this thesis possible and an unforgettable experience for me.

Firstly, I would like to express my deepest sense of gratitude to my advisor Mrs Pascale Minet for her continuous support of my PhD study, for her knowledge and immense encouragement. Her guidance helped me throughout the research and writing of this thesis. I cannot have imagine having a better advisor and mentor for my PhD study.

I would like to express my special appreciation to my co-advisor Mr Anis Laouiti, for encouraging my research and for helping me to further my career as a research scientist.

I have been honored to work with the members of Eva Team, formerly 'HIPERCOM'. Thank you: Erwan Livolant, Selma Boumerdassi, Dana Marinca, Cedric Adjih, Paul Muhlethaler and Thomas Watteyne for the pleasant working atmosphere you created. I also thank Christine Anocq for her valuable help, support and friendly company. She always made my administrative tasks so much easier.

I also am happy to have collaborated with Saoucene Mahfoudh, Erwan Livolant, Mohamed Haddad, Mohamed Amine Koulali and Nadia Boufares.

I want to address my kind regards to the many friends that I have made at Inria, for their wonderful company and sincere friendship: thank you Nesrine, Hana, Ichrak, Ines, Saoucene, Ridha, Younes, Alaa and Haykel. I am grateful to all of you, for your support as well as the fun times I have shared with you.

I am indebted to Richard James, our English teacher at Inria. His interesting lessons improved my English 'beyond measure' (in his opinion!). I would also thank him, for the considerable time and effort he spent correcting my papers and this dissertation.

I must take this opportunity to express my heartfelt gratitude to my family. Words cannot express how grateful I am for your support and kind thoughts. Their prayers for me have sustained me over the years.

Special thanks must go to my husband and best friend Bilel. He was always my support in those moments when there was no one to answer my queries, and there to cheer me up and stand by me through both the good times and the bad.

This research has been supported by The Cluster CONNEXION (digital command COntrol for Nuclear EXport and renovatION) project. I gratefully acknowledge this funding.

To my Family

Abstract

Wireless sensor networks are deployed to monitor physical phenomena. The accuracy of the information collected depends on the position of sensor nodes. These positions must meet the application requirements in terms of coverage and connectivity, which therefore requires the use of deployment algorithms.

This thesis focuses on the deployment of wireless sensor nodes: firstly when the nodes are autonomous, and secondly when they are static and the deployment is assisted by mobile robots. In both cases, this deployment must not only meet the application's coverage and connectivity requirements, but must also minimize the number of sensors needed while satisfying various constraints (e.g. obstacles, energy, fault-tolerant connectivity).

We propose several autonomous deployment algorithms, based on the virtual forces strategy to monitor 2D and 3D areas. Since the virtual forces strategy suffers from the node oscillations problem, we have designed the ADVFA algorithm that adapts the distance between neighboring sensor nodes to the number of connected nodes. ADVFA avoids useless moves in order to reduce node oscillations. We also propose the GDVFA algorithm to cope with the node oscillations problem. GDVFA is a hybrid algorithm that combines the virtual forces strategy with the grid strategy to stop node oscillations. In addition, since the monitoring area may be unknown and contain obstacles, we propose the OA-DVFA algorithm. For a 3D area, we have designed the 3D-DVFA algorithm, based on a 3D version of the virtual forces algorithm.

Autonomous deployment may be expensive when the number of mobile sensor nodes is very high. In this case, an assisted deployment may be necessary: the nodes' positions being pre-computed and given to mobile robots that place a static sensor at each position. To compute the optimized number of nodes needed to fully cover a 2D area containing obstacles, we propose the OAD-Area algorithm. We also propose OAD-PoI, to optimize the relay node positions and ensure a fault-tolerant connectivity between each Point of Interest (PoI) and the sink. Once the sensor node positions have been computed, they can be given to mobile robots to carry out the actual deployment. We adopt two approaches to optimize the deployment duration. The first one is based on game theory to optimize the length of the paths of two robots (TRDS), and the second is based on a multi-objective optimization, for multiple robots (MRDS). The objectives to be met are: optimizing the duration of the longest tour, balancing the durations of the robot tours and minimizing the number of robots used, while bypassing obstacles.

Contents

Chapter 1

General Introduction

1.1 Context and motivation

Wireless Sensor Networks (WSNs) constitute an emerging technology that has caught the interest of many researchers over the last few years. An increasing number of applications are supported by wireless sensor networks, and cover areas as diverse as structural health monitoring, smart metering, industrial process monitoring, precision farming, smart cities, control of traffic lights, smart homes, etc.

A WSN is a wireless network consisting of a set of static or mobile sensor nodes scattered over an area of interest to monitor physical or environmental conditions. These sensor nodes may be of different types such as seismic, thermal, infrared, radar etc, and they are able to monitor a wide variety of ambient conditions that include temperature, humidity, vehicular movement, lighting conditions, pressure, soil makeup, noise levels, etc.

Node deployment is a fundamental issue in WSNs. A proper node deployment scheme can significantly improve the performance of the data gathering process. Furthermore, it can extend the lifetime of WSNs by minimizing energy consumption. Depending on the size of the entity (area, barrier or point of interest) monitored, a multi-hop network may need to be deployed to enable the monitoring of this area as well as the delivery of the collected data. To meet the application requirements, the deployment of sensor nodes must ensure coverage and connectivity properties. Roughly speaking, coverage refers to the ability to detect events occurring in the entity monitored, whereas connectivity refers to the ability to report this event to a special wireless node, called the sink, in charge of processing the data gathered from the sensor nodes.

In many applications, sensors are deployed randomly in a specific area. This random deployment results in some regions being highly covered while others have just a few scattered sensors. As a result, many regions of the deployed area cannot be monitored. Such a deployment may be suitable for some applications, such as forest fire monitoring, which tolerates partial coverage in wet seasons. However, many other critical applications require full coverage of the area monitored such as monitoring temporary worksites or monitoring nuclear plants. Consequently, a redeployment algorithm is necessary to place sensors in appropriate positions to ensure full area coverage in order to detect each event occurring in this area.

Sensor deployments differ in their goals, their constraints and their implementation (e.g., centralized versus distributed). For cost reasons, most deployments aim at minimizing the number of sensor nodes deployed to achieve the application requirements. This goal is the same as minimizing the deployment cost, which mainly depends on the number

of sensor nodes deployed. Another goal that is frequently encountered in crisis situations (e.g., after a disaster) is that a wireless sensor network must be deployed as quickly as possible in order to, on the one hand, help rescuers to save victims, and on the other hand, assist in damage assessment. In such cases, the goal is to minimize the time needed to deploy an operational wireless network. Since sensors are battery equipped, the minimum time spent in the deployment will help to save energy and prolong network lifetime. This goal is also targeted in hostile environments (e.g radiation), where the exposure duration must be reduced.

More precisely, in this work we focus on deployment algorithms in WSNs:

- Goal: to ensure full coverage (of an area or Points of Interest, PoI) and maintain network connectivity.
- Under the following constraints:
  - Sensor nodes may be mobile and autonomous or they may be static. When sensor nodes are autonomous, the deployment is termed self deployment. In self deployment, sensor nodes cooperate together to compute their final positions and move to them. However, when sensor nodes are static, the deployment is computed by a central entity. Then, a human or one or multiple mobile robot(s) should place sensor nodes in their final positions.
  - Minimum number of nodes to minimize the deployment cost.
  - Connectivity: fault-tolerance. Network robustness can be ensured if at least two node-disjoint paths exist between each sensor node and the sink. Additional relay nodes may be needed between sensor nodes and the sink to enhance network robustness.
  - Presence of obstacles. In many studies, it is assumed that the entity monitored does not contain obstacles. However, this assumption is not realistic. Then, deployment algorithms should be able to cope with obstacles since obstacles prevent the physical presence of sensor nodes and may prevent the connectivity between them. Obstacles may be transparent or opaque, and their positions and shapes may be known in advance or unknown.

1.2 Main contributions

Figure 1.1 depicts the positioning of the main contributions of this PhD thesis. The figure illustrates the problem tackled, the approaches adopted to solve it and the different solutions to meet our goal. Our contributions are presented in red rectangles.

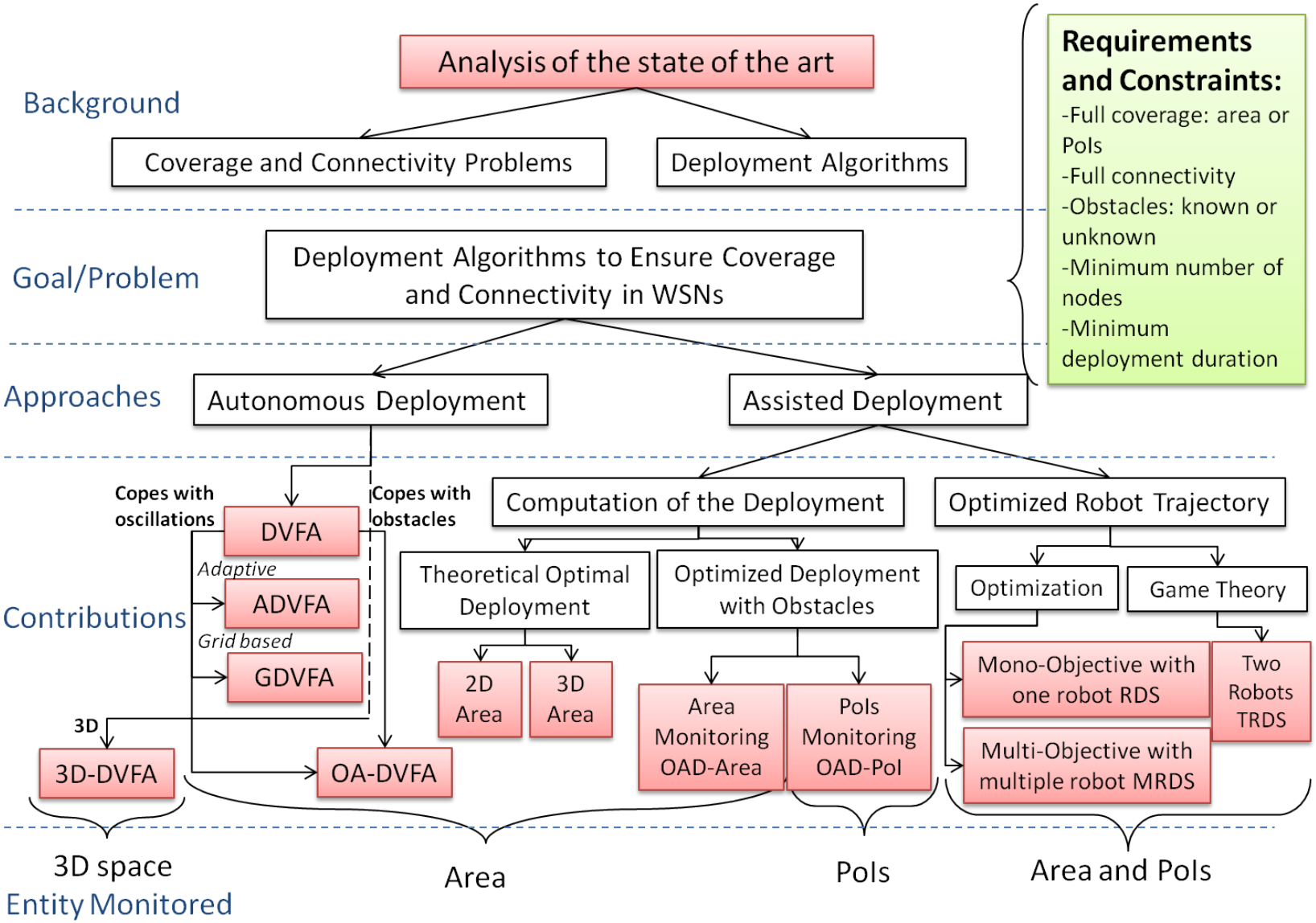


Figure 1.1: Main contributions.

Since deployment algorithms for wireless sensor networks, have been well studied, we start our study by analyzing the state of the art of coverage and connectivity problems and deployment algorithms proposed in the literature. We distinguish between different targets to monitor (i.e. area, barrier and point of interest), coverage problems (i.e. full or partial) and connectivity problems (i.e. permanent or temporary). For each problem we give a corresponding use case. Then, we propose a detailed analysis and classification of existing deployment algorithms. Following on from this study of the state of the art, we put forward some recommendations for designing a deployment algorithm. We then address two coverage problems in WSNs: the full area coverage problem and the Point of Interest (PoI) coverage problem with permanent connectivity. We propose two approaches to ensure full coverage and maintain network connectivity.

- **Autonomous deployment approach**, where sensor nodes are autonomous. To ensure full area coverage, we propose deployment algorithms based on the Virtual Forces strategy. We adopt the Virtual Forces strategy in order to take advantage of the spreading out of nodes over the whole area due to attractive and repulsive forces while maintaining network connectivity. First, we improve the Distributed Virtual Forces Algorithm (DVFA) to cope with node oscillations. We propose the Adaptative Distributed Virtual Forces Algorithm (ADVFA) to reduce node oscillations caused by the virtual forces and the border effects. Then, we propose a hybrid solution, Grid Distributed Virtual Forces Algorithm (GDVFA), based on the Grid strategy and the Virtual Forces strategy to eliminate node oscillations and save energy consumption. Since obstacles always exist in the real environment, we propose the Obstacle Avoidance Distributed Virtual Forces Algorithm (OA-DVFA) to avoid known and unknown obstacles. To deploy sensor nodes in a 3D area we propose the 3D Distributed Virtual Forces Algorithm (3D-DVFA).

- **Assisted deployment**, where sensor nodes are static and need to be deployed by a human or (multiple) mobile robot(s). We propose an optimized deployment that ensures full coverage of an irregular-shaped area containing obstacles while minimizing the total number of sensor nodes deployed. To ensure PoI coverage and connectivity, we propose an optimal deployment based algorithm that provides k-connectivity where k node-disjoint paths from each PoI to sink are ensured. This algorithm provides a robust and fault-tolerant network.

  In both cases of assisted deployment (i.e. area or PoI coverage), we define an optimization problem called the Robot Deploying Sensor nodes problem (RDS) to optimize the delay needed by the robot to place sensor nodes in their positions. Depending on the number of robots available, we propose various approaches (e.g. game theory, multi-objective optimization problem (MRDS) with genetic algorithms) to optimize robot trajectories and minimize the deployment duration.

Finally, we describe use cases in an industrial context (e.g. in nuclear power plants) where such algorithms can be applied, and we discuss how to improve their accuracy, taking into account real measurements made in the wireless sensor network.

1.3 Manuscript organization

This dissertation is organized as follows:

- **Chapter 1 introduces the context and the motivations of our work and describes our main contributions.**

Part I: State of the art

To properly understand coverage and connectivity issues, the various constraints impacting the deployment, as well as the different types of deployment algorithms existing in the literature, we provide a comprehensive study of the state of the art in this part.

- **Chapter 2 presents a state of the art on coverage and connectivity problems in WSNs.**
- **Chapter 3 analyzes deployment algorithms in WSNs.**

Part II: Models and theoretical computation for an optimized deployment in 2D and 3D

An optimal deployment is a deployment that ensures full coverage and maintain network connectivity of the entity monitored while using the optimal number of sensor nodes. To obtain such a deployment, some constrains on sensing and communication range should be satisfied and sensor node positions should respect an appropriate pattern. In this part, we provide theoretical models and computations of the 2D and 3D optimal deployments.

- **Chapter 4 presents the different models of** sensing and communication ranges, area to be monitored and obstacles in both 2-dimension (2D) and 3-dimension (3D) deployments.
- **Chapter 5 proposes a theoretical computation** of on the one hand the optimal deployment in a 2-dimension area and on the other hand of the optimized deployment in a 3-dimension area.

Part III: Autonomous deployment

When sensor deployment is autonomous, all sensor nodes are mobile, able to communicate and cooperate together to determine their final position in the area considered. The virtual forces strategy is adopted in the autonomous deployment. Due to its principles, sensor nodes are able to spread in the whole 2D or 3D area and to be uniformly deployed. However, the virtual forces strategy suffers from node oscillations, where sensor nodes still oscillate even if full coverage is ensured. Autonomous deployment based on virtual forces is studied in this part, where we propose algorithms to cope with node oscillations and the presence of known or unknown obstacles. We also extend the deployment algorithm based on virtual forces to operate in 3D space.

- **Chapter 6 presents self deployment algorithms** based on the virtual forces strategy: DVFA, ADVFA, GDVFA, OA-DVFA and 3D-DVFA. Three problems are studied in this chapter: the node oscillations problem, the presence of obstacles (known or unknown) and 3D deployment.

Part IV: Assisted deployment

When the deployment is assisted, sensor node positions should be computed by a central entity and then given to mobile robots in charge of placing sensor nodes at their positions. In this part, we first propose a solution to compute an optimized deployment that ensures area coverage and network connectivity. Then, we optimize the trajectory of robots deploying sensor nodes in an area containing obstacles.

- **Chapter 7 proposes two centralized algorithms for an optimized deployment in the presence of obstacles:** the first one, called OAD-Area, aims at ensuring full area coverage and connectivity, whereas the second algorithm, called OAD-PoI, aims at ensuring PoIs coverage and connectivity.

- **Chapter 8 describes two solutions to optimize the trajectory of the robot(s)** in charge of placing sensor nodes in their positions and minimize the deployment duration. The first solution, called TRDS, is based on game theory approach and the second one, called MRDS, is based on multi-objective optimization approach.

Part V: Discussion and conclusion

- **Chapter 9 shows how to extend our solutions when some constraints due to the real environment exist, and then concludes this dissertation and presents our perspectives.**

Part I

State of the art

Chapter 2

Coverage and connectivity issues in WSNs

2.1 Introduction

A Wireless Sensor Network (WSN) consists of a number of sensor nodes working together to monitor a given entity (e.g. area, barrier, point of interest). The main functionalities of a sensor node are: sensing the environment and reporting the data it gathers to a special node called the sink. Hence, the monitoring task depends on two major issues:

- Coverage of the entity to allow sensor nodes to detect events,
- Network connectivity to allow the events detected to be reported to the sink.

In this chapter we focus on different types of coverage and connectivity problems in WSNs. We start by giving some representative use cases matching different monitoring applications. Then, we detail the different coverage and connectivity problems in WSNs. After that we present the relationship between coverage and connectivity based on the values of the sensing range and communication range. Finally, we conclude.

2.2 Definition of coverage and connectivity problems in WSNs

In this section, we detail the different coverage and connectivity problems in wireless sensor network.

2.2.1 Coverage problems

An area is said to be covered if and only if each location in this area is within the sensing range of at least one active sensor node.

In our work, we distinguish three types of coverage problems : Area coverage, Point coverage and Barrier coverage as illustrated in Figure 2.1.

2.2.1.1 Area coverage

In the area coverage problem, the goal is to cover the whole area. Depending on the application requirements, full or partial coverage may be required. However, if the number of sensors is not sufficient, full coverage cannot be achieved and the goal becomes maximizing the coverage rate.

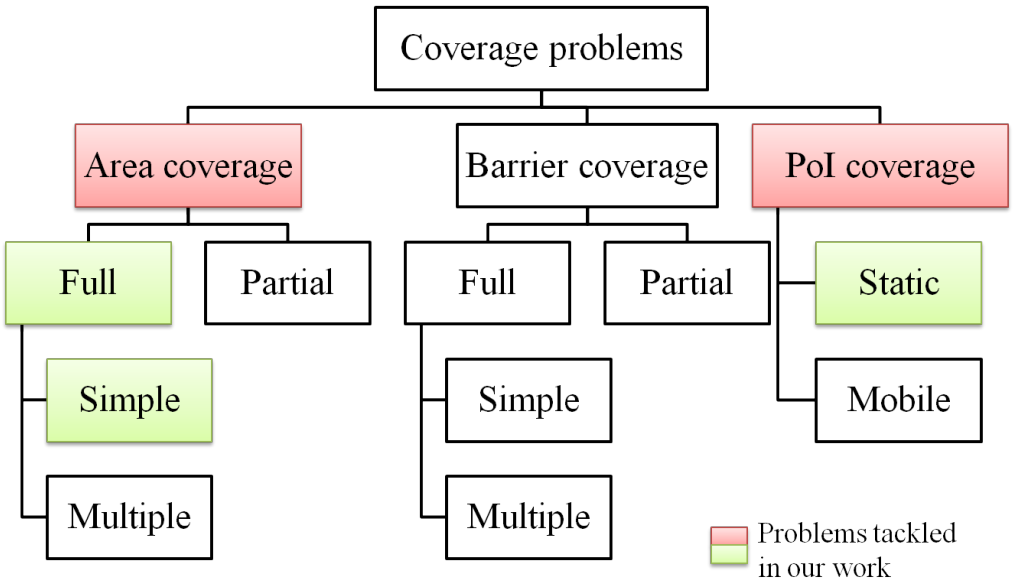


Figure 2.1: Coverage problems.

- Full coverage

Applications such as battlefield monitoring require full area coverage. In these applications, every location is covered by at least one sensor node (1-coverage) or by $k > 1$ sensor nodes (k-coverage). Deploying sensors over a large area while ensuring full coverage and network connectivity may be expensive. However, full coverage with connectivity provides the best surveillance quality. In the following we detail *one*-coverage defined as simple coverage and k-coverage defined as multiple coverage, depending on the degree of robustness required by the application. Figure 2.2 illustrates simple coverage and multiple coverage, the disk around the sensor node depicts its coverage area.

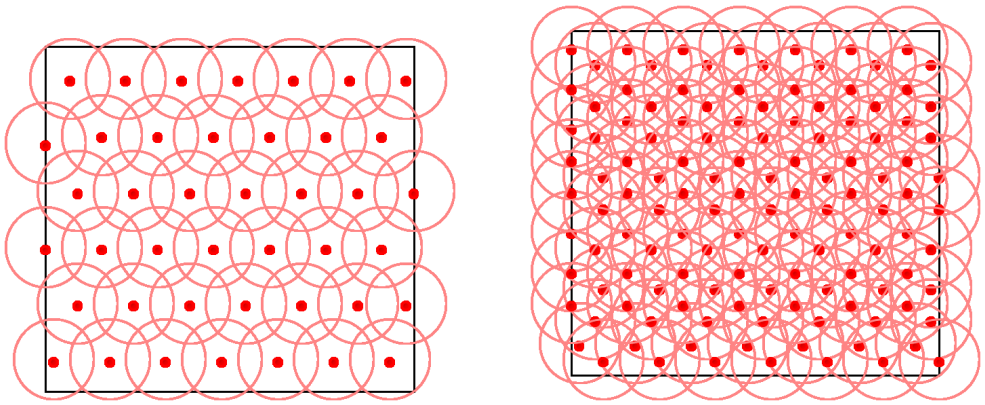
– Simple coverage

In WSNs, it is necessary to ensure full coverage of the area considered while deploying the minimum number of sensor nodes. This can be satisfied by covering every location in the field using at least one sensor node. The information detected in this location should be reported to the sink. Many studies aim to minimize the number of nodes deployed while ensuring coverage and connectivity. For instance, the triangular lattice deployment provides full coverage, connectivity and uniform deployment using the minimum number of sensor nodes.

– Multiple coverage

Multiple coverage is defined as an extension of simple coverage and is denoted by k-coverage. It is specific to applications such as distributed detection, mobility tracking, monitoring in high security areas and military intelligence in a battlefield. Since the failure of a single node may result in the loss or corruption of important data, one degree of coverage is not sufficient for these applications. Such applications require highly accurate

information in order to provide fault tolerance and allow the right decisions to be made. The k-coverage deployment is defined as a sensor deployment pattern where each point in the area is covered by at least k deployed sensor nodes, which means that, k-coverage tolerates at least $k-1$ node failures while maintaining coverage.



a Simple coverage. b Multiple coverage.

Figure 2.2: Full area coverage.

- Partial coverage

In some applications, full coverage of a given area is not required, in which case partial coverage ensuring a given degree of coverage is sufficient and acceptable. Partial coverage can be defined as the set of sensor nodes that cover at least θ percent of the entire area and is referred to as θ-coverage where $0 < \theta < 1$. Generally, environment monitoring applications require only partial coverage. An example of such an application is temperature-sensing applications where it is sufficient to sense the temperature of 80% of the region to know the temperature in this region. Another example is forest fire applications where full coverage of the forest is required in the dry season whereas only an 80% coverage rate is required in the rainy season. Partial coverage is a way of reducing energy consumption of sensor nodes and prolonging network lifetime since the number of sensor nodes deployed is less than the number required to fully cover the area considered. Figure 2.3 depicts sensor deployment ensuring partial coverage.

2.2.1.2 Point coverage

In many applications, monitoring the whole area might be unnecessary and it is sufficient to monitor only some specific points. Each specific point should be covered by at least one sensor node. Consequently, the deployment cost will decrease because of the smaller number of sensors used compared to the number required to cover the entire area. Examples of point of interest monitoring, include monitoring of enemy troops and bases, and capturing real-time video material of possibly mobile targets. In such applications, mobile

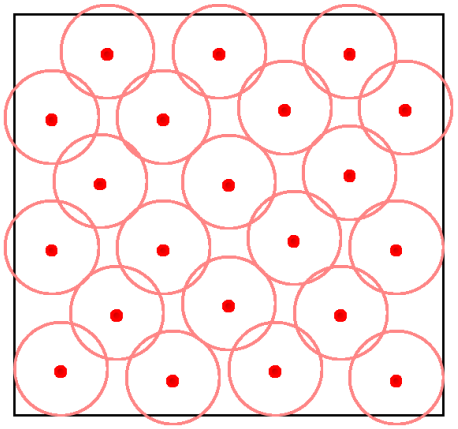


Figure 2.3: Partial coverage.

flying sensors can be deployed to monitor a point of interest. The PoIs can be either fixed or mobile.

- Fixed PoI

A PoI is fixed if it always has the same location. It is simpler to cover a fixed PoI with prior knowledge of its position than to follow a mobile PoI. Figure 2.4 depicts an example of static PoI monitoring. In this example sensor nodes do not only cover the PoI but also maintain connectivity with the sink in order to report detected events.

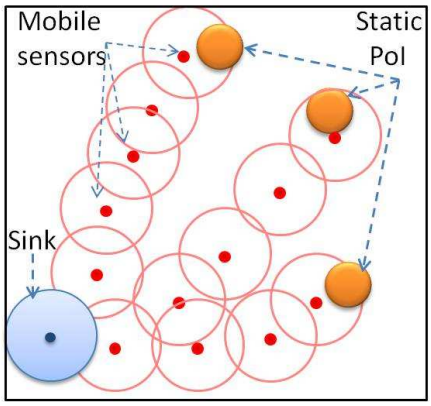


Figure 2.4: Static PoIs coverage.

- Mobile PoI

A point of interest is considered mobile if it changes its location. We distinguish two solutions to cover such a mobile PoI. If mobile sensors are used, then they should be deployed in such a way as to cover this mobile PoI and keep track of it when it moves to a new

position. If static sensors are deployed, they should be placed such that for each new position of the PoI there is at least one sensor node that can cover it.

The monitoring of these points can be permanent (each point is permanently monitored by at least one sensor) or not. In the latter case, a mobile sensor should visit this point to collect its data.

2.2.1.3 Barrier coverage

In several important applications, sensors are not designed to monitor events inside the area considered but to detect intruders that attempt to enter this area. Examples of such applications involving movement detection are the deployment of sensors along international borders to detect illegal intrusion, around forests to detect the spread of forest fire, around a chemical factory to detect the spread of lethal chemicals, and on both sides of a gas pipeline to detect potential sabotage. Barrier coverage, which guarantees that every movement crossing a barrier of sensors will be detected, is known to be an appropriate model of coverage for such applications. There are two types of barrier coverage: full barrier coverage or partial barrier coverage.

- Full barrier coverage

A barrier is fully covered if every location along this barrier is covered by at least one sensor node, as it is shown in Figure 2.5.

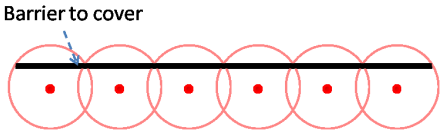


Figure 2.5: Full barrier coverage.

- Partial barrier coverage

When the number of sensors is insufficient to fully cover the barrier, sensor nodes will provide partial coverage. The deployment algorithm should ensure that by moving the sensor nodes along the barrier, they will be able to detect an intruder trying to cross this barrier, with a probability that is higher than a given threshold.

2.2.2 Connectivity problems

Two sensor nodes are said to be connected if and only if they can communicate directly (one-hop connectivity) or indirectly (multi-hop connectivity). In WSNs, the network is considered to be connected if there is at least one path between the sink and each sensor node in the area considered.

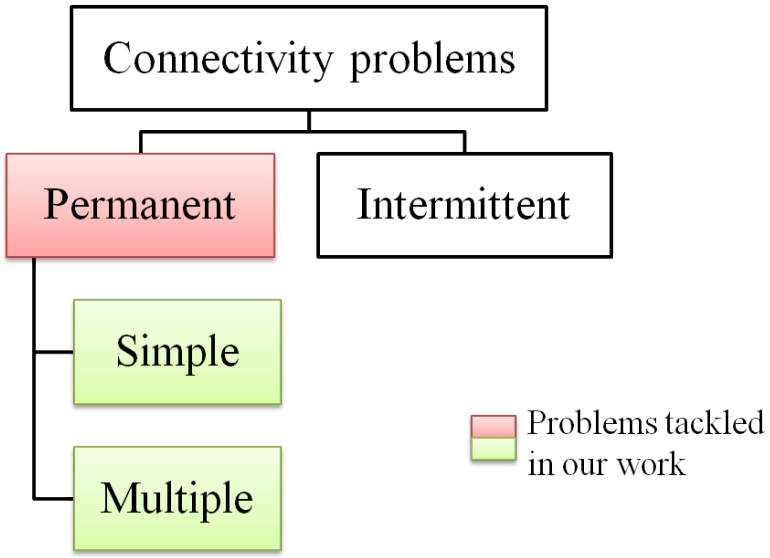


Figure 2.6: Connectivity problems.

To monitor a specific area it is not enough to ensure coverage without considering connectivity. When an event is detected, it should be reported to a sink. Consequently, it is necessary to ensure connectivity between the sensor nodes and the sink in order to guarantee the transfer of information to the sink. There are two types of network connectivity: full connectivity and intermittent connectivity as illustrated in Figure 2.6.

2.2.2.1 Full connectivity

Since connectivity is essential to guarantee the transfer of information, it should have the same degree of importance as coverage. Thus, to efficiently monitor a given area, many applications require not only full coverage but also full connectivity in order to collect information and report it.
As we saw in the previous section dealing with full coverage, full network connectivity can also be either simple (1-connectivity) or multiple (k-connectivity). In addition, full connectivity can be maintained during the deployment procedure or it can be provided only when sensors have been deployed in the area. In the following, we use the term connectivity to represent full connectivity.

- Simple versus Multiple connectivity

Full connectivity is said to be simple if there is a single path from any sensor node to the sink.

Full connectivity is termed multiple if there are multiple disjoint paths between any sensor node and the sink.

- Preserved connectivity versus connectivity at the end of the algorithm

Considering only initial sensor deployment where all the nodes are connected to each other and to the sink, the deployment algorithm is said to preserve connectivity if and only if at any time during the deployment, there is a path connecting every sensor node to the sink. On the other hand, if the deployment algorithm ensures connectivity at the end of the algorithm, connectivity can be lost during the deployment process. However, at the end of its execution, the deployment algorithm should guarantee full connectivity.

2.2.2.2 Intermittent connectivity

In some applications, it is not necessary to ensure full connectivity in the area considered. It is sufficient to guarantee intermittent connectivity by using a mobile sink that moves and collects information from disconnected nodes. There are two types of intermittent connectivity: the first one uses only one or several mobile sinks and the second uses a mobile sink and multiple throwboxes (Cluster heads).

- Isolated nodes

When the communication range, R, is less than the sensing range, r, full coverage can be achieved but without maintaining connectivity between neighboring nodes. Consequently, these nodes will be isolated. One solution to collect the information detected from isolated nodes is to use one or several mobile sinks. One or several nodes are in charge of visiting any sensor node that is not connected to the sink.

- Connected components

In any connected component, all sensor nodes of this component are connected to each other. However, they are disconnected from nodes in another connected component and they may also be disconnected from the sink. To take advantage of the connectivity within a connected component, a throwbox, illustrated in Figure 2.7 by green nodes, can be assigned to each connected component. A throwbox has the task of collecting the information from each node belonging to its component. Then, one or several nodes, also called mobile sinks (the blue node in Figure 2.7) are in charge of visiting the throwbox of each connected component.

In this section, we studied the different coverage and connectivity problems in WSNs. In the following section, we give a representative use case for each of those problems.

2.3 Representative use cases

Depending on the application requirements, we can distinguish the following use cases (UC) dealing with coverage and connectivity, and representative of most applications:

UC1 *monitoring of a temporary industrial worksite* requires full area coverage, permanent network connectivity and a uniform deployment of sensor nodes to reduce data gathering delays and provide a better balancing of node energy.

UC2 *forest fire detection* requires full area coverage in dry seasons and only 80% in rainy seasons. Permanent connectivity is required in both cases so firefighters can be alerted.

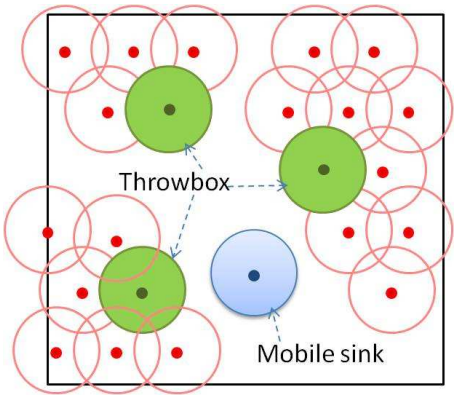


Figure 2.7: Intermittent connectivity using a mobile sink and throwboxes.

UC3 *detecting and tracking of intruders in restricted areas.* Such applications require full area coverage; furthermore, the most critical zones should be covered by more than one sensor node (i.e. multiple coverage). Permanent connectivity is also required.

UC4 *monitoring of endangered species at some water points*: the idea is to obtain statistics about the number of individuals of this species from the number of individuals visiting the water point. A full or partial belt of sensor nodes is established along or around the water point, depending on its size. Intermittent connectivity is usually sufficient.

UC5 *detection of intruders crossing a barrier* (e.g. the border of a country, doors or windows in an apartment). Such applications require a barrier coverage with permanent connectivity. Depending on the application requirements, one or several barriers are needed, the latter case being called multiple barrier coverage.

UC6 *air pollution monitoring in a smart city.* Partial area coverage is sufficient and intermittent connectivity can be compliant with the application requirements.

UC7 *instantaneous snapshot of measures taken at locations predefined by the application.* In precision agriculture, the goal is to detect the occurrence of diseases in the crops. In a smart city, the goal is to track an air pollutant. Such applications require the coverage of static points of interest. Permanent connectivity may be not needed. Intermittent connectivity can be provided by mobile robots (e.g. tractors for precision agriculture).

UC8 *tracking of wild animals or a truck fleet with embedded sensors.* In such a case, different technologies can be used to track these mobile points of interest (e.g. Argos beacons for animals, 3G/4G systems for trucks). Depending on the application requirements, connectivity may be intermittent (e.g. for animals) or permanent (e.g for a fleet of trucks).

UC9 *health monitoring of isolated workers, disabled people or elderly.* They are considered as mobile Points of Interest (PoIs) that must be permanently covered. Permanent connectivity is required.

All these uses cases will enable us to classify the coverage and connectivity problems encountered in the literature (see Table 8.1), according to the criteria defined more precisely in Section 2.2.
With the emergence of smart cities, different use cases can coexist simultaneously. For instance, air pollution monitoring, surveillance of parking lots, public lighting control, and pollutant tracking are examples of sensor deployments that are likely to be very common in our cities in the near future.

		Area coverage			Barrier coverage			PoI coverage	
		Full		Partial	Full		Partial	Static	Mobile
		Simple	Multiple		Simple	Multiple			
Connectivity	Permanent	**UC1**	**UC3**	**UC2**		**UC5**	**UC4**	**UC8**	**UC9**
	Intermittent			**UC6**	**UC4**		**UC4**	**UC7**	**UC8**

Table 2.1: Classification of use cases.

2.4 Coverage and connectivity problems with regard to R and r

Some deployment algorithms only work when a given relationship exists between the radio range R and the sensing range r. For instance, if $R \geq 2r$, it is sufficient to ensure full coverage, and connectivity will be provided as a consequence. In the following, we study the different cases considered in the literature.

- Case $R \geq 2r$: Full coverage implies connectivity

In (1) and (2), the authors prove that when $R \geq 2r$, the full coverage of a convex area implies full network connectivity. This result is extended to k-coverage and k-connectivity in (2). Then, using this assumption, it is sufficient to ensure full coverage, and connectivity will be a consequence.

- Case $R \geq \sqrt{3}r$: Full coverage implies connectivity

In (3), it is proved that when $R \geq \sqrt{3}r$, ensuring full coverage implies full connectivity. Moreover, the number of sensors needed is optimal when the triangular lattice is used as a deployment pattern. For instance, in (4), the authors propose a deployment algorithm where each sensor node should be placed in a vertex of an equilateral triangle of edge $\sqrt{3}r$.

- Case $R = r$

An optimal deployment algorithm is proposed in (5) to ensure full coverage and 1-connectivity when $R = r$. In this algorithm, sensor nodes are deployed along a horizontal line, with

each two neighboring nodes at a distance of r. Adjacent lines are at a distance of $(\frac{\sqrt{3}}{2}+1)r$. In such a deployment, full coverage is ensured, but only sensor nodes located on the same line are connected. That is why the authors propose adding a sensor node between each two adjacent lines in order to connect them, such that these nodes form a vertical line, thereby ensuring 1-connectivity. The optimality of this deployment in terms of the number of sensor nodes is proved in (3).

- Case $R < \sqrt{3}r$

When $R < \sqrt{3}r$, full coverage does not imply network connectivity. Network connectivity is necessary to report information and it is a vital part of the monitoring task. Thus, ensuring connectivity while maximizing the area coverage becomes the goal of the deployment algorithm. The deployment algorithm proposed in (5) which deploys sensor nodes in horizontal lines and connects these lines by placing sensor nodes between two adjacent lines, is generalized in (3), as illustrated in Figure 2.8. In addition, this deployment is optimal when the distance between neighboring sensor nodes on the same line R and the distance between two adjacent lines is $r + \sqrt{r^2 - \frac{R^2}{4}}$.

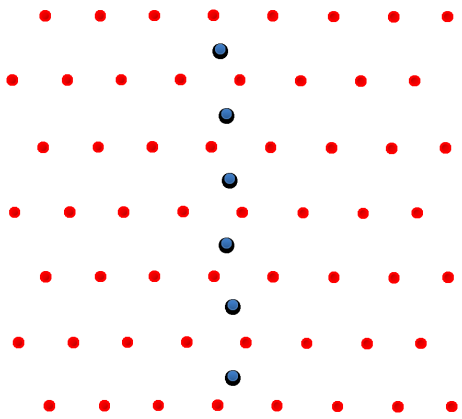


Figure 2.8: Sensor deployment with added sensors to ensure connectivity.

- Case arbitrary R and r

In (6), the authors propose an algorithm that aims at preserving network connectivity while maximizing area coverage. Starting with an initial deployment where all the sensor nodes are connected to the sink, a virtual force algorithm is applied in order to redeploy sensor nodes in the area considered. As the sensing and radio ranges do not meet the assumption $R \geq \sqrt{3}r$, when sensor nodes move to their new positions they check whether they are still connected to the sink. If they are not, they move towards the sink until connectivity is established. This algorithm preserves full network connectivity during the deployment process and tries to maximize the area coverage with any given values of R and r. In (7), the authors propose a deployment algorithm that aims at ensuring full coverage and full network connectivity of an area containing obstacles of different shapes.

The authors propose dividing the area into two different types of region: small regions or large regions which may contain boundaries and obstacles. As there are no assumptions concerning R and r, in the small regions (like a belt), sensors are deployed along the bisectors of this region and are separated by $r_{min} = min\{R, r\}$. In the large region, sensor nodes are deployed in rows. The distances which separate sensor nodes and rows are determined according to the values of R and r.

Goal(s)	Relationship between R and r	Deployment pattern: examples
Full coverage (Coverage implies connectivity)	$0 < \frac{R}{r} \leq \frac{1}{2}3^{3/4}$	Hexagonal grid (3)
	$\frac{1}{2}3^{3/4} \leq \frac{R}{r} \leq \sqrt{2}$	Square grid (3)
	$\sqrt{2} \leq \frac{R}{r} \leq \sqrt{3}$	Rhomboid pattern (3)
	$\frac{R}{r} \geq \sqrt{3}$	Triangular lattice (3)
1-Full or partial coverage by horizontal lines 2-Connectivity by an additional vertical line	$R = r$	- Horizontal lines + a node between two adjacent lines (5)
	$R < r$	- Horizontal lines + a node between two adjacent lines (3) Optimal when $R < \sqrt{3}r$, distance between nodes R and distance between adjacent lines $= r + \sqrt{r^2 - \frac{R^2}{4}}$
Ensuring connectivity and maximizing coverage	No assumptions	- Floors (6)
Full coverage and Full connectivity	No assumptions	- Dividing the area into small and large regions (7) Sensors are deployed along the bisectors of small regions and in rows in the large regions

Table 2.2: Relationship between r and R.

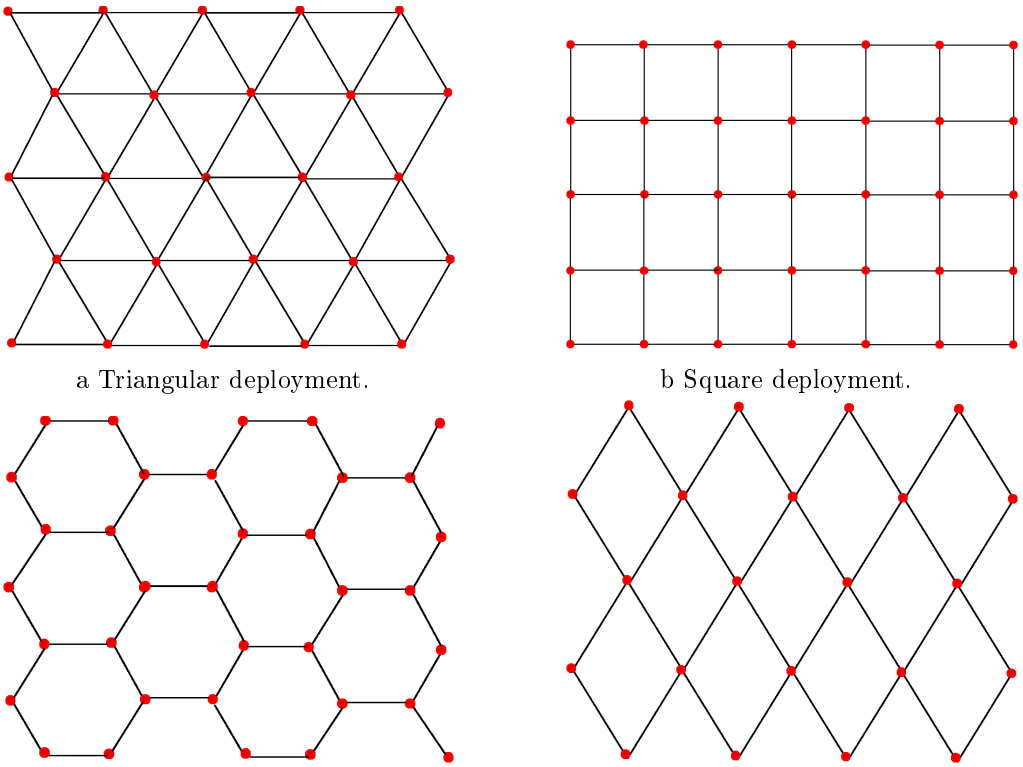
2.5 Coverage and connectivity with regard to regular optimal deployment

Sensor nodes can be deployed in a regular pattern. This pattern can be a triangular lattice, a square grid, a hexagonal grid or a rhomboid grid. For each pattern, the authors in (8) specify a condition that ensures coverage of the area and consequently guarantees network connectivity.

- If $R \geq r$ and the hexagonal grid pattern is used, then full area coverage is ensured and the network is connected.
- If $R \geq \sqrt{2}r$ and the square grid or rhomboid pattern is used, then full area coverage is ensured and the network is connected.
- if $R \geq \sqrt{3}r$ and the triangular lattice pattern is used, then full area coverage is ensured and the network is connected. The triangular lattice is the optimal deployment pattern to ensure full area coverage and guarantee network connectivity.

These conditions are studied in (3) with regard to the optimal number of sensor nodes and the regular pattern used. It was proved that when:

- $0 < \frac{R}{r} \leq \frac{1}{2}3^{3/4}$, the hexagonal grid is the best deployment pattern (i.e. it requires the minimum number of sensor nodes). See Figure 2.9c.
- $\frac{1}{2}3^{3/4} \leq \frac{R}{r} \leq \sqrt{2}$, the square grid is the best deployment pattern. See Figure 2.9b.
- $\sqrt{2} \leq \frac{R}{r} \leq \sqrt{3}$, the rhomboid pattern is the best deployment pattern. See Figure 2.9d.
- $\frac{R}{r} \geq \sqrt{3}$ the triangular lattice is the best deployment pattern. See Figure 2.9a.



a Triangular deployment.

b Square deployment.

c Hexagonal deployment.

d Rhomboid deployment.

Figure 2.9: Regular deployment patterns.

2.6 Conclusion

Coverage and connectivity issues are well studied in the literature. Existing surveys (9; 10; 11; 8; 12; 13; 14) introduced basic concepts related to coverage and connectivity. From these surveys we distinguished different problems related to coverage and connectivity in WSNs. In this chapter we provided comprehensive definitions of coverage and

connectivity with their possible variants. These variants depend on the latency and robustness requirements that differ in the applications considered, leading to representative use cases. These definitions of coverage and connectivity are valid for both 2D and 3D problems.

The focus of the next chapter will be the deployment algorithms in WSNs. For each coverage and connectivity problem defined in this chapter, we will list some deployment algorithms found in the literature.

Chapter 3

Deployment algorithms in WSNs

Contents

3.1 Introduction

The coverage and connectivity problems in WSNs depend on the locations of the sensor node. These locations determine the percentage of coverage in the entity monitored and whether network connectivity is maintained. Clearly, to meet the application requirements, the location of sensor nodes should be carefully studied. Many deployment algorithms are proposed in the literature to determine the appropriate sensor node locations. However, these deployment algorithms may vary according to the strategy used, the coverage problem, the connectivity problem, the number of sensor nodes needed, etc. In this chapter, we give a global analysis of the deployment problem by discussing the impacting factors, detailing the common assumptions and models adopted in the literature. We also propose some performance criteria to evaluate deployment algorithms. Moreover, we discuss various deployment algorithms which cope with area coverage, barrier coverage and Points of Interest (PoIs) coverage. We dedicate an entire section to issues and recommendations regarding coverage and connectivity problems which may be helpful to choose the most suitable deployment algorithm.

3.2 Analysis of the criteria of deployment algorithms

In this section, we analyze the various factors that have a positive or negative impact on sensor deployment. We discuss the common assumptions and models found in the literature before focusing on the relationship between the sensing range, r, and the communication range, R, which have a great impact on the behavior of the deployment algorithm. We end this section by defining performance criteria for evaluation purposes.

3.2.1 Factors impacting the deployment

Several factors impact the deployment and determine how satisfactory the application is. They concern:

- *The assumptions and models used concerning r the sensing range and R the communication range.* Such assumptions and models are discussed in the next section. The discrepancy between these oversimplified models and reality may explain why the results obtained are not those which might be expected. The values of r and R determine the minimum number of sensors needed to fully cover the entity monitored (i.e. area, barrier or PoIs). The deployment algorithms that use exactly this number are said to be optimal. Depending on the relationship between r and R, detailed in Section 2.4 Chapter 2, some algorithms either work or they do not. Others are valid whatever the relationship between r and R may be, but are not, however, optimal in all cases.

- *The number of sensor nodes available for the deployment* and the dimensions of the entity monitored will determine whether this number is sufficient to fully cover the entity. It is usually assumed that this entity has a regular shape (e.g. rectangle, disk, etc). However, the reality is often more complex an involves irregular borders.

- *The sensor nodes' ability to move is a determining factor.* If sensor nodes are unable to move, the only possible deployment is an assisted one, in which a mobile robot for example is used to place the static sensor nodes at their final location. If on the other hand, each sensor node is autonomous and able to move, autonomous deployment is carried out, yet it should be noted that in such a case, the sensor nodes' movement will consume more energy than is used for communication during the deployment.

- *The initial topology* may require some extensions to the deployment algorithm. For instance, if the initial topology comprises several disconnected components and a centralized deployment algorithm is used, a mobile robot should be used to collect the initial positions of the nodes needed by the centralized deployment algorithm to compute the final positions of these nodes and this information should be disseminated to them. If on the other hand, a distributed deployment algorithm is chosen, this algorithm should include a neighborhood discovery phase as well as a spreading phase to allow sensor nodes to quickly discover other connected components.

- *The energy of sensor nodes* is difficult or impossible to renew, and this fact is of great importance. In the deployment phase, the main reason for energy consumption is the movement of the nodes, whereas in the data gathering phase it is communication between the nodes. In both phases, energy-efficient techniques must be used.

- *The presence of obstacles* makes the deployment more complex: no sensor node should be placed such that an obstacle prevents its being located. Hence, the obstacles must be detected and a strategy must be used by the deployment algorithm to bypass the obstacles. Furthermore, if the shape of the entity monitored is complex with irregular borders, some extensions to the deployment algorithm will be needed.

- *The quality of the data gathering* required by the application may lead to a uniform and regular deployment. Such a deployment provides smaller data gathering delays (15), a better time and space consistency of the data gathered, which leads to a more accurate snapshot of the measures taken.

- *The positioning system* may introduce some inaccuracy in the position of the nodes; such a positioning error is very common with GPS. To meet the application requirements, the deployment algorithm should not accumulate positioning errors during the deployment.

3.2.2 Common assumptions and models

The common assumptions and models found in the literature concern:

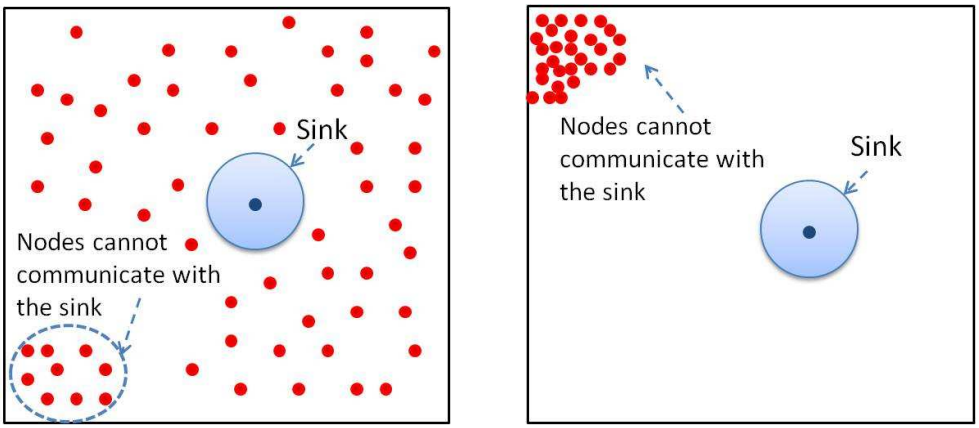
- *Communication:*

– A unit disk graph model is generally adopted, where any two nodes whose Euclidean distance from each other is less than or equal to the communication range R, have a communication link: they are able to communicate in both directions. This binary model is, however, too simple and does not match the real world. Some authors have introduced more complex models where the probability of success falls less abruptly when the distance increases towards to R (16).

– A consequence of the unit disk graph model is that any wireless link is assumed to be symmetric. This assumption is not always true in the real world.

– A frequent assumption is that all sensor nodes have the same communication range. Sensor nodes may differ in their age, their manufacturer, and their communication capacity. Hence some sensor nodes may have a higher transmission range than others.

– The initial topology considered in centralized deployment algorithms is usually connected with the sink. This may not be the case in the real world (see the discussion in Section 3.2.1). In distributed deployment algorithms, the initial topology is generally random, as it facilitates the spreading of nodes, leading to shorter convergence delays. For instance, Figure 3.1a depicts an initial topology where some sensor nodes are unable to communicate with the sink. Figure 3.1b depicts another initial topology where all the sensor nodes are grouped at an entry point but unable to communicate with the sink.



a Random Topology. b Entry point topology.

Figure 3.1: Intial disconnected topology.

- *Sensing:*

– A unit disk graph model is used to model the sensing of a sensor node. It is assumed that any event occurring within the disk of radius the sensing range r, centered at the sensor node will be detected. This assumption may well, however, prove over optimistic in the presence of obstacles, for instance.

– The homogeneity of sensors (i.e. the same sensing model with the same sensing range) is generally assumed. Which may not be the case in the real world.

- *The presence of obstacles:*

– Most authors assume that the entity to be monitored is flat and that nodes can move freely without obstacles. Such an assumption cannot be made for rescue applications after a disaster, for instance.

3.2.3 Criteria for performance evaluation

Each pattern may suit some application requirements. The question is then how to evaluate and select the best one. Different evaluation criteria have been introduced:

- *coverage:* (e.g. area, barrier, point of interest) is the main criteria to evaluate the efficiency of the algorithm. Usually, coverage is computed as follows: the area to cover is divided virtually into LxW grid units, where L is the length and W the width of the area considered. A grid unit is considered to be covered if and only if its centered point is covered by at least one sensor node. The coverage rate is computed as the percentage of grid units covered.

- *connectivity:* is also important. The type of connectivity (i.e. full or intermittent) is application dependent. For some applications, maintaining full connectivity is required in order to report any detected event immediately to the sink. Other applications with fewer constraints require intermittent connectivity: usually a data mule to collect data from disconnected sensor nodes.

- *convergence and stability:* convergence is evaluated by the convergence time defined as the time needed to achieve the required coverage and connectivity. In distributed deployment algorithms, convergence may be difficult to reach because of node oscillations. In addition, the stability of the deployment is an important criterion that may be used to detect the completion of the deployment.

- *energy and distance traveled:* during the deployment, the main cause of energy consumption is the movement made by the nodes. That is why the total distance traveled by the nodes must be measured, as this measure reflects the energy consumed. Obviously, minimizing the total distance traveled leads to savings in energy. Notice that the convergence and stability performance has a strong impact on the distance traveled and the energy consumed. Once the deployment has been carried out and the nodes are stationary, data gathering takes place. The main cause of energy consumption in this phase is communication. To maximize network lifetime,

node activity scheduling can be used to make nodes sleep when they are not needed for data gathering.

- *communication overhead:* comes from the control messages exchanged between the nodes to organize the deployment and the data gathering. In the case of contention-based medium access, collisions imply retransmission and increase the overall bandwidth and energy consumption. The aim is to reduce this overhead.

- *uniformity, regularity and optimality of the deployment:* if space consistency of the measures taken is expected, a uniform deployment is needed: all the nodes (except the border ones) should have the same number of neighbors. Similarly, if the measures should be taken at equidistant positions, a uniform and regular deployment is needed. Usually, such a deployment reproduces the same geometric pattern (e.g. triangle, hexagon, square, etc). Depending on the relationship between r and R, some patterns are optimal (see Section 2.4 in Chapter 2). This optimality is useful because it requires the smallest number of sensor nodes to meet the application requirements. A uniform and regular deployment is also mandatory when the application requires time and space consistency of the data gathered.

3.3 Area coverage and connectivity algorithms

In this section, we study area coverage and connectivity algorithms with regards to analysis criteria presented above.

3.3.1 Full coverage

Many deployment algorithms aim to ensure full coverage of the area to be monitored. These algorithms are classified into three strategies. We distinguish the forces-based strategy, the grid-based strategy and the computational geometry-based strategy.

3.3.1.1 Forces-based strategy

The forces-based strategy is known for its simple deployment principle. This principle is based on virtual forces that can be attractive, repulsive or null. In this strategy, a sensor node should maintain a fixed threshold distance called D_{th} from its 1-hop neighbors. Then, if the distance separating two neighboring nodes is greater than D_{th}, an attractive force is exerted, whereas if this distance is less than D_{th}, a repulsive force is exerted. Otherwise, the force is null since the distance separating the neighboring sensor nodes is equal to D_{th}, the required distance. This principle is illustrated in Figure 3.2, where $\overrightarrow{F_{ij}}$ denotes the force exerted by sensor node j on sensor node i.

The virtual forces algorithm (VFA) is proposed in (17) as a centralized redeployment algorithm to enhance an initial random deployment. In the initial deployment, any sensor node is able to communicate with the sink in a one-hop or multi-hop manner. Then, the sink computes the appropriate new position of each sensor node based on the coverage requirements and using the virtual forces mechanism. In this work, obstacles exert a

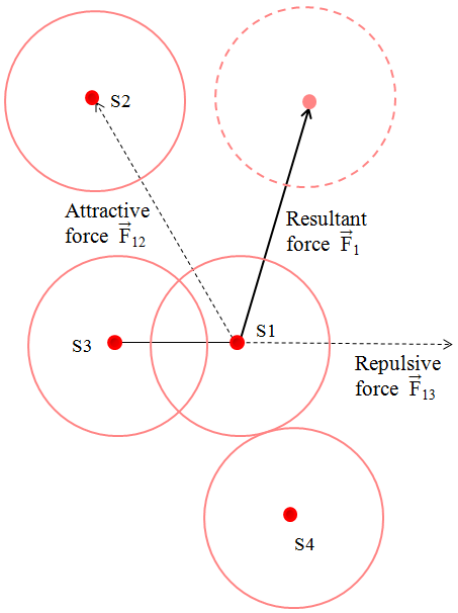


Figure 3.2: Forces based strategy.

repulsive force and an area of preferential coverage exerts an attractive force on sensor nodes. During the execution of the virtual forces algorithm, sensor nodes do not move. It is only when they receive their final positions from the sink that they move directly to them. VFA is a centralized algorithm that offers a good coverage rate of the area considered while maintaining network connectivity. However, a central entity must know the initial positions of all sensor nodes, compute their final positions and disseminate these positions to all sensor nodes. This principle is problematic when network connectivity is not initially ensured. Furthermore, when the network is very dense, this algorithm can perform poorly due to the time required to gather the initial positions of the sensor nodes.

To cope with the scalability problem, distributed versions of VFA have been proposed in the literature. For instance, the extended virtual forces-based approach proposed in (18) copes with two drawbacks of the virtual forces algorithm: connectivity maintenance and nodes stacking problems (i.e. where two or more sensor nodes occupy the same position). The connectivity maintenance problem occurs when the communication range is low, $\frac{R}{r} < 2.5$. The authors therefore propose adding an orientation force which is exerted only if the node has fewer than 6 neighbors. This force aims to keep the angle formed by one node and its two neighbors equal to $\frac{\pi}{3}$ in order to provide reliable connectivity and eliminate coverage holes. The authors observe a stacking problem, where several nodes are located in almost the same position. This is because the coefficient of the attractive forces is not well tuned. As a solution, the authors propose an exponential force model to adjust the distance between a node and its distant neighbors. However, the threshold value of $\frac{R}{r} = 2.5$ is not explained and how connectivity is maintained is not specified. Furthermore, the additional orientation force may induce node oscillations.

IVFA, Improved Virtual Force Algorithm, and EVFA, Exponential Virtual Force Algorithm are two distributed deployment algorithms proposed in (19). EVFA aims at speeding up convergence because forces increase exponentially with the distance between sensors. IVFA limits the scope of virtual forces: only nodes in radio range of a given node exert virtual forces on it. Furthermore, the stacking problem is solved by using a very small attractive force with regard to the repulsive force. IVFA converges to a steady state faster than the basic virtual forces algorithm, and defines a maximum movement at each iteration to reduce useless moves and save energy.

Usually, the virtual forces strategy is used to ensure full area coverage as the attractive and repulsive forces spread sensor nodes over the whole area and consequently achieve a high coverage rate rapidly. Furthermore, this strategy is used in (6) with the goal of preserving network connectivity. This deployment algorithm, called CPVF, Connectivity-Preserved Virtual Force, is used to monitor an unknown area with an arbitrary ratio $\frac{R}{r}$. To achieve this, a sink periodically broadcasts a message to neighboring sensors which in turn flood the message to all connected nodes. A sensor node is considered to be disconnected from the network if it does not receive the flooding message. Then, it moves toward the sink in order to reconnect. This algorithm induces a high overhead in terms of messages broadcast over the network to check the connectivity of the nodes with the sink. This paper also proposes a floor-based scheme to improve global network coverage by reducing overlapping. This scheme is based on the division of the area into equidistant floors (distant of $2r$) and encourages sensors to stay on the floor lines. Sensor nodes are added in a column between floor lines to ensure connectivity. Although this work aims at preserving network connectivity when the ratio $\frac{R}{r}$ is arbitrary, it requires a high number of sensor nodes, as illustrated in Figure 3.3, because the inter-floor distance is fixed to $2r$ for any value of R and r.

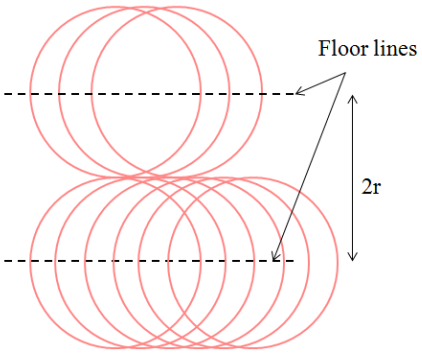


Figure 3.3: Floor based deployment.

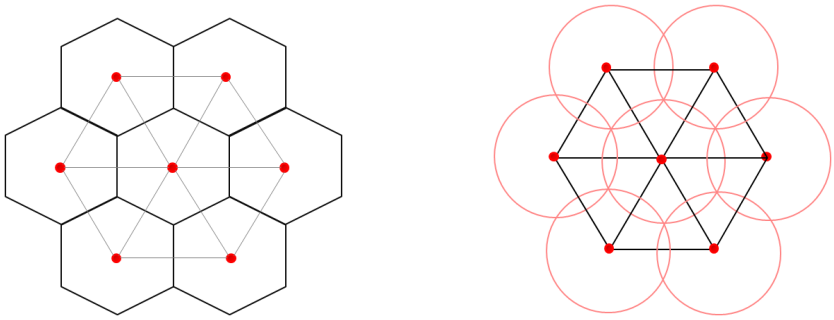
3.3.1.2 Grid-based Strategy

The grid-based strategy provides a deterministic deployment where the positions of the sensor nodes are fixed according to a special grid pattern such as a triangular lattice, a

square grid or a hexagonal grid (see Figures 3.4b, 3.5 and 3.6 respectively). Then, the area is divided into virtual cells and depending on the deployment algorithm used, sensor nodes are located either in cell vertices or at the cell center.
The grid deployment is also a regular deployment pattern as all the generated grid cells have the same shape and size. The regular deployment pattern is studied in (20) in order to provide multiple coverage (p-coverage) and multiple connectivity (q-connectivity) using the triangular lattice, square or hexagonal pattern. The value of p and q are provided by adjusting the distance separating sensor nodes and limiting the ratio $\frac{R}{r}$. A comparative study of regular pattern performance in terms of the number of nodes required is also provided to achieve 1, 3 and 5-coverage and q-connectivity. With the ratio $\frac{R}{r} \geq \sqrt{3}$, the triangular lattice is better than the square grid, which, in turn, is better than the hexagonal grid. However, with the value of $\frac{R}{r} < \sqrt{3}$, the triangular lattice becomes the worst. Multiple coverage and connectivity with regard to the regular deployment pattern is also studied in (21). The authors propose optimal deployment patterns to ensure full coverage and q-connectivity when $q \leq 6$ for certain values of $\frac{R}{r}$. They consider the hexagonal deployment pattern as a universal basic pattern that can generate all optimal patterns. Then, they present different forms derived from the hexagonal pattern by changing the edge length and the angle between adjacent edges.
When the applications require time and space consistency of the measures taken by sensor nodes regularly distributed over the area, the regular deployment pattern can be a good solution to provide a high level of coverage and connectivity with a minimum number of sensor nodes.
In the following we present some research studies proposing a regular deployment pattern based on a triangular lattice and a square grid.

Triangular grid

In (22), it was proved that the triangular lattice shown in Figure 3.4b offers the smallest overlapping area and requires the smallest number of sensor nodes. When the triangular lattice is used as a deployment pattern, each sensor node occupies a hexagonal cell. However, the deployment is not considered to be a hexagonal deployment (see Figure 3.6) since a sensor node is at the center of a hexagon and neighboring sensors form a triangular pattern (see Figure 3.4). The authors in (23), for instance, propose a deployment algorithm called HGSDA that deploys sensor nodes in a triangular lattice. This deployment starts by dividing the area into small hexagonal cells and each cell center corresponds to a sensor position. Although the cells are hexagonal, sensor nodes are deployed in a triangular lattice since the distance between two neighbors is $\sqrt{3}r$ and there is a sensor node at the cell center. HGSDA identifies redundant sensor nodes in order to place them in empty hexagonal cells. Since the size of a hexagonal cell is computed according to sensor sensing range and the area size, full coverage is achieved using the smallest number of sensor nodes. This algorithm is carried out by a sink. Then, all the sensor nodes receive their final position from the sink and move to it. HGSDA is a centralized algorithm that ensures full coverage using the minimum number of sensor nodes while ensuring simple connectivity with the sink in the final deployment. This centralized algorithm can only



a Triangular lattice with hexagonal cells.

b Triangular lattice.

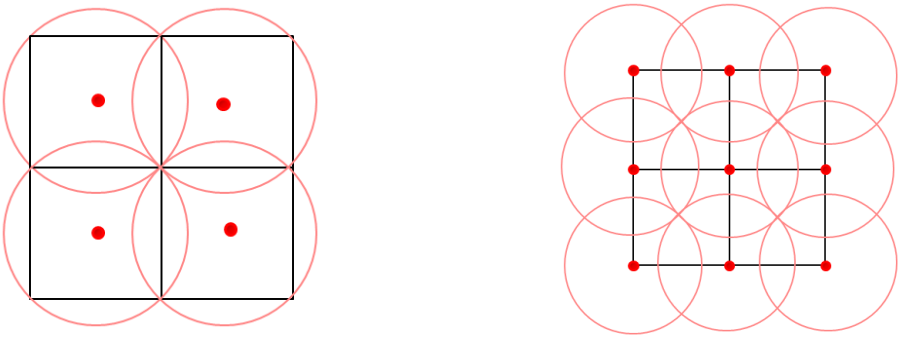
Figure 3.4: Triangular lattice.

be used if connectivity with the sink is ensured in the initial deployment. The same deployment pattern is presented in (24), but in a distributed version. However, at the beginning of the deployment, the area has not yet been divided into hexagonal cells. An initiative sensor node starts by snapping itself at the center of the first hexagonal cell and selects six sensor nodes in its vicinity to snap them in the adjacent hexagonal cells. The selected sensor nodes move to their cells and in turn select other sensor nodes to occupy their adjacent cells. Then, hexagonal cells are built progressively in a distributed way: the hexagonal side length is equal to the sensing range. Since the sensor occupies the center of the cell, the triangular lattice is used as the deployment pattern.
The deployment algorithm C^2 proposed in (25) is a triangular lattice based strategy where a sensor node occupies a hexagonal cell. Hexagonal cells are built progressively in a distributed manner by sensor nodes. This algorithm proceeds in two phases. In the first phase, called cluster heads selection, the sink which is the first cluster head in the area considered, starts by building its hexagonal cell and defines its position as the cell center. The distance between the cell center and one of the vertices is $\frac{R}{3}$ and the distance between two neighboring cell centers is $2\frac{R}{3}$ in order to maintain network connectivity during the deployment process. Then, the sink determines the center of each neighboring cell and informs sensor nodes in its neighborhood. The nearest sensor node to the cell center is selected as a cluster head of its hexagonal cell, and it should move towards its cell center. In turn, the new cluster heads define the center of their neighboring cells. The second phase is called node balancing and the goal is to improve area coverage by balancing the number of sensor nodes between cells. To do so, if the difference between sensor nodes in two neighboring cells is greater than 1, some sensor nodes will move to the cells with a deficit number of nodes. In this deployment algorithm, a hexagonal grid is used to ensure full coverage and maintain full connectivity. Energy saving is achieved by selecting a cluster head for each cell and balancing the number of sensor nodes between adjacent cells. This algorithm performs well when the sink is located at the center of the area and all the nodes are grouped around the sink.

Square grid

The square grid strategy is used in (26) where the area to be monitored is divided into square cells, as shown in Figure 3.5. Each cell represents the maximum square size that is covered by one sensor node. Each sensor node occupies a cell center to cover the corresponding square cell. If an empty cell exists, neighboring sensor nodes should decide to which one they will move in order to cover it, such that if new empty cells appear, they will be around the sink. Redundant nodes should move toward the sink so as to cover empty cells that can occur along the path to the sink.

A grid-based approach is also used for robot-assisted sensor deployment. As an example in (27), a robot places sensor nodes at the vertices of a square cell. Then, each sensor node colors itself white if it is adjacent to an empty cell and black otherwise. Neighboring sensor nodes exchange *hello* messages to inform each other about white nodes (empty cells) and maintain a back pointer corresponding to the nearest empty cell along the backward path of the robot. Then, the robot backtracks this back pointer to drop sensor nodes in the empty cells. This algorithm guarantees full coverage in a failure free environment using a mobile robot in a square grid.
It is assumed that the robot carries enough sensors to heal any coverage hole (i.e. empty cell) that is detected. Such strategies are used when the sensor nodes are static, and a mobile robot is used to ensure coverage by repairing any coverage hole detected by the sensor nodes. The new problem is that of detecting coverage holes and optimizing the robot's trajectory.



a Sensors in cell centers.

b Sensors in cell vertices.

Figure 3.5: Grid Based Strategy.

3.3.1.3 Computational geometry based Strategy

The computational geometry strategy is used to solve problems based on geometrical objects: points, polygons, line segments, etc. The Voronoi diagram and Delaunay triangulation are two computational geometry methods used in WSNs to solve static problems. The Voronoi diagram is a method of partitioning the area into a number of polygons based

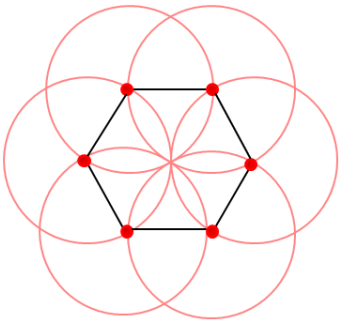
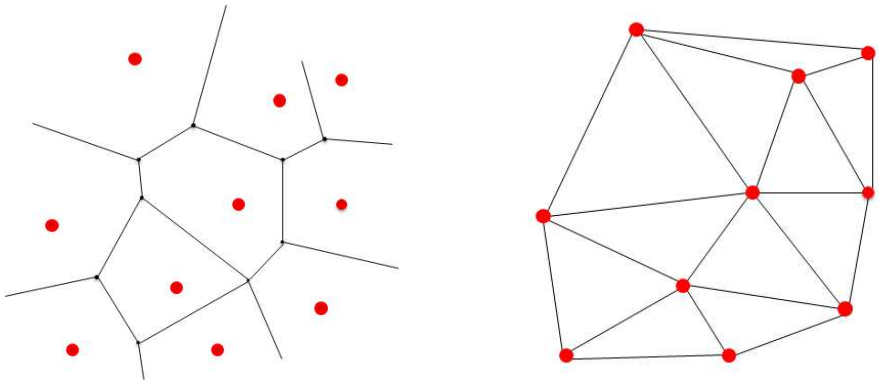


Figure 3.6: Hexagonal pattern.

on distances to a specific discrete set of nodes. Each node occupies only one polygon and is closer to any point in this polygon than any other node in the neighboring polygons (see Figure 3.7b). These polygons can be obtained by drawing the mediator of each two neighboring nodes. Consequently, the edges of the polygons are equidistant from neighboring nodes. Delaunay triangulation is the dual graph of the Voronoi diagram (see Figure 3.7a). It can be constructed by connecting each two neighboring points in the Voronoi diagram whose polygons share a common edge. The Voronoi diagram and Delaunay triangulation are used in WSNs to deal with coverage hole problems. The occurrence of coverage holes after the deployment of sensor nodes in a given area can be considered as one cause of a low coverage rate. By detecting and healing these holes, the coverage rate can be maximized.

Deployment algorithms based on the Voronoi diagram

Some schemes proposed are based on Voronoi diagram to detect coverage holes. Sensor nodes are able to construct their Voronoi polygons based on location information received from their neighbors. Due to these Voronoi polygons, nodes can detect coverage holes. Then, they move in order to reduce or eliminate these holes while maximizing the coverage rate of the area.
In (28), three distributed moving algorithms are proposed: VEC, VOR and Minimax algorithms. The VECtor based algorithm (VEC) is inspired by the behavior of electromagnetic particles. When two electromagnetic particles are too close to each other, an expelling force pushes them apart. VEC pushes sensor nodes away from a densely covered area. In contrast to the VEC algorithm, the VORonoi based algorithm (VOR) pulls sensor nodes to the sparsely covered area. The Minimax algorithm is similar to VOR. It fixes coverage holes by moving sensor nodes closer to the furthest Voronoi vertex. However, it does not go as far as VOR in order to avoid situations in which a vertex that was originally closer now becomes the furthest. Minimax chooses the node target position as the point inside the Voronoi polygon whose distance to the furthest Voronoi vertex is minimized. Minimax and VOR do not ensure uniform coverage of the final deployment since the algorithm stops as soon as full coverage is obtained. Moreover, if the number of sensors is not sufficient to



a Voronoi diagram. b Delaunay triangulation.

Figure 3.7: Computational geometry approaches.

cover the whole area, node oscillations may occur.

Deployment algorithms based on Delaunay Triangulation

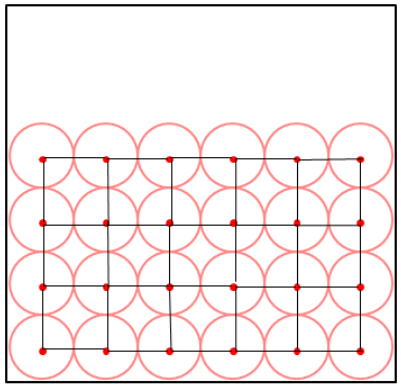
In (29), a centralized algorithm is proposed to cope with the boundaries and obstacle coverage problem. In their paper, the authors propose a deterministic sensor node placement to ensure full coverage of an area containing obstacles of arbitrary shapes. Sensor nodes are deployed in a triangular lattice over the whole area as if there were no obstacles. Then, sensor nodes inside the obstacles are eliminated and so coverage holes may occur around these obstacles. To deal with this problem, Delaunay triangulation is used to partition these coverage holes into triangles of edges less than r, and then, a sensor node is placed in one of the triangle vertices to cover it.

Other computational geometry deployment algorithms

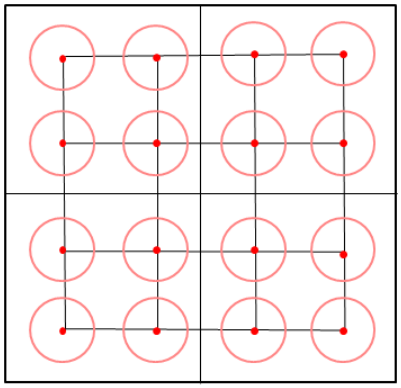
Another study based on a computational geometry strategy is proposed in (30) to detect any coverage hole and calculate its size. In this work, the authors do not rely on a Voronoi diagram or Delaunay triangulation, but, they propose a triangular oriented diagram called HSTT that connects static sensor nodes such that every three neighboring nodes form a triangle. Using a HSTT diagram, coverage holes can be detected and the required number of mobile sensors to heal these holes can be determined. Although this HSTT diagram presents some advantages compared to a Voronoi diagram, such as its simplicity and its accuracy when computing the size of the coverage holes, it requires a high level of energy consumption to achieve its goal.

3.3.1.4 Other deployment strategies

Other deployment strategies exist. They include off-line optimization algorithms which compute off-line the best position of each sensor node with the goal of ensuring the coverage



a A large uncovered area.



b Regular distribution of uncovered area.

Figure 3.8: Partial coverage.

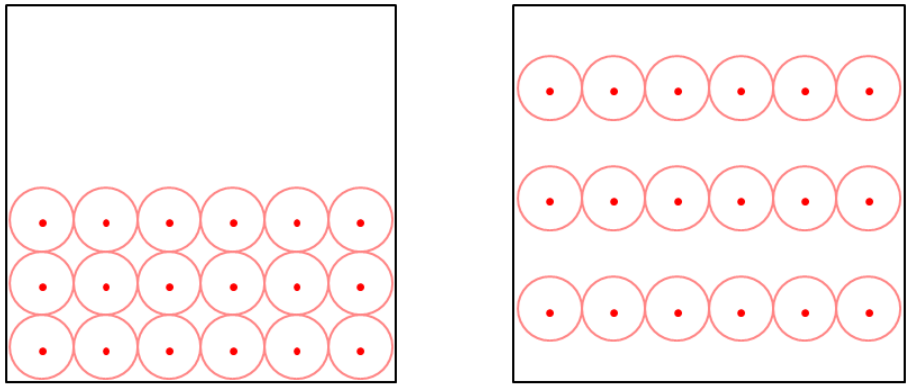
and connectivity required by the application. For this purpose, they employ optimization techniques, usually based on a linear programming of the problem considered. They discretize the area of interest and decide for each point in the area whether a sensor should be located there or not, taking into account the application requirements (e.g. maximum number of sensors, maximum cost, etc). See for instance (31).

3.3.2 Partial coverage

The area coverage problem has been widely studied in the literature. As we have pointed out previously, a great deal of effort has focus on the issues of full area coverage. However, only a few studies have focused on partial area coverage.
Generally, partial coverage is one solution to prolong network lifetime when full coverage is not required. The foremost requirement in this case is that the coverage rate provided should be higher than some predefined bound which is a specific parameter fixed by the application. The goal is to cover at least θ percent of the area considered while maintaining a connected graph between these nodes. Partial coverage is useful to measure the temperature and humidity, to detect smoke and to provide early warning of a forest fire (32), for instance.
In addition, to avoid a large area being uncovered (see Figure 3.8a), the uncovered areas should be regularly distributed (see Figure 3.8b). For that purpose, the authors in (33) propose dividing the area to be monitored into subregions of equal size. The goal is then to cover θ-percent of each subregion.

3.3.3 Intermittent connectivity

The deployment algorithms presented above ensure full or partial coverage with permanent connectivity. When permanent connectivity is not required, intermittent connectivity is provided, exploiting the mobility of some nodes. The strategies differ in:



a Bad distribution. b Good distribution.

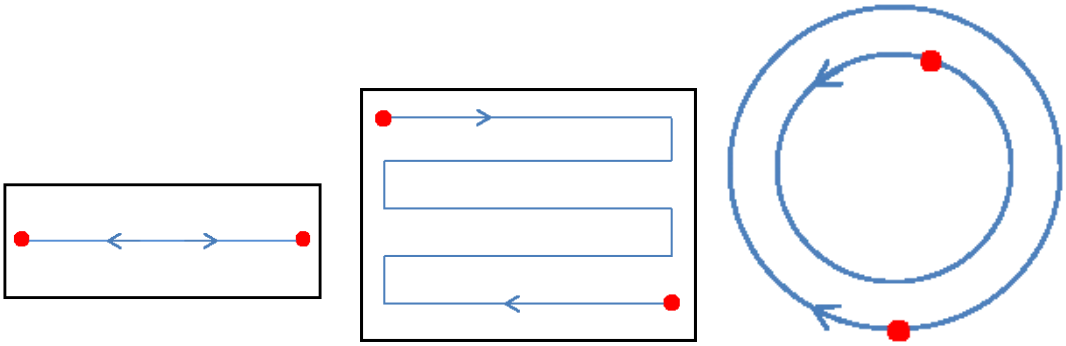
Figure 3.9: Different distributions of an uncovered area.

- the number of mobile nodes: one mobile node or several. If several, how do the mobile nodes coordinate their action to visit nodes and gather their data?
- the trajectory type of mobile nodes:
  - a fixed predefined geometrical trajectory like a line or a circle, for instance. This trajectory do not visit node positions but it allows mobile node to communicate with sensor nodes during its movement.
  - a trajectory that depends on node positions. It should go through all the nodes or a subset of nodes depending on the deployment architecture (e.g. clustering).

More particularly, we distinguish:

Mobile sink with multiple cluster heads (throwboxes): In (34), a large number of sensor nodes are randomly dispersed in a square area. These sensor nodes are grouped into clusters and a cluster head is elected for each one. Obviously, sensor nodes are connected to their cluster head in order to report the detected information to it. The cluster head has the role of storing this information and waiting for the mobile sink. A moving strategy for the mobile sink is proposed to collect the information detected over the whole area while minimizing energy consumption. The mobile sink starts from a fixed point, follows a specific trajectory to visit each cluster head and gathers information, and finally it returns to its starting point. Intermittent connectivity is provided using a mobile sink communicating with the cluster heads and coverage is maximized.

Ferries: a ferry is a mobile robot that has a geometrical trajectory like a line or circle. Sensor nodes can be randomly deployed with no connectivity with the sink. The ferry will act as a relay between sensor nodes and the sink to ensure communication, distribution and gathering of the data collected by the nodes. Based on this principle, (35) studies the ferry trajectory that may be a line, path (multiple) or annular, as depicted in Figure 3.10.



a Line ferry. b Path ferry c Two annular ferries.

Figure 3.10: Different Ferry trajectories.

Its goal is to optimize the route of the ferries that collect information from the sensor nodes.

3.3.4 Summary

Area coverage has been widely studied in the literature. Table 3.1, presents a summary of different studies that focus on area coverage. We have distinguished three deployment strategies: a force-based strategy, a grid-based strategy and a computational geometry-based strategy. Based on the studies cited previously, we can observe that force-based strategies exhibit many advantages:

- The simplicity of the basic principle, which performs well both in centralized and distributed versions. In the distributed version, all the nodes apply the same algorithm and play the same role. The distributed version is based only on local information (coordinates of the nodes and their neighbors). It allows nodes to progressively discover their environment and react to changes in this environment without the need for a central entity to manage these changes.

- The uniformity of the redeployment obtained: the density obtained is nearly the same and the same distance is maintained between the neighboring nodes.

- The coverage obtained is generally very good. However, in the distributed version it is achieved at the expense of nodes moving over large distances. This is due to node oscillations that occur even when maximum coverage has been reached. Such oscillations cause high energy consumption and are detrimental to the network lifetime.

- With the enhancements brought by many authors ((4), (18) and (19) for instance), maximum coverage is achieved more quickly.

Nevertheless, some issues remain unsolved, such as the previously mentioned node oscillations and the detection of the end of the distributed algorithm.

The grid-based strategy has the following advantages:

- It provides a regular deployment with deterministic positions of sensor nodes (e.g. a triangular lattice, square pattern, etc), if a virtual grid is used.
- It requires a minimum number of sensor nodes to achieve the required coverage. The optimal deployment pattern (i.e. the pattern requiring the minimum number of sensor nodes) varies according to the relationship between R and r.
- It can easily achieve k-coverage and connectivity.
- It exists in centralized and distributed versions.

Generally speaking, the distributed version is more complex. If a virtual grid is not used, a sophisticated management of grid cells is needed ((24),(25)). The complexity of this strategy comes from managing the movement of the nodes and the positions of newly built cells. Coverage holes may appear.

Computational geometry-based strategies aim at improving the area coverage by healing previously detected coverage holes. Like the other strategies, computational geometry based strategies exist in centralized and distributed versions. The main drawback lies in the complexity of detecting coverage holes and computing the new nodes' positions. Furthermore, the new deployment obtained is not always uniform.

In addition, all these strategies have been enhanced to deal with the existence of obstacles within the network area. A better adaptability to the environment is still a challenge.

There are two types of wireless sensor networks, depending on the mobility of sensor nodes. If the sensor nodes are mobile, all the redeployment strategies (virtual forces strategy, grid based strategy and computational geometry strategy) can be considered as autonomous deployment. Otherwise, sensor nodes are static and mobile robots are used to put the sensor nodes in their final position. In this case the redeployment is said to be assisted.

3.4 Barrier coverage and connectivity algorithms

Intruder detection and border monitoring are two important applications of WSNs. Barrier coverage is considered to be an appropriate model for such applications. A deployment of sensor nodes along a barrier is necessary to detect an intruder crossing, for example, an international border, or a protected industrial area. Depending on the application requirements and the number of sensor nodes provided, this deployment can ensure either full barrier coverage or partial barrier coverage.

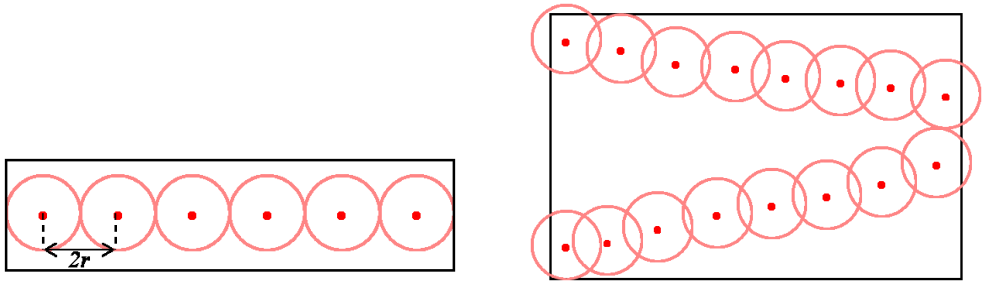
3.4.1 Full barrier coverage

Full barrier coverage can be either simple or multiple. It is simple, if there is just one barrier that is fully covered by sensor nodes. The barrier coverage is multiple if there are

Table 3.1: Area coverage.

Area coverage					
Protocol	**Coverage problem**	**Connectivity problem**	**Strategy**	**Cent/Dist**	**Specific assumptions**
VFA (17)	Full coverage	Permanent connectivity	Forces based	Centralized	
Extended VFA(18)	Full coverage	Permanent connectivity	Forces based	Distributed	$\frac{R}{r} > 2.5$ and $\frac{R}{r} < 2.5$
IVFA (19)	Full coverage	Permanent connectivity	Forces based	Distributed	
EVFA (19)	Full coverage	Permanent connectivity	Forces based	Distributed	
DVFA (4) (36)	Full and uniform coverage	Permanent connectivity	Forces based	Distributed	$R \geq \sqrt{3}r$
CPVF (6)	Maximized coverage	Permanent connectivity	Forces based	Distributed	arbitrary R and r
Push&Pull (24)	Maximized coverage	Permanent connectivity	Forces based	Distributed	Triangular lattice
VFCSO(37)	Full coverage	Full connectivity	Forces based Grid based	Centralized	Square grid $R \geq \sqrt{5}r$ Node activity scheduling
(20)	Full coverage Multiple	Permanent connectivity Multiple	Grid based	Distributed	
(21)	Full coverage Multiple	Permanent connectivity Multiple	Grid based	Distributed	
HGSDA (23)	Full coverage	Permanent connectivity	Grid based	Centralized	Triangular lattice $R \geq \sqrt{3}r$
C^2 (25)	Full coverage	Permanent connectivity	Grid based	Distributed	Triangular lattice Energy saving
(26)	Full coverage	Permanent connectivity	Grid based	Distributed	Square pattern
(27)	Full coverage	Permanent connectivity	Grid based	Distributed	Square pattern Static node Assisted by robot
(33)	Partial coverage	Permanent connectivity	Grid based	Cent/Dist	
VEC, VOR and Minimax (28)	Maximized coverage	Permanent connectivity	Computational geometry based	Distributed	Voronoi diagram
(29)	Full coverage	Permanent connectivity	Computational geometry based	Centralized	Delaunay triangulation Obstacles
(30)	Full coverage	Permanent connectivity	Computational geometry based	Centralized	Static nodes
(34)	Full coverage	Intermittent connectivity	Random	Centralized	Robot collector Cluster head Energy saving
(35)	Full coverage	Intermittent connectivity	Random	Centralized	Ferries
(38)	Full coverage	Permanent connectivity	Random	Distributed	Node activity scheduling
(39)	Full coverage	Permanent connectivity	Random	Centralized	Node activity scheduling Connected graph based
(2)	Full coverage Simple-Multiple	Permanent connectivity	Random	Distributed	$R \geq 2r$ Node activity scheduling
(40)	Full coverage	Permanent connectivity	Random	Distributed	Arbitrary R and r Node activity scheduling
(41)	Partial coverage	Permanent connectivity	Random	Dist/Cent	Node activity scheduling

k successive barriers of sensor nodes.
The authors in (42) were the first to address the problem of providing the minimum number of deployed sensor nodes to ensure simple or multiple barrier coverage. They define a simple barrier coverage by a belt of successive sensor nodes such that their sensing areas overlap. Multiple barrier coverage is defined by the fact that every two successive barriers have two overlapping sensor nodes, as depicted in Figure 3.11b. Based on a theoretical study, the authors prove that the optimal number of sensor nodes deployed along a barrier is $\frac{l}{2r}$, where l is the length of the barrier and r the sensing range. Then, every two successive sensor nodes are at a distance of $2r$ in order to optimize the overlapping (see Figure 3.11a). To ensure full barrier coverage, two types of deployment algorithms can be used, depending on whether the sensor nodes are static or mobile.



a Optimal 1-barrier coverage. b The above zone is 2-barrier covered.

Figure 3.11: Barrier coverage.

3.4.1.1 Static sensor nodes

When sensor nodes are static, they are generally deployed uniformly over the whole area based on a Poisson Point Process model. Using this kind of deployment, barrier coverage can be provided by selecting a chain of overlapping sensor nodes. However, when static sensor nodes are dropped from an aircraft, they will deviate from their expected location due to mechanical inaccuracy or environmental factors such as wind, terrain characteristics, etc. To cope with this problem, (43) proposes a concentrated deployment of sensor nodes along the deployment line with some random offsets, using for example aircraft (see Figure 3.12). This distribution is called LNRO, Line based Normal Random Offset distribution, and in terms of barrier coverage, it outperforms the Poisson model when the random offset in LNRO is relatively small compared to r.

3.4.1.2 Mobile sensor nodes

A deployment strategy to ensure (simple or multiple) barrier coverage using mobile sensor nodes is proposed in (44). This strategy consists in dividing the area into virtual lines (i.e. barriers) where the number of virtual lines matches the desired robustness of barrier coverage. In each line, sensor nodes should occupy grid points at a distance of $2r$. Starting from a random deployment in a rectangular area, mobile sensor nodes should execute two

phases to reach their final positions. In the first phase, each sensor node moves vertically to reach a line. Then, in the second phase, it moves horizontally along the line to a predetermined grid point position. When each grid point is occupied by a sensor node, full barrier coverage is provided.

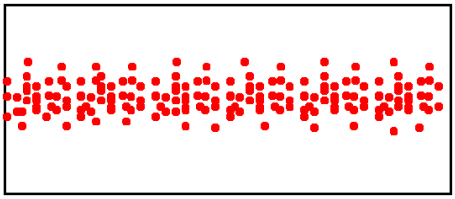
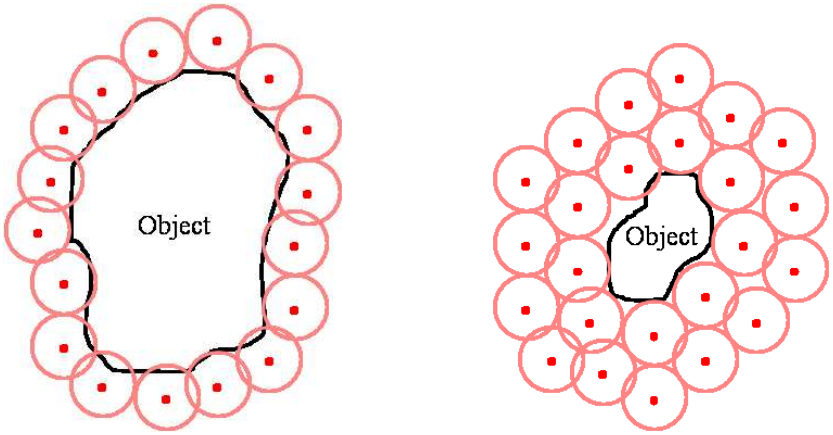


Figure 3.12: LNRO barrier deployment.

(45) focuses on finding and healing barrier holes using mobile sensor nodes. This work is an extension of (43). After the deployment, sensor nodes may fail due to many factors, such as battery depletion, environmental conditions or malfunctioning. Then, a redeployment is needed to heal coverage holes. The algorithm proposed proceeds in two phases. In the first phase, it scans the network from the beginning to the end of the barrier to check for coverage holes. The second phase consists in computing which sensor nodes should move to which position such that the total distance traveled by the nodes is minimized. This algorithm takes advantage of the LNRO distribution as all the sensor nodes are concentrated along a line, as depicted in Figure 3.12, allowing quick and easy replacement of failed nodes.



a Big object: 1-barrier coverage.

b Small object: 2-barrier coverage.

Figure 3.13: Dynamic object.

The monitored object may be dynamic, (i.e. changing its shape). As a consequence, sensor nodes have to move to modify the belts they form around the object to be monitored. In (46) the problem of mobile barrier for dynamic coverage is formulated as: for a given

number n of sensor nodes, how do sensor nodes move with the objective to minimize the total distance traveled under the constraint that the number of barriers is maximized at any time ? Sensor nodes are placed around the dynamic object, neighboring sensors are at a distance less than or equal to $2r$ forming a belt around the dynamic object without any coverage holes. The authors assume that $R \geq 2r$, in order to ensure full connectivity. A dynamic belt region provides k-mobile barrier coverage if and only if there are k vertex disjoint belts in its coverage graph. The maximum number of barriers k changes in response to changes in the dynamic object: k becomes smaller when the dynamic object becomes larger, as illustrated in Figure 3.13.

3.4.2 Partial barrier coverage

In the barrier coverage problem, the optimal number of nodes (denoted m points) required to fully cover the barrier, can be determined based on the sensors' sensing range and the barrier length. However, if the number of available nodes is less than optimal, the barrier coverage problem will be formulated as how to move n mobile sensor nodes to monitor n points among the m points so as to maximize the average intruder detection while minimizing the average sensor movement distance. To solve this problem, two algorithms, PMS and CSP, are proposed in (47). PMS, periodic monitoring scheduling, lets sensor nodes monitor each point of the barrier periodically, regardless of any arrival by an intruder and without any coordination between sensors. Each sensor moves to the point j and stays there for T time slots. Then, it moves to point $mod(j + n, m)$ and stays there, also for T time slots. This is repeated until all the sensors run out of energy. CSP, Coordinated Sensor Patrolling, is a centralized algorithm that uses the temporal correlation of intruder arrival times. CSP runs in two steps. Firstly, it selects the point with the highest priority of intruder arrival to be monitored at the current time. Then, it determines how to move sensors to the selected point while minimizing the total distance traveled, using the information collected in the past time slot. It has been shown that the CSP algorithm outperforms PMS.

3.4.3 Summary

The barrier coverage problem generally relates to critical applications such as intruder detection which require special attention. A high degree of robustness (multiple barrier coverage) is normally chosen for critical applications to ensure the efficiency and reliability of the monitoring task.
Furthermore, the zone monitored, such as a battlefield or international borders very often includes obstacles and is not always flat in such applications, and many environment constraints may be involved. Obstacles can also occur in the monitoring barrier. The solutions proposed in the literature do not take into account these constraints which have a negative impact on the deployment algorithm.
The issue of connectivity is very important in critical applications since it allows information to be reported to the sink. All the papers cited in this section, assume that connectivity between neighboring nodes and with the sink is ensured: $R \geq 2r$. However,

Table 3.2: Barrier coverage.

Protocol	Coverage problem	Connectivity problem	Strategy	Cent/Dist	Specific assumptions
(44)	Full coverage Simple-Multiple	Permanent connectivity	Grid based	Distributed	Mobile sensors
(43)	Full coverage	Permanent connectivity	Random	Centralized	Random offset $< r$
(45)	Full coverage	Permanent connectivity	Random	Centralized	
MBC (46)	Full coverage Simple-Multiple	Permanent connectivity	Deterministic	Distributed	$R \geq 2r$ Dynamic object
CSP (47)	Partial coverage	Intermittent connectivity	Probabilistic	Centralized	
PMS (47)	Partial coverage	Intermittent connectivity	Probabilistic	Centralized	
(48)	Full coverage	Permanent connectivity	Random	Centralized	Node activity scheduling

in real deployments, this condition is not always met. Therefore, strategies to ensure connectivity should be provided.
Sensor nodes may be dropped randomly, trying to follow a barrier line (e.g. (43)). In this case, coverage can be improved by a centralized algorithm, as in (45) in charge of detecting and healing holes in barrier coverage. However, when coverage holes are present, the central entity may fail to collect all the sensor nodes' positions since these holes may produce disconnected components.
Table 3.2 presents different studies that focus on barrier coverage.

3.5 Point coverage and connectivity algorithms

The last type of coverage is the coverage of Points of Interest (PoI). Examples of applications include the detection of some static or moving target, using the smallest number of sensors. We distinguish between static PoIs and dynamic PoIs.

3.5.1 Static PoI

In (49), the authors are interested in the deployment of mobile sensors to cover predefined PoIs, while maintaining connectivity with the sink. The sink has the task of disseminating information about the PoI locations to the sensors as well as collecting the information reported from the sensors about the events happening at the PoI. The basic idea of this deployment algorithm for PoI coverage is as follows: initially all the sensors are within radio range of the sink. All the sensors run the same algorithm but the motion decision is taken individually by each sensor node. The sensors move toward one predefined point that could be the PoI or the barycenter of the PoIs. Then, they form straight lines between the PoI and the sink. The distance the sensors move is bounded in order to maintain connectivity. When all the sensors are in positions, the PoI is covered by one sensor in the line (i.e. the PoI is within the sensing range of a sensor in the line). The strategy of this deployment algorithm minimizes the number of sensors used to maintain connectivity by using the Relative Neighborhood Graph (RNG).

If multiple PoIs exist in the area considered, two approaches can be adopted:

- Random PoI deployment: the sensor chooses one of the PoIs at random;

- Barycenter PoI deployment: Every sensor calculates the barycenter of all the PoIs and the sink to cover it. Then each sensor chooses a PoI at random and covers it.

In (50), a distributed deployment scheme is proposed where mobile sensors nodes move following concentric circular paths (ferries with annular trajectories) that cover static PoIs (See Figure 3.14). The goal of this work is to ensure PoI coverage and that the events are reported to the sink. This sink is located at the barycenter. Two neighboring circular paths are at a distance of R. The authors assume that $R \geq 2r$ and mobile sensors have no global knowledge of the PoIs in the area. This work combines three aspects which are: PoI discovery, PoI coverage and connectivity with the sink. To achieve these three aspects, a mobile sensor should move constantly to execute the PoI discovery task. Then, it should adjust its movement velocity with sensors in the neighboring circular paths to satisfy the constraints regarding coverage and connectivity with the sink in order to be able to report the information it has gathered about the PoIs.

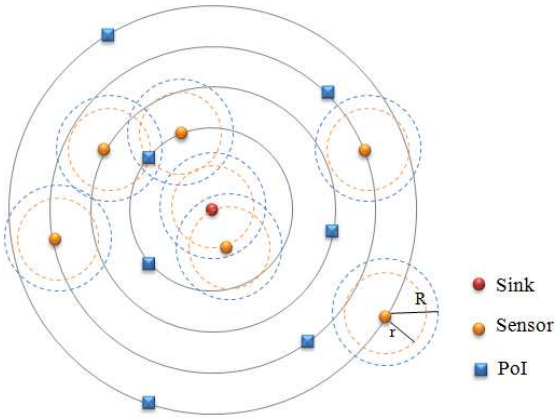
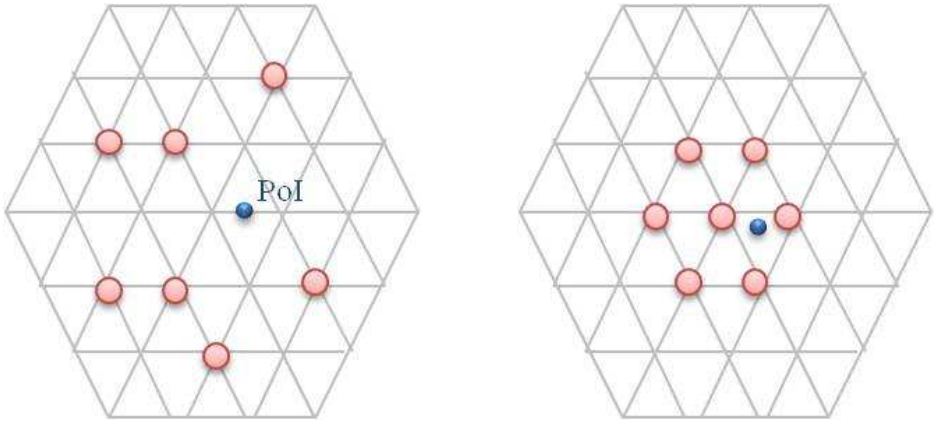


Figure 3.14: PoI coverage using annular ferries.

Temporary coverage of Multiple PoIs is studied in (51) and is called the sweep coverage problem as sensor nodes sweep between PoIs and cover them periodically. The authors propose distributed algorithm DSWEEP to address this problem. A sensor node covers a PoI for a determined duration and then moves on to a new one. When a sensor node is moving, it encounters other sensor nodes and exchanges information that serves to decide which PoI should be monitored next. This deployment algorithm requires a small number of sensor nodes to cover a large number of PoIs. DSWEEP provides temporary coverage and partial network connectivity.
In some applications, the PoI, as well as the area surrounding it, need to be covered. In (52) a localized autonomous deployment algorithm is proposed to meet this goal. This algorithm is based on a virtual triangular lattice grid of edge $\sqrt{3}r$ to maintain connectivity since it is assumed that $R \geq \sqrt{3}r$. Sensor nodes are autonomous and know the position of the PoI. They move through the triangular vertexes and organize themselves by respecting rules that avoid collisions between sensors, to reach the vertices around the PoI. Based on

this principle no coverage holes will occur if all the vertexes around the PoI are occupied by sensor nodes.



a Initial deployment. b Final deployment.

Figure 3.15: PoI coverage using Grid.

3.5.2 Mobile PoIs

In the case of mobile PoIs, the authors of (49) propose three strategies to reach the mobile PoI:

- In the first strategy, sensor nodes move back to the sink before deploying toward the new location of the PoI. This strategy provides a high coverage quality but increases the deployment duration and the amount of energy consumed.
- In the second strategy, sensors try to move directly toward the new location of the PoI without going back to the sink. This strategy reduces the time needed to cover the new PoI but also reduces the coverage quality as it requires a greater number of sensors to maintain connectivity.
- In the third strategy, a sensor moves to the straight line between the sink and the new location of PoI, then it follows the line toward the PoI. This strategy provides a higher coverage quality and reduces the time needed to cover the PoI.

3.5.3 Summary

Any PoI needs only one sensor to be covered. If permanent connectivity is required, a sufficient number of sensor nodes are deployed to ensure connectivity with the sink. However, if intermittent connectivity is sufficient, one sensor node will cover a PoI, and a mobile node (that could be the sink or a collector Robot) will operate like a data mule. This can be a solution to deploy a minimum number of sensor nodes and save energy.
When the PoI is static, a static sensor node can be used to cover it. If the PoI is mobile,

Table 3.3: Point of Interest coverage.

Protocol	Coverage problem	Connectivity problem	Strategy	Cent/Dist	Specific assumptions
(49)	Full coverage	Permanent connectivity	Forces based	Distributed	RNG for connectivity
(50)	Temporary coverage	Intermittent connectivity	Random	Distributed	Static PoI Ferries $R \geq 2r$
DSWEEP (51)	Temporary coverage	Intermittent connectivity		Distributed	
(52)	Full coverage	Permanent Connectivity	Grid based	Distributed	$R \geq \sqrt{3}r$

however, autonomous sensor nodes are deployed to track the PoI and avoid the need of a robot to pick up and deploy sensor nodes each time the position of the PoI changes.
Although any PoI can be covered by just one sensor, a zone of interest may require several sensors when the zone is larger than the disk covered by a sensor. When many sensor nodes are deployed to cover a zone (area) of interest, they are usually deployed in varying densities: a high density in the center of the zone of interest and then the density decreases with the distance to the center of the zone.
Table 3.3 presents different studies that focus on PoIs coverage.

3.6 Node activity scheduling with regard to coverage

Assuming an initial deployment of static sensor nodes which meets the application requirements (e.g. full or partial coverage), the node activity scheduling problem consists in determining a connected set of active nodes to ensure the application requirement. Only the nodes in this set are active, the other nodes are in sleep state in order to save energy, thereby maximizing the network lifetime. The problem here is not to find the appropriate sensor node positions but only to select which sensor nodes will be active to maximize coverage and connectivity. Figure 3.16 depicts an example where the blue sensor nodes are sleeping, while coverage and connectivity are ensured by the active sensor nodes in red.

We distinguish two categories of node activity scheduling with regard to coverage:

3.6.1 Node activity scheduling based on message exchanges between neighbors

Sensor nodes rely on message exchanges to decide which sensor nodes should be in an active state while others are sleeping, with the goal of ensuring full coverage and saving energy. This mechanism can either be centralized, where a central entity collects all the nodes' positions and assigns a state (active or sleep) to each node, or distributed, where neighboring sensor nodes exchange messages to decide which of them will be active while the others are sleeping.
An example of a centralized algorithm is given in (39). This work is based on the construction of a connected subgraph of sensor nodes based on local information. It focuses on

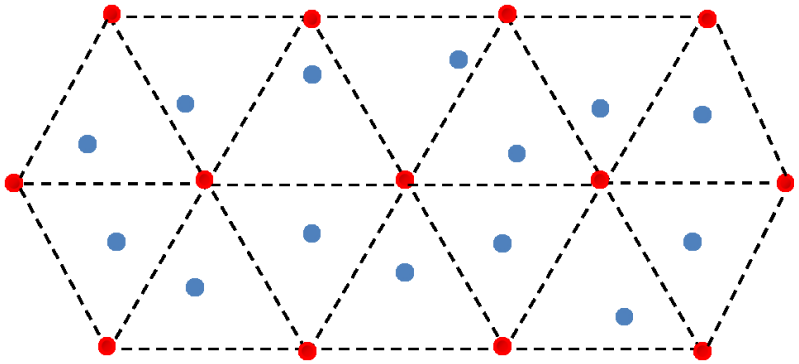


Figure 3.16: Node Activity Scheduling.

finding the smallest subset of sensor nodes that ensures full coverage of the area monitored while maintaining connectivity with the sink.
Another centralized algorithm is proposed in (48) to build a camera barrier from an initial arbitrary deployment of camera sensors. The aim is to guarantee that each point of the barrier is fully covered visually. The method consists in building a graph of nodes where each of them covers a small subregion, and each two adjacent nodes are connected. The idea is to select a path from one boundary to another such that the nodes of the path are full-view covered. Only nodes belonging to that path are active.
Node activity scheduling based on message exchanges is also adopted to ensure partial coverage. In (41), a centralized algorithm is proposed to ensure partial coverage. It aims to select the smallest number of nodes to monitor p-percent of the area. The authors also propose a distributed algorithm that determines a set of nodes to cover p-percent of the considered area. The main idea of these two algorithms is to divide the whole area into sub-regions and select specific nodes, while respecting certain criteria (for example, a starter node selects its furthest neighbor) in order to cover p-percent of each sub-region.
CCP, Coverage Configuration Protocol (2) is a distributed algorithm based on message exchanges to provide the degree of coverage required by applications when $R \geq 2r$. In CCP, depending on information about its sensing neighbors, a sensor node can be in a sleep state to save energy, a listen state to collect neighboring messages and decide its new state, or an active state to sense the environment. Without assuming $R \geq 2r$, CCP cannot guarantee network connectivity. In (2), CCP is combined with SPAN (53) to achieve both coverage and connectivity when $R < 2r$. SPAN is a connectivity maintenance protocol. This protocol connects all active nodes via a communication backbone, and connects inactive nodes to at least one active node. Then, when $R < 2r$, network connectivity is ensured.
Several other distributed protocols are proposed in (54), to ensure area coverage with a low communication overhead. In these protocols sensor nodes select a waiting time for each round and receive neighboring messages which are used to compute the area coverage. If the sensing area of a sensor node is not fully covered, the node should stay in an active state during the current round and announce its state when its waiting time expires.

Sensing range and radio range may be different and they may also differ between sensor nodes. The authors in (40) adopt this assumption and aim to minimize the number of active nodes queried in the region that is fully covered. Then, each sensor node should determine whether it switches to an active state to respond to the query request originating from the sink, based on information collected from its neighbors.

3.6.2 Node activity scheduling based on implicit coordination

Implicit coordination algorithms are proposed to save the energy of sensor nodes, assuming full coverage and connectivity. Such algorithms are distributed and based on a grid. Each node knows from its position in the grid whether it must be active or it can sleep. An example is given in (38) for a square pattern and a hexagonal pattern: each sensor node located in the vertex of the grid switches to the active state, while the other nodes sleep. Another example of a square pattern is given in by VFCSO, Virtual Force-Based Coverage Optimization Strategy (37). VFCSO is a dynamic deployment algorithm that aims at ensuring full area coverage using a minimum number of sensor nodes while reducing energy consumption. In this work, the area considered is divided into square cells with edges equal to r. Many sensor nodes may be in the same square cell. Starting from a random deployment, the virtual forces strategy is applied by sensor nodes belonging to the same square cell. Only one node in each cell will be active, the others should switch to the sleep state: the active node being the closest sensor node to the center of the cell with the highest residual energy. Both full coverage and network connectivity are guaranteed in this work as $R \geq \sqrt{5}r$.

3.7 Guidelines for selecting a deployment algorithm

In this section we set out guidelines we used in our scientific approach presented in the following chapters. These guidelines may in general help the designers to select a deployment algorithm that meets their application requirements.
We consider two main questions:

- What does the application require in terms of coverage and connectivity?
- Which assumptions and constraints are given?

In the following we discuss various ways to answer these questions.
► **Definition of the coverage and connectivity problem that must be solved:**

- *Coverage*
  – If the goal is to monitor an area, then the problem concerns area coverage, which may be full or partial.
  If it is to detect barrier crossing, the problem is barrier coverage, which, again, may be full or partial.
  If the goal is to track/monitor a target, the problem deals with point of interest (PoI) coverage. The PoI may be static or mobile.
  – If coverage must be full and the degree of robustness required by the application

is high, multiple coverage is needed, otherwise simple coverage is sufficient.
– If long delays to detect an event are tolerated by the application, the coverage of any point can be temporary. Otherwise, it is permanent.

- *Connectivity*
  – If short delays to report detected events to the sink are required by the application, permanent connectivity must be ensured. Otherwise, intermittent connectivity is sufficient.
  – If the application needs a high degree of robustness, multiple paths to the sink should be maintained. Otherwise, a simple path is sufficient.

- *Type of deployment*
  – depending on the application requirements, a uniform and regular deployment should be provided, based on a pattern (see Section 2.4).

► **Assumptions and constraints**

In most cases, the designer will be faced with multiple assumptions and constraints that must be taken into account when selecting the appropriate deployment algorithm. These include:

- *Environment*
  – The dimensions and position of the area, barrier or PoI to cover should be provided in order to compute the minimum number of sensor nodes required. If this number is large, the deployment algorithm must be scalable. The initial topology influences the deployment algorithm, especially when some sensor nodes are disconnected, or when they are all grouped together at an entry point (see the discussion in Section 3.2.1).
  – The choice of the radio propagation model must be compliant with the environment (e.g. free space or confined) which may suffer from perturbations caused by other wireless networks (e.g. WiFi) or electronic devices (e.g. microwaves), and may also contain obstacles.
  – In the presence of obstacles, detection and bypassing techniques should be provided.

- *Sensor nodes*
  – Mobility: sensor nodes may be mobile and autonomous, and this condition is necessary for autonomous-deployment. On the other hand, static nodes should be assisted in their deployment by a mobile robot.
  – The sensing range r, the communication range R and the associated models: for more details see Section 3.2.2. Furthermore, the relationship between r and R will be used to select the appropriate deployment algorithms in Table 2.2.
  – The number of sensor nodes must be sufficient to meet the application requirements, otherwise the problem is intractable.
  – Energy: if sensor nodes are equipped with a battery, the deployment algorithm must be energy efficient.

- *The sink*
  The sink is in charge of collecting the data generated by the sensor nodes deployed. It can be static or mobile. If the sink is static, either it is connected to sensor nodes, or a mobile robot visits the disconnected sensor nodes to collect their data and report them back to the sink. If the sink is mobile, it moves to collect data.

► **Recommendations:**

- *Coverage problem*
  Depending on the application needs, the problem may be an area, barrier or PoI (Point of interest) coverage problem.

- *Relationship between the transmission range R and the sensing range r*
  The relationship between R and r influences the choice of the solution. If, for instance, $R \geq \sqrt{3}r$, it is sufficient to solve the coverage problem to obtain connectivity as a consequence of coverage. If the transmission range R is strictly less than the sensing range r, a distributed deployment would require a smaller target distance between sensor nodes than that required by full coverage of the area. Hence, a higher number of sensor nodes will be used, leading to a more expensive solution. If the designer has a small budget, he/she will prefer a centralized solution with a mobile robot/agent to deploy the sensor nodes to their final position, and to collect data from these nodes in the data gathering phase. Similarly, such a solution is also preferred when the application tolerates delays (e.g. delay tolerant networks, ferries). In contrast, if a permanent path must exist from any sensor node to the sink, additional sensor nodes will be required to ensure this permanent connectivity.

- *Centralized versus Distributed solution*
  Depending on the area/barrier size, a centralized/distributed solution will be preferable. If the monitoring requires a high number of sensors, a distributed solution is chosen because of its better scalability, provided that the energy constraints are taken into account, as discussed below. A centralized solution requires that the central entity in charge of the deployment computation has perfect knowledge of the positions of all the sensor nodes. If the initial topology is disconnected, a mobile robot is needed to collect the initial positions of all the disconnected nodes to compute the final deployment. If all the sensor nodes are static, the centralized solution is the only possible one. A mobile robot is needed to deploy the sensor nodes to their final positions.

- *Energy constraints*
  When sensor nodes are equipped with a battery, energy efficient techniques should be used, and it is important that the scheduling allows nodes to sleep, for energy saving purposes. Another advantage of node activity scheduling is to make the deployment adaptive to varying coverage requirements, ranging from full to partial. However, the energy consumed by nodes movements is considerable and should be limited. For instance, node oscillations occurring in some distributed solutions should be avoided.

If the designer wants to keep the energy of sensor nodes for data gathering, a mobile robot/agent should be used to deploy the sensor nodes to their final positions.

- *Uniform and regular deployment*
  A uniform and regular deployment reduces the energy consumed during the data gathering phase and minimizes the data gathering delay. Moreover, it provides better time and space consistency of the measures reported to the sink.

- *Obstacles*
  An area/barrier with obstacles needs mechanisms to detect obstacles and strategies to bypass them, as well as ensuring the required coverage.

3.8 Conclusion

In this chapter, we focused on deployment algorithms dealing with coverage and connectivity in WSNs. We provided indications for analyzing deployment algorithms and evaluating their performances. We distinguish two types of deployment algorithms depending on the mobility of the sensor nodes: autonomous deployment for mobile sensor nodes and assisted deployment for static sensor nodes deployed by mobile robots. Deployment algorithms are designed to meet application requirements such as coverage, connectivity, latency and robustness. We established a classification of deployment algorithms based on these requirements. In fact, the deployment of sensor nodes and sinks can be considered as the first step in the design of a data gathering application. As a second step, node activity scheduling is used to optimize energy consumption by switching off redundant nodes to maximize the network lifetime, while ensuring the coverage and connectivity required by the application. Finally, we gave some guidelines on selecting the deployment algorithms that are best-suited to the application requirements. With regard to these guidelines, we present in Table 3.4 the constraints and assumptions considered in our deployment algorithms proposed in the next chapters.

Table 3.4: Constraints and assumptions considered in our deployment algorithms

	Area	**PoIs**
Coverage	Full; Simple	Full; Simple
Connectivity	Full; Simple	Full; Multiple
R versus r	$R \geq \sqrt{3}r$	$R \geq \sqrt{3}r$
Deployment algorithm	Centralized; Distributed	Centralized
Type of deployment	Based on the optimal deployment (Triangular tessellation)	
Energy	- optimizes deployment duration - avoids node oscillations - turns off redundant nodes	- optimizes deployment duration minimize the length of the path betwwen each PoI and the sink
Sensor nodes	Mobile; Static	Static
Obstacles	Known; unknown	Known
Strategy	Virtual Forces; Grid based	Grid based

Part II

Models and theoretical computation for an optimized deployment in 2D and 3D

Chapter 4

Models for an optimized deployment

Contents

4.1 Introduction

In this chapter we present the sensing model and the radio transmission model which we adopted in our work for both 2D and 3D problems. We also present the models of the entity to be monitored and obstacles adopted for 2D and 3D deployments.

4.2 Models for 2D deployment

4.2.1 Sensing range and communication range in 2D

In a 2D area, the wireless sensors are all assumed to have the same sensing range denoted r and the same radio range R. The sensing model and the radio transmission model are the classical disk see Figure 4.1. For the sake of simplicity, we also assume that the condition $R \geq \sqrt{3}r$ is met. This condition guarantees that any deployment of wireless sensor nodes ensuring full coverage also ensures full connectivity.

4.2.2 The area considered and obstacles in 2D

The area to fully cover is considered as a polygon which may or may not be convex (see Figure 4.3). This polygon is defined by its edges. These edges constitute the borders of the area. We distinguish two types of borders:

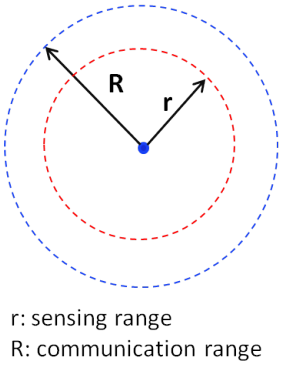


Figure 4.1: Sensing and communication ranges in the disk model

- Transparent border: a transparent border does not prevent the sensing activity of sensor nodes.
- Opaque border: an opaque border prevents nodes from sensing the area located behind the border: a sensor node s can cover a point u within its sensing disk if and only if u is in the line of sight of s (see Figure 4.2).

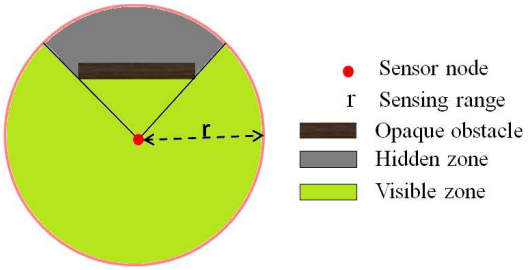
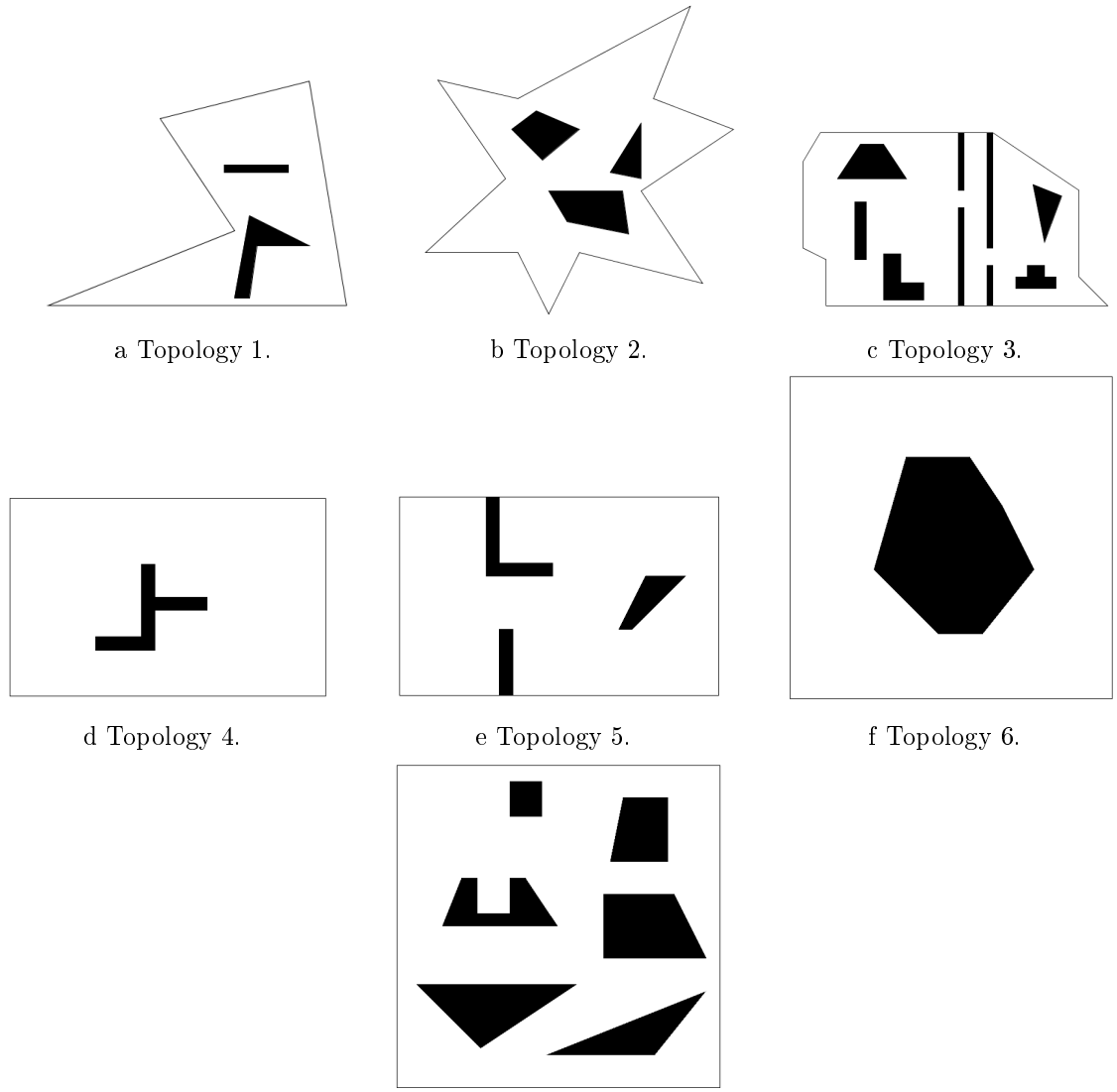


Figure 4.2: Hidden zone caused by an opaque obstacle.

The area considered usually has obstacles, see for instance Figure 4.3 illustrates the different topologies adopted in our studies. No sensor node must be located within an obstacle. As for borders, we distinguish two types of obstacles: transparent and opaque. An obstacle is defined by the edges of its polygon which may be irregular in shape and not convex.

- Transparent obstacles, have no impact on either the sensing range or the communication range of nodes. They only prohibit nodes from crossing them.
- Opaque obstacles, like transparent obstacles block the node from moving. However, they may also cause hidden zones. A hidden zone is an area within the sensing range and the communication range of a sensor node (see Figure 4.2). If it is within the sensing range it prevents an event occurring in this zone from being detected. Otherwise, it prevents communication between nodes at a distance less than or equal to the communication range, if only one of them is in the hidden zone.



a Topology 1. b Topology 2. c Topology 3.

d Topology 4. e Topology 5. f Topology 6.

g Topology 7.

Figure 4.3: 2D area

4.3 Models for 3D deployment

In the literature, most studies use a 2D space to model the area considered. However, in the real environment the area to be monitored is not always flat. Also, in some applications where (the third dimension) the height should not be neglected, 2D monitoring is not sufficient. Applications requiring 3D monitoring include: home automation, precision agriculture (fruit tree plantation, olive groves), environmental monitoring.

4.3.1 Sensing range and communication range in 3D

In 3D, sensing range and communication range are modeled by two spheres of radius r and R respectively (see Figure 4.4).

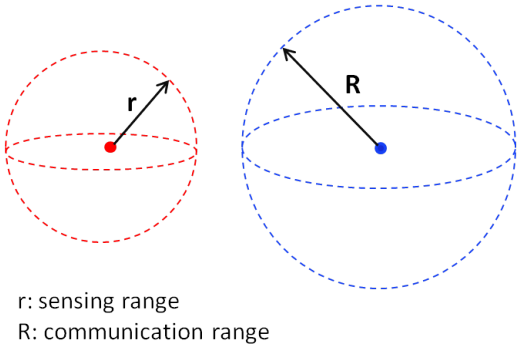
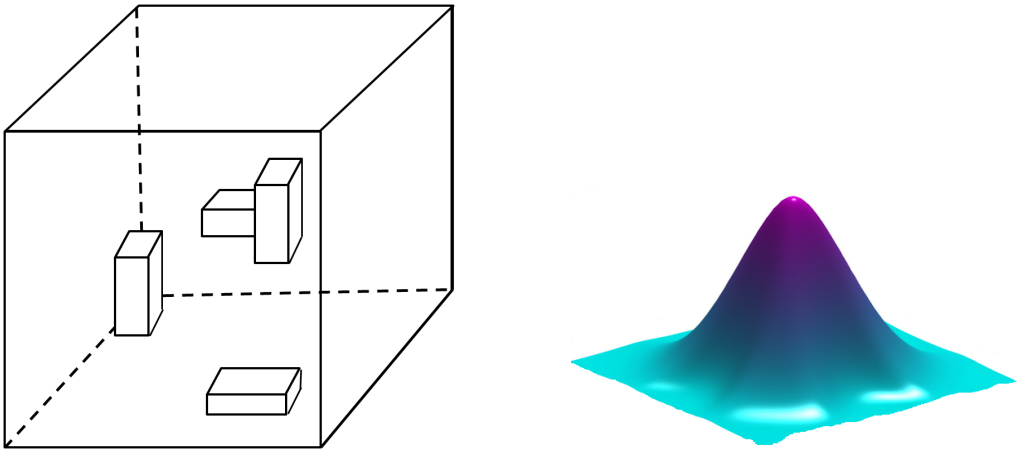


Figure 4.4: Sensing and communication ranges modeled by spheres

4.3.2 The area considered and obstacles in 3D

In a 3D space, the area to cover can be either a polyhedron like that illustrated in Figure 4.5a, or a surface that is not flat (e.g. a mountain) like that illustrated in Figure 4.5b. When the area to cover is a polyhedron, a sensor node can be located at any position inside the polyhedron. However, when the target to cover is a 3D surface, the sensor node should follow the shape of this surface. This problem is also called 3D surface covering. An obstacle is modeled by a polyhedron or a juxtaposition of polyhedra. Obstacles are also present in a 3D space and they may be transparent or opaque, as in a 2D space.



a Topology 1. b Topology 2.

Figure 4.5: 3D area.

4.4 Conclusion

The models defined in this chapter will be used in the following chapters, where we present solutions for autonomous or assisted deployments in a 2D or 3D space.

Chapter 5

Theoretical computation of an optimized deployment in 2D and 3D

Contents

5.1 Introduction

An optimal deployment is a deployment which minimizes the number of sensors used. It is required in many applications where the number of available sensor nodes is limited. It is defined by the exact position of the nodes and their number.

In this chapter, we not only focus on optimal deployment in a 2D area but also optimized deployment in a 3D space. We start by proposing the computation of the optimal positions and number of nodes to cover a rectangle. Then, we show how to compute the optimized number of nodes to cover a cube.

5.2 Theoretical computation of an optimal 2D deployment

As proved in (22), an optimal placement of sensors in a 2D area offering full coverage can be obtained by a triangular lattice, as illustrated in Figure 5.1. If the targeted deployment is the optimal one, each sensor node will have six neighbors at a distance D_{th}. The optimal deployment is obtained with an equilateral triangular lattice, (see Figure 5.1) where each sensor node has 6 neighbors at the same fixed distance D_{th}. Each sensor node occupies a vertex of an equilateral triangle. In Figure 5.1, a circle of radius r around a sensor node denotes its sensing area.

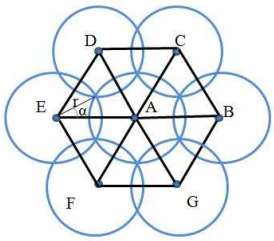


Figure 5.1: Triangular lattice deployment.

Figure 5.2 represents three sensors A, B and C in the optimal deployment. The coverage area of each sensor is represented by a disk of radius r. The centers of these three disks form an equilateral triangle ABC since these sensors are neighbors and are separated by the same distance D_{th}.

5.2.1 Target distance in the optimal deployment

We now compute the target distance D_{th} in the optimal deployment.
Let M be the point of intersection of these three disks. AM is the radius r of the circle whose center is A. Since H is situated in the medium of AC then MH is the mediator of AC.

To compute the value of D_{th}, we consider the angle $H\widehat{A}M$, denoted by α (see Figure 5.2). As $\cos\alpha = \frac{AH}{AM} = \frac{\frac{D_{th}}{2}}{r}$, we can deduce:

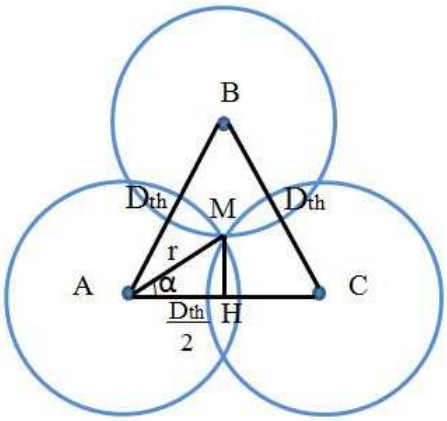


Figure 5.2: Basic pattern in an optimal deployment.

$$D_{th} = 2rcos\alpha \tag{5.1}$$

In the optimal deployment, the angle $C\widehat{A}B$ is equal to $\frac{\pi}{3}$, because of the equilateral triangle. Since α is half of the angle $C\widehat{A}B = \frac{\pi}{3}$, we have $\alpha = \frac{\pi}{6}$.

Consequently,

$$\boxed{D_{th} = \sqrt{3}r \text{ in the optimal deployment}} \tag{5.2}$$

To ensure network connectivity, the communication range R must be higher than the distance separating two neighboring sensors (i.e $R \geq D_{th}$) . Therefore, when the optimal deployment is reached, we have:

$$R \geq \sqrt{3}r$$

Coverage and connectivity are closely related. In fact, if the sensing range r and the transmission range R meet $R \geq \sqrt{3}r$, then it is sufficient to ensure coverage, as connectivity is a consequence.

5.2.2 Optimal number of sensors to cover a given area

To determine the optimal number of sensors required to achieve the full coverage, we consider the optimal deployment illustrated in Figure 5.3 in an area of length L and width W. It is based on an equilateral triangular lattice of edge D_{th} (see triangle ABC in Figure 5.3). Since in the optimal deployment of sensors, the pattern of the first line is reproduced identically at each odd line and similarly the pattern of the second line is reproduced identically at each even line, we compute the number of sensors in odd lines and even lines (see Figure 5.3). We then compute the total number of lines and finally deduce the total number of deployed sensors.

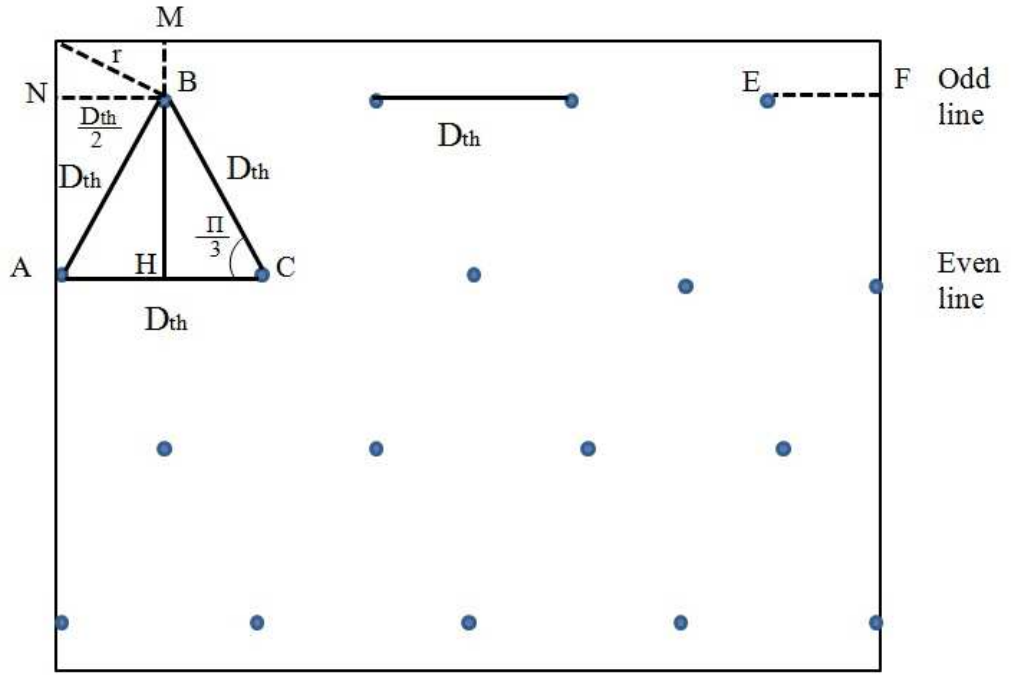


Figure 5.3: Optimal deployment of sensors.

• **Number of sensors in odd lines**

In the first line, and in any odd line, the first sensor is located at a distance $\frac{D_{th}}{2}$ (represented by NB in Figure 5.3) from the left boundary of the area. On a line all sensors are uniformly distributed at a distance of D_{th}. Let $N_{s,o}$ be the number of sensors in odd lines. Let $\delta_{s,o}$ be an integer equal to 0 or 1 computed as follows:

$$N_{s,o} = \lfloor \frac{L - \frac{D_{th}}{2}}{D_{th}} \rfloor + 1 + \delta_{s,o} \tag{5.3}$$

$$with\ \delta_{s,o} = \begin{cases} 1 & if\ L - D_{th} - \lfloor \frac{L - \frac{D_{th}}{2}}{D_{th}} \rfloor D_{th} > 0 \\ 0 & otherwise \end{cases}$$

$\delta_{s,o}$ is equal to 1 when the distance between the last sensor in the line and the right boundary (represented by EF in Figure 5.3) is higher than $\frac{D_{th}}{2}$.

• **Number of sensors in even lines**

In even lines, the first sensor is located at the left boundary of the given area.
Let $N_{s,e}$ be the number of sensors in even lines. We have

$$N_{s,e} = \lfloor \frac{L}{D_{th}} \rfloor + 1 + \delta_{s,e} \tag{5.4}$$

$$with\ \delta_{s,e} = \begin{cases} 1 & if\ L - \frac{D_{th}}{2} - \lfloor \frac{L}{D_{th}} \rfloor D_{th} > 0 \\ 0 & otherwise \end{cases}$$

$\delta_{s,e}$ is equal to 1, if the distance between the last sensor and the right boundary is higher than $\frac{D_{th}}{2}$.

• Number of sensor lines

The first line starts at a distance BM from the top of the area considered (see Figure 5.3). The computation of BM is done in the triangle NBM of Figure 5.3.
$BM^2 + (\frac{D_{th}}{2})^2 = r^2$
As $D_{th} = 2rcos\alpha$, then $BM^2 = r^2(1 - cos^2\alpha)$
And then, $BM = rsin\alpha$.
The distance between lines is represented by BH in Figure 5.3.
$BH = D_{th}sin\frac{\pi}{3} = \frac{\sqrt{3}}{2}D_{th}$. Finally, we get:
$BH = \sqrt{3}rcos\alpha$
Consequently the number of lines denoted by N_l is given by:

$$N_l = \lfloor \frac{W - rsin\alpha}{\sqrt{3}rcos\alpha} \rfloor + 1 + \delta_l \tag{5.5}$$

$$with\ \delta_l = \begin{cases} 1 & if\ W - 2rsin\alpha - \lfloor \frac{W - rsin\alpha}{\sqrt{3}rcos\alpha} \rfloor \sqrt{3}rcos\alpha > 0 \\ 0 & otherwise \end{cases}$$

• Number of sensors

The total number of sensors in a given area is the sum of the total number of sensors in odd lines and the total number of sensors in even lines denoted by N_{opt} is:

$$N_{opt} = \lfloor \frac{N_l}{2} \rfloor N_{s,e} + \lceil \frac{N_l}{2} \rceil N_{s,o}$$

$$N_{opt} = \lfloor \frac{\lfloor \frac{W - rsin\alpha}{\sqrt{3}rcos\alpha} \rfloor + 1 + \delta_l}{2} \rfloor (\lfloor \frac{L}{D_{th}} \rfloor + 1 + \delta_{s,e})$$

$$+ \lceil \frac{\lfloor \frac{W - rsin\alpha}{\sqrt{3}rcos\alpha} \rfloor + 1 + \delta_l}{2} \rceil (\lfloor \frac{L - \frac{D_{th}}{2}}{D_{th}} \rfloor + 1 + \delta_{s,o}) \tag{5.6}$$

5.2.3 Computation of the effective distance

We now assume that N, the number of operational sensor nodes, is given with $N \geq N_{opt}$. Our goal is now to obtain a uniform redeployment in a given area $L * W$, using all the N sensors. This uniform redeployment is also based on a triangular lattice, where any node

is at a distance D_{eff} from its adjacent neighbors.
According to Equation 5.6, we get:

$$N = \lfloor \frac{\lfloor \frac{W - r\sin\alpha}{\sqrt{3}r\cos\alpha} \rfloor + 1 + \delta_l}{2} \rfloor (\lfloor \frac{L}{D_{eff}} \rfloor + 1 + \delta_{s,e})$$

$$+ \lceil \frac{\lfloor \frac{W - r\sin\alpha}{\sqrt{3}r\cos\alpha} \rfloor + 1 + \delta_l}{2} \rceil (\lfloor \frac{L - \frac{D_{eff}}{2}}{D_{eff}} \rfloor + 1 + \delta_{s,o}) \quad (5.7)$$

In this work, we use the mathematical software Maple to solve Equation 5.7. Knowing the size of the area considered, we deduce the value of D_{eff} while varying N, the number of operational nodes.

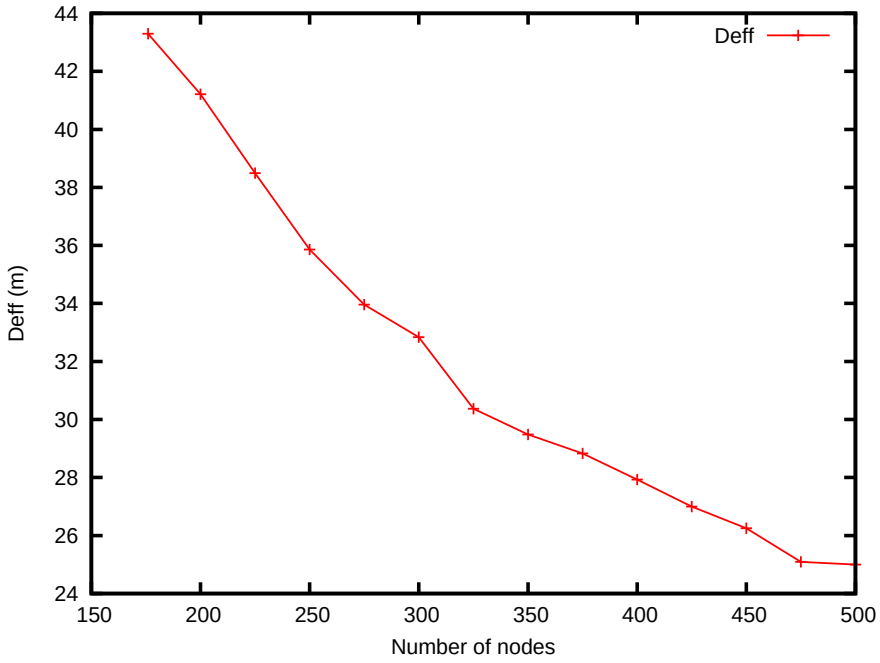


Figure 5.4: The effective optimal distance.

Figure 5.4 depicts the value of D_{eff} for a $500m$x$500m$ area and a sensing range $r = 25m$. The optimal value D_{th} is equal to $43.3m$ and is obtained for 178 nodes.
As expected, the distance D_{eff} decreases when the number of nodes increases. This corresponds to a higher density of nodes in the area considered.

5.3 Theoretical computation of an optimized 3D deployment

In 2D area coverage, triangular tessellation was proved to be the optimal strategy in terms of the number of sensor nodes number needed. This property cannot be generalized in 3D area coverage due to the big difference between two 2D and 3D deployment problems.

5.3.1 Best polyhedron tessellation for 3D space

To determine the optimal 3D tesselation, we need to answer the following question: what is the best way to place nodes in 3D such that the number of nodes required to ensure full 3D coverage is minimized ? Unfortunately, the optimal tesselation has not yet been determined. However, a truncated octahedron tesselation has been proved to be the most efficient (55). The authors in (55) demonstrated that the use of Voronoi tessellation to create truncated octahedron cells is the best strategy to achieve full coverage of a 3D area using the minimum number of nodes. The study was derived from Kelvin's conjecture. However, according to Kelvin and Kepler, the optimality proof for truncated octahedron tessellation is still not proven. The truncated octahedron (see Figure 5.5e) has 14 faces, of which are 8 are regular hexagons, and 6 are squares,so, a node has 14 neighbors.
Other patterns including different polyhedra: rhombic dodecahedron, hexagonal prism and cube, are also presented in (55), see Figure 5.5. The authors define the volumetric quotient to determine the best polyhedron. Let V be the volume of the polyhedron and r the maximum distance from its center to any vertex, then the volumetric quotient is:

$$\frac{V}{\frac{4}{3}\pi r^3} \tag{5.8}$$

The authors proved that the truncated ocatahedron gives the best volumetric quotient with the value of 0.68329 and provides the optimized number of nodes.

In addition to these polyhedra, we consider the regular dodecahedron where each node has 12 neighbors at the same distance, as illustrated in Figure 5.5d.

According to Equation 5.8, the volumetric quotient of the regular dodecahedron is equal to 0.666, which is very close to the value provided by the dodecahedron. We extracted Table 5.1 from (55) and completed it with the dodecahedron. This table presents a comparison between the different polyhedra in terms of volumetric quotient and the number of nodes compared to the truncated octahedron.

We observe that the regular dodecahedron requires a number of nodes that exceeds the number required by the truncated octahedron by only 2.59% which is a very interesting result that we will use in the next chapter.

Table 5.1: Volumetric quotient and number of nodes

Polyhedron	Volumetric quotient	Number of nodes needed compared to truncated octahedron
Cube	0.36755	85.9% more
Hexagonal Prism	0.477	43.25% more
Rhombic Dodecahedron	0.477	43.25% more
Regular Dodecahedron	0.666	2.59% more
Truncated Octahedron	0.68329	—

To meet the connectivity requirement, we study the constraint on transmission range in 3D for each polyhedron. Table 5.2 depicts the minimum value for the transmission range to ensure that two nodes occupying the center of adjacent polyhedra are able to

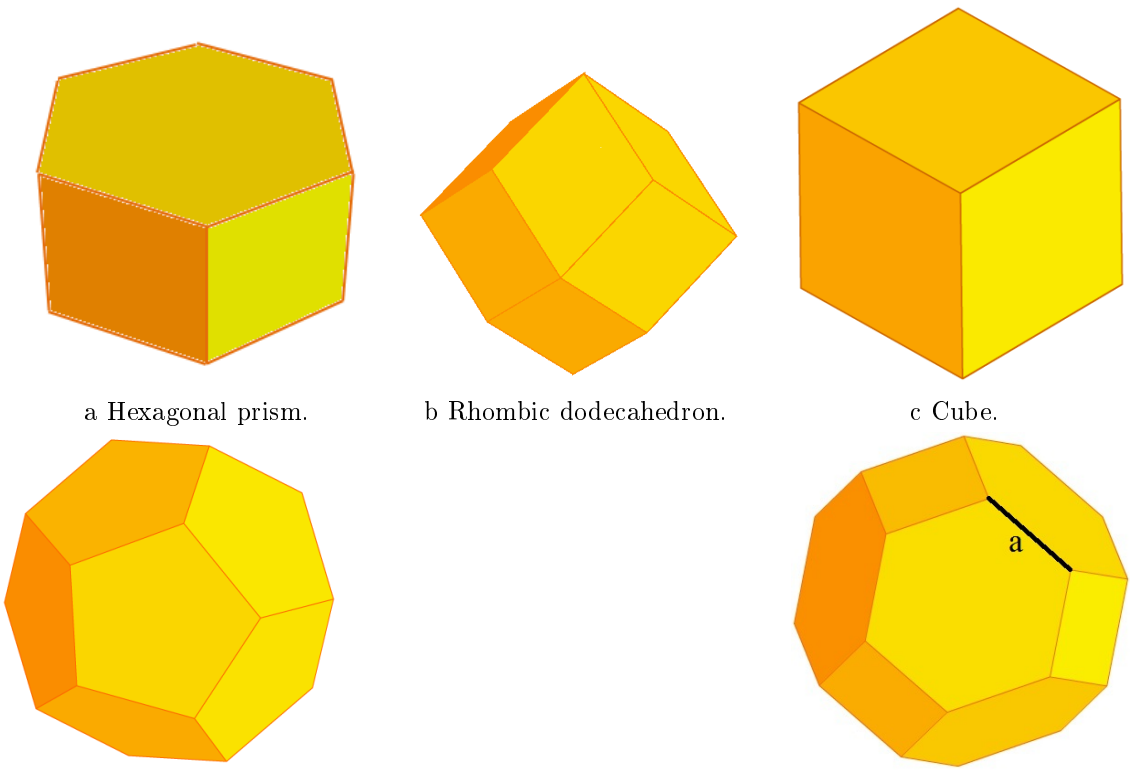


Figure 5.5: 3D geometric shapes.

communicate. The truncated octahedron provides the highest values on the three axes x, y and z, immediately followed by the regular dodecahedron.

Table 5.2: Transmission range in terms of sensing range

Polyhedron	x axis	y axis	z axis	Minimum R
Cube	1.1547 r	1.1547 r	1.1547 r	1.1547 r
Hexagonal Prism	1.4142 r	1.4142 r	1.1547 r	1.4142 r
Rhombic Dodecahedron	1.4142 r	1.4142 r	1.4142 r	1.4142 r
Regular Dodecahedron	1.5893 r	1.5893 r	1.5893 r	1.5893 r
Truncated Octahedron	1.7889 r	1.7889 r	1.5492	1.7889 r

5.3.2 Optimized number of nodes to cover 3D space

The authors in (55), propose equations to determine node positions for the cube, the hexagonal prism, the rhombic dodecahedron and the truncated octahedron placement strategies. However, they do not consider the border effects. Figure 5.6 depicts the truncated octahedron tessellation without considering the border effects.
To determine the optimized number of sensors required to achieve full coverage, we consider

the truncated octahedron tessellation.
In our study, we assume that the 3D space considered is a cube. Then, border effects are taken into account when calculating the minimum number needed to cover the whole cube.
Let a be the edge of the truncated octahedron (see Figure 5.5e).
The distance between two opposite rectangular faces is equal to $2\sqrt{2}a$.
The distance between two opposite hexagonal faces is equal to $\frac{\sqrt{6}}{a}$.
The radius of the sphere or distance between each vertex and the center of the truncated octahedron is $r = \frac{\sqrt{10}a}{2}$

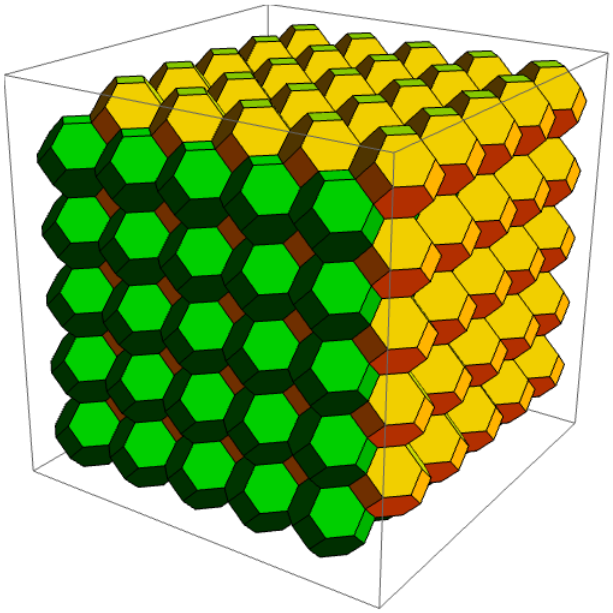


Figure 5.6: Truncated octahedron pattern

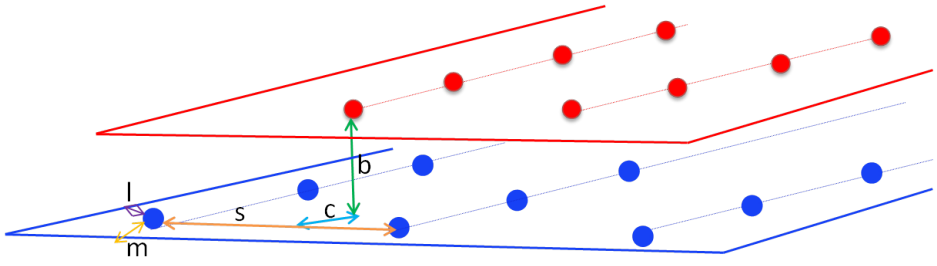


Figure 5.7: Node placement

In Figure 5.7, a node can be located either in a blue plane or a red plane. Each two neighboring nodes of the same plane are adjacent on the hexagonal faces, then $s = 2a$. In order to cover the border (left border in Figure 5.7), the nodes should be located at a distance $l = \frac{a}{2}$ from the border. The two planes do not start at the same value of the

x-axis. The red plane is shifted by $c = 1.281867a$ from the blue plane. The distance between two consecutive plans is $b = 1.1019a$.

The blue plane starts at $m = c - \frac{\sqrt{6}a}{2}$ on x-axis.

Let L, W and H be the length, the width and the height of the rectangular parallelepiped considered.

- Computation of the number of nodes in the blue plane
  - Number of nodes per column:

$$Nnode_{blue} = \lfloor \frac{L-m}{\sqrt{6}a} \rfloor + 1 + \delta. \tag{5.9}$$

$$with\ \delta = \begin{cases} 1 & if\ L - m - \lfloor \frac{L}{\sqrt{6}a} \rfloor \sqrt{6}a > \frac{\sqrt{6}a}{2} \\ 0 & otherwise \end{cases}$$

  - Number of columns per blue plane:

$$Ncol_{blue} = \lfloor \frac{W-l}{4a} \rfloor + 1 + \delta. \tag{5.10}$$

$$with\ \delta = \begin{cases} 1 & if\ W - l - \lfloor \frac{W}{4a} \rfloor 4a > \frac{4a}{2} + \frac{a}{2} = \frac{5a}{2} \\ 0 & otherwise \end{cases}$$

- Computation of the number of nodes in the red plane
  - Number of nodes per column:

$$Nnode_{red} = \lfloor \frac{L - \frac{\sqrt{6}}{2}}{\sqrt{6}a} \rfloor + 1 + \delta. \tag{5.11}$$

$$with\ \delta = \begin{cases} 1 & if\ L - \frac{\sqrt{6}}{2} - \lfloor \frac{L}{\sqrt{6}a} \rfloor \sqrt{6}a > \frac{\sqrt{6}a}{2} \\ 0 & otherwise \end{cases}$$

  - Number of columns per red plane:

$$Ncol_{red} = \lfloor \frac{W-l-2a}{4a} \rfloor + 1 + \delta. \tag{5.12}$$

$$with\ \delta = \begin{cases} 1 & if\ W - l - 2a - \lfloor \frac{W}{4a} \rfloor 4a > \frac{4a}{2} + \frac{a}{2} = \frac{5a}{2} \\ 0 & otherwise \end{cases}$$

- Computation of the number of nodes to cover a cube

– The number of planes

$$N_{plans} = \lfloor \frac{H}{1.1019a} \rfloor + 1 + 2 + \delta. \tag{5.13}$$

The term + 2 accounts for the two plans inserted to avoid coverage holes on the top and bottom of the cube.

$$with\ \delta = \begin{cases} 1 & if\ H - \lfloor \frac{H}{1.1019a} \rfloor 1.1019a > 0.36r \\ 0 & otherwise \end{cases}$$

– The total number of sensors in a given cubic area, denoted by N_{3D}, is the sum of the total number of sensors in the red plane and the total number of sensors in the blue plane. It is equal to:

$$N_{3D} = \lfloor \frac{N_{plans}}{2} \rfloor Nnode_{red} * Ncol_{red} + \lceil \frac{N_{plans}}{2} \rceil Nnode_{blue} * Ncol_{blue}$$

5.4 Conclusion

The use of an optimized deployment to monitor a given area allows the network to be optimized in terms of the number of sensors used and the energy consumed.
In this chapter, we adopted the triangular lattice tessellation in 2D to provide an optimal deployment using the least number of sensor nodes. We proposed a computation of the exact number of nodes needed to cover a full rectangular area. We also proposed a 3D optimized deployment based on the truncated octahedron tessellation to cover a rectangular parallelepiped.

These theoretical computations are used in the following chapters for optimized autonomous and assisted deployments.

Part III

Autonomous deployment

Chapter 6

Virtual forces based algorithms

Contents

6.1 Introduction

In the previous chapters, we studied the state of the art with regard to coverage and connectivity problems, different deployment strategies, assumptions, constraints and models adopted, which depend on the applications requirements. Here, we are interested in deployment algorithms, that ensure full coverage and connectivity of a given area, while taking into account many constraints such as, energy consumption and presence of known and unknown obstacles.

In this chapter, we focus on autonomous deployment based on virtual forces. There are many reasons for choosing virtual forces as a strategy to move mobile sensor nodes. First, the principle of virtual forces allows sensor nodes to cooperate and compute their appropriate positions in a distributed way, in which case, the deployment algorithm can be considered as being completely distributed. Second, virtual forces favor the spreading of nodes over the whole area, meaning that, full area coverage can be reached quickly. Third, if the target distance between neighboring sensor nodes is maintained, then, on the one hand, network connectivity will also be maintained and, on the other hand, the optimal deployment based on the triangular lattice can be established if a steady state is reached. Finally, the principle of the virtual forces is very simple, and has a low computation cost. Hence, the virtual forces strategy provides a distributed deployment that matches the optimal deployment and ensures full area coverage while maintaining network connectivity, as we will see in this chapter.
As shown in the state of the art, many studies based on virtual forces exist. However, they may differ in the parameters adopted and the attractive and repulsive forces formula.

Starting from a previous version of DVFA (4), Distributed Virtual Forces Algorithm, designed by our team to deploy sensor nodes in the area considered, we study how to tune the parameters of this algorithm to obtain very good results.

Unfortunately, the virtual forces algorithm in its distributed version suffers from some weaknesses:

- Node oscillations, due to the fact that each sensor node cannot have exactly 6 neighbors according to the triangular tessellation (3) at a distance of exactly D_{th} (e.g. border effect, number of sensor nodes higher that the required number).
- Tuning of parameters K_a (attractive coefficient) and K_r (repulsive coefficient): when K_a is high, the attractive force is great and may cause the stacking problem (i.e. two or more sensor nodes occupy the same position). When K_r is high, the new position of a sensor node can be at a distance greater than the communication range. Hence, a sensor node may be disconnected from the sink due to a large value of the repulsive force.
- End of the algorithm: the algorithm of the virtual forces ends when a steady state has been reached where all the nodes have stopped moving. However, due to node oscillations, the end of the virtual forces algorithm is its distributed version is still a problem.
- Energy consumption: during the execution of the virtual forces algorithm, most of the energy consumed by the sensor nodes is due to the nodes movements. Node oscillations induce high energy consumption and do not contribute to increasing area coverage.

We can conclude that many drawbacks of the virtual forces algorithm are related to node oscillations. In this chapter, we deal with these drawbacks and propose three variants of DVFA that cope with node oscillations. Since obstacles exist in a real environment, they should be taken into account when designing a deployment algorithm. That is why we propose a variant of DVFA that ensures full area coverage and network connectivity when known and unknown obstacles exist.

In this chapter, we start by presenting the principles of DVFA and its performance evaluation in Section 6.2. Then, in Section 6.3 we propose two deployment algorithms called ADVFA and GDVFA to cope with node oscillations. In Section 6.4, we propose a deployment algorithm called OA-DVFA that deals with both node oscillations and the presence of known and unknown obstacles in the area to be monitored. Finally, in Section 6.5, we show how to use virtual forces strategy in a 3D space and we propose 3D Distributed Virtual Forces Algorithm (3D-DVFA).

6.2 DVFA: Distributed Virtual Forces Algorithm

6.2.1 DVFA principles

DVFA (4), Distributed Virtual Forces Algorithm, is a distributed sensors deployment algorithm applying the virtual forces approach. The goal of DVFA is to ensure full coverage

of the area while maintaining network connectivity. Autonomous sensor nodes move according to the virtual forces exerted on them by their neighboring sensor nodes. The idea is to maintain the target distance D_{th}, the distance threshold, between two neighbors. Knowing the dimensions of the area to cover, the algorithm computes D_{th} as the result of Equation 5.2 in Chapter 5, assuming that the number of nodes is higher than or equal to N_{opt} given in Equation 5.6 in Chapter 5. It is worth noting that, if the number of operational sensors is smaller than N_{opt}, DVFA maximizes the coverage that can be obtained with this number.

In DVFA, each node repeats the following steps: neighborhood discovery, virtual forces computation and moves to its new position, as shown in Figure 6.1. More precisely, it proceeds as follows:

- **Step 1:** Each node s_i periodically sends a *Hello* message that contains its position obtained from a GPS and its 1-hop neighbors with their positions. This message allows the node to discover the positions of its 1-hop and 2-hop neighbors.
- **Step 2:** Each node s_i computes the forces exerted on it by its 1-hop and 2-hop neighbors. The force exerted by s_j on s_i where s_j is any 1-hop or 2-hop neighbor of s_i is:
  - Attractive if $d_{ij} > D_{th}$, where d_{ij} is the Euclidean distance between s_i and s_j. We have $\overrightarrow{F_{ij}} = K_a(d_{ij} - D_{th})\frac{(x_j - x_i, y_j - y_i)}{d_{ij}}$, where K_a is a coefficient in $[0, 1)$, (x_i, y_i) and (x_j, y_j) are the coordinates of s_i and s_j respectively;
  - Repulsive if $d_{ij} < D_{th}$. We have $\overrightarrow{F_{ij}} = K_r(D_{th} - d_{ij})\frac{(x_j - x_i, y_j - y_i)}{d_{ij}}$, where K_r is a coefficient in $[0, 1)$;
  - Null if $d_{ij} = D_{th}$.
- **Step 3:** Each node s_i computes the resulting force exerted on it: $\overrightarrow{F_i} = \sum_j \overrightarrow{F_{ij}}$.
- **Step 4:** Each node s_i moves to its new position (x'_i, y'_i) with $x'_i = (x_i +$ x-coordinate of $\overrightarrow{F_i})$ and $y'_i = (y_i +$y-coordinate of $\overrightarrow{F_i})$. Before moving, each node s_i sends a *Bye* message containing its new position. This message allows its neighbors to update their 1-hop and 2-hop neighbor table. The *Bye* message decreases the convergence time of DVFA.

To maintain network connectivity and limit the total distance traveled by each node at each iteration, the distance to the new position can never exceed a fixed threshold *Lmax*. *Lmax* reduces oscillations in sensor moves and then enables nodes to save energy.
The *Hello* period must be larger than the time needed to compute DVFA and to travel the distance *Lmax*, as shown in Figure 6.1.
We notice that DVFA does not need the knowledge of the exact number of operational nodes. For this reason, DVFA uses the value of D_{th} computed for the minimum number of nodes needed to fully cover the given area.

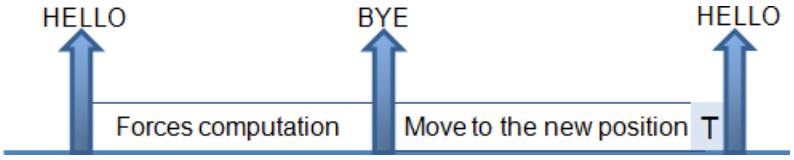


Figure 6.1: The three steps in an iteration of DVFA.

6.2.2 Performance evaluation

6.2.2.1 Simulation parameters

We implemented the DVFA algorithm as an agent in the NS2 simulator and performed simulations for different wireless sensor networks. Simulation parameters are given in Table 6.1. The values of Ka and Kr were experimentally determined to increase the area coverage and the convergence of the centralized virtual forces algorithm (4). We use the $Hello$ period value of $2s$. The IEEE 802.11b MAC protocol was used as many mobile robots are equipped with such an interface. Furthermore, this assumption makes sense, knowing that the evolution of the IEEE 802.15.4 MAC protocol and its performances are getting closer to the 802.11b protocols. Using these parameters, we compute the value of D_{th} in the optimal triangular lattice using Equation 5.2. The obtained value is $D_{th} = 43.3m$. From Equation 5.6, we compute $N_{opt} = 178$. For these simulations, we use a number of sensor nodes equal to $250 > N_{opt}$.

6.2.2.2 Simulated topologies

During the monitoring of the area, the data gathering process can partially fail if the network is disconnected, specially when sensor nodes have to cooperate to report the information detected to the controller robot. That is why we study the topologies depicted in Figures 6.2b and 6.2d. Each of them corresponds to a temporary worksite application considered in this PhD thesis.

- **Disconnected topology:** In the disconnected topology, several disconnected islands of connected nodes exist in the temporary worksite (see Figure 6.2b). This topology corresponds to several groups operating in the same worksite, but in non contiguous zones.

- **Four entry points topology:** The initial topology depicted in Figure 6.2d, corresponds to a scenario where different teams organize themselves to monitor the worksite starting from different entry points (four entry points in our case).

Moreover, the presence of coverage holes during deployment causes a problem for the data gathering process since data corresponding to coverage holes are missing. Hence, we will study, in addition, the two topologies depicted in Figures 6.2a and 6.2c.

- **Random topology:** In the random topology, sensor nodes are randomly scattered over the worksite (see Figure 6.2a).

Table 6.1: Simulation parameters.

Topology	
Sensor nodes	250 or 220 for different initial topologies
Area size	500m x 500m
Speed	5m/s
Simulation	
Result	average of 30 simulation runs
Simulation time	5000s
MAC	
Protocol	IEEE 802.11b
Throughput	2 Mb/s
Radio range R	50 m
Sensing range r	25 m
DVFA	
Ka	0.001
Kr	0.56
Hello period	2s
$Lmax$	$D_{th}/6$

- **Failed topology:** The topology depicted in Figure 6.2c presents a uniform deployment where some sensor nodes have failed. These failures are due to, for instance, battery depletion.

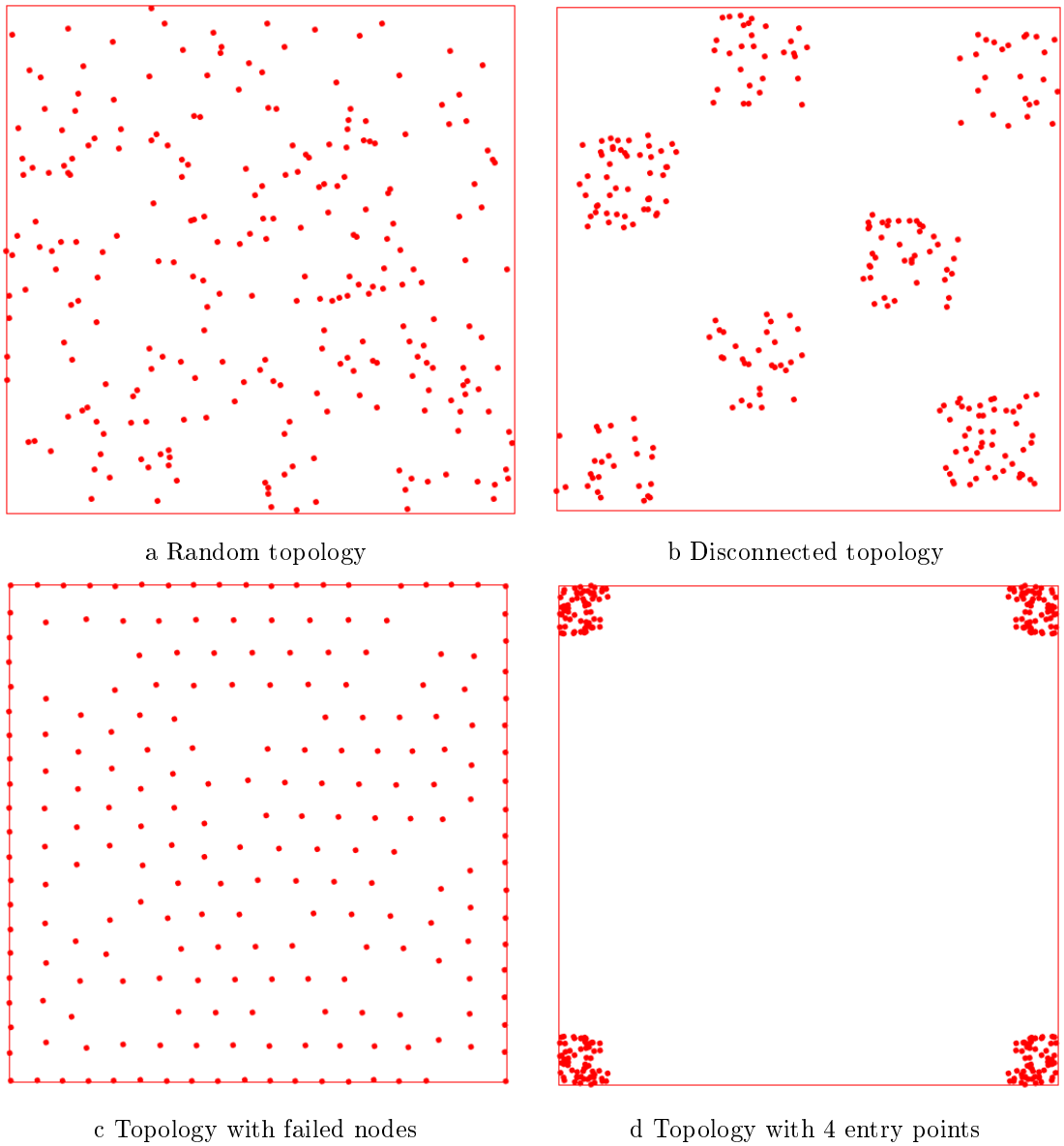
For each initial topology, 30 random configurations are simulated. The figures given in this chapter show the average value with the standard deviation.

6.2.2.3 Computation of the coverage

To compute the coverage rate, we virtually divide the network area into LxW grid units. A grid unit is considered to be covered if and only if its centered point is covered by at least one sensor node. The coverage rate is computed as the percentage of grid units covered. In the performance evaluation, we evaluate the coverage rate dynamically as a function of time. This evaluation is done in a centralized way using the nodes' positions at the current time.

6.2.2.4 Computation of the distance traveled

During the deployment, most of the energy consumption is caused by the sensor nodes' movements. In our simulations, we did not directly measure the energy consumed during



a Random topology

b Disconnected topology

c Topology with failed nodes

d Topology with 4 entry points

Figure 6.2: Initial topologies.

deployment; however, we did evaluate the total distance traveled by the sensor nodes. As the energy is proportional to this distance, the values of the distance traveled in the following sections reflect the energy consumed during deployment.

6.2.2.5 Simulation results

Figure 6.3 illustrates the final deployment obtained with DVFA for an initial topology with four entry points (Topology d in Figure 6.9b), providing a quasi-uniform deployment with a 99.9% coverage rate. Figure 6.4a depicts the coverage rate over time for the four

initial topologies. The first 500s are crucial to improve the coverage rate. After this time, the additional gain is small and almost null. For all topologies, DVFA achieves a very good coverage, it reaches 99.9% for the four toplogies depicted in Figure 6.2.

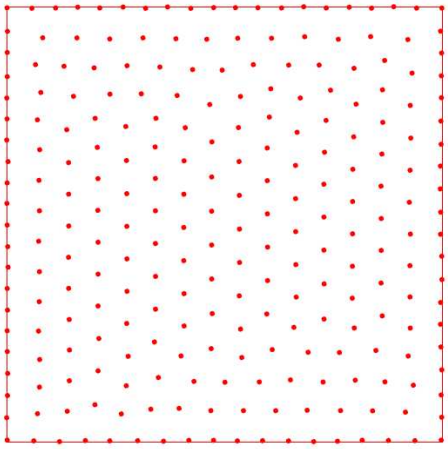
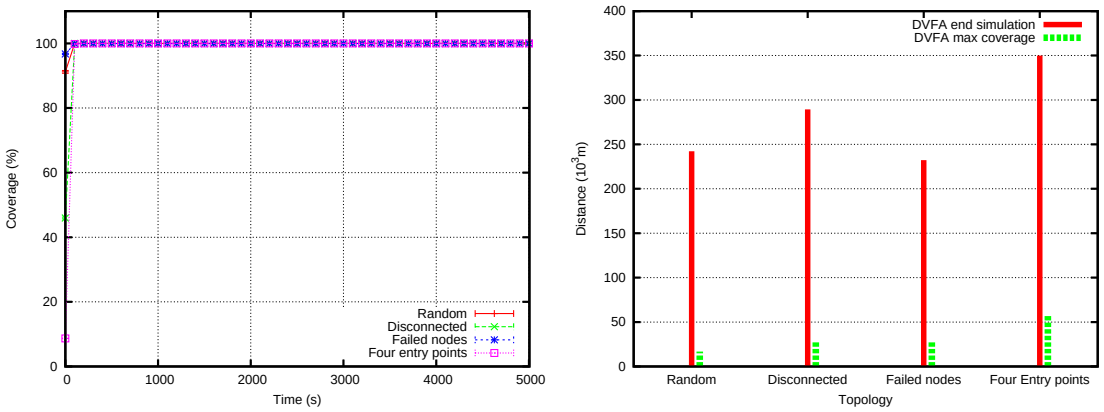


Figure 6.3: Final deployment of topology 4 with DVFA.



a Coverage rate as a function of time.

b Distance traveled by nodes.

Figure 6.4: DVFA evaluation: coverage rate and distance traveled.

We now evaluate the total distance traveled by nodes in DVFA, as shown in Figure 6.4b. We observe a very big gap between the total distance traveled by nodes during the simulation and the distance traveled by nodes when the maximum coverage is reached for the first time. This gap can be explained by node oscillations. In fact, even if the maximum coverage rate is reached, the nodes continue to run the DVFA algorithm and move accordingly. These oscillations lead to energy waste.

6.2.3 Summary

DVFA is a distributed algorithm that favors the spreading of sensor nodes over the whole area to provide full area coverage while maintaining network connectivity. However, it suffers from a major problem. Nodes move continuously, oscillating between different nearby positions, even when the maximum coverage has been reached. This stems from the fact that a node does not know when the maximum coverage of the area has been reached. Indeed, it is difficult to distinguish between a local optimum and a global one, and this problem is still an open issue.
To cope with the node oscillations problem, we made improvements to DVFA. In the following section, we propose two deployment algorithms called ADVFA and GDVFA.

6.3 How to cope with node oscillations

In this section we propose two virtual forces based deployment algorithms that deal with node oscillations. The first one is called ADVFA, Adaptive Distributed Virtual Forces Algorithm. ADVFA reduces node oscillations. The second deployment algorithm is GDVFA, Grid Distributed Virtual Forces Algorithm, that stops node oscillations.

6.3.1 ADVFA: Adaptive Distributed Virtual Forces Algorithm

Simulations of DVFA, show that using an inappropriate D_{th} independent of the number of nodes amplifies the oscillation phenomenon. With this D_{th}, it becomes impossible to obtain an equilibrium where virtual forces are null, leading to high energy waste.
We try to overcome this problem by proposing ADVFA: an Adaptive Distributed Virtual Forces Algorithm, which adapts the distance between neighbors to the total number of nodes.

6.3.1.1 ADVFA principles

ADVFA is a fully distributed deployment algorithm ensuring full coverage of the area. Unlike DVFA, the target distance between two neighbors is not fixed but varies as a function of the number of nodes discovered. ADVFA is highly adaptive to any environment: it adjusts its target distance according to the newly discovered connected components. The goal is to obtain a homogeneous deployment to avoid oscillations using the most appropriate distance between two neighbors depending on the number of nodes.

Like DVFA, ADVFA is based on the four steps defined in Section 6.2.1.An additional message, called *Component*, is exchanged periodically between 1-hop neighbors to compute the number of connected operational nodes discovered in the area. The *Component* message sent by a node s_i determines the operational nodes already discovered in its connected component. These operational nodes are represented in the *Component* message by a bitmap: the j^{th} bit represents the node s_j. If it is equal to 1, node s_j is present in the connected component of s_i. See Figure 6.5 for an example of such a bitmap. Initially, each node s_i marks the i^{th} bit to 1 in its *Component* message and sends it. Upon reception of the *Component* messages, node s_i makes an OR operation between its own message and

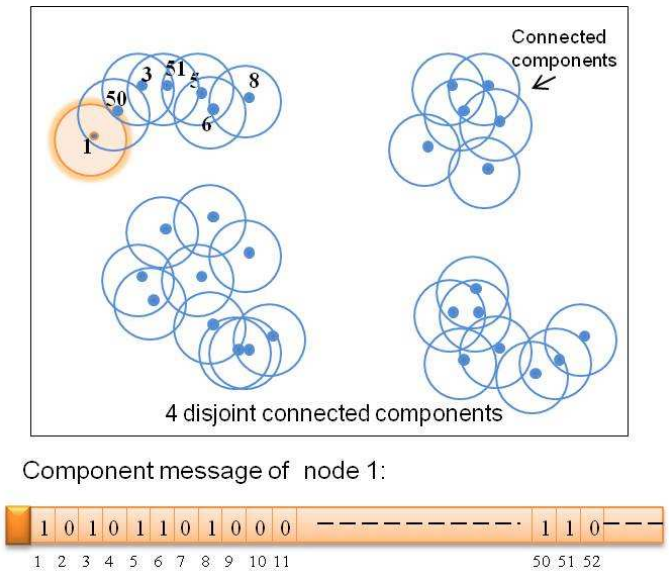


Figure 6.5: Bitmap of node 1 in its *Component* message.

all *Component* messages received and sends it in the next period. Consequently, node s_i is able to determine N, the number of operational nodes in its connected component by counting the number of marked bits:

- If $N \leq N_{opt}$ then $D_{eff} = D_{th}$ where N_{opt} is the optimal number of nodes needed to fully cover the given area and computed according to Equation 5.6 in Chapter 5, and D_{eff} is the expected distance between two neighbors.
- If $N > N_{opt}$ then D_{eff} is the solution of Equation 5.7 in Chapter 5.

ADVFA allows connected components to be discovered and merges of them. In fact, the first contact between two disjoint components will allow the exchange of *Component* messages with their different bitmaps included. Thus, the corresponding D_{eff} is immediately deduced and broadcast in the new connected component resulting from the merge.

Some nodes may fail due, for instance, to energy depletion. To take into account node failures occurring during the deployment algorithm, the bitmap is periodically recomputed from scratch to remove failed nodes. A re-computation of the bitmap of a connected component is triggered by an elected node (e.g., the node with the smallest address in this component).

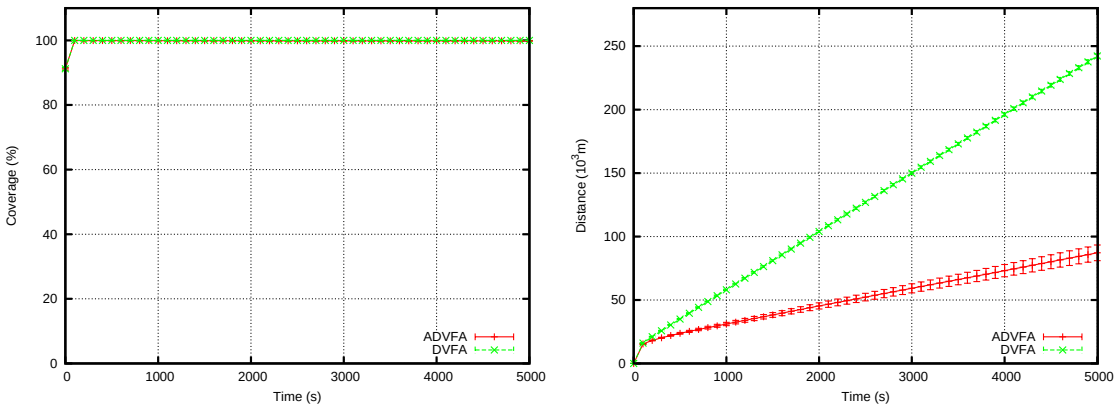
6.3.1.2 Comparative evaluation between ADVFA and DVFA

In this series of simulations, the period of *Component* messages is fixed to 5s. A short period of *Component* messages is needed to track the number of connected nodes already discovered. ADVFA adapts its parameters to this number in order to maintain the appropriate distance between neighboring nodes, thereby avoiding useless moves. As long as

new operational nodes are discovered, the target distance is updated. Depending on the adaptivity requirement, we may reduce the frequency of *Component* messages in order to save bandwidth.

The parameters used in the simulations are the same parameters as those defined in Table 6.1. We make a comparative evaluation between ADVFA and DVFA in terms of coverage rate and distance traveled. The coverage rate and distance traveled for ADVFA are computed as explained in Sections 6.2.2.3 and 6.2.2.4.

Random topology: Figure 6.6a shows that ADVFA and DVFA provide an excellent coverage rate of 99.9%. This is due to the principle of virtual forces that contributes to maintain the target distance between neighboring nodes, and results is sensor nodes occupy the whole area leading to this result. This result is achieved at the cost of a total distance traveled depicted in Figure 6.6b. We observe that ADVFA considerably reduces this distance by 64%.



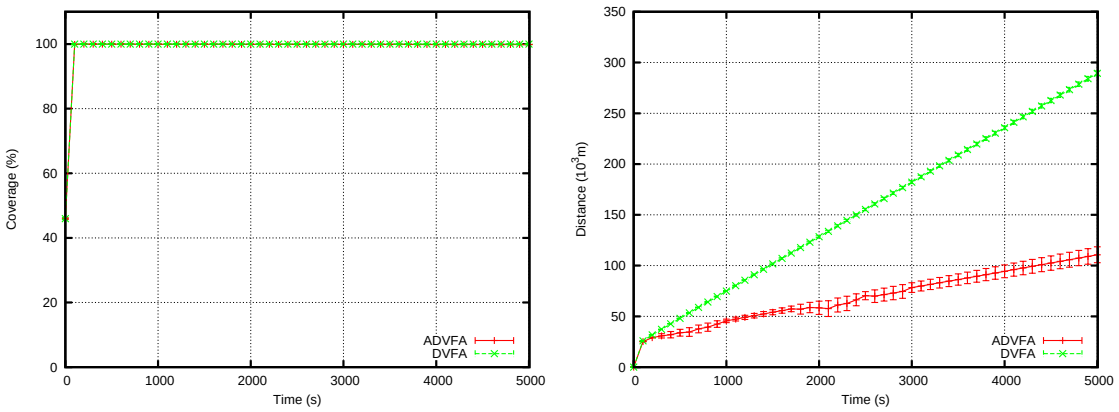
a Coverage rate as a function of time.

b Distance traveled by nodes.

Figure 6.6: ADVFA evaluation: Random topology

Disconnected topology: Figure 6.7a shows that, after a short time, full coverage is achieved by both DVFA and ADVFA. However ADVFA has the merit of reaching this coverage with a smaller total distance traveled. As shown in Figure 6.7b, this distance is reduced by 61% compared to DVFA. As a conclusion ADVFA maintains full coverage like DVFA, but increases the network lifetime by reducing energy consumption.

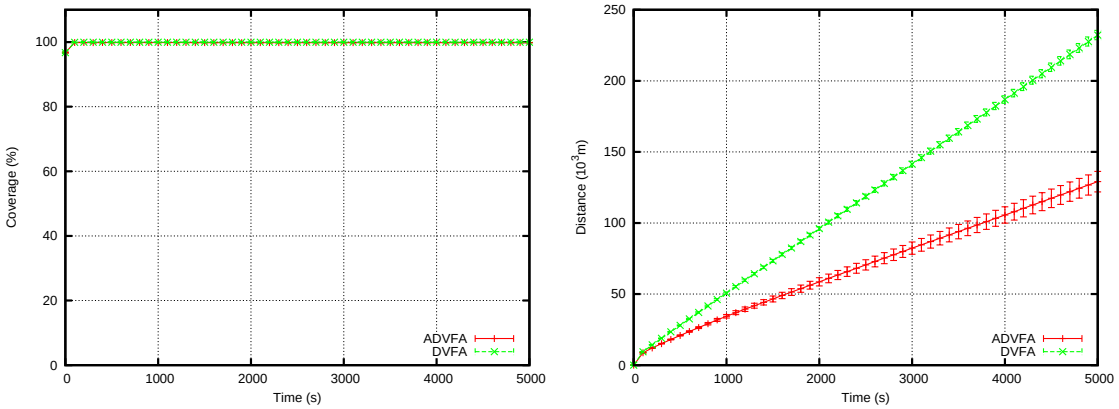
Topology with failed nodes: In the monitoring area, sensor nodes may fail due to their battery depletion. These failures are detected by both algorithms that use *Hello* messages to discover node neighborhood. However only ADVFA adapts the target distance to the new number of operational nodes. This is made possible by the exchange of the *Component* message that is periodically updated. We observe that ADVFA and DVFA achieve full coverage rate as depicted in Figure 6.8a. However, the distance traveled is considerably



a Coverage rate as a function of time. b Distance traveled by nodes.

Figure 6.7: ADVFA evaluation: Disconnected topology

smaller with ADVFA (see figure 6.8b). This is due to a target distance computed with the effective number of operational nodes, leading to a more stable redeployment.
ADVFA is robust with regard to node failures: it is able to adapt to the number of operational nodes that it progressively discovers. This quality of ADVFA can be very important for applications where sensors can be damaged during their initial drop or can fail because of energy depletion.

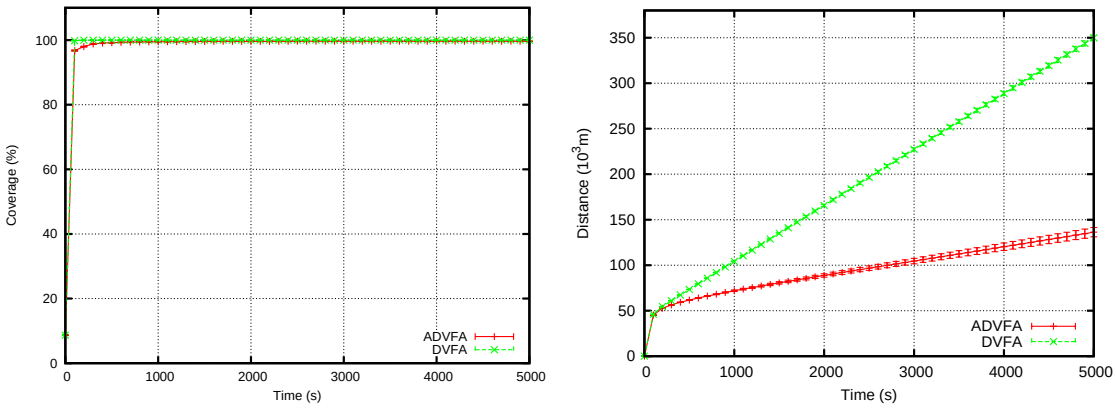


a Coverage rate as a function of time. b Distance traveled by nodes.

Figure 6.8: ADVFA evaluation: Failed topology

Topology with four entry points: Figure 6.9a depicts a very good coverage rate for both ADVFA and DVFA. However ADVFA considerably reduces the total distance traveled by nodes.In Figure 6.9b, we can observe, that the distance traveled by DVFA increases rapidly to reach $300km$ at the end of the deployment, whereas the distance

traveled by ADVFA does not exceed $140km$. Hence, ADVFA is more energy efficient than DVFA.



a Coverage rate as a function of time.

b Distance traveled by nodes.

Figure 6.9: ADVFA evaluation: Four entry points topology

ADVFA is a good solution to provide full area coverage and guarantee network connectivity while saving energy. As shown previously, ADVFA reduces node oscillations by adapting the target distance to the number of nodes deployed. Although the distance traveled by nodes is considerably reduced, nodes continue to oscillate. To cope with this problem, sensor nodes should stop moving when they are in the appropriate position. This goal can be met if the final positions of the nodes are predetermined.
To deal with this problem we propose a redeployment algorithm called GDVFA, Grid Distributed Virtual Forces Algorithm, which combines the virtual forces strategy to move sensor nodes with the grid-based strategy to stop them and save energy.

6.3.2 GDVFA: Grid Distributed Virtual Forces Algorithm

6.3.2.1 GDVFA principle

The GDVFA algorithm is a hybridization of the virtual forces strategy and the grid strategy. GDVFA, like DVFA, is based on virtual forces to move sensor nodes and maintain the target distance D_{th} between neighboring nodes. The new position of a sensor node is computed according to the sum of the forces exerted on it by its 1-hop and 2-hop neighbors. As we showed for DVFA, the distance the node can move is limited to a fixed threshold called $Lmax$ in order to reduce the distance the nodes travel at each iteration. The originality of GDVFA lies in the use of grid based strategy: we propose dividing the area into similar virtual cells. Our target is to incite nodes to occupy the centers of cells. Hence, redundant nodes are those that do not occupy the center of a cell. They can easily be detected and switched to a sleep state to save energy. Furthermore, any node whose neighboring cell centers do not change can stop moving. In this way, the energy consumed is reduced. These two enhancements are detailed in Subsection 6.3.2.3.

GDVFA proceeds in two phases, both of which consist of a set of iterations. At each iteration, each node executes the four steps defined in Section 6.2.1. Each iteration has a duration of a *Hello*-period.

Phase 1 executes the simple DVFA to spread sensor nodes over the whole area while ensuring a uniform density. During this phase, each node moves in step 4 to the new position computed by virtual forces. At the end of phase 1, the nodes are deployed over the whole area while offering a good coverage and uniform density. Nevertheless, DVFA suffers from high energy consumption due to node oscillations.
The aim of *Phase 2* is essentially to cope with this drawback by adopting the grid strategy. In this phase, step 4 (see Section 6.2.1) is replaced by the following step:

- **Step 4':** each node determines the cell containing its new position. If the center of this cell is empty and this node has the smallest identifier among all the nodes in this same cell, then it moves to the cell center. Otherwise, this sensor node moves to the new position determined by the resultant force. Notice that a node that occupies the center of a cell can leave it if and only if its neighbors exert on it an attractive or repulsive force.

The benefit of the first phase is that is spreads the nodes over the whole area rapidly, in a predefined amount of time (the spreading factor described in 6.3.2.5), before switching to the second phase offering stability and convergence. For example, in the performance evaluation in section 6.3.2.4, this spreading factor is equal to $100s$ for a topology where nodes are randomly scattered all over the network area. Any sensor node is assumed to know the value of the spreading factor, a parameter of GDVFA.

6.3.2.2 Cell definition

In this section, we explain how to define the virtual grid in GDVFA by giving the equations used to determine the cell center relative to a sensor node of coordinates (x, y) in the grid. The size of the cells is computed with regard to the sensing range in order to ensure full area coverage. As shown in Chapter 2, when $R \geq \sqrt{3}r$ and full area coverage is ensured, network connectivity is consequently ensured. In our work, network connectivity is therefore guaranteed since we assume $R \geq \sqrt{3}r$.

In GDVFA, we choose to use rectangular cells to simplify the computation of the cell which any node belongs to, knowing the coordinates (x, y) of the sensor node, the values of L and W of the network area and the value of the sensing range, r. Sensor nodes should occupy the center of these cells. Figure 6.10 depicts the grid cells with a rectangular shape. Each non-border cell has a length equal to D_{th} and a width of $\frac{3r}{2}$. Furthermore, each non-border cell has 6 neighboring cells. However, the sensor nodes are deployed in triangular lattice of edge D_{th}. As the GDVFA algorithm deploys sensor nodes in a triangular lattice, then each three neighboring nodes, the vertices of the same triangle, should fully cover this equilateral triangle while minimizing their overlapping. This overlapping can be a single point which corresponds to the barycenter of the triangle. Since the distance between the barycenter of this triangle and each vertex of this triangle is r, the distance between two neighboring nodes, an edge of the triangle, is $D_{th} = \sqrt{3}r$. This also corresponds to the

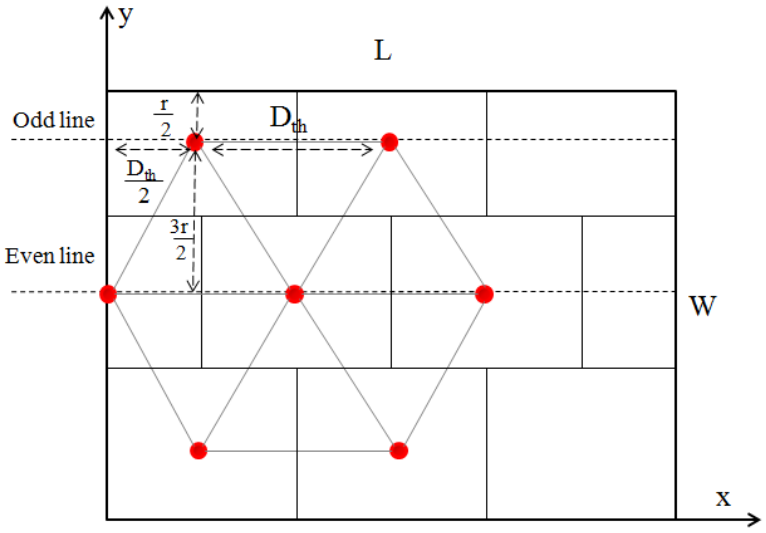


Figure 6.10: Grid in GDVFA.

length of the cell. The width of the cell is equal to the height of the equilateral triangle $\frac{3r}{2}$.
The cell centers of the same floor constitute a single line. We distinguish between odd and even lines. In any odd line, the first sensor node should be located at a distance $\frac{D_{th}}{2}$ from the left border line. In any even line the first sensor node is located on the left border line. Border cells are a particular case. They are built to ensure full area coverage while minimizing the number of sensor nodes. That is why the first line is at a distance $\frac{r}{2}$ from the top border line.
Let (x, y) be the coordinates of any sensor node s_i in the grid. We want to determine which cell s_i belongs to. This cell is defined by a line number, denoted $line(x, y)$, and a column number, denoted $col(x, y)$.

$$line(x,y) = \lfloor \frac{y - \frac{r}{2}}{\frac{3r}{2}} \rfloor + 1 + \delta_l \tag{6.1}$$

$$with\ \delta_l = \begin{cases} 1 & if\ y - \frac{5r}{4} - \lfloor \frac{y-\frac{r}{2}}{\frac{3r}{2}} \rfloor \frac{3r}{2} > 0 \\ 0 & otherwise \end{cases}$$

For the computation of the column, we distinguish two cases:

- Case $(line(x, y) modulo 2) = 0$:

Then, the node of coordinates (x, y) occupies the $col_e(x, y)^{th}$ cell in an even line computed as follows:

$$col_e(x,y) = \lfloor \frac{x}{\sqrt{3}r} \rfloor + 1 + \delta_e \tag{6.2}$$

$$with\ \delta_e = \begin{cases} 1 & if\ x - \frac{D_{th}}{2} - \lfloor \frac{x}{D_{th}} \rfloor D_{th} > 0 \\ 0 & otherwise \end{cases}$$

Hence, the coordinates of the cell center are (x_c, y_c)

$$with \begin{cases} x_c = (col_e(x,y) - 1)D_{th} \\ \\ y_c = (line(x,y) - 1)\frac{3r}{2} + \frac{r}{2} \end{cases}$$

- Case $(line(x,y) modulo 2) \neq 0$:

Then the node of coordinates (x, y) occupies the $col_o(x,y)^{th}$ cell in an odd line computed as follows:

$$col_o(x,y) = \lfloor \frac{x - \frac{D_{th}}{2}}{D_{th}} \rfloor + 1 + \delta_o \tag{6.3}$$

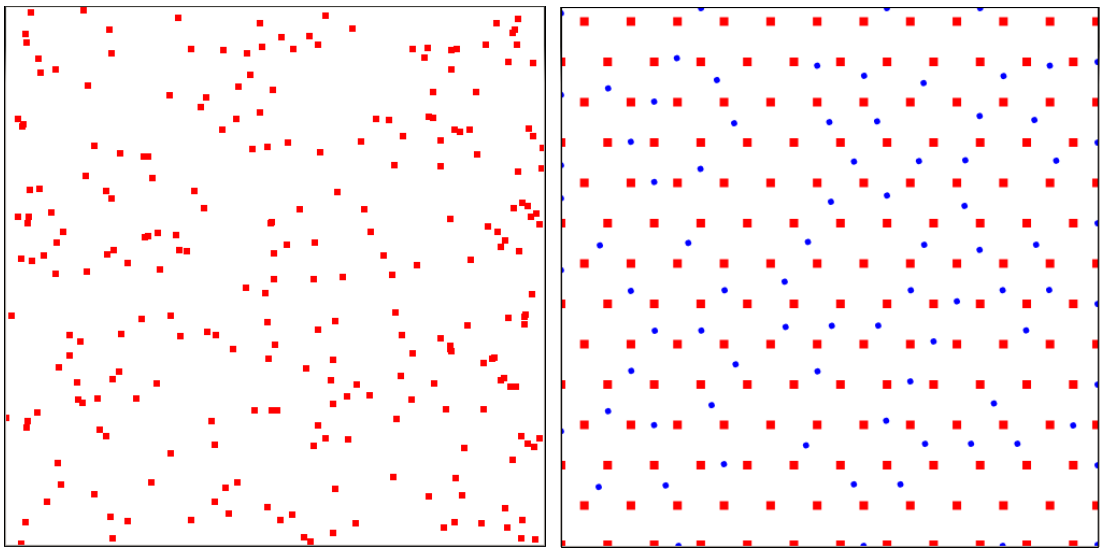
$$with\ \delta_o = \begin{cases} 1 & if\ x - D_{th} - \lfloor \frac{x - \frac{D_{th}}{2}}{D_{th}} \rfloor D_{th} > 0 \\ 0 & otherwise \end{cases}$$

and $D_{th} = \sqrt{3}r$.
Hence, the coordinates of the cell center are (x_c, y_c)

$$with \begin{cases} x_c = (col_o(x,y) - 1)D_{th} + \frac{D_{th}}{2} \\ \\ y_c = (line(x,y) - 1)\frac{3r}{2} + \frac{r}{2} \end{cases}$$

6.3.2.3 Stopping condition and detection of redundant nodes

By definition a node is said to be in stop state in an iteration if and only if it does not move during this iteration due to the stopping condition. However, to keep the required property of reactivity to topology changes (e.g. node departure, empty cells detected), the stopping condition is always verified at each iteration.
Stopping condition: the nodes occupying the center of its 6 *neighboring cells and its cell have not changed during three consecutive iterations.*
At each iteration, any node computes the resultant of the virtual forces exerted on it, and checks the stopping condition. If the node has not stopped and the stopping condition is true, the node stops. Furthermore, a previously stopped node moves in an iteration if and only if: either the resultant of the virtual forces differs from the previous one (e.g. the arrival of a new neighbor), or the stopping condition is no longer true.
A node that has stopped without occupying a cell center is said to be *redundant*. Redundant nodes are used to replace failed or depleted nodes. For the initial deployment depicted in Figure 6.11a, GDVFA, provides the final deployment shown in Figure 6.11b for 250 nodes. The red nodes are active nodes, whereas the blue nodes with small points are redundant nodes which sleep to save energy and prolong the network lifetime.



a Initial random deployment b Final deployment.

Figure 6.11: GDVFA with 250 nodes.

6.3.2.4 Comparative evaluation between DVFA and GDVFA

We evaluate the performance of GDVFA based on various random topologies with different numbers of nodes (200, 250 and 300). We fix the spreading factor for the different random topologies to $f = 100s$. The remaining simulation parameters are those used for the evaluation of DVFA (see Table 6.1). The coverage rate and the distance traveled for GDVFA are computed as explained in Sections 6.2.2.3 and 6.2.2.4.
In Figure 6.12, we can see that GDVFA reaches a 100% coverage rate, while considerably reducing the total distance traveled by the nodes, as shown in Figures 6.13a, 6.13b and 6.13c, for different numbers of nodes 200, 250 and 300. The total distance traveled by the nodes increases with time and with the number of nodes in DVFA (see Figures 6.13a, 6.13b and 6.13c).
We can see that the slope of the total distance curve is much steeper when the number of nodes exceeds the optimal number needed to fully cover the network area. This can be explained by the fact that DVFA tries to enforce an equilateral pattern deployment, where the edge of the triangle is D_{th}, computed for the optimal number of sensors. Hence, the node oscillations increase when the number of sensors increase beyond this optimal number. The nodes continue to move even when the 100% coverage rate is reached. Unlike DVFA, GDVFA provides very good performances with regard to the distance traveled: this distance remains constant a short time after the end of the Phase 1, because GDVFA eliminates node oscillations by stopping nodes. Hence, with GDVFA energy consumption is reduced.
Simulation results show that the stopping condition is very efficient. In fact, with GDVFA all the nodes stop whatever the number of nodes (see Figure 6.14a).
We can distinguish two types of stopped nodes:

- the ones occupying the center of cells. These nodes are needed to ensure full coverage;
- the other ones are redundant nodes (see Figure 6.14b).

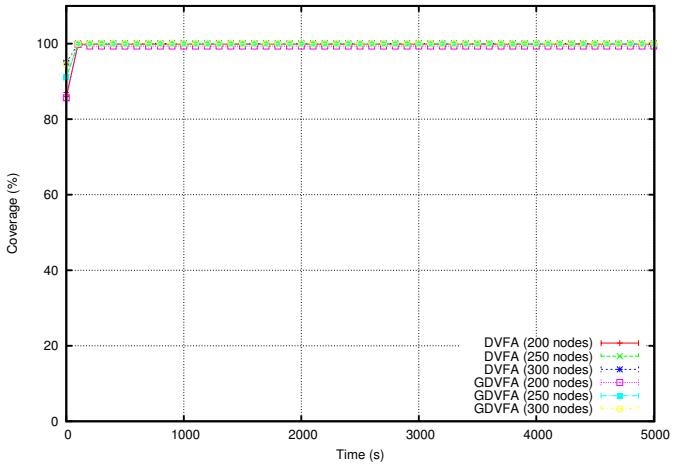


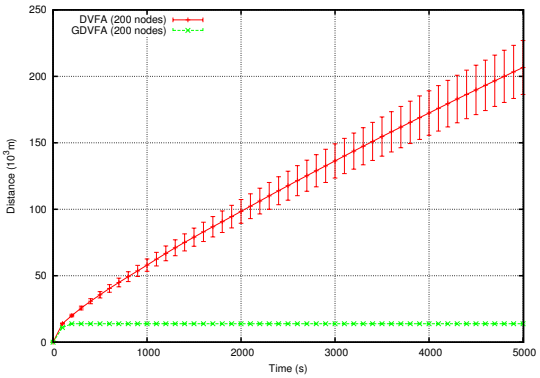
Figure 6.12: Coverage with GDVFA and DVFA.

GDVFA enables the distributed detection of redundant nodes (see Figure 6.14b). Such nodes can be turned off to save energy and increase the network lifetime. The detection of redundant nodes can also be used to repair coverage holes or replace energy depleted nodes. Figure 6.14c depicts the cumulative stopping time that allows nodes to save energy by reducing useless moves and prolonging network lifetime.

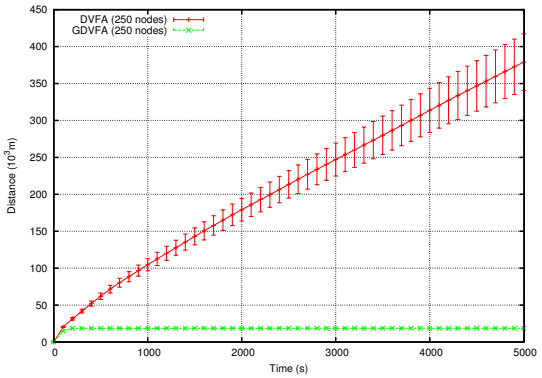
6.3.2.5 Impact of the spreading factor

Up to now, we have assumed that the initial deployment was random and the sensor nodes were scattered over the entire network area. Simulation results reported in Section 6.3.2.4, show that $f = 100s$ is sufficient to obtain a final coverage rate of nearly 100% at the end of the simulation. For many applications, this initial deployment is not representative; initially all the sensor nodes are grouped together at a single entry point of the area. In this case, $f = 100s$ may be insufficient to obtain the full coverage, even if the number of sensors is sufficient to obtain this full coverage. We can establish a lower bound for this factor in such an initial topology as follows. Let $W * L$ be the length and the width of the rectangular area considered. In DVFA and GDVFA, any sensor node cannot move more than $Lmax$ during each $Hello$-period. The lower bound on the spreading factor is computed, taking into account the time needed by a sensor to reach the furthest position from the initial one in the topology. For instance, in a topology with a single entry point at the corner of the rectangular area, which can be considered as one of the worst cases, we get:

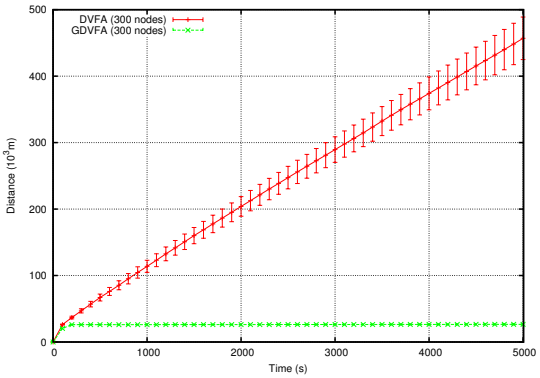
$$\frac{\sqrt{L^2+W^2}}{Lmax} * Hello\ period$$



a Distance traveled by 200 nodes.



b Distance traveled by 250 nodes.



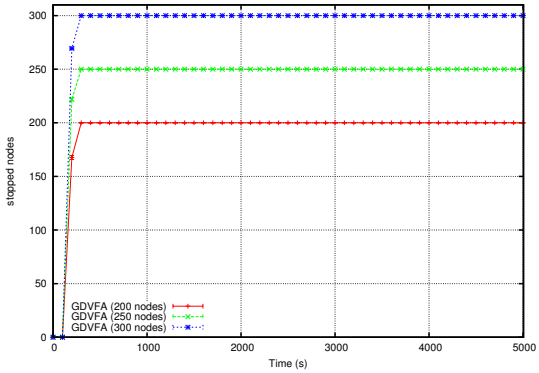
c Distance traveled by 300 nodes.

Figure 6.13: GDVFA evaluation: Total distance traveled by nodes with GDVFA and DVFA.

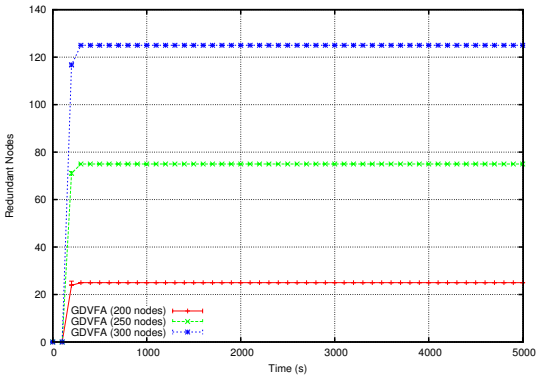
With the simulation parameters given in Table 6.1, we get $f \geq 194s$. We now study the impact of the spreading factor on the performances achieved by GDVFA. Figure 6.15a shows that a spreading factor of $100s$ is not sufficient to ensure the nodes are spread over the whole area. Consequently, the area coverage remains limited to 63%. A spreading factor larger than $200s$ allows a 100% coverage rate to be reached. As illustrated in Figure 6.15b, the total distance traveled by the nodes increases with the spreading factor. This is due to the oscillations caused by DVFA, which occur even when full area coverage has been obtained. As a conclusion, the choice of the spreading factor is very important for the performances of GDVFA, expressed in terms of coverage rate and distance traveled. To save energy, we recommend choosing the smallest value that ensures full coverage, (e.g. $250s$ in our example).

6.3.3 Summary

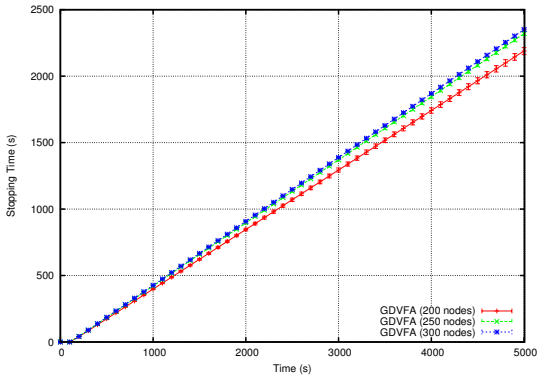
In this section, we proposed two deployment algorithms based on virtual forces that enable the spreading of nodes as quickly as possible with the minimum energy consumption while



a Number of stopped nodes.

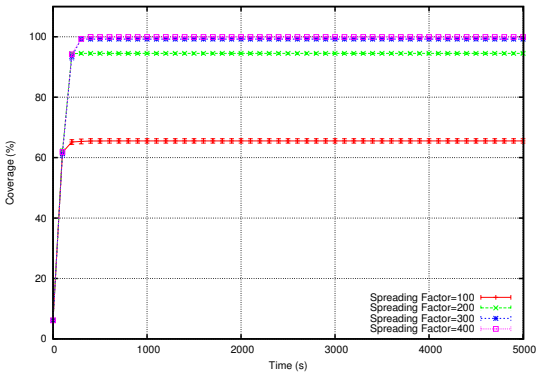


b Number of redundant nodes.

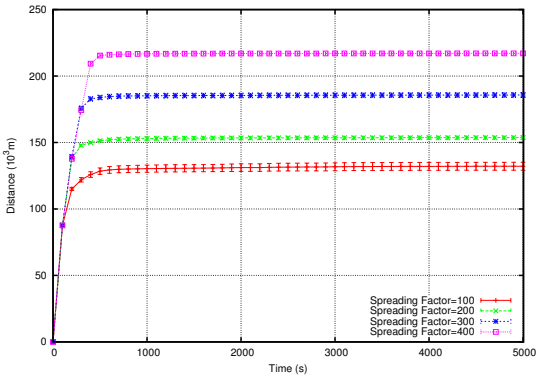


c Cumulative stopping time.

Figure 6.14: GDVFA evaluation: Stopped and redundant nodes.



a Coverage rate.



b Distance traveled.

Figure 6.15: GDVFA evaluation: Impact of the spreading factor on the coverage rate and the total distance traveled by nodes.

avoiding node oscillations.

- ADVFA is a deployment algorithm that adapts to the number of nodes and the presence of disconnected components. Due to its mechanism, ADVFA considerably reduces nodes oscillations.

- GDVFA is based on both virtual forces strategy and grid strategy. It is able to stop node oscillations and determine redundant nodes, so, they can be switched to sleep state. GDVFA is an energy efficient autonomous deployment algorithm.

However, the area considered may contain obstacles. In the following section, we show how to cope with the presence of known and unknown obstacles.

6.4 How to cope with the presence of known or unknown obstacles

6.4.1 Obstacles and deployment algorithms

In the literature, many studies focus on the deployment of wireless sensor nodes in an area containing known obstacles. However, very few studies deal with unknown and unpredictable obstacles. This situation corresponds to the requirement of many applications such as monitoring a post-disaster area and damage assessment.
The principle of virtual forces must be enhanced to cope with obstacles. For instance, the authors of (17) and (6) propose a virtual force algorithm as a sensor node deployment strategy to enhance the coverage rate of the area. In this study, a repulsive force is exerted by the obstacle on sensor nodes. Despite the high level of coverage rate obtained by this solution, the total knowledge of, on the one hand, the area considered and, on the other hand, the obstacles' shape and position is required. Two other solutions based on virtual forces and which cope with unknown obstacles, are presented in (56). Both solutions aim to maintain network connectivity between sensor nodes and the sink. Since obstacles may exist in the area, the authors propose using the right-hand rule to bypass the obstacles. The idea is to move a sensor node along a straight line toward its new position; when an obstacle is detected, the right hand maintains contact with the obstacle until this sensor node gets back to the straight line. The two solutions proposed are not only based on virtual forces but also on other strategies that require broadcasting messages to maintain network connectivity. These solutions therefore induce a high overhead in terms of messages broadcast in the network to check the connectivity of the nodes with the sink. Moreover, the right-hand rule proposed to avoid obstacles, may not be efficient with some shapes of obstacles. We notice that both solutions favor network connectivity at the expense of full area coverage. In this section, we focus on the deployment of autonomous sensor nodes, based on the virtual forces principle, in an area that may contain unknown obstacles. We propose OA-DVFA Obstacles Avoidance Distributed Virtual Forces Algorithm which not only avoids obstacles, but also deals also with node oscillations problem as it uses the grid strategy to stop nodes moving.

To be more representative of a real environment, we have to take into account the existence of obstacles. The principle of virtual forces in DVFA does not consider the presence of obstacles in the area. We distinguish two types of obstacles: transparent obstacles and opaque obstacles (see Chapter 4). Furthermore, obstacles may be known in advance and their position and shape can be taken into account before starting the placement of nodes at their appropriate positions. In contrast, unknown obstacles are discovered dynamically

when a mobile node coming close to an obstacle detects it. The trajectory of the mobile node is then modified during the deployment. The mechanism used by OA-DVFA to cope with obstacles is valid for both transparent and opaque obstacles and also when obstacles are unknown.

6.4.2 OA-DVFA: Obstacles Avoidance Distributed Virtual Forces Algorithm

6.4.2.1 OA-DVFA principles

OA-DVFA, like DVFA, is based on virtual forces to move sensor nodes and maintain the target distance D_{th} between neighboring nodes. The new position of a sensor node is computed according to the sum of the forces exerted on it.

To avoid node oscillations and stop the movement of sensor nodes, OA-DVFA uses a virtual grid strategy, like GDVFA. The idea is to divide the area into cells whose centers match the optimal deployment as if no obstacles were present. Nodes are incited to occupy these centers when they are reachable (i.e. not inside obstacles). Then, sensor nodes in cell centers should perform the monitoring task whereas, the others are considered as redundant nodes and can switch to a sleep state to save energy. However, in the presence of obstacles, not only the nodes in cell centers should be active, but so should some nodes which are around the obstacles and whose cell centers are inside the obstacle (see for instance Figure 6.16). The others can switch to a sleep state.
More precisely, OA-DVFA proceeds in three phases:

Phase 1: Node Spreading

Nodes spread over the whole area based on the virtual forces principle while avoiding known or unknown obstacles. Then, like in DVFA, at each iteration, each node executes the four steps defined in 6.2.1. The fourth step is adapted to OA-DVFA to cope with obstacles. So, when the new position is within an obstacle, the sensor node will move toward this position until it detects the obstacle. Then, it stops at a certain distance from the obstacle's border.

Phase 2: Stop Node Oscillations

In a virtual cell matching the optimized deployment, the node with the smallest identifier moves to the cell center if it is unoccupied.

Phase 3: Node Activity Scheduling

After a pre-computed time, each node decides to stay active or switch to sleep state to save energy. This decision is taken according to the following rules:

- Nodes in cell centers stay in an active state.
- Nodes whose cell centers are occupied by other nodes switch to a sleep state.

- For all nodes whose cell center is inside an obstacle:
  - Only the closest node to the cell center remains in an active state,
  - The others switch to a sleep state.

The neighborhood of a sensor node may change due to obstacles. Some neighboring nodes will no longer be neighbors due to the presence of opaque obstacles. The number of active nodes is not the same depending on whether obstacles are opaque or transparent. We do not propose an additional condition to deal with opaque obstacles since the OA-DVFA principle is still valid. Figures 6.16 and 6.17, show how OA-DVFA copes with both opaque and transparent obstacles. Small squares (red in the center of the cell, black otherwise) denote active sensor nodes, whereas redundant nodes in a sleep state are denoted by blue disks. In the case of a transparent obstacle (see Figure 6.16), only one sensor node per cell, the closest to the cell center, stays active, the others are considered to be redundant nodes and should switch to a sleep state. However, when the obstacle is opaque (see Figure 6.17), at least one node stays active in a cell. Since opaque obstacles block communication between nodes, two nodes may be in the same virtual cell without being neighbors. Then, both of them decide to stay active: see for instance the nodes within the orange circles.

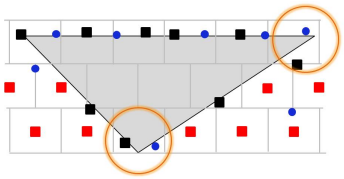


Figure 6.16: Transparent Obstacle.

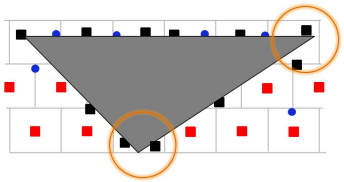


Figure 6.17: Opaque Obstacle.

6.4.2.2 How to Run OA-DVFA for known obstacles

When obstacles are known, the spreading time, called $Phase1\_Spread\_Time$ and defined as the time needed to execute the node Spreading Phase, can be estimated in advance. All the nodes know the value of the spreading time, a parameter of OA-DVFA. They all enter Phase 2 after this time, followed by Phase 3. The $Phase1\_Spread\_Time$ is equal to $1500s$ for Topology 1 and $4000s$ for Topology 2. The execution of OA-DVFA is illustrated in Figure 6.18.

6.4.2.3 How to Run OA-DVFA for unknown obstacles

When obstacles are unknown, the spreading time cannot be estimated in advance. Sensor nodes should cooperate to decide when to stop the spreading phase. This decision strongly

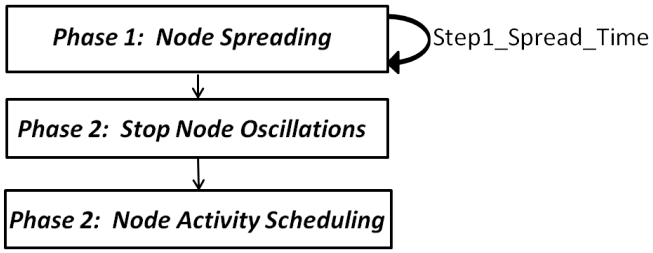


Figure 6.18: Known obstacles: OA-DVFA.

depends on the coverage rate: as long as the coverage rate increases, node spreading should continue. Since each node is able to determine the virtual cell to which it belongs, this information is used to estimate the coverage by computing the number of cells visited by sensor nodes. For this purpose, nodes exchange a bitmap message, where each node sets to 1 the bit corresponding to its cell. When a node receives the bitmap of its neighbors it updates its bitmap by making a logic OR between its bitmap and the bitmap received. The coverage is estimated by the number of times 1 appears in the bitmap. However due to the presence of obstacles, not all the cells can be occupied by sensor nodes. We notice that if the number of visited cells in the bitmap increases, the coverage rate also increases, and node spreading should continue. Otherwise, the Spreading Phase ends.
To reduce the overhead, the exchange of bitmap messages is limited. Initially, nodes spread without exchanging bitmap messages during a time $Initial\_Spread\_Time$. This time must be higher than or equal to the $\frac{Diagonal}{Lmax} * Hello\_Period$ to allow nodes to reach the corner opposite to the sink. In the performance evaluation of Section 6.4.2.4, we get $Initial\_Spread\_Time \geq \frac{500*\sqrt{2}}{5} * 2.9 = 410s$.
After this time, nodes continue to spread while exchanging bitmap messages during $Bitmap\_Spread\_Time$. After this time, all the nodes check whether the number of visited cells remains constant. If so, they end the Spreading Phase. Otherwise, they continue to spread during $Spread\_Time$ but without exchanging bitmap messages.
The time sequence diagram of OA-DVFA is given in Figure 6.19.

6.4.2.4 Performance Evaluation

For this performance evaluation we consider an area of $500mx500m$ with obstacles occupying 20% of this area. We distinguish two topologies: Topology 1 (see Figure 6.20a) with only one obstacle and Topology 2 with 6 obstacles (see Figure 6.20b). Both topologies have the same total surface of obstacles. These two topologies allow us to evaluate the impact of the number, shape and position of obstacles. The parameters used in the simulations are presented in Table 6.2. We evaluated the performance of OA-DVFA in terms of coverage rate and distance traveled. We also evaluate the number of active nodes. The coverage rate and the total distance traveled by nodes are computed as shown in Sections 6.2.2.3 and 6.2.2.4.

Figure 6.22a and Figure 6.22b illustrate the coverage rate as a function of time when all obstacles are known (see Figure 6.22a) and when all the obstacles are unknown (see

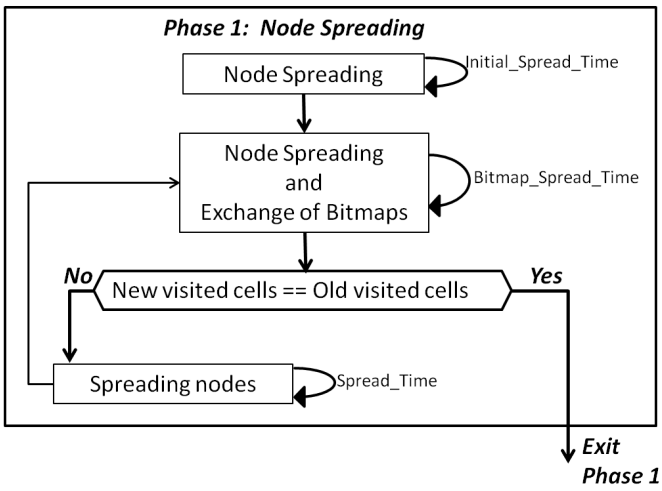
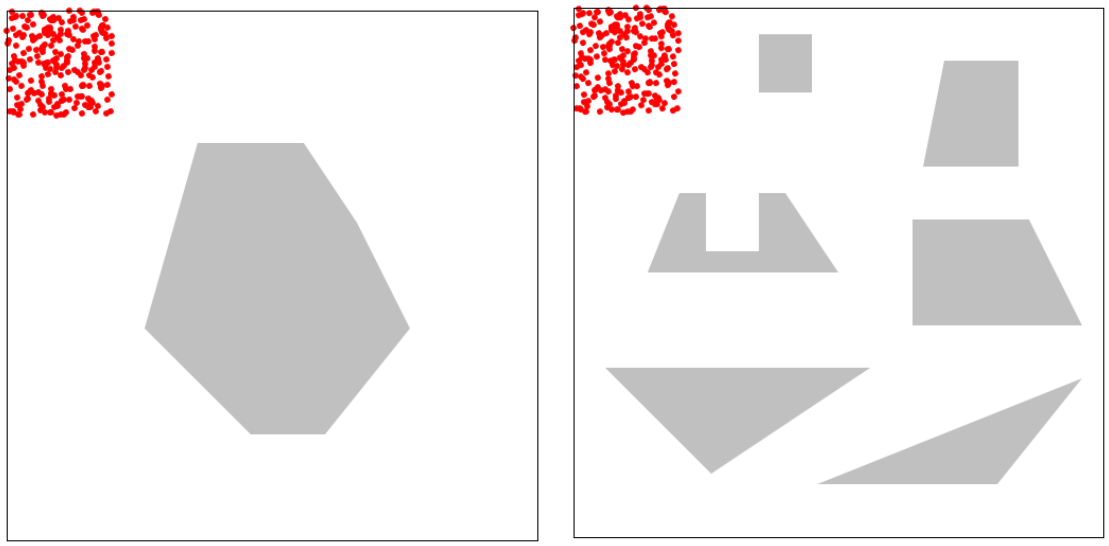


Figure 6.19: Unknown obstacles: OA-DVFA.

Table 6.2: Simulation parameters

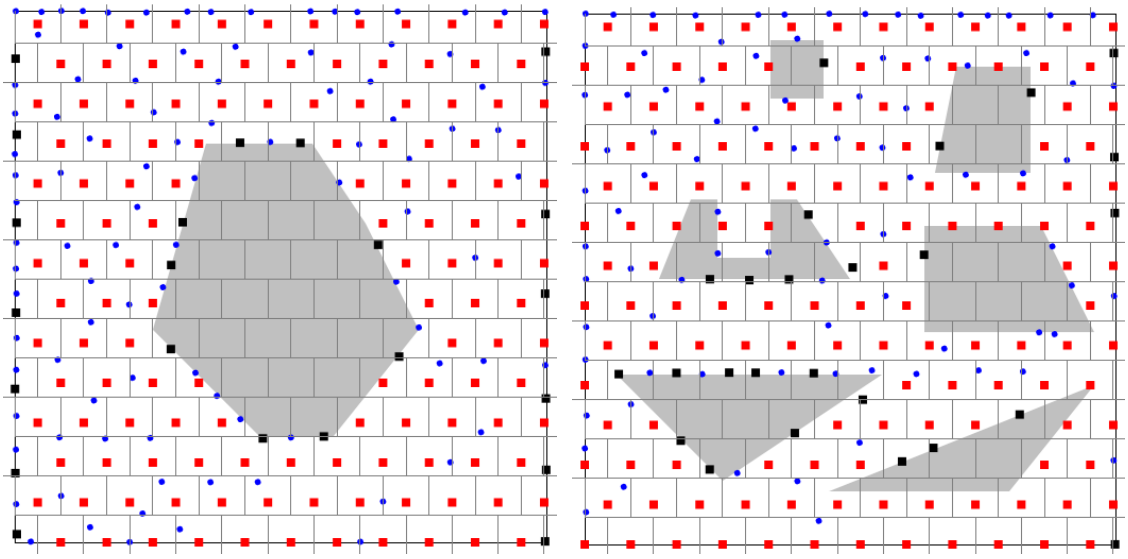
Topology	
Sensor nodes	250 grouped in the corner close to the sink
Area size	500m x 500m
Speed	2m/s
Simulation	
Result	average of 30 simulation runs
Simulation time	5000s or 8000s
MAC	
Protocol	IEEE 802.11b
Throughput	2 Mb/s
Radio range R	50 m
Sensing range r	25 m
OA-DVFA	
K_a	0.001
K_r	0.56
Hello period	2.9s
$Lmax$	$5m$
$Initial\_Spread\_Time$	500s
$Bitmap\_Spread\_Time$	100s
$Spread\_Time$	200s

Figure 6.22b). In each figure, the coverage rate is depicted for Topologies 1 and 2. We observe in Figure 6.22a that full coverage is reached in both topologies when all the obstacles are known. As expected, Topology 2, the most complex topology, requires a



a Topology 1. b Topology 2.

Figure 6.20: Intial deployment.

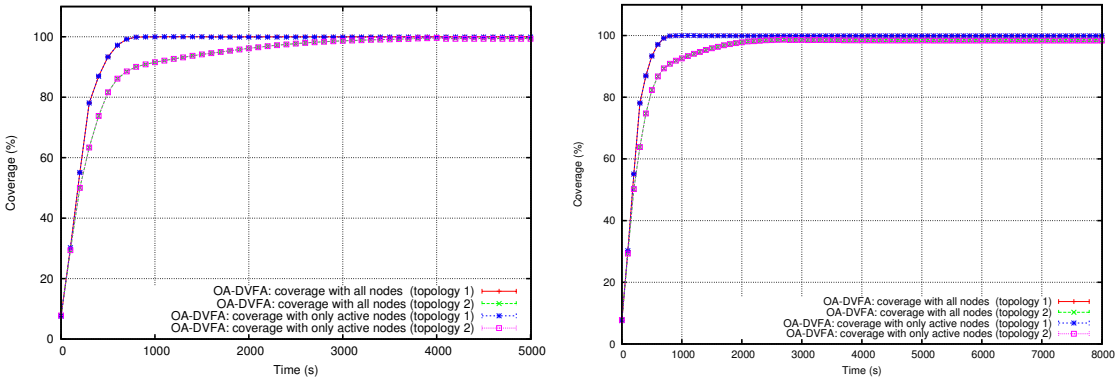


a Topology 1. b Topology 2.

Figure 6.21: Final deployment.

longer time (i.e. 4000s) to reach a 100% coverage rate, whereas the deployment in Topology 1 is much faster, requiring only 1000s. This highlights the impact of the number, shape and position of obstacles.
When we focus on unknown obstacles (in Figure 6.22b), full coverage is reached with Topology 1. With Topology 2, OA-DVFA provides a coverage rate of 98%, which is a very good result for a complex topology.

When obstacles are unknown, sensor nodes do not know the number of virtual cells that should be covered (i.e they do not know how many cells are occupied by obstacles). Since Topology 2 is complex, the stopping condition in the node spreading phase of OA-DVFA may be true even if all the cells have not yet been visited. OA-DVFA stops even if coverage is 98%. This can be explained by the following observation. At the beginning of the algorithm, the density of the nodes is high and so the repulsive forces are high. Hence, the spreading of nodes is quick. Closer to the stability point, the virtual forces are lower and so the spreading of the nodes becomes slow. In addition, the spreading of the nodes can be slowed down by the presence of obstacles that create a narrow lanes in the area considered. To limit the distance traveled, and hence the energy consumed by the nodes, we prefer to stop sensor nodes prematurely rather than allowing them to move for a longer time and only gaining 2% of coverage.
As a conclusion, OA-DVFA succeeds in providing a very good coverage rate, even when obstacles are discovered dynamically. As expected, obtaining a high coverage rate requires more time when the obstacles are unknown.

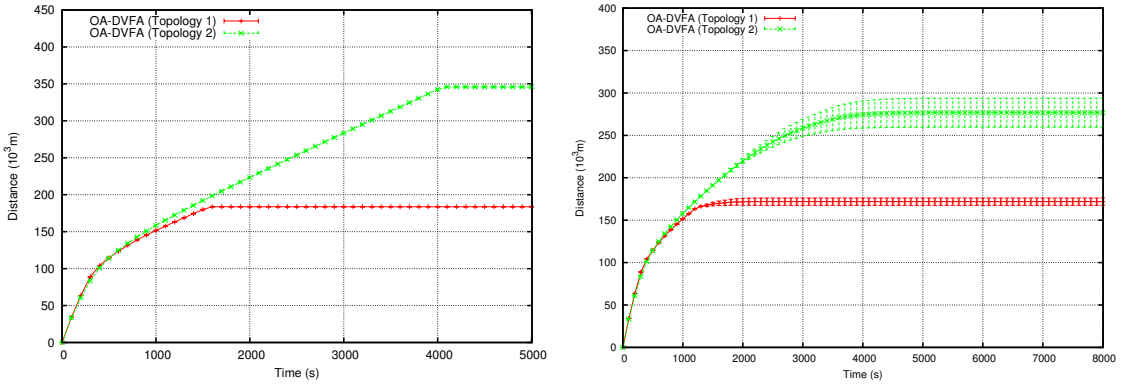


a Known obstacles. b Unknown obstacles.

Figure 6.22: Coverage as a function of time.

We now focus on the distance traveled by nodes. We depict the accumulated distance traveled by all nodes during their deployment. We can see that in Figure 6.23a when the obstacles are known, and in Figure 6.23b when the obstacles are unknown, all the nodes stop moving according to Phase 2 of OA-DVFA. Hence the total distance traveled remains constant after this time. We conclude that OA-DVFA avoids node oscillations, an inherent drawback in virtual forces-based algorithms.

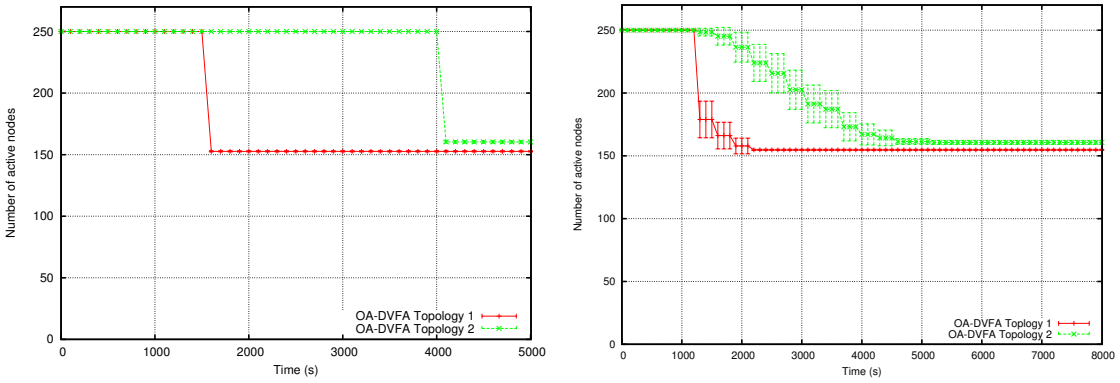
Since the area may contain unknown obstacles, the number of sensor nodes required cannot be computed in advance. Consequently, the number of sensor nodes initially present is higher than the number that is actually necessary.
To save energy, OA-DVFA includes node activity scheduling where only nodes needed to ensure full area coverage are active, and the others switch to a sleep state. As illustrated in Figure 6.22a and Figure 6.22b, the coverage rate obtained by only active nodes (i.e. 151 active nodes in Topology 1 and 155 active nodes in Topology 2 in Figure 6.24a) is the same as if all the nodes (i.e. 250 nodes for both topologies in Figure 6.24a) were active.



a Known obstacles. b Unknown obstacles.

Figure 6.23: Total Distance traveled by nodes as a function of time.

When the obstacles are unknown, we get results that are very close to those with known obstacles, in terms of the number of active nodes as depicted in Figure 6.24b. However, the process may take more time.
Considering Phase 2 and Phase 3, we can conclude that OA-DVFA is an energy-efficient self-deployment algorithm.



a Known obstacles. b Unknown obstacles.

Figure 6.24: Number of active nodes.

6.4.3 Summary

In this section, we proposed the OA-DVFA algorithm that spreads sensor nodes over the whole area as quickly as possible, while avoiding known and unknown obstacles and stopping node oscillations.

In a 3D space, the virtual force strategy can be used to spread sensor nodes over a volume (e.g. such as a cube). In the following section we show how to extend virtual forces strategy to perform in 3D.

6.5 How to use virtual forces in 3D

The improvements made in WSNs have led to the emergence of new networks known as Mobile WSNs. These mobile WSNs are better able to meet the requirements of more realistic applications such as 3D applications.
Some 2D deployment approaches could be extended to perform in 3D space. The virtual forces strategy, or our DVFA, can be extended to operate in 3D space.
The 3D virtual forces strategy is like the 2D strategy: each sensor node can exert attractive and repulsive forces, on its neighbors, the strength of the force being dependent on the distance separating them. If this distance is higher than the distance threshold D_{th}, then an attractive force is exerted. If it is smaller than D_{th}, then a repulsive force is exerted. Otherwise, the exerted force is null. Each node moves according to the resultant force. Figure 6.25 illustrates an example of virtual forces exerted on sensor S_1 in 3D space.

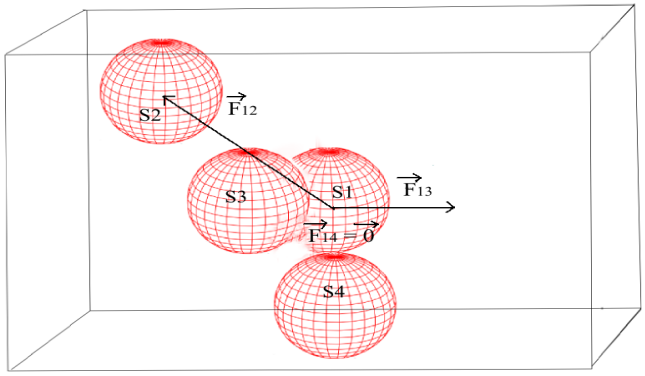


Figure 6.25: Example of 3D virtual forces exerted on node S1

As shown in Chapter 4, in a 3D space, the sensing zone and the communication zone of a sensor node are assumed to be spheres centered on the sensor node and of radius r and R, respectively.

6.5.1 3D-DVFA: 3D Distributed Virtual Forces Algorithm

The 3D-DVFA algorithm like DVFA is based on virtual forces strategy. However, it aims to deploy sensor nodes in 3D space to ensure coverage and maintain network connectivity.

6.5.1.1 3D-DVFA principles

The 3D-DVFA algorithm works as follows. Each sensor node within the network runs the following algorithm that proceeds by iterations but does not require node synchronization. Let s_i denote any sensor node and (x_i, y_i, z_i) be its coordinates. At each iteration, each node broadcasts a *Hello* message. In the *Hello* message, each node sends its position and the node it hears in order to perform the neighborhood discovery. Then, each sensor node is able to determine its 1-hop neighbors and 2-hop neighbors, and compute its new position according to the forces exerted on itself by its 1-hop and 2-hop neighbors.

Let d_{ij} denote the Euclidean distance between the sensor nodes s_i and s_j. d_{ij} is given by $\sqrt{(x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2}$.
The force exerted by sensor s_j on sensor s_i is

- an attractive force if $d_{ij} > D_{th}$ and is given by:

$$\vec{\mathcal{F}_{ij}} = K_a \cdot (d_{ij} - D_{th}) \cdot \frac{(x_j - x_i, y_j - y_i, z_j - z_i)}{d_{ij}}$$

- a repulsive force, if $d_{ij} < D_{th}$ and is equal to

$$\vec{\mathcal{F}_{ij}} = K_r \cdot (D_{th} - d_{ij}) \cdot \frac{(x_i - x_j, y_i - y_j, z_i - z_j)}{d_{ij}}$$

- null otherwise, $(d_{ij} = D_{th})$

Hence, the resultant force $\vec{\mathcal{F}_i}$ on s_i is computed as the sum of these forces exerted by its 1-hop and 2-hop neighbors:

$$\vec{\mathcal{F}_i} = \sum_j \vec{\mathcal{F}_{ij}}$$

Then, node s_i moves according to the resultant force to its new position. The new position of sensor s_i is given by $(x'_i\ , y'_i, z'_i\ )$ with $x'_i = x_i + F_{ix}$, $y'_i = y_i + F_{iy}$ and $z'_i = z_i + F_{iz}$.

Since the role of D_{th} is very important in the principle of virtual forces, it should be well tuned. In 2D deployment, D_{th} is computed according to the optimal deployment based on the triangular tessellation where each node, located at a triangle vertex, has 6 neighbors. However, in 3D space, the optimized deployment is provided by the truncated octahedron tessellation (55). The truncated octahedron (see Figure 6.26b) has 14 faces, of which 8 are regular hexagons, and 6 are squares, so, a node has 14 neighbors. If the neighbors are adjacent on a square face, the target distance is equal to $4R_s/\sqrt{5}$. In contrast, if they are adjacent on an hexagonal face, the target distance is $2R_s\sqrt{3}/\sqrt{5}$. When the truncated octahedron is used, two target distances are maintained between neighboring nodes depending on their respective positions. However, since in the virtual forces strategy, only one target distance is maintained between neighboring nodes, we do not adopt the truncated octahedron tessellation as a deployment pattern. We prefer a regular tesselation requiring a unique D_{th}.
The regular dodecahedron(see Figure 6.26a) is a regular polyhedron composed of 12 equally sized regular pentagons. Since it is regular, a node in the center of the dodecahedron has 12 neighbors at the same distance. In our study, we adopt the regular dodecahedron to determine the target distance in order to apply the virtual forces strategy. Figure 6.27 shows the regular dodecahedron tesselation where the node in the center of the dodecahedron has 12 neighbors.

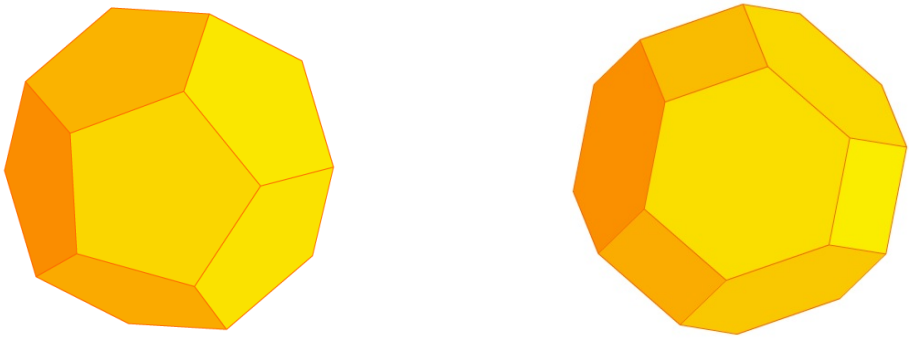
Let a be the edge length of a regular dodecahedron. The radius of the circumscribing sphere that intersects the dodecahedron at all vertices, represents the sensing range r and

is equal to:

$$r = a\frac{\sqrt{3}}{4}(1+\sqrt{5}). \tag{6.4}$$

The target distance is equal to:

$$Dth = a\sqrt{\frac{5}{2}+\frac{11}{10}\sqrt{5}}. \tag{6.5}$$



a Regular Dodecahedron. b Truncated octahedron.

Figure 6.26: 3D geometric shapes.

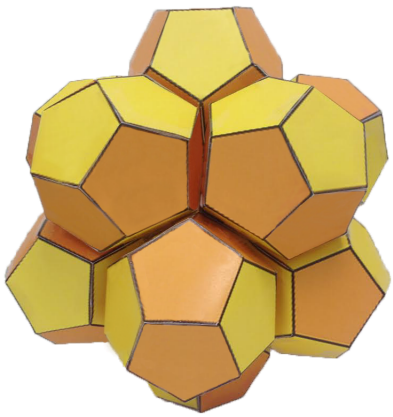


Figure 6.27: Regular Dodecahedron tesselation

The regular dodecahedron is, on the one hand, a regular shape that fits the virtual forces strategy and, on the other hand it, minimizes the total number of nodes needed to ensure full coverage of 3D area, as shown in Table 5.1 of Chapter 5.

6.5.1.2 Performance evaluation

To evaluate performances of the 3D-DVFA algorithm, we implemented it in the Network Simulator NS3 version 3.20.

Table 6.3 illustrates the simulation parameters used to evaluate the 3D-DVFA algorithm.

Table 6.3: Simulation parameters

Toplogy	
Sensor nodes	250
Area size	100mx100mx100m
MAC Layer	
Protocol	IEEE 802.11b
Throughput	2 Mb/s
Transmission range R_c	26m
Sensing range R_s	14m
Simulation	
Result	average of 20 simulation runs
Simulation time	500s
3D-DVFA	
Ka	0.004
Kr	0.25
Hello period	2.4s
Dth	22.27m
$Distmax$	4

We evaluate the coverage ratio, provided by the 3D-DVFA algorithm, with regards to initial configuration and number of nodes deployed.
To determine the coverage ratio, we divide the whole space into $100m$x$100m$x$100m$ unit cubes. Hence, the coverage ratio is calculated as the ratio of unit cubes whose center is covered by at least one sensor.
As 3D-DVFA algorithm is run by mobile and autonomous sensor nodes, the energy consumption due to sensor moves may be high. That is why, we evaluate the total distance traveled by nodes.

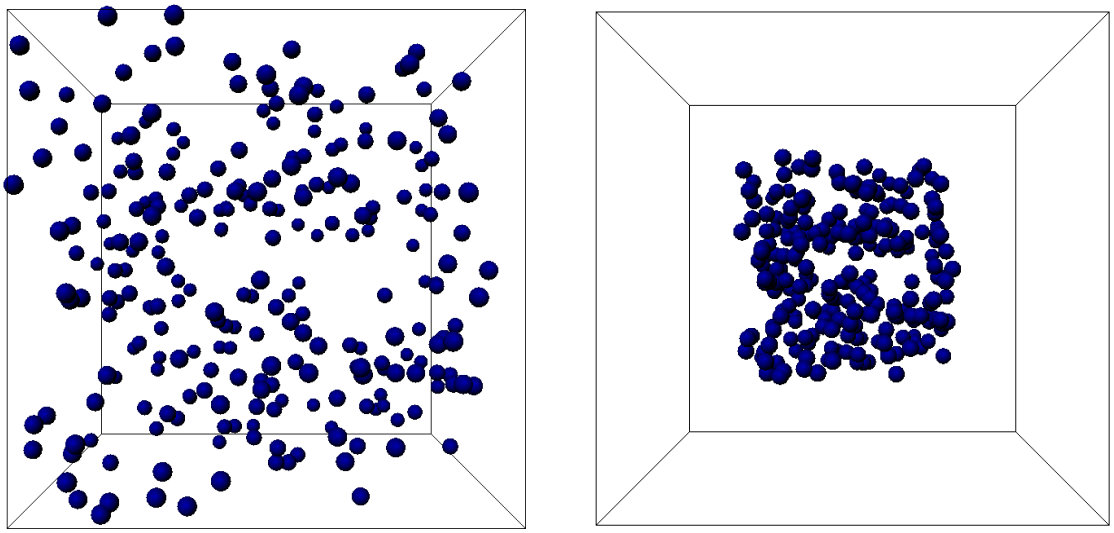
We evaluate the coverage rate and the distance traveled using two initial configurations:

- Random configuration: Sensor nodes are scattered randomly in the whole space. See Figure 6.28a.
- Centered configuration: Sensor nodes are grouped initially around the center of the space, i.e $x \in [25, 75], y \in [25, 75], z \in [25, 75]$. See Figure 6.28b.

Figure 6.29 illustrates the final deployment using 250 nodes, obtained with both initial configurations.

Evaluation of the coverage rate

Figure 6.30a illustrates the coverage rate obtained with 3D-DVFA as a function of time using 250 nodes in random configuration and centered configuration. In both configurations, sensor deployment based on 3D-DVFA provides full coverage of the 3D area considered (see



a Initial random configuration. b Initial centered configuration.

Figure 6.28: Initial Configurations.

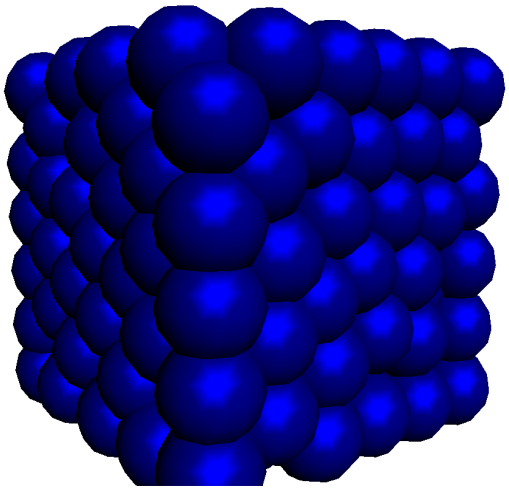
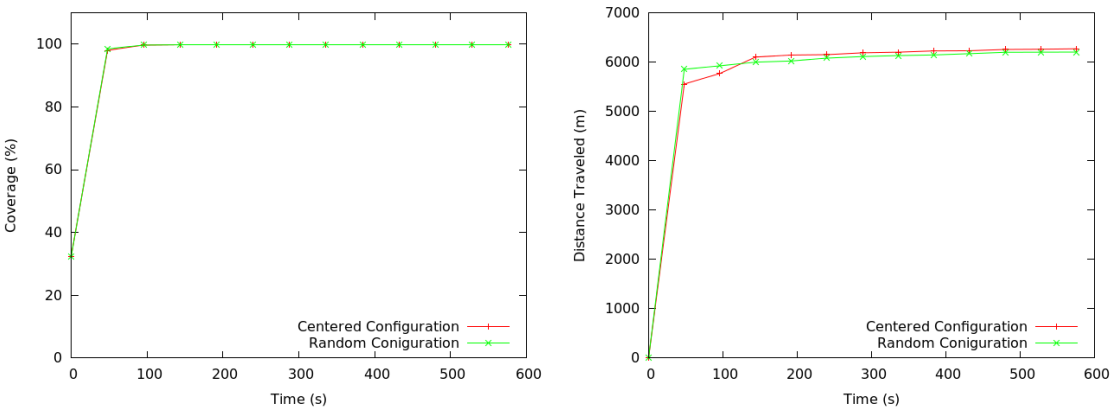


Figure 6.29: 3D deployment based on Virtual forces and regular dodecahedron.

Figure 6.29). Figure 6.30a shows that the coverage rate reaches $99,99\%$ at time $t = 150s$ in both configurations and this rate is still maintained during the remaining simulation time. This can be explained by the spreading of nodes caused by virtual forces. Due to virtual forces, sensor nodes are able to spread in the whole 3D area, reach very quickly full coverage, while maintaining network connectivity.



a Coverage rate as a function of time. b Total distance traveled by nodes.

Figure 6.30: Performance evaluation of 3D-DVFA.

Evaluation of the total distance traveled by nodes

Figure 6.30b illustrates the total distance traveled by nodes as a function of time. The value presented in Figure 6.30b is the cumulative distance. We can observe that the distance traveled by nodes increases rapidly until $t = 100s$ with random configuration and $t = 150s$ with centered configuration, after this time the curve of the distance traveled for both configurations increases very slowly. Then, when full coverage has not been reached yet, nodes move in order to reach their final positions. However, when full coverage is ensured, nodes oscillate due to, for instance, border effects and the number of nodes deployed that may be higher than the optimal one.
As expected, the distance traveled by nodes in random configuration is less than the one obtained in the centered configuration. This can be explained by the fact that in the centered configuration sensors are grouped in the center of the 3D area, and in the random configuration they are already spread in the whole 3D area. Thus, sensor nodes move much more with the centered configuration to reach full coverage than with random configuration.

6.6 Conclusion

In this chapter, we focused on the deployment of autonomous sensor nodes in the area considered. We adopted the virtual forces strategy to move mobile sensor nodes. We dealt with three problems, the first being related to the virtual forces strategy and its main drawback: node oscillations. We proposed two deployment algorithms ADVFA and GDVFA that overcome with this drawback. The ADVFA algorithm reduces node oscillations by adapting the target distance maintained between neighboring nodes to the total number of connected nodes. The ADVFA algorithm has been published in (57). GDVFA stops node oscillations by using the grid-based strategy. Then, sensor nodes are encouraged to occupy cell centers and stop moving. The GDVFA algorithm has been published in (58). The second problem studied in this chapter is the presence of known or

unknown obstacles in the area. We proposed OA-DVFA, a virtual forces based deployment that autonomously deploys sensor nodes while discovering and avoiding obstacles. OA-DVFA is a deployment algorithm that deals with both the node oscillation problem and the presence of obstacles. The third problem studied is the use of virtual forces in a 3D space. We proposed 3D-DVFA that deploys sensor nodes in a cube based on virtual forces. The 3D-DVFA algorithm has been published in (59). Table 6.4, summarizes the algorithms proposed in this chapter.

The deployment of autonomous sensor nodes is very important and useful in some situations, such as damage assessment and hostile environments (e.g. a radioactive zone). However, a high number of autonomous and mobile sensor nodes may be too expensive. For this reason, in the next chapter we focus on assisted deployment where sensor nodes are static and a human or a mobile robot is in charge of placing them in their appropriate positions which been computed previously. We focus not only on how to find these positions but also on how to optimize the trajectory of a robot responsible for visiting these positions and placing sensor nodes.

Table 6.4: Autonomous deployment for area coverage and network connectivity

	Area	**Obstacles**	**Energy-efficiency**	**Fault-tolerant** with regard to coverage and connectivity
ADVFA	unknown		– reduces node oscillations – adapts to the number of nodes	– link and node failures – disconnected network component
GDVFA	known		– stops node oscillations	–link and node failures
OA-DVFA	unknown	known/ unknown	– stops node oscillations	– link and node failures
3D-DVFA	known	no	no	– link and node failures

Part IV

Assisted deployment

Chapter 7

Optimized deployment in the presence of obstacles

Contents

7.1 Introduction

In this chapter, we focus on two types of coverage problem: area coverage and Points of Interest (PoIs) coverage.

- For the area coverage problem, our goal is to deploy wireless sensor nodes in an arbitrary, realistic area of irregular shape, and containing obstacles that may be opaque. In Section 7.2, we propose a simple projection-based method, called OAD-Area, that tends to minimize the number of sensor nodes needed to fully cover such an area.

- For the PoIs coverage problem, we aim to ensure a fault-tolerant connectivity between each PoI and the sink, while minimizing the total number of relays deployed and the length of each path between the PoIs and sink. Obstacles are also taken into account. The problem is called RNP: Relay Nodes Placement and is described in Section 7.3. In order to achieve our goal, we propose a solution based on the optimal deployment, called OAD-PoI. Each position in the optimal deployment may be a position for relay node placement. This solution ensures fault-tolerant connectivity, even in the presence of obstacles.

7.2 First problem: full area coverage and connectivity

In our work, we consider wireless sensors that must be deployed to fully cover a given 2D area of irregular shape with the presence of obstacles.

Our goal is to minimize the number of sensors needed to achieve full coverage of the area given, denoted $\mathcal{A}$. Full coverage of $\mathcal{A}$ means that any event occurring in $\mathcal{A}$ is detected by at least one sensor node. The deployment of wireless sensor nodes is computed by a single entity that takes as inputs the vertices of the polygon defining $\mathcal{A}$, as well as for each obstacle, the vertices of its polygon.

We consider transparent and opaque obstacles (see Chapter 4). Opaque obstacles are much more complex to handle than transparent ones and require the deployment of additional sensors to eliminate coverage holes. That is why in this section, we focus on the deployment of wireless sensor nodes in an irregular area with transparent and opaque obstacles and propose a projection-based method, Optimized Assisted Deployment to monitor an Area (OAD-Area), that tends to minimize the number of sensor nodes needed to fully cover this area.

7.2.0.3 Related work

The vast majority of approaches encountered in the literature adopt the optimal deployment based on triangular tessellation. Then, sensors nodes located within an obstacle or outside the border of the area to cover are eliminated. This elimination usually causes coverage holes. The existing approaches differ in the way they heal the coverage holes. We distinguish the following two approaches:

• *Contour-based approaches* like (60; 61): these approaches deploy sensor nodes at a constant distance along the border of the area and along the border of each obstacle in order to heal coverage holes occurring on these contours. The distance between two successive sensor nodes deployed successively on a given contour is computed from the sensing range. Such approaches are simple but may require a high number of sensors when there are many irregular borders, as shown in (29). In contrast to the method we propose, coverage holes that are not adjacent to the area border or the obstacle border are not detected as shown in Figure 7.1.

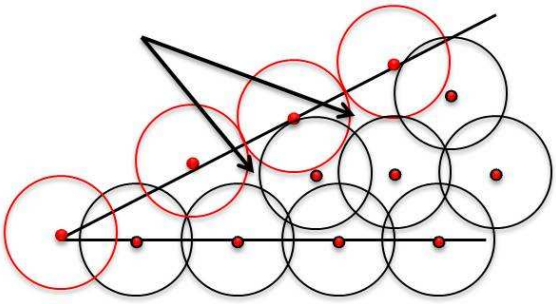


Figure 7.1: Coverage hole that is undetected by a contour-based method.

• *Delaunay-triangulation-based approaches* like (29): these approaches use Delaunay

triangulation to detect coverage holes, and then place sensor nodes at some vertices of the triangles defined using a vertex coloring technique. However, the resulting complexity may be high, due to the presence of two modules: (a) determining of coverage holes followed by (b) computing sensor locations, which may be greedy in terms of computation resources. In contrast to this approach, our method determines the sensor locations without searching for coverage holes. To reduce the number of sensors, our method eliminates redundant sensor nodes.

7.2.1 Optimized deployment in an irregular area

In this section, we propose a deployment algorithm to cope with the irregular shape of an area. In this first coverage problem, we consider any irregular 2D area and assume that there are no obstacles and that the border of the area is transparent.

7.2.1.1 Principle

Our projection-based method proceeds as follows:

1. We start with the optimal deployment in the rectangle circumscribing the given area $\mathcal{A}$, see Figure 7.3a.

2. Sensor nodes that are outside $\mathcal{A}$ are eliminated, which may cause coverage holes: see Figure 7.3b.

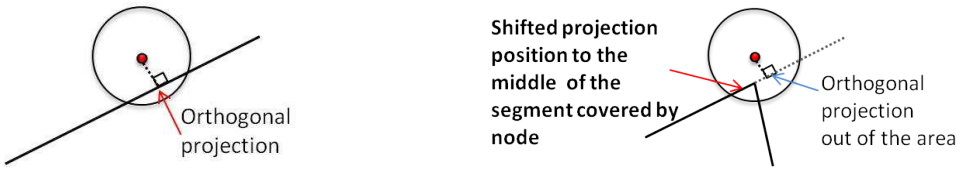
3. For each sensor node s located outside the area at a distance strictly less than r from a border, we check whether the border segment initially covered by s is still covered by other sensor nodes within $\mathcal{A}$, even if s is eliminated. Otherwise s is orthogonally projected on the border, see Figure 7.3c. Due to this projection technique, illustrated in Figure 7.2, we can guarantee that the zone initially covered by the eliminated sensor node s, is still covered after the projection of s.

4. Finally, to optimize the number of sensor nodes needed, we check whether some of them are providing redundant coverage, in which case, they can be eliminated in that case. They can be eliminated if and only if the intersection of $\mathcal{A}$ and the zone they covered is fully covered by other sensor nodes that are retained (see Figure 7.3d).

It should be noted that the projection of a sensor node is not always on the border considered as shown in Figure 7.2b. In this case, the position of the projected node is shifted to the middle of the border segment covered by this node in order to heal coverage holes.

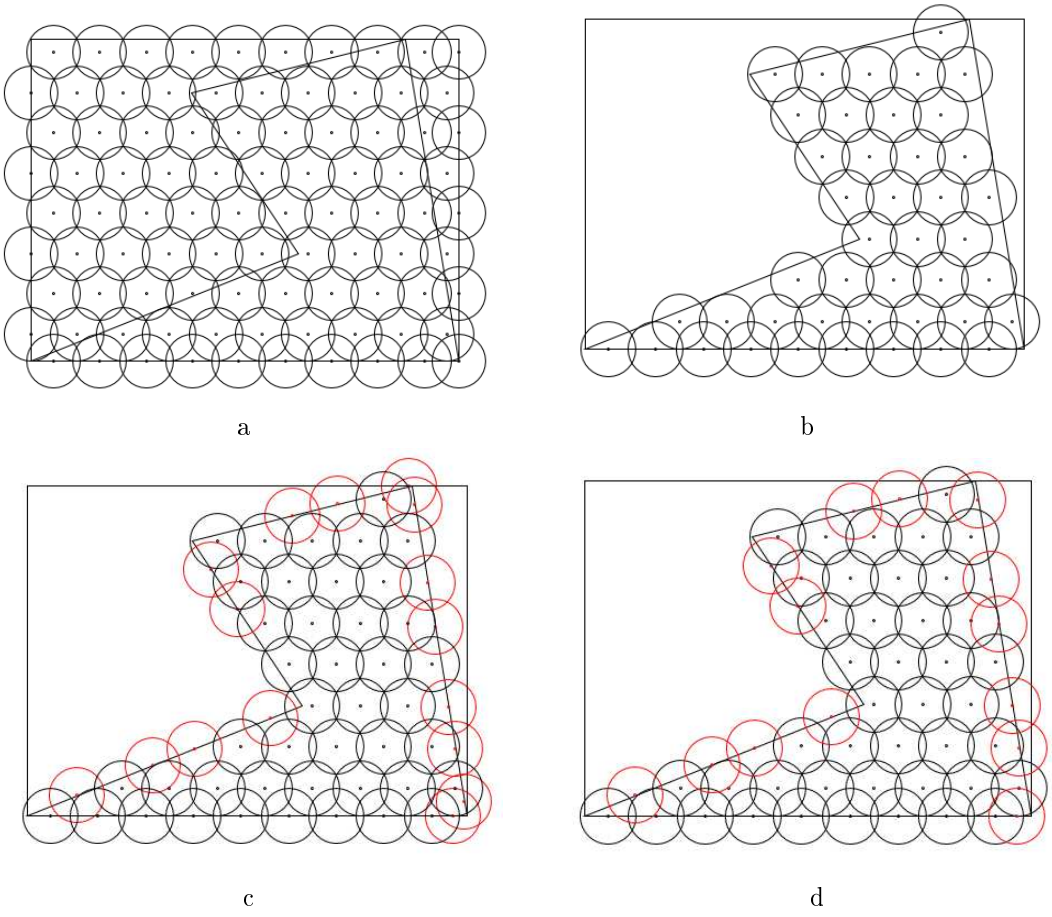
7.2.1.2 Upper bound on the number of sensors required

We now establish $OurMax_N$, an upper bound on the number of sensors needed by our method to fully cover an irregular-shaped area with transparent borders. This bound does not take into account the elimination of redundant sensor nodes done in step 4. Let $Outr$ denote the set of sensor nodes outside $\mathcal{A}$ at a distance less than r from a border and N_{Outr}



a b

Figure 7.2: Projection technique.



a b

c d

Figure 7.3: Principles of our method.

denote the cardinality of $Outr$. We recall that N_{In} denotes the number of sensor nodes within $\mathcal{A}$. Since the sensor nodes that are added come from the projections of nodes in $Outr$ and these projections are done only on edges whose distance is less than r, we then have our upper bound:

$$OurMax_N = N_{In} + \sum_{P \in Outr} \sum_{e \in edge(\mathcal{A})} 1_{distance(P,e)<r} \tag{7.1}$$

with $1_{distance(P,e)<r} = 1$ if $distance(P, e) < r$ and 0 otherwise.

7.2.1.3 Comparative evaluation

In this subsection, we compare the performances of our method (projection-based) with the contour-based method on the boot configuration depicted in Figure 7.3d. This boot configuration has a circumscribing rectangle of size $20r$ x $18r$. In Sections 7.2.1 and 7.2.2, we study more complex configurations.
The contour-based method chosen applies the method explained previously to each edge of $\mathcal{A}$. Hence, it needs N_b sensor nodes to fully cover an edge of length L (see Equation 8.1).

We vary the sensing range (i.e. r, $r/2$, $r/4$), whereas the dimensions of the area are kept constant. We study the impact on the number of sensor nodes required to fully cover the area. As expected, when the sensing range decreases from r to $r/2$, the number of sensor nodes increases from 57 to 184. The lower bound Min_N suggests that this number should be multiplied by 4. We observe a multiplication by 3.23. Respectively, when the sensing range decreases from r to $r/4$, the number of sensor nodes increases from 57 to 652. Similarly, the lower bound Min_N suggests that this number should be multiplied by 16. We observe a multiplication by 11.46. This can be explained by the irregularity of the border. The contour-based method adds 35 nodes on the border, whereas our method adds only 14 nodes. Hence, the contour-based method leads to a total number of sensor nodes of 78, which is higher than is required by our method (i.e. 57 nodes).

We observe that the upper bound $OurMax_N$ is very close to the number of sensor nodes needed by our method: we obtain an upper bound of 60, 188 and 656 whereas the exact number is 57, 184 and 652. We also notice that the gap between the real number of sensor nodes and the number given by the contour-based method drastically increases when the sensing range decreases. It reaches 21, 39 and 73 sensor nodes when the sensing range is equal to r, $r/2$ and $r/4$. This would increase the deployment cost by a factor of 36% for a sensing range r, for instance.

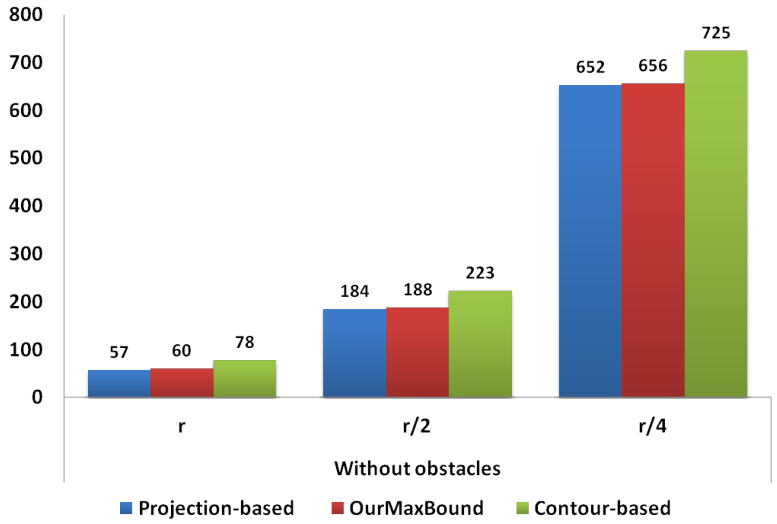


Figure 7.4: Impact of the sensing range on the number of sensor nodes (Boot configuration without obstacles).

7.2.2 Optimized deployment in an irregular area with opaque obstacles

In this section, we propose a deployment algorithm to cope with the hidden zone due to opaque obstacles. In this third coverage problem, we consider any irregular 2D area that includes obstacles and assume that some obstacles and/or some borders of the area are opaque. This may result in hidden zones (see Figure 4.2), and sensor nodes must be added to cope with them.

7.2.2.1 Principle

In the presence of opaque borders or opaque obstacles, our method checks whether a hidden zone (see step 4 below) exists. If so, sensor nodes are added. More precisely, the method proceeds according to the following steps:

1. We start with the optimal deployment in the rectangle circumscribing the given area $\mathcal{A}$, see Figure 7.5a.

2. Sensor nodes that are outside $\mathcal{A}$ or inside the obstacles $\mathcal{O}$ are eliminated, which may cause coverage holes, see Figure 7.5b.

3. For each sensor node s outside the area at a distance strictly less than r from a border of the area, we check that the border segment initially covered by s is still covered by sensor nodes within $\mathcal{A}$, even if s is eliminated. Otherwise s is orthogonally projected on the border. We proceed similarly with any sensor node s inside an obstacle at a distance strictly less than r from a border of the obstacle, see Figure 7.5c.

4. For each sensor node s remaining after step 2, we check whether it is the only sensor node covering a zone in $\mathcal{A}\backslash\mathcal{O}$ that becomes hidden because of the opacity of a border or an obstacle. If so, a new sensor node is added as the projection of s in the zone it should cover (see Figure 7.5d).

5. Finally, redundant sensor nodes are eliminated.

7.2.2.2 Upper bound on the number of sensors required

We now extend our previous bound on the maximum number of sensor nodes needed by our method in the presence of opaque obstacles or opaque borders. To deal with obstacles, our method projects nodes within an obstacle at a distance less than r from an edge of the obstacle. That is why, we add a third term to account for obstacles.

We also add a fourth term to deal with opaque borders and opaque obstacles.

$$OurMax_N = N_{In} + \sum_{P \in Outr} \sum_{e \in edge(\mathcal{A})} 1_{distance(P,e)<r} +$$

$$\sum_{P \in InObstr} \sum_{e \in edge(\mathcal{O})} 1_{distance(P,e)<r} +$$

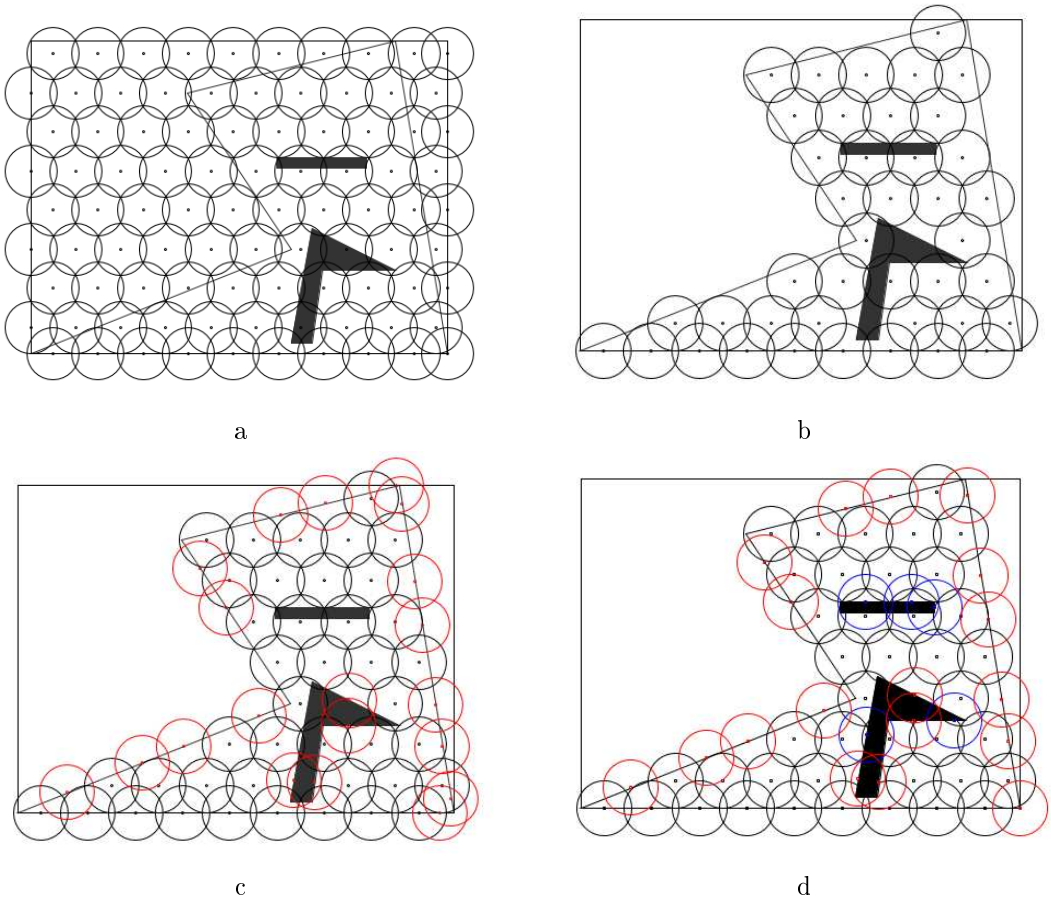


Figure 7.5: Principles of our method.

$$\sum_{P \in In} \sum_{e \in Opaque_edge(\mathcal{A} \cup \mathcal{O})} 1_{distance(P,e)<r}, \tag{7.2}$$

where In denotes the set of sensor nodes that remain after the elimination carried out in step 2 and $1_{distance(P,e)<r} = 1$ if $distance(P, e) < r$ and 0 otherwise.

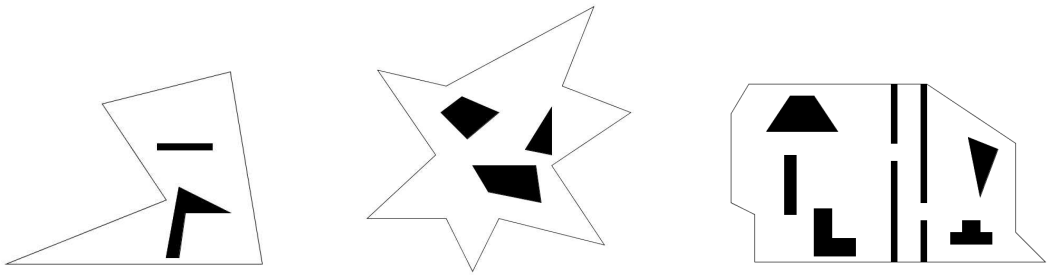
7.2.2.3 Comparative evaluation

We consider different configurations with opaque obstacles to compare our method with the contour-based method. The configurations are varied:

- The boot configuration with obstacles, (see Figure 7.6a) with the circumscribing rectangle of size $20r$ x $18r$. This configuration is the simplest one we study.

- The star configuration. This configuration is representative of an area having a complex shape with many salient angles. Its circumscribing rectangle is of size $24r$ x $28r$ (see Figure 7.6b).

- The warehouse configuration, (see Figure 7.6c) with the circumscribing rectangle of size $28r$ x $18r$. This configuration is representative of an indoor area with several rooms and many obstacles.



a b c

Figure 7.6: Configurations studied (Boot, Star, Warehouse).

The contour-based method needs to deploy 93 sensor nodes in the boot configuration, 140 sensor nodes in the star configuration and 197 in the warehouse configuration. Notice that the contour-based method does not distinguish between opaque and transparent obstacles.

In the boot configuration, our method needs only 64 sensor nodes, 5 of which are used just to cover hidden zones. These sensor nodes are depicted in blue in Figure 7.5d. Our method out performs the contour-based method by 48%.

In the star configuration, our method needs only 105 sensor nodes, 4 of which are used to cover hidden zones. Our method saves 33% of the deployment cost compared to the contour-based method.

In the warehouse configuration, our method needs 134 sensor nodes, 22 of which are added to cover hidden zones. Our method saves 47% of the deployment cost compared to the contour-based method.

When we vary the sensing range from r to $r/2$ and $r/4$, we still observe that our method outperforms the contour-based method, as depicted in Figures 7.7, 7.8 and 7.9. The bound $OurMax_N$ always provides a very good approximation of the number of sensors required by our method.

The comparative evaluation reported in Sections 7.2.1 and 7.2.2 has the merit of quantitatively evaluating the impact of the complexity of the area (i.e. with/without obstacles, opaque/transparent borders, opaque/transparent obstacles) on the number of sensor nodes needed to obtain full coverage. The bound we computed $OurMax_N$ is very accurate, whatever the configuration, and our method always outperforms the contour-based method. Furthermore, we noticed the strong impact of border edges and obstacle edges whose length is smaller than $r\sqrt{3}/2$ on the number of edges required by a contour-based method.

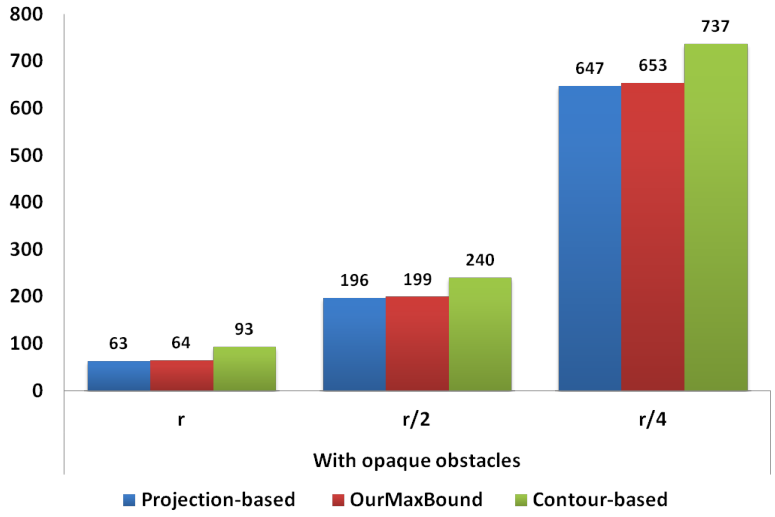


Figure 7.7: Impact of the sensing range on the number of sensor nodes (Boot configuration).

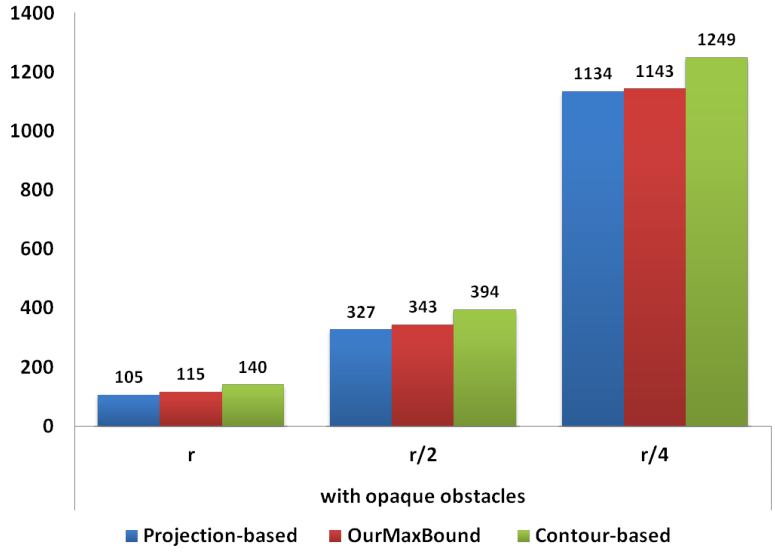


Figure 7.8: Impact of the sensing range on the number of sensor nodes (Star configuration).

7.2.3 Summary

In this section, we focused on ensuring full coverage and connectivity of a given area. We proposed OAD-Area, a projection-based method, that computes an optimized deployment of a given area containing obstacles. We showed that OAD-Area optimizes the number of nodes deployed in complex configurations. This is due to its principle which is based on the optimal deployment.

In the following section, we are interested in ensuring coverage and connectivity of

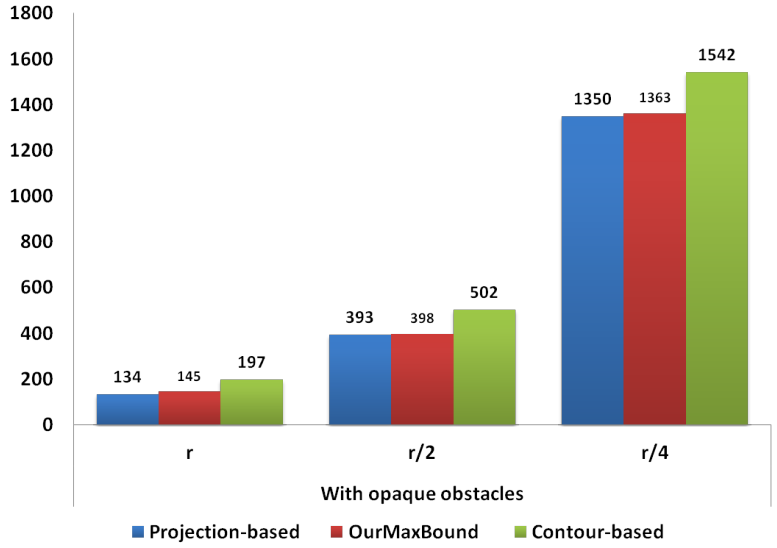


Figure 7.9: Impact of the sensing range on the number of sensor nodes (Warehouse Configuration).

Points of Interest. We propose OAD-PoI which is also based on the optimal deployment to ensure fault tolerant connectivity in the presence of obstacles.

7.3 Second problem: PoI coverage and connectivity

We now focus on the coverage of Points of Interest (PoIs)where the nodes sensing each PoI must be connected to the sink. In this section, we propose the Optimized Assisted Deployment to monitor PoIs (OAD-PoI). OAD-PoI is based on the optimal deployment to ensure fault tolerant connectivity while minimizing the total number of relay nodes deployed and minimizing the path length between each PoI and the sink.

7.3.1 Related work

To minimize the number of relay nodes (i.e, additional wireless nodes deployed to ensure connectivity with the sink), one strategy consists in using the property of the Steiner point in a triangle. For instance, the algorithm given in (62) builds the minimum Steiner tree on the convex hull of the points of interest. It proceeds iteratively. At each iteration, the algorithm selects the points of interest that have not yet been considered and the Steiner point of the previous iteration (if any), all these points belonging to the outermost convex hull not yet considered. The Steiner point of every three consecutive points of the convex hull is computed (see Figure 7.11 for an illustrative example). The Steiner points are the optimized locations of the relay nodes. The algorithm stops when all the PoIs have been selected. Additional relay nodes are added if necessary (i.e. when the distance is greater than the communication range) on each straight line connecting each PoI to its Steiner point, as well as between any Steiner point obtained at iteration i and its Steiner point

obtained at iteration $i+1$. Many other algorithms based on the Steiner points principle exist in the literature (63).

To tolerate $k-1$ failures of wireless links or nodes, k-connectivity has been introduced to tolerate $k-1$ node or link failures. The authors of (64) focus on k-connectivity in a WSN while minimizing the number of relay nodes. The solution proposed takes advantage of overlapping node communication areas to place a relay node at the intersection of overlapping communication areas in order to achieve connectivity. Hence, this relay node is within transmission range of at least two other nodes. We will see in Section 7.3.5 how this principle is adapted to cope with obstacles.

Another study (65) focuses on the problem of deploying fault tolerant relay nodes in heterogeneous wireless sensor networks where sensor nodes and relay nodes have different communication ranges. The authors use the Steinerization of edges to create a path between two sensor nodes. The idea is to start by deploying two relay nodes, each at a distance equal to the minimum communication range between sensor nodes and relay nodes, from each path extremity. Then, additional equidistant relays are added along the remaining path between the two relays deployed. Han et al. (65) formalized the relay node placement problem that minimizes the number of relay nodes deployed to ensure that there exist $k \geq 1$ node-disjoint paths between every pair of nodes, a node being a sensor node or the sink. If $k > 1$, node placement is said to be fault-tolerant. The authors proposed approximation algorithms to solve these NP-hard problems.

Misra et al. (66) studied constrained relay node placement, where the relay nodes can only occupy a set of candidate locations and calculated the number of relay nodes needed to connect each sensor node with $k = 1$ *or* 2 sink(s) through k node-disjoint paths. If $k = 2$ the relay node placement is said to be survivable. Misra et al. (66) propose approximation algorithms to solve these problems.

However, our problem is different: we are interested in ensuring efficient connectivity between each PoI and the sink. We do not focus on connectivity between PoIs but, for reasons of efficiency we want to minimize the length of the paths connecting each PoI with the sink. Misra (66) and Han (65) do not minimize the length of the path of each PoI to the sink, but rather the total weight of the tree including all the PoIs, where the weight between two nodes is equal to the number of relays needed to ensure connectivity between them.

7.3.2 Definition of relay node placement problems

In this section we focus on wireless sensor networks deployed to cover some given points of interest, achieve connectivity with the sink and be robust against link and node failures. More precisely, we want to minimize the number of relays deployed, as well as the maximum length of paths connecting each PoI with the sink, because the transfer of any message on a longer path consumes more bandwidth and more energy, and these resources are limited in a wireless sensor network. Since the reliability of a path is equal to the product of the reliability of each link composing it, a long path is less reliable than a short one, assuming that all the links have a similar reliability. Hence, to maximize robustness, we will favor short paths from any PoI to the sink. In addition, the end-to-end

delivery delay depends on the number of hops involved. That is also why short paths are favored, provided that they are able to ensure the quality of service (QoS) required by the application.

Before defining the Relay Node Placement problem (RNP), we first define our notations.
Let $\mathcal{P}$ denote the set of PoIs that must be covered. We have $\mathcal{P} = \{P_1, P_2, P_n\}$, with $n \geq 1$.
Let P_0 be the sink.
Let R be the communication range of relays and sensor nodes.
Let $L(i)$ be the length of the path from any PoI P_i to the sink, with $i \in [1, n]$.
Let N_r be the number of relay nodes deployed to ensure connectivity of each PoI with the sink.

With regard to relay node placement, we distinguish two types of problems:

- **The relay node placement problem (RNP): to minimize the number of relay nodes deployed, as well as the maximum length of the paths connecting each PoI to the sink:**

$$min\{N_r \cdot max_{i\in[1,n]} L(i)\}. \tag{7.3}$$

  This is the simplest problem.
  We also define a variant of this problem where relay nodes cannot be placed in certain locations: relay node placement is constrained by the presence of obstacles and the border of the area considered. On the one hand, the presence of obstacles constrains the placement of relay nodes: places within an obstacle are forbidden. On the other hand, the presence of obstacles may result in hidden nodes, which may break connectivity.
  The constrained relay node placement problem (C-RNP): to minimize the number of relay nodes deployed in an area with obstacles, as well as the maximum length of paths connecting each PoI to the sink:

$$min_{obstacle}\{N_r \cdot max_{i\in[1,n]} L(i)\}. \tag{7.4}$$

  where obstacles are taken into account (i.e. inaccessible places and connectivity loss).

- **The fault-tolerant relay node placement problem (FT-RNP): to minimize the total number of relay nodes deployed, as well as the maximum lengths of primary paths and secondary paths connecting each PoI to the sink, respectively.** Each PoI is connected to the sink via k node-disjoint paths:

$$min\{N_r \cdot max_{i\in[1,n]} Lp(i) \cdot max_{i\in[1,n]} Ls(i)\}. \tag{7.5}$$

  where $Lp(i)$ is the length of the primary path from PoI P_i to the sink, $Ls(i)$ is the length of the secondary path from PoI P_i to the sink, and N_r the total number of relay nodes deployed to ensure k-connectivity of each PoI with the sink.

Similarly, we can define a variant, where the fault-tolerant relay node placement is constrained by obstacles. **The constrained fault-tolerant relay node placement problem (C-FT-RNP): to minimize the number of relay nodes deployed in an area with obstacles, as well as the maximum length of primary paths and secondary paths connecting each PoI to the sink, respectively:**

$$min_{obstacle}\{N_r \cdot max_{i\in[1,n]}Lp(i) \cdot max_{i\in[1,n]}Ls(i)\}. \quad (7.6)$$

where obstacles are taken into account(i.e. inaccessible places and connectivity loss).

We assume a disk-based model for radio communication. All the nodes, (i.e. relay nodes and sensor nodes) have the same communication range R. Two nodes at a distance less than or equal to R are able to communicate with each other if no obstacles are present. Obstacles prohibit the presence of sensor nodes in certain locations and may prevent direct communication between sensor nodes. We distinguish two types of obstacles: opaque and transparent. The sensing and communication models are those presented in Chapter 4. Similarly, the obstacles are modeled as in Chapter 4. Thus, two sensor nodes that are within radio range may be unable to communicate with each other due to the presence of an obstacle.

7.3.3 Solution for Relay Node Placement: RNP

In this section, we assume there is neither link/node failures, nor obstacles. We will see later how to relax these assumptions. We present three solutions based on heuristics: an intuitive solution based on the straight line, a solution based on the Steiner point and, finally, our proposed solution based on the optimal deployment grid.

7.3.3.1 An intuitive solution: The Straight-Line heuristic

The straight-line-based algorithm is the simplest solution and being the most intuitive one, we propose it as a baseline for comparison. It is based on classical wired deployment where each PoI is linked to the sink by a straight line cable. Here we simply propose to replace wires by a set of relay nodes along the path between each PoI and the sink. This algorithm deploys a relay node every R meters on the straight line between a PoI and the sink. Hence, each PoI is connected to the sink by the shortest path, as illustrated in Figure 7.10 where 14 PoIs are connected to the sink.

However, this solution has two main drawbacks. First, it is not robust: the failure of a single node or link on any path to a PoI may disconnect the PoI concerned. Second, no relay nodes are shared between the PoIs which means that the number of relays deployed may be very high.

7.3.3.2 A solution based on relay sharing: the Steiner-Point

By definition, the Steiner point S of three points A, B and C is the point that minimizes the sum of the distance to the three vertices of the triangle ABC. Hence, we have, for any point P, $d(A,S) + d(B,S) + d(C,S) \leq d(A,P) + d(B,P) + d(C,P)$, where $d(A,B)$

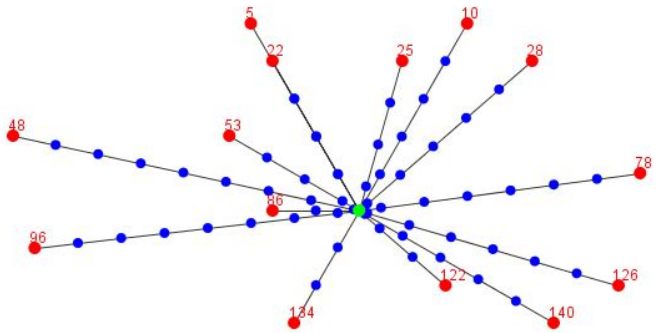


Figure 7.10: The Straight-Line Algorithm for 14 PoIs.

denotes the distance between A and B, see Figure 7.11 for an illustration. Notice that the Steiner point of the three points A, B and C is B itself if the angle (A,B,C) is higher than or equal to 120 degrees.

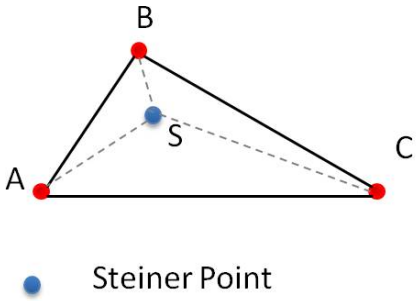


Figure 7.11: The Steiner point S of A, B and C.

The Steiner-Point-based algorithm builds a path from each PoI represented in red to the sink in green using the closest neighbor which may be another PoI, a Steiner Point (in blue) or simply a relay node, as illustrated in Figure 7.12 where 14 PoIs in red are connected to the sink in green. An initial consequence is that this algorithm enables PoIs to share some relay nodes, thereby reducing the total number of relay nodes needed, as we will see in Section 7.3.3.4. The second consequence is that the path from a PoI to the sink may lead away from the sink before getting closer to it, like, for instance, the path originating at node 78 in Figure 7.12. This phenomenon is evaluated by the path length from each PoI to the sink in Section 7.3.3.4.

7.3.3.3 Our solution: the Optimal-Deployment-Based Algorithm

The Optimal-Deployment-based algorithm uses the virtual optimal deployment in the circumscribing rectangle which includes all the PoIs. In this deployment, nodes are placed according to a triangular lattice (see Chapter 5). For each PoI, the shortest path is built only from relay nodes belonging to the optimal deployment grid. In the final deployment, only relay nodes that are used by at least one PoI are retained. This solution favors both

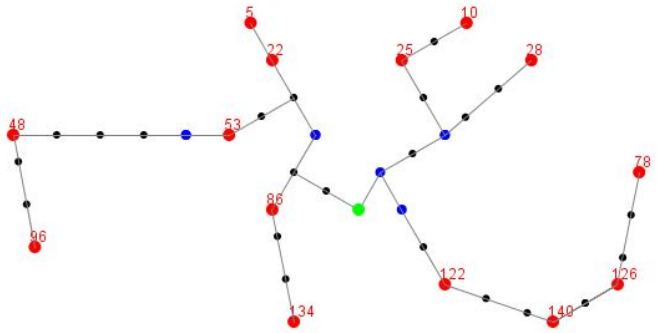


Figure 7.12: The Steiner-Points-Based Algorithm for 14 PoIs.

the sharing of relay nodes (in blue) between PoIs (in red) and short paths to the sink (in green), as illustrated in Figure 7.13 where 14 PoIs are connected to the sink.

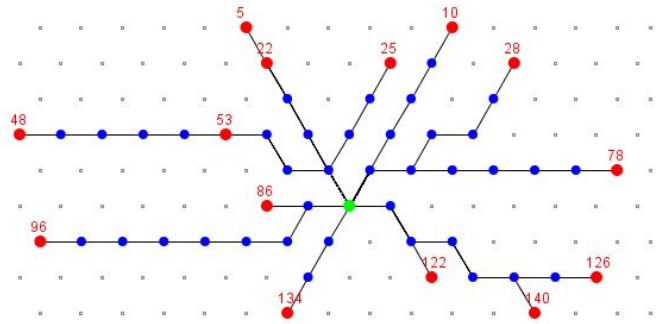


Figure 7.13: The Optimal-Deployment-Based Algorithm for 14 PoIs.

7.3.3.4 Performance Evaluation

For the performance evaluation of the three solutions described above, we developed our own simulation tool in Java and implemented the three solutions. The choice of a Java simulation tool was motivated by the need to obtain fast performance results, bearing in mind that these results do not depend on the network communication protocols used by the WSN in question. We consider different configurations where the number of PoIs + the sink varies from 8, 15, 22, 35 to 45. These PoIs are deployed in a 500m x 500m area. The communication range R satisfies $R \geq \sqrt{3}r$, where r is the sensing range of the nodes. We fix $R = 34.64m$. In this performance evaluation, the sink is assumed to be at a fixed location, at the center of the area.

a) Performance metrics

We compare the three solutions using the following metrics:

- Total number of nodes deployed: we want to know the number of additional relays deployed to ensure connectivity of each PoI with the sink.
- Number of shared nodes: if a node belongs to at least two paths originating at different PoIs, it is considered to be shared.
- Path length to the sink: we measure the average and maximum length of the paths connecting each PoI to the sink.
- Average node degree: we evaluate the average number of one-hop neighbor nodes per node (i.e. the average number of nodes located within the transmission range of the node considered).
- RNP index: we define the RNP index of a relay node placement as $RNP\ index = N_r \cdot max_{i\in[1,n]}L(i)$. This gives an indication on both the number of relays used and the maximum length of the paths connecting the PoIs to the sink.

b) Number of Sensor Nodes Needed

Figure 7.14 depicts the total number of nodes deployed for each configuration, highlighting the number of additional nodes, also called relay nodes because they are deployed only to provide connectivity with the sink. Simulation results show that the Straight-Line-based algorithm deploys the highest number of relay nodes, whatever the number of PoIs. For instance, for 45 PoIs, the number of additional nodes deployed by the Straight-Line-based algorithm is 3.7 times higher than that needed by the Optimal-Deployment-based algorithm.

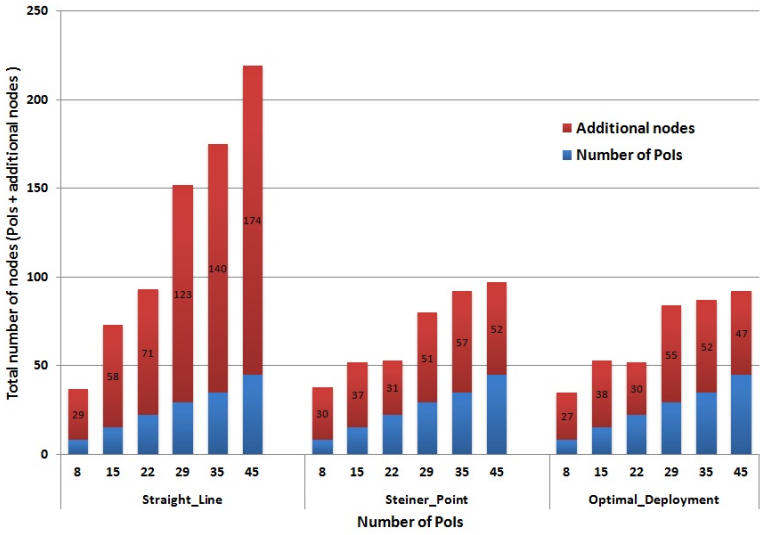


Figure 7.14: Total and additional nodes deployed.

With regard to this metric, the Optimal-Deployment-based algorithm minimizes the total number of nodes deployed. We also notice that when the number of PoIs increases,

the number of additional nodes may decrease. This can be observed in Figure 7.14 for 35 and 45 PoIs.

Unlike the Steiner-Point and the Optimal-Deployment based algorithms, the Straight-Line based algorithm does not share any relay nodes between paths connecting different PoIs to the sink. As a consequence, the total number of nodes deployed is higher, see Figure 7.15.

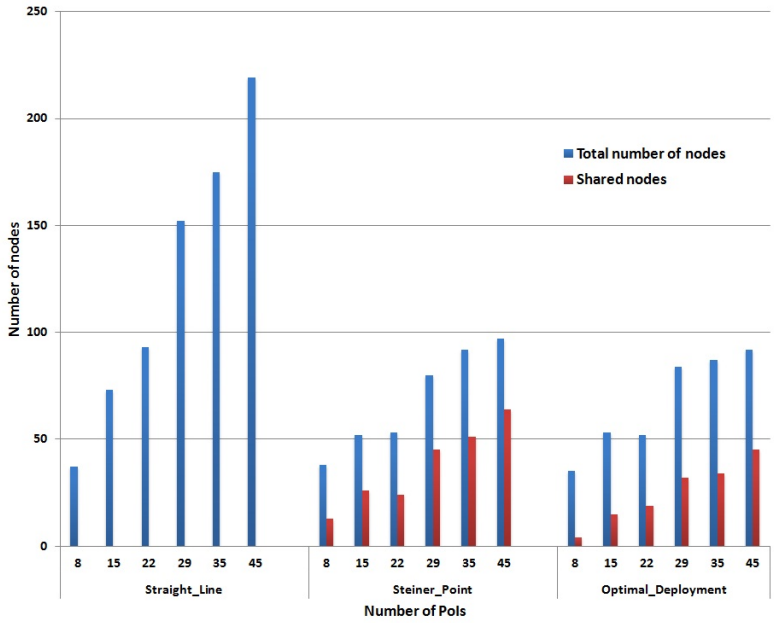


Figure 7.15: Total and shared nodes deployed.

c) Path Length to the Sink

Simulation results depicted in Figure 7.16, show that the Steiner-Point-based algorithm always provides longer paths than the Straight-Line and Optimal-Deployment based algorithms, both in terms of maximum and average path lengths. This is due to the principle of the Steiner-Point algorithm that connects PoIs together. In other words the connectivity of each PoI with the sink is a consequence and not the goal of this algorithm, the main goal being to reduce the number of nodes deployed. However, the Optimal-Deployment based algorithm provides results very close to those given by the Straight-Line algorithm; which gives the shortest routes.

d) Computation of the RNP index

Table 7.1 shows that the RNP index strongly increases with the number of PoIs for the straight line solution. It increases less strongly with the Steiner point solution, whereas the increase is only moderate for the optimal deployment based solution. In all the configurations tested, the optimal deployment based solution provides the smallest RNP index. For instance, for 45 PoIs it is 3 times less than the straight line solution.

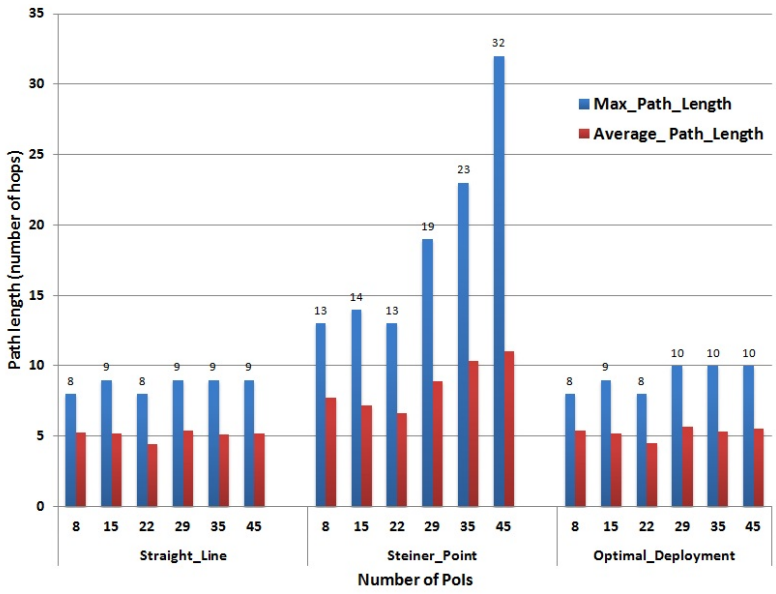


Figure 7.16: Maximum and average path length to the sink.

Table 7.1: RNP index for RNP solutions

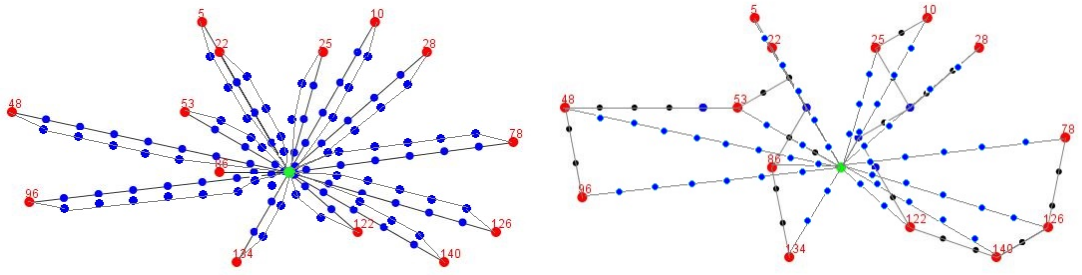
	RNP index		
Number of nodes	Straight-Line based	Steiner point based	Optimal deployment based
8	232	390	**216**
15	522	518	**342**
22	568	403	**240**
29	1107	969	**550**
35	1260	1311	**520**
45	1566	1664	**470**

7.3.4 Solution for Fault-tolerant RNP: FT-RNP

Assuming that link and/or node failures may occur, we now show how to improve the robustness of the three algorithms described in Section 7.3.3. To cope with node and/or link failures, an additional path is built from each PoI to the sink. For any PoI and for any algorithm considered, the first path to the sink obtained by the algorithm is called the primary path, whereas the others, obtained as explained in this section, are called secondary paths.

7.3.4.1 The Straight-Line Algorithm

The robustness of the Straight-Line algorithm is ensured by providing k-connectivity. This algorithm replicates each shortest path $k-1$ times. Each PoI appears to be at the end of



a Straight-Line. b Steiner-Point.

Figure 7.17: 2-connectivity.

a petal, whose other end is the sink, as depicted in Figure 7.17a, where 2-connectivity is provided. This algorithm remains very simple but no relay node is shared by the PoIs to reach the sink.

Furthermore, we observe a high concentration of nodes around the sink when the number of PoIs increases. This may induce, a high level of interference.

7.3.4.2 The Steiner-Point-Based Algorithm

Since in the basic version presented in Section 7.3.3, no redundancy is provided, there is no robustness: the failure of a link or node prevents data from at least one PoI reaching the sink. To achieve 2-connectivity, the straight line path from each PoI to the sink is added (see Figure 7.17b). Hence, there are no additional shared nodes compared with the basic version with only one path per PoI.

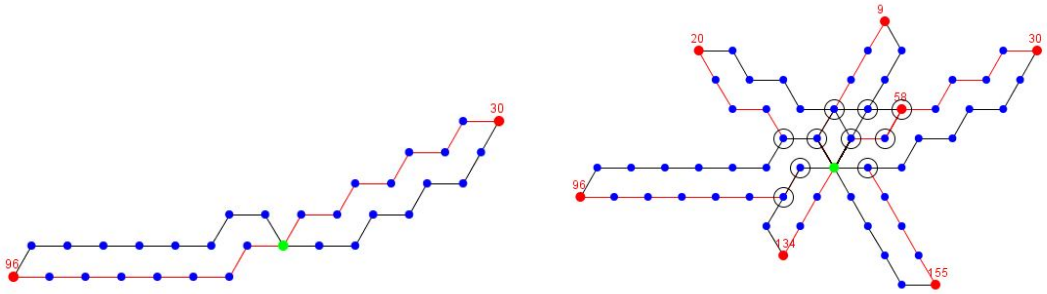
7.3.4.3 The Optimal-Deployment-Based Algorithm

This solution is made robust by adding one node-disjoint shortest path for each PoI to the sink. This new path shares no nodes with the primary path of the PoI in question, as shown in Figure 7.18a. However, it may share nodes or links with the primary or secondary path of another PoI, thus reducing the total number of nodes deployed. Figure 7.18b depicts shared nodes in black circles: at least two paths originating from different PoIs use this node to reach the sink.

In the triangular lattice of the optimal deployment, each non-border node has 6 neighbor nodes. Consequently, we can obtain any k-connectivity with $k \leq 6$. If a higher connectivity is required, another grid structure should be used.

7.3.4.4 Performance Evaluation

Having enhanced these these three solutions to achieve 2-connectivity, we now compare their performances for various configurations. In addition to the metrics given in Section 7.3.3.4, we add a new metric: the node degree. The RNP index is modified to take



a Two paths. b Shared nodes.

Figure 7.18: 2-connectivity with the Optimal-Deployment.

fault-tolerance into account. By definition, a fault-tolerant relay node placement has an FT-RNP index $= N_r \cdot max_{i\in[1,n]}Lp(i) \cdot max_{i\in[1,n]}Ls(i)$.

a) Number of Sensor Nodes Needed

As concerns the total number of relay nodes deployed, simulation results show that the Optimal-Deployment-based algorithm strongly minimizes the total number of relay nodes deployed, as illustrated in Figure 7.19. For instance, for 45 PoIs, the Optimal-Deployment-based algorithm requires a number of additional nodes that is more than 4.3 times less than the Steiner-Point based algorithm, thus considerably reducing the deployment cost. We also note that when the number of PoIs increases, the number of additional nodes used by the Optimal-Deployment-based algorithm may decrease. This is shown in Figure 7.20.

Simulation results depicted in Figure 7.20 show that for both the Steiner-Point and the Optimal-Deployment based algorithms, the number of shared nodes increases with the number of PoIs. Moreover, with the Optimal-Deployment based algorithm, the deployment around the sink becomes very close to the optimal one.

b) Path Length to the Sink

Figure 7.21 shows that for each algorithm considered, the maximum path length is identical when maintaining one path or two-paths with either the Steiner-Point or the Straight-Line algorithm. For the Optimal-Deployment algorithm, the secondary path has a length that is either equal to that of the primary path or greater by one hop. To reduce the data gathering delays in a WSN deployed according to the Steiner-Point algorithm, we recommend exchanging the role of primary and secondary paths by using the Straight-Line path as the primary path.

c) Node Degree

In the optimal deployment based on a triangular lattice, each non-border node has exactly 6 neighbor nodes. As a consequence, the degree of any node is upper bounded by 6 for

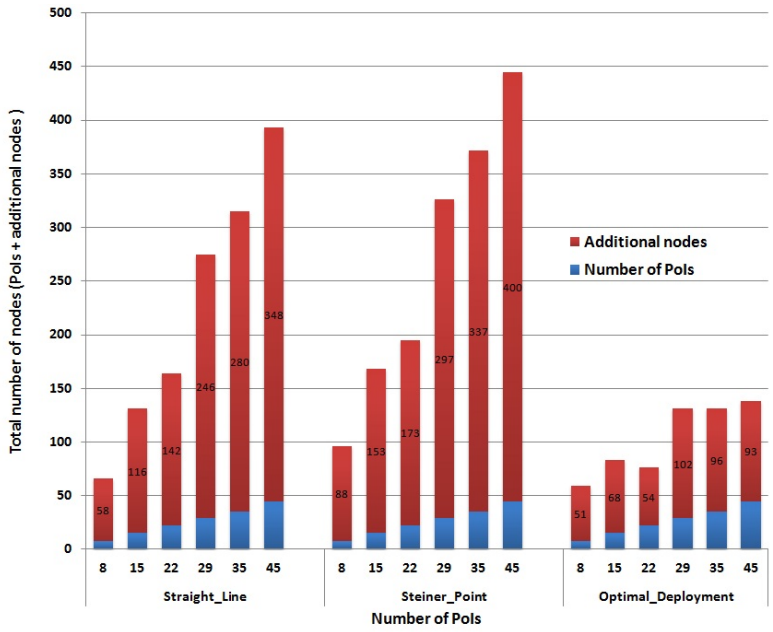


Figure 7.19: Total and additional nodes deployed for 2-connectivity.

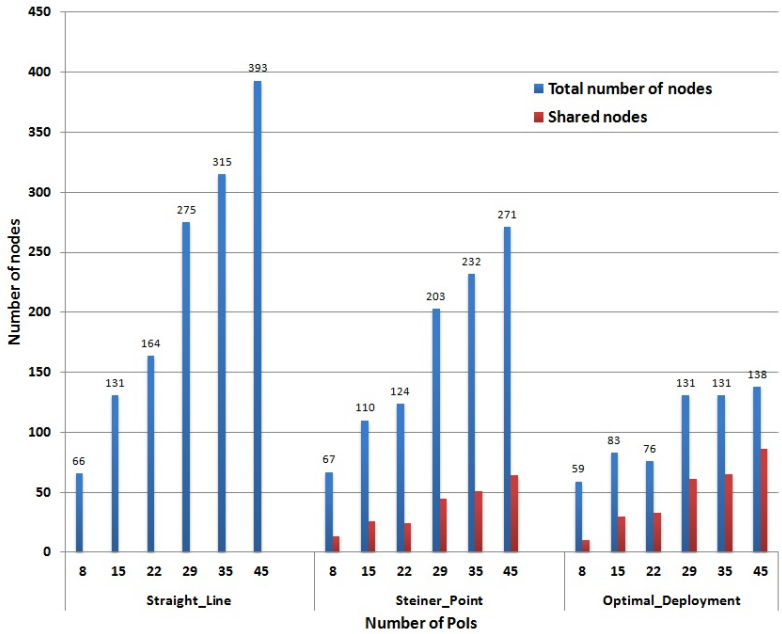


Figure 7.20: Total and shared nodes deployed for 2-connectivity.

any number of paths $k \leq 6$. Simulation results depicted in Figure 7.22 show that for one path, the average node degree remains in the interval $[2, 3]$ for all the numbers of PoIs tested, whereas for two paths, it remains in the interval $[4, 5]$. However, with the Straight-Line algorithm, the node degree strongly increases with the number of PoIs, even for a single path. This is due to the very high density of nodes close to the sink and

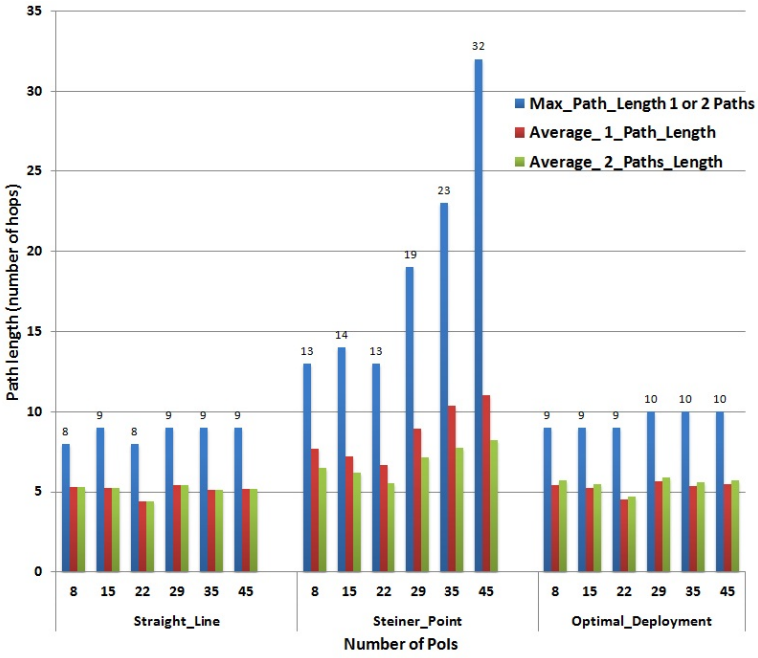


Figure 7.21: Maximum and average path length to the sink for 1 and 2-connectivity.

the non-sharing of nodes between the paths. Furthermore, the Steiner-Point algorithm provides the smallest average node degree, because paths are not built toward the sink but between PoIs and relay nodes. More precisely, the sink is considered as a PoI and not as the target destination of any path originating at a PoI. For this reason, there is no concentration of nodes around the sink with the Steiner Point algorithm, which is not the case with the Straight-Line and the Optimal-Deployment algorithms, as depicted in Figures 7.12, 7.10 and 7.13 respectively.

d) Computation of the FT-RNP index

Table 7.2 shows that the optimal deployment based solution provides the smallest FT-RNP index in fault-tolerant RNP. This is due to the sharing of relay nodes and the minimized length of both primary and secondary paths.

7.3.5 Solution for Constrained fault-tolerant RNP: CFT-RNP

In the previous sections, the PoIs and the sink were located in an area that did not contain any obstacles. However, in some applications, this assumption should be relaxed since obstacles may exist. In this section, we focus on ensuring k-connectivity between PoIs and the sink in an environment where obstacles are present.

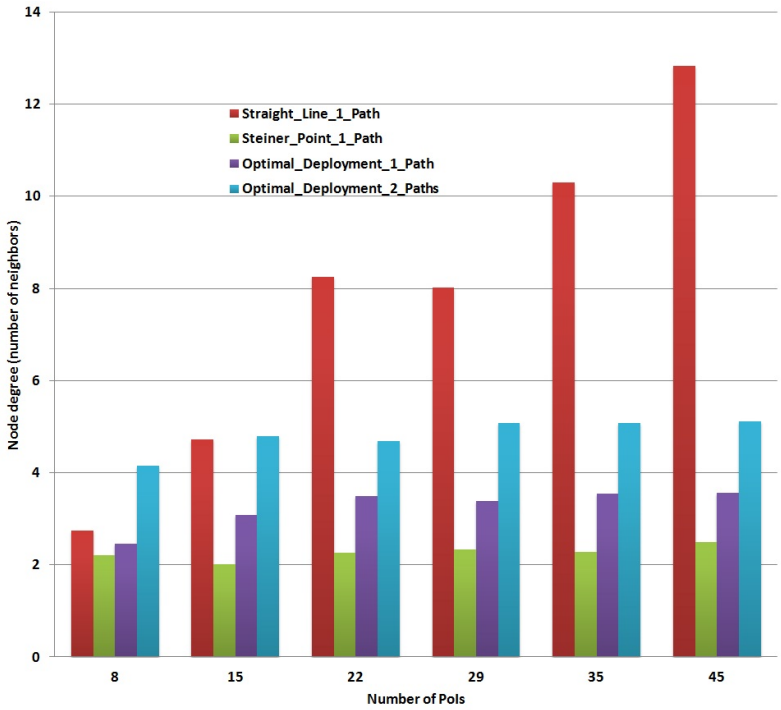


Figure 7.22: Node Degree.

Table 7.2: FT-RNP index for fault-tolerant RNP solutions.

	FT-RNP index		
Number of nodes	Straight-Line based	Steiner point based	Optimal deployment based
8	3712	9152	**3672**
15	9396	19278	**5508**
22	9088	17992	**3888**
29	19926	50787	**10200**
35	22680	69759	**9600**
45	28188	115200	**9300**

7.3.5.1 The Straight-Line Algorithm

The Straight-Line algorithm, which provides the minimum number of relay nodes, cannot be applied to ensure network connectivity in the presence of obstacles since obstacles may exist on the straight line between the PoI and the sink. However, this solution can be enhanced to cope with obstacles. Retaining the basic principle of this method, the relay nodes are deployed along a straight line between the PoI and the sink. The presence of an obstacle on this line is analog to the problem of void handling in geographic routing (67). One possible solution would be to follow the left-hand rule to bypass the obstacle(s). However, such a solution is not optimal in terms of path length and the

number of additional nodes deployed.

7.3.5.2 The Steiner-Point based Algorithm

The Steiner-Point based algorithm cannot cope with the presence of obstacles. Since the computation of the Steiner Point position takes no account of the shape of the area or the presence of obstacles, the Steiner Point position could be inside an obstacle. If this position is changed, the mathematical property is lost. Therefore, we do not consider any enhancement of this solution to cope with obstacles.

7.3.5.3 The Optimal-Deployment based algorithm

When there are no obstacles in the area considered, the virtual grid of the optimal deployment ensures full area coverage and network connectivity. In this case, at least one path to the sink can be ensured. On the other hand, in the presence of obstacles, not only coverage and connectivity holes may occur, but isolated PoIs may also exist. In fact, when we compute the optimal deployment in an area containing obstacles, nodes that belong to the virtual grid and whose location is inside obstacles are removed, which may lead to coverage and connectivity holes occurring around obstacles. Depending on the PoI's position and the sink's position, these coverage holes may result in isolated PoIs, particularly if the PoI is surrounded by obstacles.

In Section 7.2, we proposed a solution based on the optimal deployment to ensure full area coverage and network connectivity in the presence of opaque obstacles. We healed coverage holes caused by obstacles by deploying additional nodes in these coverage holes. This final deployment which can cope with obstacles is used as our new virtual grid. Using this virtual grid and the principle of the Optimal-Deployment based algorithm, network connectivity can be ensured between each PoI and the sink, as depicted in Figure 7.23.

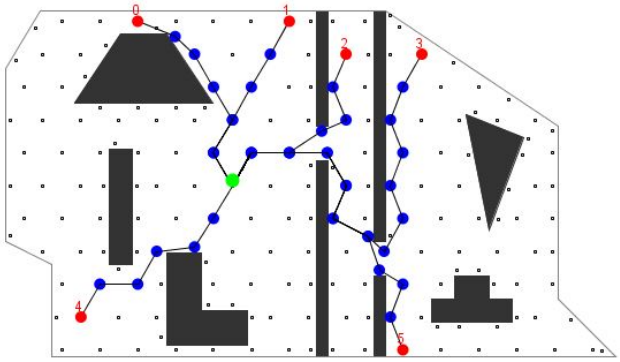


Figure 7.23: Connectivity between each PoI and the sink in the presence of obstacles.

in order to support k-connectivity in the presence of obstacles, we may obtain a network like that depicted in Figure 7.24 for 2-connectivity. There are two paths with disjoint nodes to connect each PoI to the sink, and so, the failure of nodes on a single path does not

disconnect a PoI. However, we can observe two problems:
− bypassing the obstacle leads to a secondary path that is much longer than the primary path (see, for instance, PoI 5 at the bottom right in Figure 7.24).
− there is a gap between the primary and the secondary paths preventing any node on the primary path from communicating with a node on the secondary path. In Figure 7.24 we can see a relay node on the primary path of PoI 4 that has no neighbor on the secondary path due to the gap between the two paths.

For each relay node on the secondary path we need to have at least one neighbor on the primary path. As a consequence, any node on the primary path can bypass its successor using a node on the secondary path. To cope with the gap problem, the secondary path should be built using the neighbors of all the relay nodes on the primary path instead of all the deployed nodes. Due to the presence of obstacles, some neighbors of the virtual grid may not exist or may not be able to communicate with each other. That is why we propose the rule depicted in Figure 7.25, where a relay node is added to build the secondary path. The location of this node is critical. First, it should communicate with its downstream neighbor on the secondary path. Second, it should communicate with a relay node of the primary path. Finally, it should communicate with:

- either its upstream neighbor on the secondary path, if one exists, as depicted in Figure 7.25 case 2,
- or the upstream neighbor of the relay node or the primary path, as illustrated in Figure 7.25 case 3.

Figure 7.26 shows the final deployment of relay nodes after applying this rule. We can observe that for all the PoIs, any node on the primary path can communicate with a node on the secondary path. Also, we can see the relay node added in orange on the secondary path of PoI 5 which solves two problems: bypassing the obstacle and overcoming the gap between the two paths.

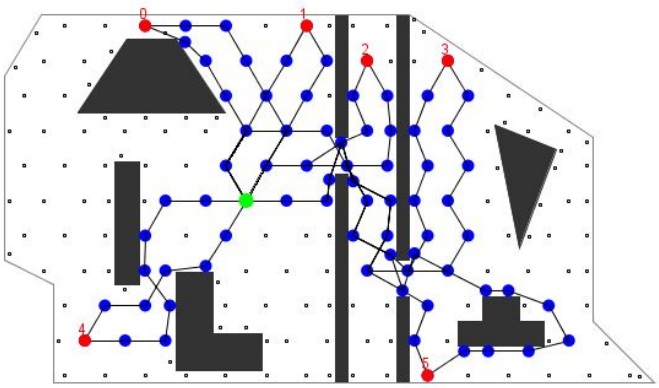


Figure 7.24: 2-Connectivity between each PoI and the sink in the presence of obstacles, with problems caused by missing relay nodes.

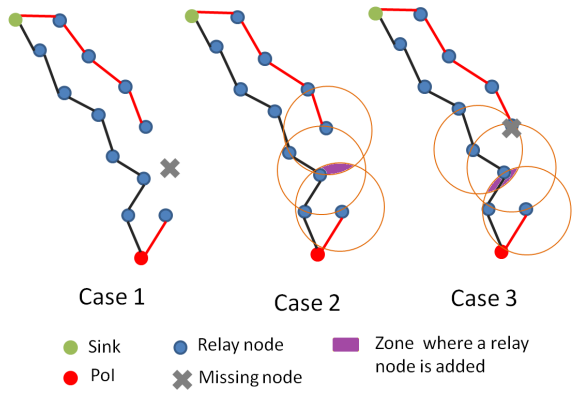


Figure 7.25: Rule to cope with missing relay nodes when obstacles exist.

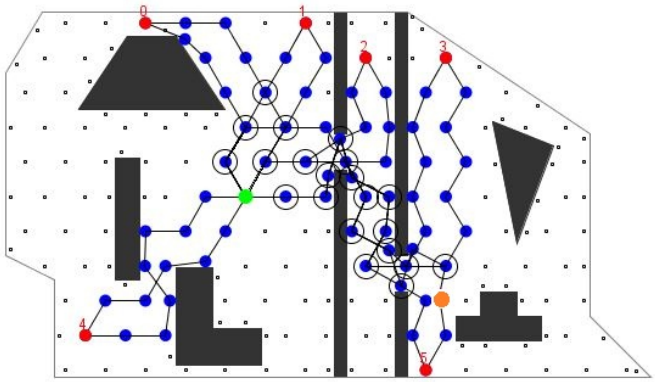


Figure 7.26: 2-Connectivity between each PoI and the sink in the presence of obstacles.

7.3.5.4 Performance Evaluation

In this section, we evaluate the impact of the presence of obstacles in the area considered in our solution. We use two configurations: 6 PoIs and 15 PoIs.

The performance evaluation metrics are the total number of relay nodes deployed, the number of shared nodes, the average path length, the maximum path length and the RNP index. In Figure 7.27, the total number of relay nodes deployed when two paths are needed is less than twice the total number of relay nodes when one path is needed. This is true both with and without obstacles, and is due to the high number of nodes that are shared between paths from different PoIs. For instance, this number reaches 44% of the total number of nodes deployed for 15 PoIs when obstacles exist and two paths are required. The presence of obstacles tends to increase the number of shared nodes in narrow lanes.

The average path length value in the presence of obstacles is close to the average path length value when there are no obstacles. This is due to the fact that our solution favors a secondary path which is close to the primary one.

The maximum path length depends on the shape of the obstacles. Although the primary and secondary paths are close, the secondary path may be longer than the primary

path. This is due to the number and location of the neighbors of all the relay nodes of the primary path.

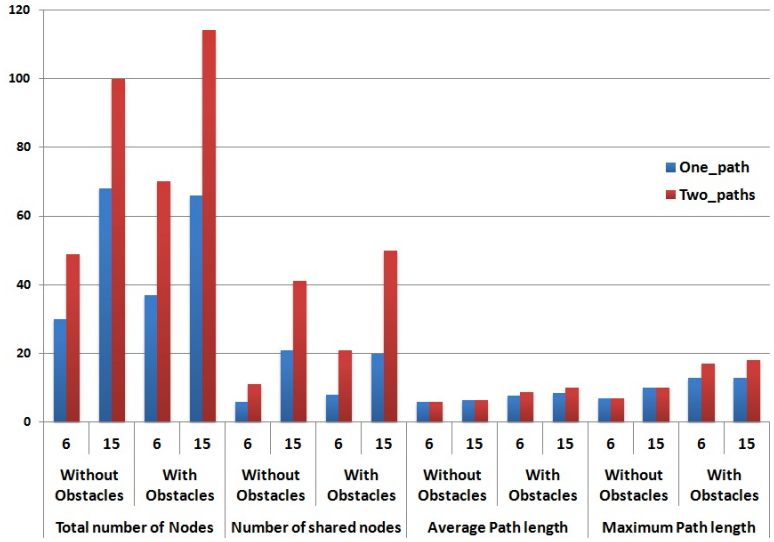


Figure 7.27: Evaluation of the impact of obstacles.

Table 7.3 shows the strong impact of the presence of obstacles on the RNP index. In addition, maintaining several paths is much more expensive since paths must bypass obstacles.

Table 7.3: Comparison of the RNP index for constrained and unconstrained FT-RNP solutions.

	RNP index		FT-RNP index	
Number of nodes	One path		Two paths	
	Without obstacles	With obstacles	Without obstacles	With obstacles
6	168	403	2107	14144
15	530	663	8500	23166

7.3.6 Summary

In this section, we studied the relay node placement problem. We proposed OAD-PoI to ensure full coverage and maintain fault tolerant connectivity while optimizing the total number of relay nodes deployed. OAD-PoI is efficient in the presence of opaque obstacles.

7.4 Conclusion

In this chapter, we studied two coverage problems: area coverage and PoI coverage in a constrained environment (e.g. an irregular area, presence of obstacles that may be opaque, node failures). For area coverage, we propose a projection-based method, OAD-Area, to ensure full area coverage and maintain network connectivity. OAD-Area has been published in (68). For PoI coverage we propose an optimal deployment-based algorithm, OAD-PoI, to solve the RNP problem and ensure a fault-tolerant connectivity. In addition, we define a new metric, called the RNP index, to evaluate the number of relays needed multiplied by the maximum length of the path from any PoI to the sink. If fault-tolerant connectivity is required, the new metric is called the FT-RNP index and takes into account the maximum length of both the primary path and the secondary path. Table 7.4, summarizes the algorithms proposed in this chapter.

The positions of sensor nodes computed for the area coverage problem and the PoI coverage problem can be given to one or multiple mobile robot(s) in charge of placing the static wireless sensor nodes at their optimized location. This will be the focus of the following chapter.

Table 7.4: Computation of an optimized deployment to ensure area/PoI coverage and network connectivity

	Entity considered	**Obstacles**	**Energy-efficiency**	**Fault-tolerant** with regard to connectivity
OAD-Area	known area	known	– Minimizes the number of sensor nodes deployed	– no
OAD-PoI	known PoIs	known	– Minimizes the number of relay nodes deployed	– ≥ 2 paths toward the sink

Chapter 8

Optimization of Robot Trajectories

Contents

8.1 Introduction

The assisted deployment can be divided into two steps: the first step consists in the computation of the deployment (i.e. computing the appropriate node positions) and the second step consists in the placement of sensor nodes by human(s) or mobile robot(s) in the area to be monitored. The problem of the first step is how to optimize the deployment in terms of the number of nodes while satisfying coverage and connectivity requirements (e.g. full, partial). We solved this problem in the previous chapter by proposing two optimized deployments: the first one to ensure area coverage and connectivity, and the second one to ensure PoI coverage and connectivity. The problem of the second step, is how to optimize robot trajectories, in terms of duration, when placing the sensor nodes at their precomputed positions. This is the focus of this chapter.

We aim to minimize the time needed by multiple robots to deploy static sensor nodes in an area that may contain obstacles. We propose two approaches to solve this problem:

- Assisted deployment with two robots; we provide a solution based on a game theory approach in Section 8.3 called Two Robot Deploying Sensor nodes (TRDS).

- Assisted deployment with multiple robots; we propose a formal definition of the multi-objective optimization problem called Multi Robot Deploying Sensor nodes (MRDS), and we provide a solution based on genetic algorithms in Section 8.4.

In the case of a single robot, we give a formal definition of the optimization problem called Robot Deploying Sensor nodes, RDS (see Appendix A), and then a solution computed by the 2-Opt heuristic and the genetic algorithm.

8.2 Related work

The cost of the deployment may be very expensive due to the large number of mobile sensors needed to cover the whole area. In such a case, it is worth using mobile robots which are able to place this large number of static nodes at their appropriate positions.

In assisted deployment, we distinguish between two different situations where mobile robots are in charge of deploying static sensor nodes.
In the first situation, the robot has two tasks: on the one hand it should move and discover the area considered, and on the other hand, place sensor nodes at their position to ensure the required coverage and maintain network connectivity. A robot has to follow predefined rules to move in the area and place the sensor nodes. This strategy is proposed in (69) where one robot follows a spiral movement policy to deploy static sensor nodes along its trajectory. The goal is not to optimize the robot's trajectory but to ensure full area coverage and network connectivity using the minimum number of sensor nodes. In addition, some movement policies are defined to enable the robot to bypass obstacles. In a similar context, the authors in (60), propose a serpentine movement policy with an obstacle handling policy and a boundary policy. The robot has to follow the serpentine movement policy while placing static sensor nodes separated by the optimal distance to reduce the total number of sensor nodes. To conclude, in such a situation, the policies proposed in the two papers cited enable the robot to visit the whole area, while avoiding obstacles and placing sensor nodes.
In the second situation, sensor node positions are precomputed and given to the robot(s). In this situation, each position should be visited by exactly one robot and one sensor node should be placed at each position computed. Here, the problem is different: the goal is no longer to discover the area and compute sensor node positions, but rather how to optimize the duration necessary to deploy these sensor nodes.

In this chapter, we are interested in the second situation. We focus on minimizing the time needed by the robots to deploy all the sensor nodes in an environment with known obstacles, and to return to their starting position.

In the next section, we show how to minimize the deployment duration using two robots. A game theory approach is proposed.

8.3 Two-robot assisted deployment: based on a game theory approach

In this section, we adopt a game theory approach in order to determine the tours of two robots in charge of deploying sensors at some points of interest (PoIs). The goal is to minimize the deployment duration while ensuring the coverage of each PoI by exactly one sensor node placed by one robot.

Figure 8.1 depicts the tour of two robots where each PoI is visited by exactly one robot and relay nodes are placed to ensure connectivity between each PoI and the sink.

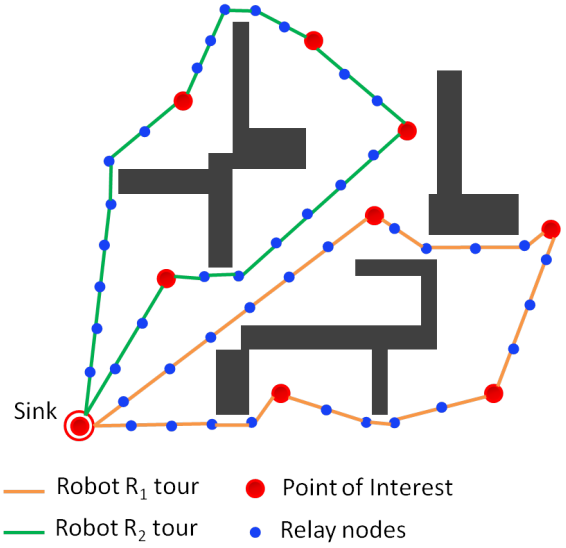


Figure 8.1: Robot tours and relay node deployment in the presence of obstacles

Game theory is a mathematical method that describes the phenomenon of conflict and cooperation between intelligent rational decision-makers. The game theory approach is used in several WSN applications including routing protocols, topology control, power control and energy saving, packet forwarding, data collection, spectrum allocation, bandwidth allocation, quality of service control, coverage optimization, WSN security, and other sensor management tasks (70). Game Theory describes the behavior of players in a game. Players may be either cooperative or non-cooperative while aiming at maximizing their payoffs from the game.
In our study, we focus on a non-cooperative game and we refer to the Nash equilibrium to determine the best solution for our game (i.e. for our problem: two robot trajectories that minimize the deployment duration).
In game theory, if each player has chosen a strategy and no player can benefit by changing his or her strategy while the other players keep theirs unchanged, then the current set of strategy choices and the corresponding payoffs constitute a Nash equilibrium.

8.3.1 Assumptions and definitions

- We assume that each robot R_i, $i = 1$ or 2, knows:
  - n, the number of PoIs. Each PoI is denoted P_i, for $i \in [1, n]$.
  - The position of each PoI.
  - The number, position and shape of the obstacles.
  - The area considered.
- Each mobile robot is able to know its position and to move to a given position.
- Both robots have the same linear speed ls and the same angular speed as. As an example, in the simulation we take $ls = 1m/s$ and $as = 10°/s$.
- Each robot R_i has the capacity to carry $C_{max,i}$ sensor nodes.
- Each robot R_i has the capacity to carry $C_{relaymax,i}$ relay nodes.
- Both robots have the same starting position, the sink denoted P_0 for simplicity, and should return to this position.
- Let S_i denote the set of strategies played by robot R_i. Any strategy $\in s_i$ played by R_i is defined by the ordered set of PoIs visited by R_i.
- To cope with obstacles, a bypassing approach is adopted as explain in the next section.

8.3.2 Deployment duration and obstacles

The deployment duration D_i of robot R_i depends not only on the time needed to travel a distance but also on the time needed to carry out changes in direction. Hence, we compute D_i for any strategy s_i as follows:

$$D_i = \sum_{j \in s_i} d_{j,j+1}/ls + \sum_{j \in s_i} a_{j-1,j,j+1}/as. \tag{8.1}$$

Where j and $j+1$ are two successive PoIs in s_i. $d_{j,j+1}$ is the distance between two successive PoIs in s_i. $a_{j-1,j,j+1}$ is the angle formed by the segments $[j-1, j]$ and $[j, j+1]$ corresponding to three successive PoIs in s_i. We notice however that the tour duration is the same when the robot visits the same nodes but in reverse order.

One or several obstacles may exist between two consecutive PoIs in the robot's tour. The tour duration increases when obstacles exist since the robot has to bypass these obstacles. We use the strategy to bypass the obstacles with the minimum duration. For each obstacle, we define as many intermediate points as the number of obstacle vertices. Then, we select the path that goes through intermediate points until reaching the PoI destination, having the minimum duration. For instance, in Figure 8.2, a direct path from A to B is impossible. The intermediate points I_1, I_2 and I_3 are the best combination in

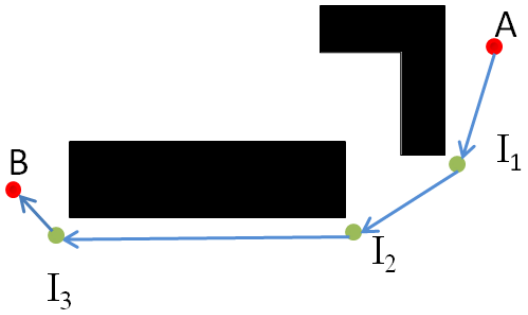


Figure 8.2: Intermediate points between sensor node positions A and B

terms of duration to go from A to B. The tour duration due to this path is computed as the sum of the duration due to any segment composing the path.

Additional intermediate positions are introduced in the tour of the robot to bypass obstacles. The deployment duration takes into account these intermediate positions. Furthermore, additional relay nodes are needed to ensure connectivity when opaque obstacles are present. The positions of these relay nodes are also taken into account.

8.3.3 Problem formalization

We proceed in two steps. In the first step, we focus on the coverage problem: the goal is only to ensure the coverage of all PoIs while minimizing the tour duration of each robot. In other words, for any given PoI, there is exactly one robot that places a sensor at its right position. In the second step, we are interested in both coverage and connectivity: each robot must, in addition, place relay nodes to ensure connectivity between each PoI with the sink.

8.3.3.1 Coverage problem

We can now model our coverage problem as a non cooperative game. A game is defined by the set of players, the set of strategies for each player and the set of payoffs corresponding to the strategies used by each player. In our case, the two players are the mobile robots. The action of each robot consists in selecting the next PoI to visit in its tour. Consequently the strategy s of a robot is an ordered set of PoIs visited by this robot.
Let R_1 and R_2 denote our two robots. Let $\{P_1, P_2, .., P_n\}$ denote the set of n PoIs that have to be monitored. Let $\mathcal{P}_i(s_i, s_{-i})$ denote the payoff of player R_i when it plays strategy s_i while the other player plays strategy s_{-i}. The payoff computation follows some rules:

- To incite robots to visit all PoIs exactly once, the payoff of a robot increases when the number of PoIs visited increases. However, the payoff of both robots becomes negative if some PoIs are visited by either no robot or both robots.

- To minimize the tour durations of the robots, the payoff of a robot increases for a given set of PoI visited when the tour duration of the robot decreases.

Algorithm 1 shows how to calculate the payoff of one robot in charge of ensuring PoIs coverage. We use a weight factor α higher than D_i to model positive outcome values. Consequently each player R_i wants to maximize its payoff $\mathcal{P}_i(s_i, s_{-i})$. Under the constraints:

- $C_i \leq C_{max,i}$, where C_i is the number of sensor nodes carried by the robot R_i.

Algorithm 1 Calculate $\mathcal{P}_i(s_i, s_{-i})$ for Coverage problem

if (*Number of PoIs visited by both* R_i *and* R_{-i} <> 0) **then**
  $\mathcal{P}_i(s_i, s_{-i}) = -1$
else
  if $((C_{max,i} + C_{max,-i}) >= n)$ **then**
    if (*Number of PoIs visited by neither* R_i *nor* R_{-i}) <> 0) **then**
      $\mathcal{P}_i(s_i, s_{-i}) = -1$
    else
      $\mathcal{P}_i(s_i, s_{-i}) = \frac{\alpha}{D_i}$
  else
    if (*Number of PoIs visited by* R_i $< C_{max,i}$) **then**
      $\mathcal{P}_i(s_i, s_{-i}) = -1$
    else
      $\mathcal{P}_i(s_i, s_{-i}) = \frac{\alpha}{D_i}$

8.3.3.2 Coverage and connectivity problem

In the coverage and connectivity problem, each robot places a relay node each time it travels a distance *Dist*. The coverage and connectivity problem differs from the coverage problem by an additional constraint on $C_{relay,i}$ the number of relay nodes placed by a robot R_i. We must have:

- $C_{relay,i} \leq C_{relaymax,i}$.

Strategies violating this constraint are eliminated.
The payoff of any strategy is computed as in Algorithm 1.

8.3.4 Problem resolution

In both games, the payoff computed for player R_i depends not only on s_i the strategy chosen by R_i but also on the strategy s_{-i} chosen by the other player R_{-i}.

A strategy profile (s_i^*, s_{-i}^*) is a Nash equilibrium if and only if no unilateral deviation of the strategy of a single player is profitable for that player. Hence, $\forall i, \forall s_i \in S_i, \mathcal{P}_i(s_i^*, s_{-i}^*) \geq \mathcal{P}_i(s_i, s_{-i}^*)$.

Nash proved the existence of at least one Nash equilibrium when mixed strategies are allowed in a game with a finite number of players and each player chooses among pure strategies.

Both problems are solved in a similar way:

- Determining all the strategies for each player.

- Eliminating all the strategies that violate the constraints. The remaining strategies are the valid strategies of any player.
- Computing the payoff for all the possible combinations of valid strategies for all the players.
- Computing the Nash equilibrium using the Gambit tool (71).

The number of strategies for robot R_i visiting q PoIs with $q \leq min(C_{max,i}, n)$ is: $C_n^q * \frac{q!}{2}$. This is because the strategies $\{P_j, P_{j+1}...P_{j+m}\}$ and $\{P_{j+m}..., P_{j+1}, P_j\}$ have the same payoff. Hence the total number of valid strategies for robot R_i is equal to:

$$\sum_{q \in \{1, C_{max,i}\}} C_n^q * \frac{q!}{2}.$$

8.3.4.1 Coverage problem

For any given strategy s_{-i}, the strategy s_i of robot R_i that maximizes $\mathcal{P}_i(s_i, s_{-i})$ in the coverage problem consists in visiting only all the PoIs that are not visited by R_{-i}, provided that R_i meets the constraint $C_{max,i}$, and selects the visit order that minimizes the deployment duration Di.

We first notice that if $C_{max,i} < \lceil n/2 \rceil$ for any $i \in [1, 2]$, it is impossible to cover all the PoIs with two robots.

In any other case, we obtain a Nash equilibrium where each PoI is visited exactly once, ensuring full coverage of all the PoIs without any redundancy.

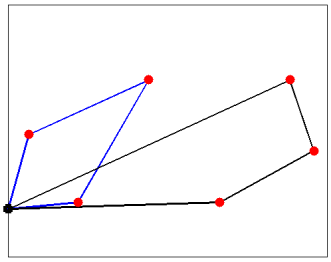
In this section, we evaluate the tour duration of the two robots deploying sensor nodes in an area with and without obstacles. We start by computing the tour duration for various values of $C_{max,i}$ and $C_{max,-i}$. The sum of $C_{max,i}$ and $C_{max,-i}$ should be higher than or equal to the number of PoIs to be covered. Then, we evaluate the duration of both tours in different configurations. These configurations are different in terms of the number and shape of the obstacles in the area.

Area without obstacles

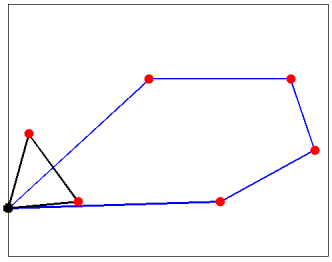
To evaluate the impact of the robot's capacity to carry sensor nodes, we vary the values of $C_{max,i}$ and $C_{max,-i}$.

	Case 1 fig. 8.3a		Case 2 fig. 8.3b		Case 3 fig. 8.3c	
	Robot 1	Robot 2	Robot 1	Robot 2	Robot 1	Robot 2
$C_{max,i}$	3	3	4	2	4	4
Deployment duration (s)	644	1056	1090	366	1090	366
PoIs visited	3	3	4	2	4	2

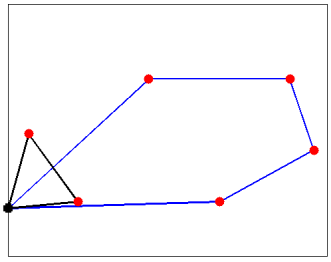
Table 8.1: Impact of $C_{max,i}$ on the deployment duration.



a $C_{max,i} = 3$, $C_{max,-i} = 3$.



b $C_{max,i} = 4$, $C_{max,-i} = 2$.

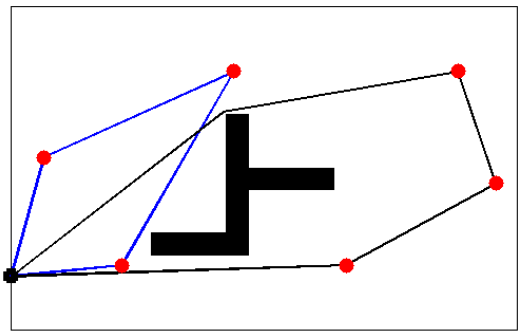


c $C_{max,i} = 4$, $C_{max,-i} = 4$.

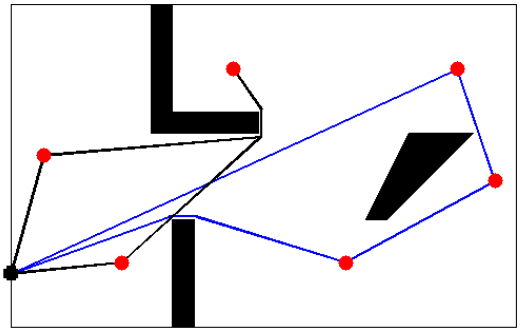
Figure 8.3: Best robot tours with different $C_{max,i}$ (No obstacles).

When $C_{max,1}$ is equal to $C_{max,2}$ and their sum is equal to the number of PoIs (See Case 1 in Table 8.1 and Figure 8.3a) the Nash equilibrium provides two robot tours that visit all the PoIs exactly once. It may not be possible to find two robot tours with the same duration due to the position of the PoIs in the area considered. However, the ones computed are the best combination to minimize the tour duration of robots.
In Figure 8.3b and Figure 8.3c, we get the same Nash equilibrium, where the first robot visits $C_{max,i}$ PoIs and the second one visits the PoIs not visited by the first one.

Area with obstacles



a $C_{max,i} = 3$, $C_{max,-i} = 3$.



b $C_{max,i} = 3$, $C_{max,-i} = 3$.

Figure 8.4: Impact of obstacles on the best robot tours.

In Figure 8.4a and Figure 8.4b, we observe that the presence of obstacles modifies the trajectory of the robots. We obtain a Nash equilibrium for the two obstacle configurations, where each robot visits 50% of PoIs. To bypass the obstacle, each robot uses intermediate points. For this reason, the tour duration increases: $1071s$ in Configuration 1 Figure 8.4a and $1077s$ in Configuration 2 Figure 8.4b instead of $1056s$, as shown in Table 8.2.

	Configuration 1 fig. 8.4a		Configuration 2 fig. 8.4b	
	Robot 1	Robot 2	Robot 1	Robot 2
$C_{max,i}$	3	3	3	3
Deployment duration(s)	644.54	1071.3	1077.1	786.69
PoIs visited	3	3	3	3

Table 8.2: Impact of the presence of obstacles on the deployment duration.

8.3.4.2 Coverage and connectivity problem

For any given strategy s_{-i}, the strategy s_i of robot R_i that maximizes $\mathcal{P}_i(s_i, s_{-i})$ in the coverage and connectivity problem consists in visiting only all the PoIs that are not visited by R_{-i}, provided that R_i meets the constraints $C_{max,i}$ and $C_{maxrelay,i}$, and selects the visit order that minimizes the Deployment duration Di.

In this problem the robot places relay nodes along its trajectory in order to maintain connectivity with the sink and the PoI previously visited. Since the robot makes a tour, it establishes two node-disjoint paths between each PoI and the sink, so, a robust network connectivity is maintained.

Area without obstacles

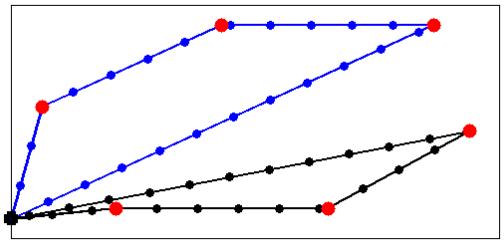


Figure 8.5: $C_{max,i} = 3$, $C_{max,-i} = 3$

In this section, we assume that the robot is able to carry a limited number of relay nodes $C_{maxrelay,i}$. In Figure 8.5, $C_{maxrelay,i} = 22$. The two robot tours are different from those illustrated in Figure 8.4 where there is no constraint on the number of relay nodes. The two robot tours in Figure 8.4 provide a smaller deployment duration than those in Figure 8.5. However, they require a number of relay nodes that is higher than $C_{maxrelay,i}$. Then, the result obtained in Figure 8.5 presents the best combination taking into account both the number of relay nodes and the deployment duration.

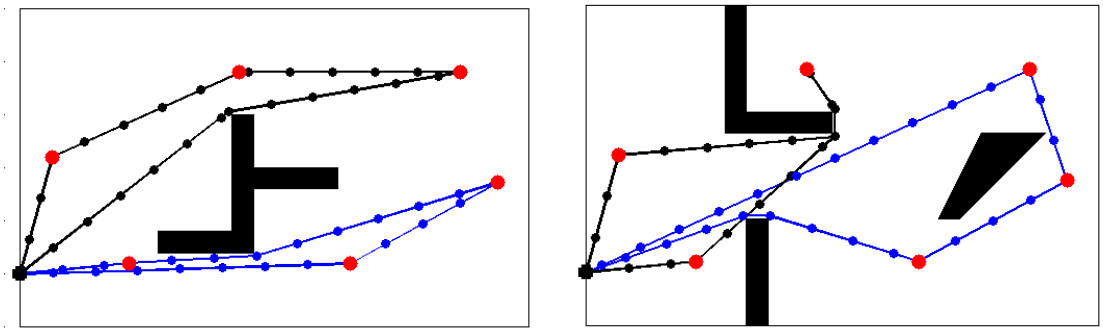
Area with obstacles

When obstacles exist, the length of each tour may increase due to the need to bypass obstacles. Then, the value of $C_{maxrelay,i}$ may increase according to the obstacle configu-

	Without Obstacles fig. 8.5	
	Robot 1	Robot 2
$C_{max,i}$	3	3
Deployment duration (s)	995	947
PoIs visited	3	3
$C_{maxrelay,i}$	22	22
$C_{relay,i}$	22	21

Table 8.3: The deployment duration with relay nodes.

ration. We fix $C_{maxrelay,i} = 23$ for Configuration 1 and $C_{maxrelay,i} = 24$ for Configuration 2, respectively (see Table 8.4). In Figure 8.6a, we observe that the two robot tours are different from those in Figure 8.4a due to the constraint on $C_{maxrelay,i}$. However, in Configuration 2 (see Figures 8.4b and 8.6b), the two robot tours do not change, whether we take the relay nodes into account or not. When we try to decrease $C_{maxrelay,i}$ to a value less than 24, we do not get a Nash equilibrium as all the strategies are eliminated due to a violation of the $C_{maxrelay,i}$ constraints.



a $C_{max,i} = 3$, $C_{max,-i} = 3$.

b $C_{max,i} = 3$, $C_{max,-i} = 3$.

Figure 8.6: Impact of obstacles on the best robot tours, with relay nodes placement.

	Configuration 1 fig. 8.6a		Configuration 2 fig. 8.6b	
	Robot 1	Robot 2	Robot 1	Robot 2
$C_{max,i}$	3	3	3	3
Deployment duration (s)	951	1014	1077	786
$C_{maxrelay,i}$	23	23	24	24
$C_{relay,i}$	21	23	24	19

Table 8.4: The deployment duration, with relay nodes and obstacles.

8.3.4.3 Summary

In Section 8.3, we aimed to find the trajectories of two mobile robots which satisfy the following constraints: all the PoIs are visited exactly once and the tour duration of the robots is minimized. Game theory is used to formalize this problem as a non-cooperative

game with two players, to find the Nash equilibria for various configurations. We studied the impact of obstacles on the deployment duration. The robot tours may differ depending on whether obstacles are present or not. However, obtaining the Nash equilibrium, provides the best combination of the two robot tours that satisfy the various constraints considered in this work.

In the next section, we study the use of a higher number of robots (i.e. > 2) to solve the same problem. However, the objective is no longer to minimize the duration of both robot tours; we focus on achieving three objectives: minimizing the longest tour duration, balancing the duration of the different tours and using the smallest number of robots.

8.4 Multi-robot assisted deployment: based on a multi-objective optimization approach

The problem of multiple robots to deploy sensor nodes can be seen as The Vehicle Routing Problem (VRP) (72), generalizes the Traveling Salesman Problem. The vehicle routing problem aims to find a set of tours that visits all positions at a minimal cost by finding the shortest path, the minimum number of vehicles, etc. The vehicles start and end their tours at the depot. Each position is visited only once, by only one vehicle, and each vehicle has a limited capacity.

Our problem, called the Multi-Robot Deploying wireless Sensor nodes (MRDS)problem, presents many similarities to the VRP problem: mobile robots correspond to vehicles and Points of Interest (PoIs) where sensor nodes should be placed to ensure the monitoring task, correspond to the positions to be visited. However, there are differences in the objectives to optimize, as we will see in the next section.

Our goal is to minimize the deployment duration of static sensor nodes, at Points of Interest (PoIs), in a given environment by $K \geq 1$ mobile robots. Since, on the one hand, robots are battery-operated, and on the other hand, the environment may be hostile (e.g. deployment in a post-crisis situation), the duration of the deployment must be as short as possible. In addition, the best balancing between robot tour duration is required. Sensor node positions are computed such that PoI coverage and network connectivity are ensured, meaning that there is at least one path from each sensor node to the sink in order to forward the collected data.

8.4.1 Problem formalization

The Multi-Robot Deploying wireless Sensor nodes (MRDS) problem is defined as follows:

Let $\{1, \dots N\}$ be the set of PoIs to be visited by robots. By convention, 0 is called the depot. It is the departure and arrival point of the robots. Let $K \geq 1$ be the number of available robots. The problem is to design a set of k tours, one tour per robot with $1 \leq k \leq K$, that:

- minimizes the longest tour duration,
- minimizes the number of robots used,

- minimizes the standard deviation of the robot tour duration.

The constraints are:

- Any robot k, with $1 \leq k \leq K$, has a limited capacity Q_k: it is unable to carry more than Q_k sensors.
- Each robot starts and ends its tour at the depot.
- Each PoI should be visited by exactly one robot.

The MRDS problem can be formulated as follows.
N: is the total number of PoIs to be visited, $K \geq 1$ is the number of available robots and K^* is the number of robots actually used. Thus, we have $1 \leq K^* \leq K$. The depot is denoted by 0, and the PoIs are denoted by 1, 2 or N.
Q_k: is the capacity of robot k.
$d_{i,j}$: is the distance required to travel from node i to node j.
l_s : is the linear speed of each robot.
a_s : is the angular speed of each robot.
$a_{i,j,t}$: is the angle formed by the segments $[i,j]$ and $[j,t]$.
The decision variables of the model are:
X_{ij}^k : is the decision variable that is equal to 1 if robot k visits PoI j immediately after PoI i, and is equal to 0 otherwise.
Y_i^k : is the decision variable that is equal to 1 if PoI i is visited by robot k and is equal to 0 otherwise.

Let TT_k be the tour duration of robot k. This duration combines the duration due to the distance traveled and the duration due to direction changes.

$$TT_k = \sum_{i=0}^{N}\sum_{j=0}^{N} d_{i,j} * X_{ij}^k/ls + \sum_{i=0}^{N}\sum_{j=1}^{N}\sum_{t=0}^{N} \hat{\theta}_{i,j,t} * X_{ij}^k * X_{jt}^k/as \tag{8.2}$$

First objective: minimizing the longest tour TT:

$$Minimize\left(TT = \max_{k\in[1,K]} TT_k\right) \tag{8.3}$$

Second objective: minimizing the number of robots used NT (i.e. the number of tours):

$$Minimize\,(NT = K^*) \tag{8.4}$$

Third objective: minimizing the standard deviation σ of the robot tour duration:

$$Minimize(\sigma = \sqrt{\frac{1}{K^*}\left(\sum_{k=1}^{K^*} TT_k^2\right) - \left(\frac{1}{K^*}\sum_{k=1}^{K^*} TT_k\right)^2}) \tag{8.5}$$

Constraints:

- Each PoI is visited by exactly one robot

$$\sum_{k=1}^{K} Y_i^k = 1 \quad \forall i \in [1, N] \tag{8.6}$$

- The number of robots used is equal to $K^* \leq K$

$$\sum_{k=1}^{K} \sum_{i=1}^{N} Y_i^k = \sum_{k=1}^{K^*} \sum_{i=1}^{N} Y_i^k = N \tag{8.7}$$

- Each robot visits a number of PoIs that is less than its capacity

$$\sum_{i=1}^{N} Y_i^k \leq Q_k \quad \forall k \in [1, K^*] \tag{8.8}$$

- Subtours are eliminated

$$\sum_{i=0}^{N} \sum_{j=0}^{N} X_{ij}^k \leq \sum_{i=1}^{N} Y_i^k \quad \forall k \in [1, K^*] \tag{8.9}$$

- Decision variables $\in \{0, 1\}$

$$X_{ij}^k \in \{0, 1\}, Y_i^k \in \{0, 1\}, \forall i \in [1, N]; \forall k \in [1, K] \tag{8.10}$$

Thus, from equations 8.3, 8.4 and 8.5, the new MRDS problem is defined as follows,

$$Minimize \left(f_{\{TT,NT,\sigma\}} = \max_{k \in K}(TT_k), K^*, \right.$$

$$\left. \sqrt{\frac{1}{K^*} \left(\sum_{k=1}^{K^*} TT_k^2 \right) - \left(\frac{1}{K^*} \sum_{k=1}^{K^*} TT_k \right)^2} \right) \tag{8.11}$$

under the constraints 8.6 to 8.10 described above.

Property 1 *A necessary feasibility condition of the MRDS problem is given by:*

$$\sum_{k=1}^{K} Q_k \geq N. \tag{8.12}$$

8.4.2 NSGA-II based approach for MRDS optimization

8.4.2.1 Overview of NSGA-II

Multi-objective optimization (also known as multi-objective programming, vector optimization and multi-criteria optimization) is an area of multiple criteria decision making, that is concerned with mathematical optimization problems involving more than one objective function to be optimized simultaneously. Optimizing a group of objective functions is not a simple task. The Multi-objective Optimization Problem (MOP) can be formulated as follows:

$$(MOP)\begin{cases} min \quad f_i(x), \quad i \in [1,m] \\ \qquad\quad s.t \\ x \in D \end{cases}$$

Where the vector $x = (x_1, \ldots, x_n)^T \in D$ is the vector of n decision variables and m is the number of objectives. D is the feasible solution space, and $f_i(x)$ is the objective function, and the vector $y = (y_1, y_2, \ldots, y_m)$ is a solution, with $y_i = f_i(x)$.

Definition 1 *For any MOP minimization, a solution $x \in D$ is said to be dominated by solution $x' \in D$ (it is denoted by $x \prec x'$) if the following conditions are satisfied:*

i) $f_i(x) \leq f_i(x')\ \forall\ i \in [1,m]$

ii) $\exists\ i \in [1,m]$ *such that* $f_i(x) < f_i(x')$

The set of optimal solutions is composed of the non-dominated vectors, often called the Pareto front and also denoted $PF^* = \{x \in D \mid \exists\ x^{'} \in D, x^{'} \prec x\}$.
In other words, the Pareto front provides the best trade off for the objectives considered. The goal of the multi-objective optimization is to find the Pareto front for a given problem. NSGA-II (73), Non dominated Sorting Genetic Algorithm, is often used to solve multi-objective optimization problems. This algorithm is a multi-objective version of the genetic algorithm in which the solutions explored are classified into Pareto-optimal fronts.

8.4.2.2 NSGA-II algorithm for the MRDS problem

NSGA-II begins with an initial population P made up of solution vectors called individuals. At each iteration, an auxiliary population Q is formed by applying the crossover and mutation operators (lines 8 to 15). Then, both the current P and the new population Q are merged together to form one set of solutions R, which will be sorted according to the non-domination and crowded comparison (line 17). Finally, only the best individuals in R can be included in the next generation and will participate in the production step while the other individuals are deleted (lines 19 to 25). These steps are repeated until the maximum number of iterations is reached.

Algorithm 2 NSGA-II algorithm for an MoP problem

Input N **N population size**
P_c **crossover probability**
P_m **mutation probability**
$Nbr\_iteration\_max$

1: $Itr \leftarrow 0$ {current iteration}
2: $P_{Itr} \leftarrow \{\emptyset\}$ {population of iteration Itr}
3: initialize $P_{Itr=0} = \{\overrightarrow{x}^{i}_{Itr=0}, , \overrightarrow{x}^{N}_{Iter=0}\}$
4: evaluate $P_{Itr=0}$
5: **while** $(Itr < Nbr\_iteration\_max)$ **do**
6: $Q_{Itr} \leftarrow \{\emptyset\}$ {new population}
7: $t \leftarrow 0$
8: **while** $(t \leq size(Q_{Itr})/2)$ **do**
9: $parents \leftarrow$ **selection**(P_{Itr})
10: $Child \leftarrow$ **crossover**$(P_c, parents)$
11: $E \leftarrow$ **mutation**$(P_m, Child)$
12: **compute_objective_values**$(Child)$
13: $Q_{Itr} \leftarrow Q_{Itr} \cup \{Child\}$
14: $t \leftarrow t + 1$
15: $R_{Itr} \leftarrow P_{Itr} \cup \{Q_{Itr}\}$
16: $R_{Itr} = \bigcup_{i=1}^{r} F_i$ where F_i is a Pareto front meeting $F_1 < F_2 < \ldots < F_r$
17: $P_{Itr+1} \leftarrow \{\emptyset\}; i \leftarrow 0$
18: **while** $(|P_{Itr+1}| + |F_i| < N)$ **do**
19: $P_{Itr+1} \leftarrow P_{Itr+1} \cup F_i$
20: $i \leftarrow i + 1$
21: **ranking**$(F_i, crowding\_distance)$
22: $Itr \leftarrow Itr + 1$
23: $P_{Itr} \leftarrow P_{Itr} \cup \{N - |P_{Itr}|\ first\ solutions\ in\ F_i\}$

8.4.2.3 Application of NSGA-II to the MRDS problem

Individual Representation

In an MRDS problem, an individual represents a possible solution: for a set of robot tours where the k^{th} robot makes the k^{th} tour, each tour is defined by the sequence of PoIs visited by the same robot. That is why we use two parts to define an individual. The first part of the individual, called the sensor-part, is a set of integers where each integer represents a PoI visited by a robot. The second part of the individual, called the robot-part, contains information on the robots. The robot-part contains as many genes as tours (i.e. robots used) described in the sensor-part. In the robot-part, the k^{th} gene is equal to the number of PoIs visited by the k^{th} robot. An example of an individual is depicted in Figure 8.7, where 3 robots are needed, robot 1 visiting 3 PoIs (4, 6 and 5 in this order). Robot 2 visiting PoIs 1 and 2, and robot 3 visiting PoIs 3, 7 and 8 in this order.

Definition 2 *An individual is said to be valid (i.e. it corresponds to a feasible solution) if only if each PoI occurs exactly once in the sensor-part (in other words, the sensor part is a permutation of the sequence* $1, 2, \ldots N$*), the sum of the numbers included in the robot-part is equal to* N*, the number of PoIs to visit, and the size of the robot-part (i.e. number of robots actually used) is less than or equal to* K*.*

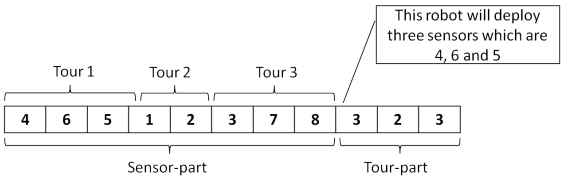


Figure 8.7: An example of an individual

Crossover

The crossover operator is one of the main parts of NSGA-II. The input of this operator consists of two solution vectors (known as parents). The output is two child vectors, which inherit certain features from both parents. For the sensor-part, we will apply PMX (74) (Partially Matched Crossover).

Mutation

After recombination, the mutation operator is applied to randomly change some genes in an individual. This operator serves as a strategy to prevent solutions from being trapped in local optima. An example of mutation is illustrated in Figure 8.8, where the 8^{th} gene of the sensor-part of the individual described in Figure 8.7 suffers a mutation. This gene corresponding to PoI 8 belongs to the third robot (see Figure 8.8(1)). After mutation, this gene belongs to the third robot (i.e. PoI 8 is visited by robot 2), see Figure 8.8(2). To make the mutated individual valid, the robot-part of the individual is updated (minus 1 in

the number of PoIs visited by robot 3 and plus 1 for robot 2), see Figure 8.8(3) depicting the new individual obtained after the mutation of gene 8.

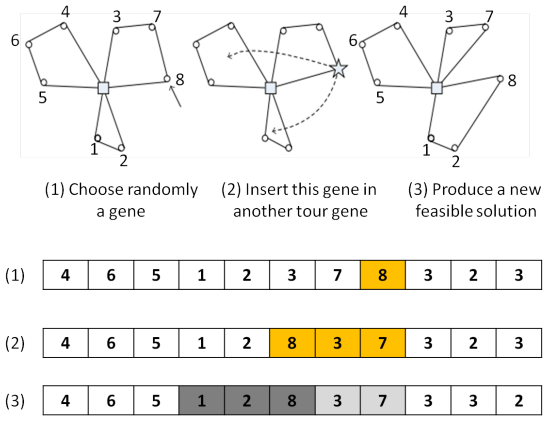


Figure 8.8: An example of a mutation operator

Selection

The selection process in NSGA-II is based on the dominance principle.

Selection of the best solutions

From experimentation, we identify a set of Pareto optimal solutions by gathering all the non-dominated solutions found in 30 independent runs. These solutions provide various tradeoffs between the three objective functions considered. We leave the designer to choose the best tradeoff. For instance, if a solution with two robots is preferred, the designer will select the solution with two robots that provides the smallest deployment duration. If several solutions provide close tour durations, the designer may opt for the solution with the smallest standard deviation.

8.4.3 Hybrid algorithm for the MRDS problem

Figure 8.9 depicts a solution found by NSGA-II that minimizes the standard deviation for 3 robots and 20 PoIs. It is obvious that the duration of each tour in this figure is not minimized. There exists a permutation of the ordered sequence of PoIs visited that improves the tour duration.

More generally, NSGA-II needs many iterations to find good solutions, especially when it starts with initial solutions that have been randomly chosen. We decided to provide NSGA-II with initial solutions that had been already optimized by a heuristic, in order to improve the quality of the solutions obtained by NSGA-II during this time. We also noticed that for any given number of robots, any permutation of the sequence of PoIs visited that decreases the tour duration tends to improve the first objective (i.e. minimizing

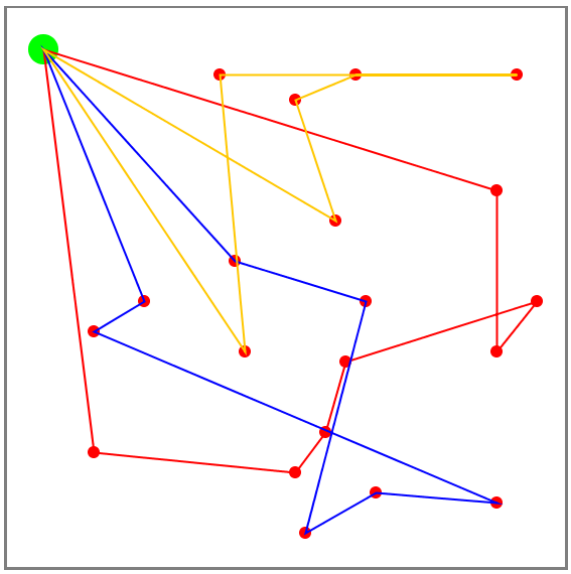


Figure 8.9: Best solution minimizing the standard deviation for 3 robots and 20 PoIs.

the longest tour duration), has no effect on the second objective (number of robots used), and has a positive impact on the third objective (the standard deviation of the robots tour durations). In other words, the new solution dominates the initial one. That is why we choose the 2-Opt heuristic (75) to improve the tour duration of any individual given to, or created by, NSGA-II. The 2-Opt algorithm (75) starts with an initial solution and tries to iteratively improve it by replacing two edges with two new ones that reduce the tour duration. This algorithm provides a local optimum based on the solution that was initially given.
To improve the performance of the NSGA-II algorithm we decided to combine it with the 2-Opt algorithm adapted to the MRDS problem, resulting in an algorithm which we called Hybrid. More precisely, instead of starting with an initial random population, the Hybrid algorithm applies the 2-Opt algorithm to optimize each individual of the initial population. In addition, at each iteration, the children obtained with the crossover operator are mutated with the gene mutation probability and then optimized by applying the 2-Opt algorithm again. As we will see in the next section, the solutions obtained by Hybrid dominate those obtained by NSGA-II when the initial solutions are randomly chosen.

8.4.4 Problem resolution

To solve the MRDS problem, we have implemented both the NSGA-II algorithm and the Hybrid algorithm using the Java programming language. We evaluated the performance of the NSGA-II algorithm and the Hybrid algorithm using 6 configurations with different numbers of PoIs $\{10, 20, 30, 40\}$ located in a $500m$ x $500m$ area. For each configuration, we conducted 30 independent simulation runs. Table 8.5 shows the simulation parameters used in each configuration, the mutation probability is equal to 0.1.

Number of nodes	Number of robots	Robot capacity	NSGA-II iterations	NSGA-II population size
10	3	10	500	40
20	4	10	500	60
30	5	10	500	80
40	6	10	500	100

Table 8.5: Simulation parameters

8.4.4.1 Deployment duration and presence of obstacles

Our goal is to evaluate the solutions provided by both NSGA-II and Hybrid in terms of the three objectives considered in the MRDS problem. Each simulation run gives a Pareto front. We then build the final Pareto front of each configuration from the 30 Pareto fronts previously obtained. Furthermore, we quantify the simulation time needed to obtain these results.

In the real environment, obstacles are always present,and their presence has a big impact on the robot tour and the deployment duration. One or several obstacles may exist between two consecutive PoIs in the robot tour, and the strategy adopted to bypass them is presented in Section 8.3.2.

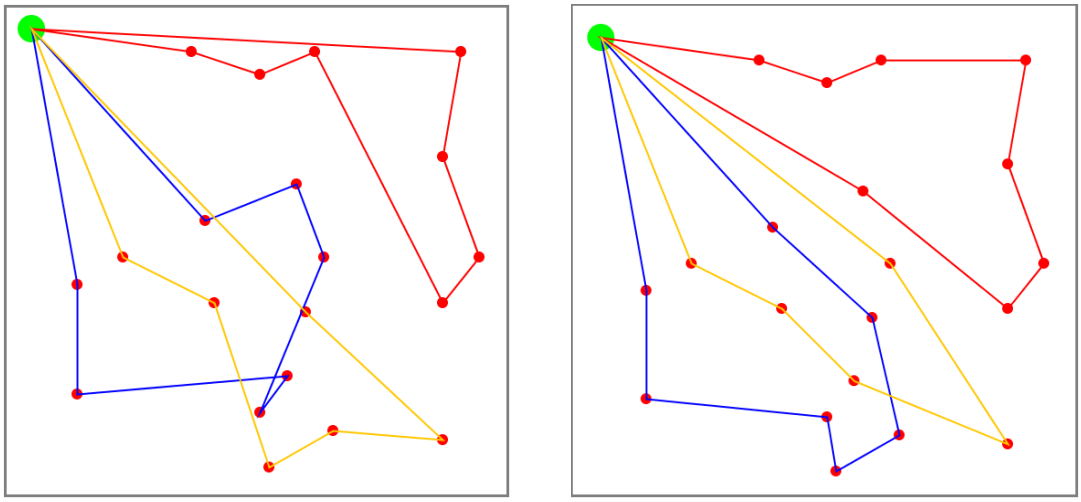
8.4.4.2 Simulation results

When the number of PoIs is small, (e.g. 20 PoIs) both NSGA-II and Hybrid algorithms provide close Pareto fronts. For instance, Figure 8.11a depicts the Pareto fronts obtained when 20 PoIs are deployed in an area without obstacles. However, when obstacles are present, the Pareto front obtained by the Hybrid algorithm is better in terms of tour duration and balanced tours (i.e. standard deviation), as shown in Figure 8.11b.

Figures 8.10a and 8.10b illustrate the best solutions from the Pareto front for 20 PoIs and 3 robots with the smallest maximum tour duration with the NSGA-II and Hybrid algorithms. NSGA-II provides a maximum tour duration of $1416s$ and a standard deviation of 25.32, whereas the Hybrid algorithm gives $1328s$ and a standard deviation of 3.7, respectively, so, the solution from the Pareto front obtained by Hybrid dominates that obtained by NSGA-II.

In large configurations, for instance when 30 or 40 PoIs should be deployed and whether obstacles are present or are not, the Pareto fronts obtained by the Hybrid algorithm outperform those obtained by NSGA-II in terms of tour duration and tours balanced. This is due to the use of the 2-Opt algorithm that prevents edges crossing in the same tour, leading to smaller tour durations and better balancing between these tours.

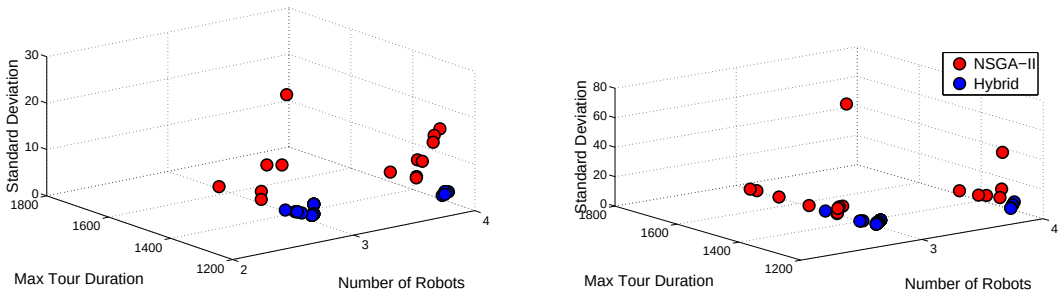
To demonstrate the distribution of non-dominated individuals on the objective space for NSGA-II and Hybrid algorithms, we considered 4 configurations (10, 20, 30 and 40 PoIs) both with and without obstacles. Figures 8.11a, 8.12a, and 8.13a depict the Pareto front obtained by gathering all the non-dominated solutions found by each algorithm in the 30 independent runs corresponding to these configurations.



a NSGA-II algorithm. b Hybrid algorithm.

Figure 8.10: Tours of 3 robots with 20 PoIs, without obstacles.

For each possible number of robots, the distribution of non-dominated solutions found by Hybrid is more localized than with NSGA-II, thereby facilitating the choice of a suitable solution.

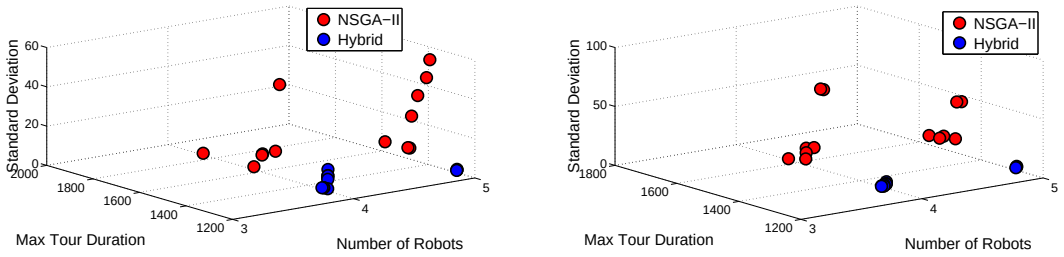


a Without obstacles. b With obstacles.

Figure 8.11: Pareto front obtained by 20 PoIs.

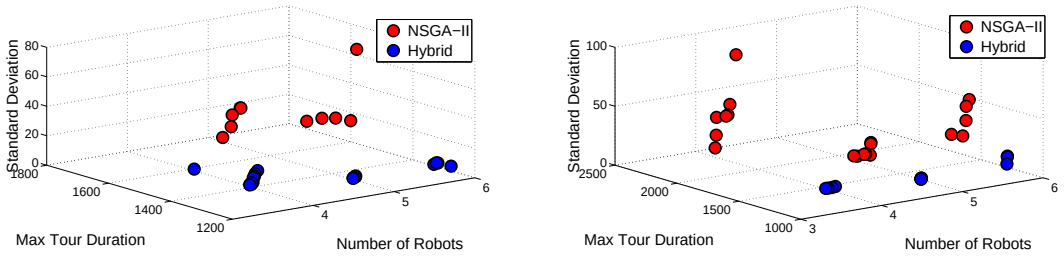
Figures 8.11a, 8.12a, and 8.13a show the improvement in the quality of the solutions obtained by Hybrid compared to NSGA-II. We can observe that, in most cases, the Pareto front of Hybrid dominates the Pareto front of NSGA-II. However, this improvement has a cost in terms of simulation time, as shown in Table 8.6. All our experiments were conducted using a desktop computer Intel Xeon E5 1620 processor with 8-Core 3.6GHz with 8 Gb of memory.

Let us study the impact of obstacles. Figure 8.14 depicts the tours of 3 robots visiting 20 PoIs in the presence of obstacles. NSGA-II provides a smallest maximum tour duration of 1443s and a standard deviation of 77, whereas Hybrid gives a smallest maximum tour duration of 1334s and a standard deviation of 4.8. Clearly, the presence of obstacles leads to longer tour durations and larger simulation times, as shown in Table 8.6. As expected,



a Without obstacles. b With obstacles.

Figure 8.12: Pareto front obtained by 30 PoIs.



a Without obstacles. b With obstacles.

Figure 8.13: Pareto front obtained by 40 PoIs.

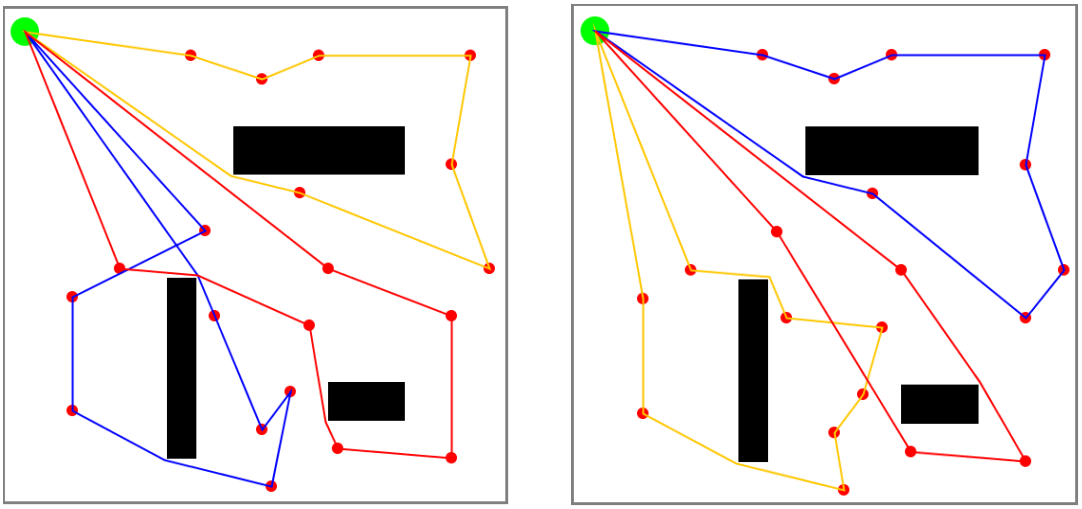
	Without Obstacles				With Obstacles			
	NSGA-II		Hybrid		NSGA-II		Hybrid	
PoI	Average	Std. deviation	Average	Std. deviation	Average	Std. deviation	Average	Std. deviation
10	32.07	1.33	75.52	3.43	841.63	71.11	1436.04	84.81
20	133.45	4.56	248	35.39	1502.09	141.38	1520.44	253.14
30	359.17	13.23	557.75	71.01	1709.11	140.2	2296.58	414.62
40	1287	32.38	1027	130.5	2870.81	243.36	3269.3	538.39

Table 8.6: The average and the standard deviation of simulation times (in seconds) obtained by NSGA-II and Hybrid algorithms

in most cases the Pareto front provided by Hybrid dominates the Pareto front given by NSGA-II.

8.4.4.3 Summary

In Section 8.4, we formalized the multi-objective optimization problem for MRDS. We solved it with the genetic algorithm NSGA-II and a Hybrid algorithm combining NSGA-II and the 2-Opt heuristic for various configurations (10, 20, 30 and 40 PoIs visited) both with and without obstacles. We showed that the Hybrid algorithm provides the Pareto front that, in most cases, dominates the Pareto front given by NSGA-II.



a NSGA-II algorithm. b Hybrid algorithm.

Figure 8.14: Tours of 3 robots with 20 PoIs and obstacles.

8.5 Conclusion

In this chapter, we focused on how to minimize the deployment duration of static sensor nodes. Several mobile robots are involved in the deployment. We considered two cases. In the first case, there are exactly two robots to deploy the sensor nodes at precomputed positions. Here, the goal is to minimize the duration of the robots' tours. We addressed this problem as a non-cooperative game with two players (TRDS), and found the Nash equilibrium. TRDS has been published in (76). In the second case, there are more than two robots. The goal is threefold: to minimize the longest tour duration, to balance the duration of tours, and to reduce the number of robots used. We addressed this problem as a multi-objective optimization problem (MRDS), and solved it with the NSGA-II and a Hybrid algorithm that combines the NSGA-II and the 2-Opt algorithm. In both cases, we studied the impact of obstacles. Table 8.7, summarizes the algorithms proposed in this chapter.

Table 8.7: Computation of robot trajectories to minimize the deployment duration

	Area	**Obstacles**	**Energy-efficiency**	**Fault-tolerant** with regard to connectivity
TRDS	known	known	– Minimizes the deployment duration	– 2 paths toward the sink
MRDS	known	known	– Minimizes the deployment duration – Balances the tour duration – Minimizes the number of robots	– 2 paths toward the sink

Part V

Discussion and Conclusion

Chapter 9

Conclusion and perspectives

Contents

9.1 Conclusion

Wireless Sensor Networks are usually deployed to monitor real-world phenomena. How accurate the information gathered will actually be, greatly depends on the manner in which the sensors are deployed and, in particular, the positions of the sensor nodes themselves. Bearing in mind that these positions must satisfy the coverage and connectivity requirements of the application in question, deployment algorithms are needed to determine the optimal positions of the sensor nodes.

9.1.1 Synthesis

This thesis mainly focuses on the deployment of sensor nodes, both when the nodes are able to position themselves autonomously, and when their deployment is assisted using mobile robots. In both cases, not only must this deployment meet the coverage and connectivity requirements of the application, but it should also minimize the number of nodes needed while meeting various constraints (e.g. obstacles, energy consumption, fault-tolerant connectivity).

An optimal deployment implies that the sensor nodes are in the best positions, such that full coverage of the area is ensured, and network connectivity is maintained. This deployment is based on a geometric pattern. In our study, we made a theoretical computation of optimal deployments in both a 2D space and a 3D space. In a 2D space, it is based on a triangular lattice; in a 3D space, it is based on a truncated octahedron tessellation.

An optimal deployment cannot, however, be achieved when the area to be covered is unknown and contains obstacles. In such a case, an autonomous deployment may well be suitable. Autonomous deployment implies that the sensor nodes are mobile, that they are able to cooperate with their neighbors to compute their final positions, and that they are capable of avoiding obstacles. In our work, we adopted a virtual forces strategy to enable the sensor nodes to spread quickly over the entire area, while maintaining network connectivity. We proposed the ADVFA, GDVFA and OA-DVFA algorithms to operate in a 2D space, and the 3D-DVFA algorithm to deploy sensor nodes in a 3D space. OA-DVFA has the advantage of being able to cope with known and unknown obstacles and also with node failures while ensuring full coverage and connectivity. OA-DVFA is based on a virtual grid corresponding to the optimal deployment to decide which sensor nodes, already spread over the whole area, are redundant and should be turned off to save energy. Therefore, OA-DVFA is an autonomous deployment algorithm that optimizes the number of active sensor nodes.

On the other hand, the area to be monitored may be known. In this case, the position and shape of the obstacles are also known, and it is preferable to compute an optimized deployment in terms of sensor nodes and coverage rate in a centralized way rather than using a large number of autonomous sensor nodes. The optimized deployment is close to the optimal deployment, but, some additional nodes may be needed to cover the irregular borders of the area and the obstacles. To monitor the whole area, we proposed a projection-based algorithm, called OAD-Area, that computes the sensor node positions in the presence of opaque obstacles, based on a virtual grid of the optimal deployment. In addition, we proposed the OAD-PoI algorithm to place sensor nodes at some specific Points of Interest (PoIs) in the area considered by building paths of relay nodes between each PoI and the sink. The OAD-PoI algorithm is also based on virtual grid of the optimal deployment to select the positions of relay nodes. This strategy favors the sharing of relay nodes between node-disjoint paths and optimizes the length of each path. Thus, the total number of relay nodes deployed is optimized. The OAD-PoI algorithm, not only determines relay node positions but also ensures a fault-tolerant connected network even in the presence of obstacles.

When the deployment is computed in a centralized way, mobile robots can be used to deploy the sensor nodes at their precomputed positions. However, due to energy constraints, the duration of these robots' trajectories should be optimized. To optimize the robot trajectories duration, we formalized our problem based on two approaches: the first model, called TRDS, following a game theory approach, to deploy sensor nodes using two robots. The second one, called MRDS, following a multi-objective optimization approach,

to deploy sensor nodes using more than two robots. We solved MRDS by implementing a multi-objective version of the genetic algorithm, NSGA-II, and the Hybrid algorithm that combines NSGA-II and 2-Opt. Both TRDS and MRDS are able to cope with the presence of obstacles.

9.1.2 Application to the Cluster Connexion project

This thesis was done as a part of the Cluster CONNEXION (digital command COntrol for Nuclear EXport and renovatION) project which aims to propose and validate an innovative architecture suitable for control systems in nuclear power plants in France and abroad. The solution set out in this manuscript could be used to tackle some of the industrial problems targeted by the Cluster CONNEXION project, notably:

The cartography of a radioactive zone in a post-crisis situation. In this context, the area where sensor nodes must be deployed is unknown and usually is hostile (e.g. presence of dangerous radiations). The goal is to set up an operational network as quickly as possible.
If mobile robots are used to discover the area and place sensor nodes at positions that are considered as points of interest, the TRDS (for two robots) and MRDS (for $\geq$2 robots) algorithms are of particular interest. On the other hand, if the sensor nodes are mobile and autonomous, an autonomous deployment can be envisioned, using the OA-DVFA algorithm.

The instrumentation of a temporary worksite. Since wiring is very expensive in nuclear power plants, it is usually kept to the strict minimum and always for permanent networks. That is why to achieve the necessary security requirements, temporary worksites are instrumented by means of wireless sensor networks which must be deployed and maintained operational as long as the temporary worksite itself. Here, we can apply the same solution as in the previous application.

The control system can detect a deviation from normal behavior. In such a case, wireless sensor nodes are deployed at some points of interest to help the control system to determine the exact cause of this deviation. An example of such a situation is the detection of a leaking valve: wireless sensor nodes are deployed upstream and downstream of the valve to be investigated. The OAD-PoI algorithm could be used to compute the relay node positions to ensure network connectivity, and the MRDS algorithm would determine the robot trajectories that minimize the deployment duration.

It is worth mentioning that although, these applications originate from control command in nuclear power plants, similar applications exist in many other industrial contexts.

9.2 Perspectives and Discussion

The deployment algorithms proposed in this manuscript have shown a very good performance. Nevertheless, several future research directions could be followed in order to

enhance these algorithms, enabling them to perform efficiently in real life situations.

We consider four future directions, three concern the sensor deployment aspect: improving the accuracy of our algorithms in the real environment with regard to sensing and communication models, 3D deployment, and implementing them in real robots. The last one deals with data gathering: even if our algorithms were originally designed for sensor deployment, however, they may be used to collect data from sensors.

All these future directions are detailed in the following.

9.2.1 More realistic models

All the algorithms proposed are based on the theoretical models presented in Chapter 5 namely, a sensing and communication range modeled by a disk and a sphere for 2D space and 3D space, respectively. However, due to the constraints imposed by a real environment; these models may fall somewhat short of reflecting reality. Then the crucial question is **how can we improve the accuracy of our algorithms in a real environment?**

Heterogeneous sensing and communication ranges We distinguish two cases where the sensing range, r, and communication range, R, may be heterogeneous. In the first case, they may be heterogeneous due to the coexistence of various types of sensor nodes, each one being deployed to achieve a specific task; for instance, sensors for fire detection and sensors for temperature measurement. These sensors have different values for r and R. In such a case, making these sensors cooperate together to ensure network connectivity is better than ensuring network connectivity for each type of sensor separately. However, unlike with OAD-PoI, relay nodes cannot be placed at equidistant positions, and therefore, when computing the relay node positions, the different existing communication ranges should be taken into account.
In the second case, they may be heterogeneous due to environment constraints such as multipath fading. In this situation, the expected r and R do not match those computed in the area considered. To cope with this problem, real measurements of r and R may be done in different zones of the area, and so, the minimum values of r and R, in each zone, could be used in order to ensure area coverage and network connectivity. In our deployment algorithm: OA-DVFA, OAD-Area and OAD-PoI, the target distance, D_{th}, maintained between neighboring nodes is based on the values of r and R. These algorithms can be improved to operate in a real environment, by adapting D_{th} to the minimum value of r and R in each zone of the area.

Computation and measurement Using TRDS or MRDS, mobile robots do not check whether network connectivity is maintained when they place sensor nodes at their precomputed positions. Since network connectivity may be lost due to environment constraints, TRDS and MRDS could be improved to guarantee network connectivity. The idea is, when the robot reaches the position of a PoI, it should check whether the communication with its previously deployed neighbors is ensured. If not, the robot should either add a new relay node or shift the position of the sensor.

Machine learning Based on machine learning, the values of r and R could be predicted using information and measurements of the area considered. The predicted values of r and R could be used in our algorithms to improve the model accuracy.

9.2.2 From 2D toward 3D

In our work, we mainly focused on sensor deployment to ensure coverage and maintain connectivity of a 2D area. However, 3D deployment is required by many applications. The question is **how can we improve our 2D deployment algorithms to perform in 3D space?**

Interconnection of different floors in buildings To interconnect different floors, our proposed algorithms, OA-DVFA, TRDS or MRDS may be used to deploy sensor nodes on each floor, and then, additional relay nodes can be deployed to maintain network connectivity between each two consecutive floors.

Surface covering Surface covering is similar to 2D deployment since sensor nodes should follow the shape of the surface that is no longer flat. The OA-DVFA algorithm can be used, for example, to monitor the snow level on a mountain. Based on the OA-DVFA algorithm, a mobile sensor node only needs to compute the direction and the distance to travel to reach its final position by following the shape of the surface.

Real 3D In our work, we proposed 3D-DVFA, which is based on virtual forces, to ensure the coverage of a cube. However, 3D-DVFA could be improved to form a 3D barrier coverage or to build a dome over the area to be protected. Such 3D deployment is required by intruder detection applications.

9.2.3 Implementation on real robots

The OA-DVFA algorithm is based on mobile and autonomous robots to ensure full area coverage and maintain network connectivity. Performance evaluations of OA-DVFA have shown that it provides very good results. Then, the question is **how can we implement OA-DVFA on real robots?**
To run OA-DVFA on real robots, certain requirements should be met. Real robots should be able to communicate with each other, and should also be equipped with the appropriate technology to detect obstacles, such as Sonar, Lidar or Radar. The same type of robot may be used to run TRDS and MRDS. Moreover, the trajectories of the robots may intersect, and so, an algorithm to handle intersecting robot trajectories is required.

9.2.4 Use of our algorithms to collect data

To collect the data sensed by sensor nodes, we can distinguish two approaches. The first one is based on network connectivity ensured by deploying relay nodes. The second one uses mobile robots that visit sensor nodes. In our work, we proposed TRDS and MRDS to determine the trajectory of the robots to deploy sensor nodes at their precomputed

positions. The question is **how can we improve TRDS and MRDS to ensure the data gathering task?**

In the case of a 2D area In a 2D area, the TRDS and MRDS algorithms could ensure both sensor deployment and data gathering. First, the robots place a sensor node at each PoI position. The different PoIs may be at a distant higher than the communication range of sensor nodes. When the connectivity is not maintained by relay nodes, TRDS and MRDS could ensure an intermittent connectivity by collecting data from each sensor node. Since, TRDS and MRDS optimize the robot trajectory durations, the data gathering duration will also be optimized.

In the case of different floors Collecting data from different floors is similar to collecting data from several 2D areas. In such a situation, relay nodes may be needed to interconnect the different floors. Then, TRDS and MRDS could be used to collect data from each floor, as in the previous case, and then, communicate with the relay nodes to collect the data from the other floors.

In the case of a 3D area In a 3D area, sensor nodes may be deployed at any location in the 3D space, and our algorithms could be applied to collect data from these sensors. In this case, drones may be used to collect data from the sensor nodes in a 3D space.

List of Publications

International Journals

1. I. Khoufi, S. Mahfoudh, P. Minet, A. Laouiti *Data gathering architecture for temporary worksites based on a uniform deployment of wireless sensors*, International Journal of Sensor Networks (IJSNET), Vol. 18 pp. 3-21 ,2015

2. I. Khoufi, P. Minet, A. Laouiti, S. Mahfoudh (in press) *Survey of Deployment Algorithms in Wireless Sensor Networks: Coverage and Connectivity Issues and Challenges*, International Journal of Autonomous and Adaptive Communications Systems (IJAACS), 2015.

International conferences

1. I. Khoufi, P. Minet, A. Laouiti *OA-DVFA: A Distributed Virtual Forces-based Algorithm to Monitor an Area with Unknown Obstacles*,IEEE Consumer Communications & Networking Conference (CCNC), Las Vegas, USA, 2016.

2. I. Khoufi, M. Hadded, P. Minet, A. Laouiti *Optimized Trajectories of Multi-Robot Deploying Wireless Sensor Nodes*, International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Abu Dhabi, UAE, 2015.

3. I. Khoufi, P. Minet, M.A. Koulali and M. Erradi *A Game Theory-Based Approach for Robots Deploying Wireless Sensor Nodes*, International Conference on Wireless Communications and Mobile Computing (IWCMC), Dubrovnik, Croatia, 2015.

4. N. Boufares, I. Khoufi, P. Minet, L. Saidane and Y. Ben Saied *Three Dimensional Mobile Wireless Sensor Networks Redeployment Based On Virtual Forces* International Conference on Wireless Communications and Mobile Computing (IWCMC), Dubrovnik, Croatia, 2015

5. I. Khoufi, E. Livolant, P. Minet, M. Hadded and A. Laouiti *Optimized trajectory of a robot deploying wireless sensor nodes*, Wireless Days (WD), Rio de Janeiro, Brazil, 2014.

6. I. Khoufi, P. Minet, A. Laouiti, and E. Livolant, *A simple method for the deployment of wireless sensors to ensure full coverage of an irregular area with obstacles*, International conference on Modeling, analysis and simulation of wireless and mobile systems (MSWiM), Montreal, Canada, 2014.

7. S. Mahfoudh, I. Khoufi, P. Minet, A. Laouiti *GDVFA: A Distributed Algorithm Based on Grid and Virtual Forces for the Redeployment of WSNs* IEEE Wireless Communication and Networking Conference (WCNC), Istanbul, Turkey, 2014.

8. S. Mahfoudh, I. Khoufi, P. Minet, A. Laouiti *Relocation of Mobile Wireless Sensors in the Presence of Obstacles*, International Conference on Telecommunications (ICT), Casablanca, Morocco, 2013.

Appendix A

A Robot Trajectory Optimization

A.1 Introduction and motivation

A mobile robot can be used to deploy static wireless sensor nodes to achieve the coverage and connectivity requirements of the applications considered. To save energy and reduce the deployment duration, the tour delay of the robot should be minimized. This delay must take into account not only the time needed by the robot to travel the tour distance but also the time spent in the rotations performed by the robot each time it changes its direction. This problem is called the Robot Deploying Sensor nodes problem, in short, RDS.

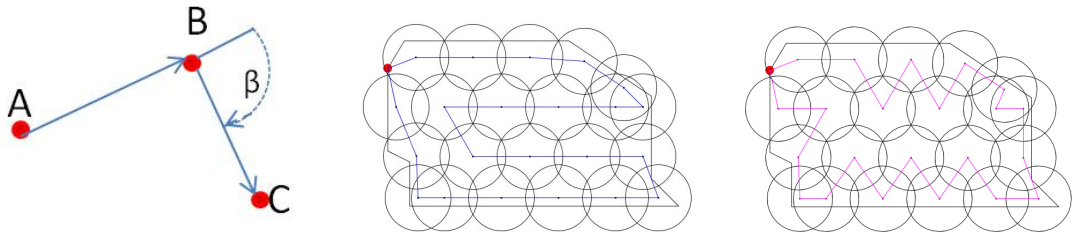
Our problem has many analogies with the well-known Traveling Salesman Problem (TSP), where the goal is to find the smallest tour visiting all the sensor node positions (representing the cities) only once, and going back to the initial position. This problem has been proved NP-hard.

However, our problem differs from the classical traveling salesman problem in that the objective is not to minimize the distance traveled but the duration of the tour, taking into account the angular speed of the robot. Consequently, the cost associated with a tour is equal to the time needed by the robot to perform its tour. It is significantly more complex to evaluate than simply the distance between two cities, B and C, visited successively, as illustrated in Figure A.1a. It should take into account the angle made by the direction the robot is traveling in, when arriving from A to B, and the direction given by BC. Let us consider an example of deployment assisted by a mobile robot. Figure A.1c shows the optimal tour of the robot obtained when only the distance is taken into account: this is the optimal tour of the TSP. We observe many direction changes in this tour. Figure A.1b depicts the optimal tour when both the distance and angle are taken into account: this is the optimal tour for RDS. This tour is smoother than the optimal TSP tour. This example clearly illustrates the difference between the optimal TSP tour and the optimal RDS tour. The optimal tour has a duration of 271.55 seconds and a distance of 2035 meters in the RDS problem, whereas it has a longer duration of 385.66 seconds, but a shorter distance of 1924 meters in the TSP problem.

A.2 Formalization of the RDS problem

The Robot Deploying Sensor nodes problem can be formulated as an Integer Linear Program (ILP).

The robot tour is modeled as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of vertices representing the node positions to be visited during the robot tour and $\mathcal{E}$ is the set of edges



a Rotation angle between cities A, B and C.

b Optimal tour with RDS.

c Optimal tour with TSP.

Figure A.1: Robot tour: comparison between TSP and RDS problems.

representing the path between node positions. The cardinal of $\mathcal{V}$ is n.

Let $d_{i,j}$ denote the distance between the node positions $i \in \mathcal{V}$ and $j \in \mathcal{V}$. Let $a_{i,j,k}$ denote the angle between the edges $(i,j) \in \mathcal{E}$ and $(j,k) \in \mathcal{E}$. Let ls and as be the robot linear speed and the robot angular speed, respectively.

We define $x_{i,j}$, where $i \in \mathcal{V}$ and $j \in \mathcal{V} \setminus \{i\}$, the utility of a path $p \in \mathcal{E}$, i.e. $x_{i,j} = 1$ if and only if p belongs to the robot tour and $x_{i,j} = 0$ otherwise. Furthermore, let $y_{i,j,k}$, where $i \in \mathcal{V}$, $j \in \mathcal{V} \setminus \{i\}$ and $k \in \mathcal{V} \setminus \{i,j\}$, be the robot rotation required at a node position. In other words $y_{i,j,k} = 1$ if and only if the rotation at node j position is effective and $y_{i,j,k} = 0$ otherwise. Finally, we denote $z_{i,j}$, where $i \in \mathcal{V}$ and $j \in \mathcal{V} \setminus \{i\}$, the robot's stock of sensor nodes available on the edge (i,j).

The objective is to minimize the time used by the robot to visit all the sensor node positions. This time takes into account the time due to both the distance and the rotation angle towards the next sensor node position:

$$min \left(\sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V} \setminus \{i\}} d_{i,j}/ls * x_{i,j} + \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V} \setminus \{i\}} \sum_{k \in \mathcal{V} \setminus \{i,j\}} a_{i,j,k}/as * y_{i,j,k} \right)$$

This objective is subject to the following constraints:

$$\forall i \in \mathcal{V}, \sum_{j \in \mathcal{V} \setminus \{i\}} x_{i,j} = 1 \tag{A.1}$$

Constraint A.1 allows only one departure for the robot from a node position.

$$\forall j \in \mathcal{V}, \sum_{i \in \mathcal{V} \setminus \{j\}} x_{i,j} = 1 \tag{A.2}$$

Constraint A.2 authorizes only one arrival for the robot at a node position.

$$\sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V} \setminus \{i\}} \sum_{k \in \mathcal{V} \setminus \{i,j\}} y_{i,j,k} = n - 1 \tag{A.3}$$

Constraint A.3 means that the robot tour includes $n-1$ rotation costs to make a complete tour.

$$\begin{gathered}\forall i \in \mathcal{V}, \forall j \in \mathcal{V} \setminus \{i\}, \forall k \in \mathcal{V} \setminus \{i,j\}, \\ y_{i,j,k} \leq (x_{i,j} + x_{j,k})/2\end{gathered} \tag{A.4}$$

Constraint A.4 ensures that any rotation cost corresponds to a pair of consecutive edges followed by the robot.

$$\forall i \in \mathcal{V} \setminus \{1\}, \forall k \in \mathcal{V} \setminus \{i,1\}, \quad y_{i,1,k} = 0 \tag{A.5}$$

Constraint A.5 guarantees that the rotation cost of the robot in its start position is not accounted for. In fact, when the robot comes back to its start position, it does not need to rotate toward any next node position, since the tour is complete.

$$\forall i \in \mathcal{V}, \forall j \in \mathcal{V} \setminus \{i\}, \quad z_{i,j} \leq (n-1) * x_{i,j} \tag{A.6}$$

Constraint A.6 denotes the maximum capacity of the robot in terms of sensor nodes.

$$\begin{gathered}\forall i \in \mathcal{V}, \\ \sum_{h \in \mathcal{V} \setminus \{i\}} z_{h,i} + \left\{ \begin{array}{lll} n & \text{if} & i = 1 \\ 0 & \text{otherwise} & \end{array} \right\} = \sum_{j \in \mathcal{V} \setminus \{i\}} z_{i,j} + 1\end{gathered} \tag{A.7}$$

Constraint A.7 expresses that the robot carries a certain number of sensor nodes. This number decreases with the number of node positions visited by the robot.

The ILP formulation of the RDS problem differs from that of the TSP problem on the following points:

- The second term in the objective function has been added to account for the time lost in rotations.
- Constraints A.3, A.4, A.5 have been introduced to deal with the robot rotation constraints.

This model of RDS adopts the following assumptions:

A1: The robot knows its location, and is able to move autonomously to any specified location in the area.

A2: The set of sensor node positions as well as the initial location of the robot, are given.

A3: The connectivity graph of sensor node positions is assumed to be complete. In other words, it is always possible for the robot to go from any sensor node position to any other sensor node position. For each pair of sensor node positions, the distance is given. For each triple of sensor node positions, the rotation angle is given.

The next three assumptions are adopted for the sake of simplicity. They will be relaxed in Section A.5.

A4: There is no obstacle in the paths between any two sensor node positions.

A5: The robot has enough capacity to carry all the sensor nodes it has to deploy.

A6: The robot has enough energy to visit all the sensor node locations in a single tour.

A.3 Proposed algorithms

We now describe the algorithms designed to solve the RDS problem, and we will compare them in the next section. We consider a algorithm that provides the exact solution and three algorithms that provide approximated solutions.

A.3.1 The exact solution

The exact solution of the RDS problem is provided by the CPLEX solver (77) using the problem formulation given in Section A.2. This exact solution will be used as a reference to evaluate the closeness to the optimal of approximated solutions. Various approximated solutions are used. The first one, called 2-Opt, is based on iterative improvement, the second one uses a genetic approach; and the third one is hybrid algorithm combining both.

A.3.2 2-Opt algorithm

We adapt the well-known 2-Opt algorithm (75) to the RDS problem. 2-Opt starts with an initial solution and tries to iteratively improve it by replacing two edges with two new ones that reduce the tour duration. This algorithm provides a local optimum based on the initial solution.

A.3.3 Genetic algorithm

A genetic algorithm takes its inspiration from the biological evolution process. To define a genetic algorithm, we have to define a selection of the initial population, the operators we use for the selection of parents, the crossover and the mutation, and, finally, the make up of the population used in the next iteration as well as the fitness function. In the traveling salesman problem, an individual is defined by an ordered sequence of the cities visited by the robot. The initial population is given by K individuals; K is a non-null natural integer, each individual being a random permutation of the C cities to visit. The first city is the initial location of the robot. Hence, it is given as an input.

Let $\mathcal{P}_i$ denote the population at the beginning of any iteration $i > 0$. The Genetic algorithm randomly selects $\lfloor K/2 \rfloor$ pairs of parents among the current population $\mathcal{P}_i$, applies the crossover operator on each pair to generate two children. Each gene (i.e. a city visited) of a child is subject to a mutation with a gene mutation probability of P_{mut}. A new population $\mathcal{P}_{new}$ is then generated. All individuals of $\mathcal{P}_i \cup \mathcal{P}_{new}$ are evaluated by

the fitness function. The fitness of an individual is equal to one over the time needed by the robot to perform this tour. The $\lfloor K/2 \rfloor$ best individuals (i.e., maximizing the fitness function) among the $\mathcal{P}_i \cup \mathcal{P}_{new}$ are selected, and they are completed by $K - \lfloor K/2 \rfloor$ individuals randomly selected from the unselected ones to constitute $\mathcal{P}_{i+1}$, the population considered in the next iteration. This principle enables the algorithm to always keep the $\lfloor K/2 \rfloor$ best individuals it has found during the $Imax$ iterations.

In (74), a genetic algorithm is built to solve the TSP. The mutation operator exchanges two genes, selected at two random positions, of an individual. The three crossover operators considered, PMX (for Partially Matched Crossover), CX (for Cyclic Crossover) and OX (for Ordered Crossover), ensure that the crossover of any two individuals is still an individual (i.e. a permutation of the C cities to visit). Furthermore, it is shown that PMX outperforms the two other crossover operators CX and OX. Hence, we select PMX as our crossover operator, using two randomly selected cross points.

A.3.4 Hybrid algorithm

The Hybrid algorithm combines the 2-Opt algorithm with a genetic one. More precisely, instead of starting with an initial random population, the Hybrid algorithm applies the 2-Opt algorithm to optimize each individual of the initial population. In addition, at each iteration, the children obtained with the crossover operator are mutated with the gene mutation probability and then optimized by applying again the 2-Opt algorithm.

A.4 Comparative evaluation

We evaluate the different algorithms presented in Section A.3 on different configurations ranging from 10 sensor nodes to 154 sensor nodes. These configurations may meet various application requirements. For instance, small configurations with less than 30 nodes are representative of temporary industrial worksites, where coverage of some interest points and connectivity with a sink must be achieved. Medium to large configurations, from 50 to 150 nodes, can represent industrial warehouses where full coverage and connectivity with a sink must be met. Small and medium configurations with less than 70 nodes can also be encountered to improve data gathering for an industrial process, to detect leakages for example.

For this performance evaluation, we take the following parameters values: $ls = 10$ meters per second, $as = 10$ degrees per second for the robot linear and angular speeds, respectively; $P_{mut} = 0.15$, $Imax = 1000$ iterations for Genetic and $Imax = 100$ for Hybrid, $K \geq 2 * C$ individuals, where C is the number of sensor nodes to deploy. Sensor nodes are deployed in the area depicted in Figure A.1c. The dimensions of the circumscribing rectangle are $530m$ x $300m$, the sensing range varies from $140m$ to $20m$ to match a number of sensor nodes from 10 to 154.

First, we compute the solutions to the TSP and RDS problems for a number of sensor node positions ranging from 10 to 154, using 2-Opt. Figure A.2 clearly shows that for very small configurations (i.e., configurations with at most 10 sensor nodes), the tours found by TSP and RDS may be the same. This is no longer the case when the number of sensor

nodes increases: the difference between the TSP solution and the RDS solution increases considerably.

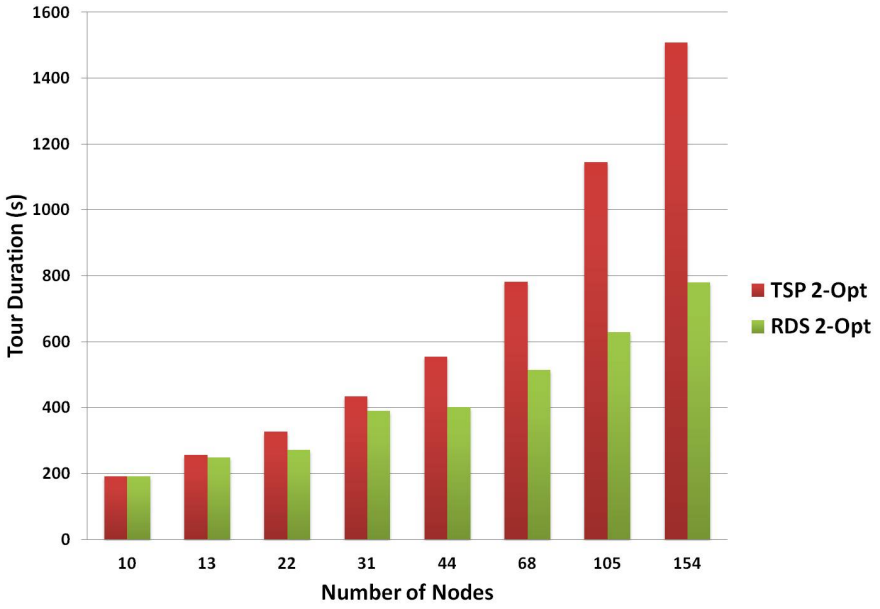


Figure A.2: Solutions found by 2-Opt for the RDS and TSP problems.

Secondly, we compare the accuracy (i.e., closeness to the optimal) of the solutions provided by each algorithm tested with small configurations (≥ 31 sensor nodes). Figure A.3 depicts the solutions given by 2-Opt, Genetic and Hybrid versus the optimal one in small configurations. When the number of nodes is higher than 13 sensor nodes, the Genetic algorithm fails to find the optimal tour in 1000 iterations. This is due to the fact that it generates many tours that are not useful. On the other hand, 2-Opt provides a solution that is close to the optimal for the configurations tested. Hybrid provides the best results as an approximation algorithm.

For large configurations, Hybrid improves on the solution found by 2-Opt as shown in Figure A.4. For instance, for 103 and 154 nodes, Hybrid decreases the tour duration from 629.1s to 628.15s and from 752.95s to 749.41s, respectively. This can be explained by the fact that 2-Opt can be blocked in a local optimal, whereas Hybrid uses mutations and crossovers to explore other tours. However, 2-Opt is better to exploit a given solution. We also observe that Genetic is very sensitive to the choice of the initial population: if it is far from the optimal, the final solution remains far from the optimal. In the configurations tested, 2-Opt improves the initial solution by at least 50%.

Another interesting result is given by the time needed by each algorithm to compute its final solution. CPLEX is run on a Quad-core Intel Xeon W3565 3.2GHz platform, whereas the other algorithms are run on a 8-core Intel i7-2760QM 2.4GHz platform. Tables A.1 and A.2 depict the computation time for each algorithm tested.

As expected, Optimal needs the largest computation time in all the configurations tested, except for 10 nodes. For example, it takes about 3 hours to solve the RDS problem

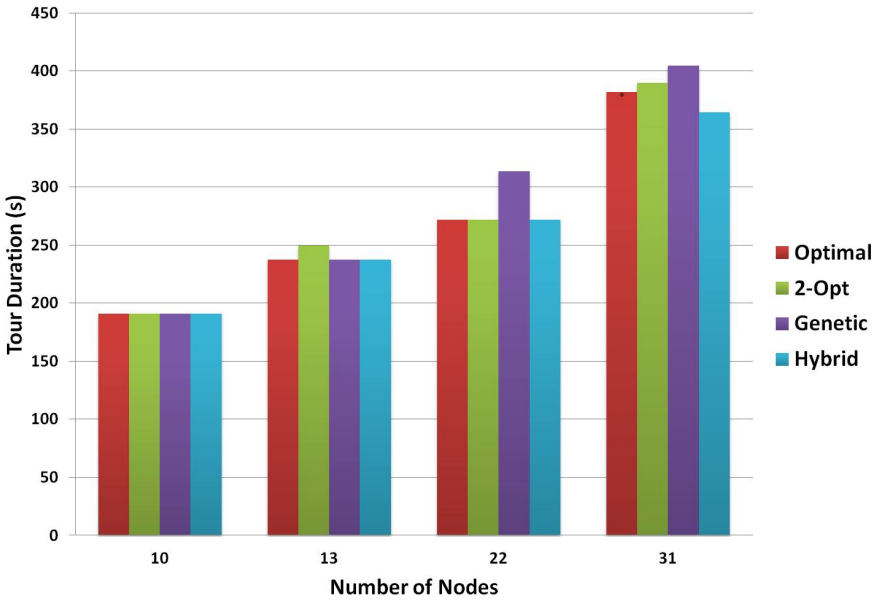


Figure A.3: Final solutions of Optimal, 2-Opt, Genetic and Hybrid for small configurations.

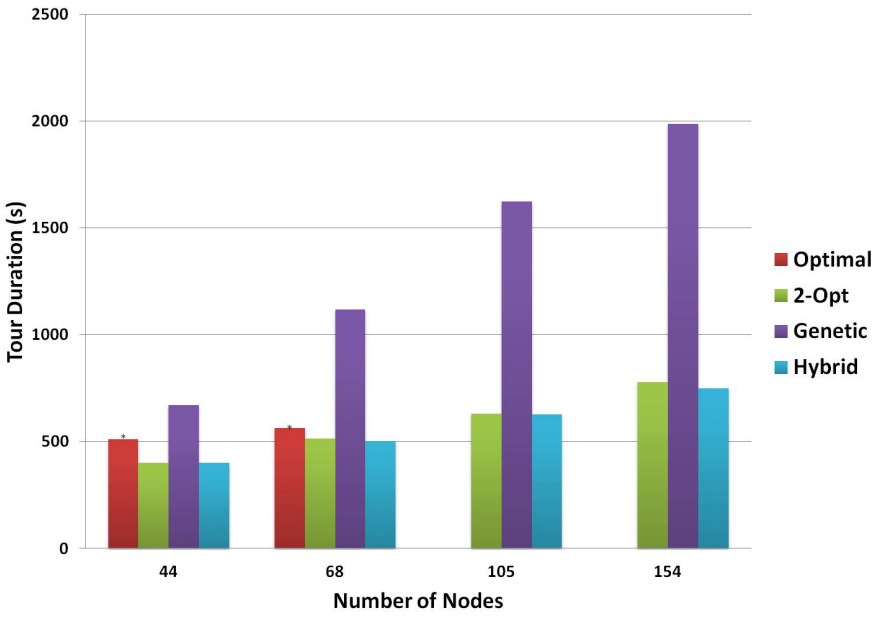


Figure A.4: Final solutions of Optimal, 2-Opt, Genetic and Hybrid for large configurations.

with 22 sensor nodes. Hybrid needs the second largest time. This is due to the calls to the 2-Opt algorithm applied first on each individual of the initial population and then on each child generated by the crossover operator. For example, it takes 636 seconds (about 6.5mn) to generate the final solution of the RDS problem with 22 sensor nodes.

Table A.1: Computation time for Optimal, 2-Opt, Genetic and Hybrid for small configurations.

Number of nodes	10	13	22	31
Optimal (s)	11.18	217.35	10866.24	87387.77*
2-Opt (s)	0.005	0.003	0.014	0.029
Genetic (s)	0.396	0.505	0.845	1.161
Hybrid (s)	2.808	6.788	44.319	276.9

Table A.2: Computation time for Optimal, 2-Opt, Genetic and Hybrid for large configurations.

Number of nodes	44	68	105	154
Optimal (s)	174828.61*	103910.62*	-	-
2-Opt (s)	0.114	0.339	2.051	3.689
Genetic (s)	2.857	7.065	18.969	39.8
Hybrid (s)	870.45	3743.66	27111.4	41267.3

The fastest algorithms are 2-Opt and, to a lesser extent Genetic. In all the configurations tested, 2-Opt is at least 10 times faster than Genetic, and this ratio reaches 100 times in small configurations. Since Genetic may provide a solution that is far from the optimal one, 2-Opt is preferred. For larger configurations, (more than 31 nodes), we did not obtain the optimal solution with CPLEX after 24 hours of computation. Since in configurations with more than 30 nodes, CPLEX needs over 24 hours, we take as a final solution, the best solution found by CPLEX in 24 hours. This solution may be a non-optimal one, as depicted in Figure A.4 by a star symbol. In all these configurations, Hybrid provides the best results. We recommend the Hybrid algorithm for large configurations because it provides the best trade-off between the optimal closeness and an acceptable computation time.

A.5 Discussion

In this section, we show how to relax the assumptions A4 (no obstacle), A5 (enough capacity) and A6 (enough energy).

A.5.1 Obstacles

Up to now we have considered a sensor deployment in an area without obstacles. However in the real life, obstacles may well be present. In this section, we show how to relax assumption A4 and extend the solutions given previously to cope with obstacles. In Chapter 7, we proposed the OAD-Area algorithm that copes with transparent and opaque obstacles, and ensures full coverage. This algorithm computes the sensor node positions that are given as input to the RDS algorithm. It is hard to compute an optimized robot trajec-

tory when taking obstacles into account. If there exist obstacles between two consecutive sensor node positions, a direct path between these two positions is impossible. Therefore, we propose a strategy to bypass these obstacles while minimizing the trajectory duration. This strategy is explained in Section 8.3.2 in Chapter 8.

A.5.2 Capacity constraint

Up to now we have assumed that the robot has the capacity to carry all its sensor nodes at the same time. If this is not the case, assumption A5 is no longer true. In such a case, the robot has to perform subtours starting at its initial position. How can we solve this new problem?

In order to handle the new carrying capacity constraint, we enhance the integer linear program of Section A.2 as follows:

Let *cap* be the robot carrying capacity in terms of the number of sensor nodes. The objective is the same as before and only three constraints are modified. Constraints A.1 and A.2 specifying that there is only one arrival at and departure from each city are relaxed to enable multiple arrivals and departures in the initial robot position. In fact, the robot must come back to its initial position to replenish its sensor node stock.

$$\forall i \in \mathcal{V} \setminus \{1\}, \sum_{j \in \mathcal{V} \setminus \{i\}} x_{i,j} = 1 \tag{A.8}$$

$$\forall j \in \mathcal{V} \setminus \{1\}, \sum_{i \in \mathcal{V} \setminus \{j\}} x_{i,j} = 1 \tag{A.9}$$

Furthermore, the capacity constraint A.6 must be updated according to the capacity parameter *cap*.

$$\forall i \in \mathcal{V}, \forall j \in \mathcal{V} \setminus \{i\}, \quad z_{i,j} \leq cap * x_{i,j} \tag{A.10}$$

Figure A.5 depicts an optimal RDS tour when the robot has to deploy 13 sensor nodes and its capacity is limited to 6 sensor nodes. This optimal tour comprises 3 subtours depicted in blue, each of them starting at the initial position of the robot.

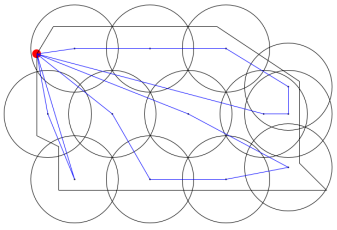


Figure A.5: Optimal RDS tour with a limited capacity of 6 sensor nodes.

Table A.3 gives the number of subtours done by the mobile robot when its capacity is 6, 5, and 4 sensor nodes respectively. When the robot capacity decreases, the number of subtour increases, as expected, leading to a longer tour duration.

Table A.3: Optimal tour with different robot capacities.

Robot capacity	13	8	6	4
Subtours (number)	1	2	3	4
Length (m)	1712	2169	2562	3048
Time cost (s)	237.64	262.98	319.11	385.70

A.5.3 Energy constraint

What happens if the maximum energy level of the robot does not allow it to visit all the sensor node locations in a single tour? Assumption A6 is no longer true. Here again, the robot proceeds by subtours, refilling its battery at its initial position which is also the starting point of each subtour. One idea could be to group sensor nodes into clusters, such that the robot deploys all the sensor nodes of the same cluster in a single subtour. It should be noted that the number of cluster members may differ from one cluster to another, since the robot consumes more energy to visit a distant cluster than to visit a close one.

A.6 Conclusion

In this appendix, we proposed RDS problem to optimize the delay needed by a mobile robot to deploy sensor nodes, taking into account the rotations performed by the robot to change its direction. The delay-optimized tour of a mobile robot may result in a reduction of at least 50% in the time needed to deploy its wireless sensor nodes. This smaller deployment time may be crucial not only in emergency applications, but also in industrial process control because the latency before the first data gathering is reduced. This benefit is obtained by using the optimal solution that can be provided by an integer linear program solver like CPLEX, for instance, in small and medium configurations. For larger configurations, however, the time needed to obtain the optimal solution using an integer linear program solver may become prohibitive. That is why we use the Hybrid algorithm, which successfully combines the exploration of the Genetic algorithm with the exploitation of 2-Opt algorithm.
RDS is the subject of the publication (78).
In large configurations, the use of multiple robots rather than a single robot will decrease the deployment duration and avoid sub tours. However, when several robots are used, our objective is not only to optimize the deployment duration but also to balance the robot tours and minimize the number of robots required. In this context, our problem is formalized as a multi-objective optimization problem, called MRDS, and solved using the NSGA-II algorithm, the multi-objective version of the genetic algorithm; and the Hybrid algorithm, which combines NSGA-II and 2-Opt, (see Chapter 8).

Bibliography

[1] H. Zhang and J. C. Hou, "Maintaining sensing coverage and connectivity in large sensor networks," *Ad Hoc & Sensor Wireless Networks*, vol. 1, no. 1-2, pp. 89–124, 2005.

[2] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill, "Integrated coverage and connectivity configuration in wireless sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems*. ACM, 2003, pp. 28–39.

[3] X. Bai, S. Kumar, D. Xuan, Z. Yun, and T. H. Lai, "Deploying wireless sensors to achieve both coverage and connectivity," in *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*. ACM, 2006, pp. 131–142.

[4] K. Mougou, S. Mahfoudh, P. Minet, and A. Laouiti, "Redeployment of randomly deployed wireless mobile sensor nodes," in *Vehicular Technology Conference (VTC Fall)*. IEEE, 2012, pp. 1–5.

[5] K. Kar and S. Banerjee, "Node placement for connected coverage in sensor networks," in *WiOpt'03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2003, pp. 2–pages.

[6] G. Tan, S. Jarvis, A.-M. Kermarrec *et al.*, "Connectivity-guaranteed and obstacle-adaptive deployment schemes for mobile sensor networks," *IEEE Transactions on Mobile Computing*, vol. 8, no. 6, pp. 836–848, 2009.

[7] Y.-C. Wang, C.-C. Hu, and Y.-C. Tseng, "Efficient deployment algorithms for ensuring coverage and connectivity of wireless sensor networks," in *International Conference on Wireless Internet*. IEEE, 2005, pp. 114–121.

[8] B. Wang, "Coverage problems in sensor networks: A survey," *ACM Computing Surveys (CSUR)*, vol. 43, no. 4, p. 32, 2011.

[9] R. Mulligan and H. M. Ammari, "Coverage in wireless sensor networks: a survey," *Network Protocols and Algorithms*, vol. 2, no. 2, pp. 27–53, 2010.

[10] A. Ghosh and S. K. Das, "Coverage and connectivity issues in wireless sensor networks: A survey," *Pervasive and Mobile Computing*, vol. 4, no. 3, pp. 303–334, 2008.

[11] G. Fan and S. Jin, "Coverage problem in wireless sensor network: A survey," *Journal of networks*, vol. 5, no. 9, pp. 1033–1040, 2010.

[12] C. Zhu, C. Zheng, L. Shu, and G. Han, "A survey on coverage and connectivity issues in wireless sensor networks," *Journal of Network and Computer Applications*, vol. 35, no. 2, pp. 619–632, 2012.

[13] S. Mahfoudh, P. Minet, and A. Laouiti, "Overview of deployment and redeployment algorithms for mobile wireless sensor networks," *Procedia Computer Science*, vol. 10, pp. 946–951, 2012.

[14] M. C. Akewar and N. V. Thakur, "A study of wireless mobile sensor network deployment," *Internatioal Journal of Computer and Wireless Communication*, vol. 2, no. 4, 2012.

[15] R. Soua, P. Minet, and E. Livolant, "Modesa: An optimized multichannel slot assignment for raw data convergecast in wireless sensor networks," in *Performance Computing and Communications Conference (IPCCC)*. IEEE, 2012, pp. 91–100.

[16] H.-L. Wang and W.-H. Chung, "The generalized k-coverage under probabilistic sensing model in sensor networks," in *Wireless Communications and Networking Conference (WCNC)*. IEEE, 2012, pp. 1737–1742.

[17] Y. Zou and K. Chakrabarty, "Sensor deployment and target localization based on virtual forces," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications.*, vol. 2. IEEE, 2003, pp. 1293–1303.

[18] J. Li, B. Zhang, L. Cui, and S. Chai, "An extended virtual force-based approach to distributed self-deployment in mobile sensor networks," *International Journal of Distributed Sensor Networks*, 2012.

[19] J. Chen, S. Li, and Y. Sun, "Novel deployment schemes for mobile sensor networks," *Sensors*, vol. 7, no. 11, pp. 2907–2919, 2007.

[20] Y.-h. Kim, C.-M. Kim, D.-S. Yang, Y.-j. Oh, and Y.-H. Han, "Regular sensor deployment patterns for p-coverage and q-connectivity in wireless sensor networks," in *International Conference on Information Networking (ICOIN)*. IEEE, 2012, pp. 290–295.

[21] Z. Yun, X. Bai, D. Xuan, T. H. Lai, and W. Jia, "Optimal deployment patterns for full coverage and k-connectivity (k 6) wireless sensor networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 18, no. 3, pp. 934–947, 2010.

[22] N. A. A. Aziz, K. A. Aziz, and W. Z. W. Ismail, "Coverage strategies for wireless sensor networks," *World academy of science, Engineering and technology*, vol. 50, pp. 145–150, 2009.

[23] J. Xiao, S. Han, Y. Zhang, and G. Xu, "Hexagonal grid-based sensor deployment algorithm," in *Control and Decision Conference (CCDC)*. IEEE, 2010, pp. 4342–4346.

[24] N. Bartolini, T. Calamoneri, E. G. Fusco, A. Massini, and S. Silvestri, "Push & pull: autonomous deployment of mobile sensors for a complete coverage," *Wireless Networks*, vol. 16, no. 3, pp. 607–625, 2010.

[25] A. Mateska and L. Gavrilovska, "Wsn coverage & connectivity improvement utilizing sensors mobility," in *Wireless Conference 2011-Sustainable Wireless Technologies (European Wireless)*. VDE, 2011, pp. 1–8.

[26] P. Park, S.-G. Min, and Y.-H. Han, "A grid-based self-deployment schemes in mobile sensor networks," in *Proceedings of the 5th International Conference on Ubiquitous Information Technologies and Applications (CUTE)*. IEEE, 2010, pp. 1–5.

[27] G. Fletcher, X. Li, A. Nayak, and I. Stojmenovic, "Back-tracking based sensor deployment by a robot team," in *IEEE Communications Society Conference on Sensor Mesh and Ad Hoc Communications and Networks (SECON)*. IEEE, 2010, pp. 1–9.

[28] G. Wang, G. Cao, and T. La Porta, "Movement-assisted sensor deployment," *IEEE Transactions on Mobile Computing*, vol. 5, no. 6, pp. 640–652, 2006.

[29] H. Tan, Y. Wang, X. Hao, Q.-S. Hua, and F. C. Lau, "Arbitrary obstacles constrained full coverage in wireless sensor networks," in *Wireless Algorithms, Systems, and Applications*. Springer, 2010, pp. 1–10.

[30] S. Babaie and S. S. Pirahesh, "Hole detection for increasing coverage in wireless sensor network using triangular structure," *IJCSI International Journal of Computer Science Issues*, vol. 9, no. 2, 2012.

[31] M. Keskin, I. Altmel, N. Aras, and C. Ersoy, "Optimal deployment, scheduling and routing for maximizing the lifetime of a wireless sensor networks with multiple mobile sinks," *Bogaziçi University*, 2013.

[32] M. Hefeeda and M. Bagheri, "Wireless sensor networks for early detection of forest fires," in *IEEE International Conference on Mobile Adhoc and Sensor Systems*. IEEE, 2007, pp. 1–6.

[33] Y. Liu and W. Liang, "Approximate coverage in wireless sensor networks," in *The IEEE Conference on Local Computer Networks, 2005. 30th Anniversary*. IEEE, 2005, pp. 68–75.

[34] E. Saad, M. Awadalla, and R. Darwish, "A data gathering algorithm for a mobile sink in large-scale sensor networks," in *International Conference on Wireless and Mobile Communications, ICWMC*. IEEE, 2008, pp. 207–213.

[35] V. Kavitha and E. Altman, "Analysis and design of message ferry routes in sensor networks using polling models," in *Proceedings of the 8th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*. IEEE, 2010, pp. 247–255.

[36] S. Mahfoudh, I. Khoufi, P. Minet, and A. Laouiti, "Relocation of mobile wireless sensors in the presence of obstacles," in *International Conference on Telecommunications (ICT)*. IEEE, 2013, pp. 1–5.

[37] Q. Xu and Q. Wang, "Coverage optimization deployment based on virtual force directed in wireless sensor networks," in *International Conference on Computer Technology and Science*. ICCTS, 2012.

[38] R. Iyengar, K. Kar, and S. Banerjee, "Low-coordination topologies for redundancy in sensor networks," in *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*. ACM, 2005, pp. 332–342.

[39] J. Carle and D. Simplot-Ryl, "Energy-efficient area monitoring for sensor networks," *IEEE Computer*, no. 2, pp. 40–46, 2004.

[40] J.-P. Sheu, S.-C. Tu, and C.-H. Yu, "A distributed query protocol in wireless sensor networks," *Wireless Personal Communications*, vol. 41, no. 4, pp. 449–464, 2007.

[41] Y. Wu, C. Ai, S. Gao, and Y. Li, "p-percent coverage in wireless sensor networks," in *Wireless Algorithms, Systems, and Applications*. Springer, 2008, pp. 200–211.

[42] S. Kumar, T. H. Lai, and A. Arora, "Barrier coverage with wireless sensors," in *Proceedings of the 11th annual international conference on Mobile computing and networking.* ACM, 2005, pp. 284–298.

[43] A. Saipulla, C. Westphal, B. Liu, and J. Wang, "Barrier coverage of line-based deployed wireless sensor networks," in *INFOCOM.* IEEE, 2009, pp. 127–135.

[44] A. Saipulla, B. Liu, G. Xing, X. Fu, and J. Wang, "Barrier coverage with sensors of limited mobility," in *Proceedings of the eleventh ACM international symposium on Mobile ad hoc networking and computing.* ACM, 2010, pp. 201–210.

[45] A. Saipulla, C. Westphal, B. Liu, and J. Wang, "Barrier coverage with line-based deployed mobile sensors," *Ad Hoc Networks*, vol. 11, no. 4, pp. 1381–1391, 2013.

[46] L. Kong, Y. Zhu, M.-Y. Wu, and W. Shu, "Mobile barrier coverage for dynamic objects in wireless sensor networks," in *International Conference on Mobile Adhoc and Sensor Systems (MASS).* IEEE, 2012, pp. 29–37.

[47] S. He, J. Chen, X. Li, X. Shen, and Y. Sun, "Cost-effective barrier coverage by mobile sensor networks," in *INFOCOM, Proceedings IEEE.* IEEE, 2012, pp. 819–827.

[48] Y. Wang and G. Cao, "Barrier coverage in camera sensor networks," in *Proceedings of the Twelfth ACM International Symposium on Mobile Ad Hoc Networking and Computing.* ACM, 2011, p. 12.

[49] M. Erdelj, T. Razafindralambo, and D. Simplot-Ryl, "Covering points of interest with mobile sensors," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 1, pp. 32–43, 2013.

[50] M. Erdelj, V. Loscri, E. Natalizio, and T. Razafindralambo, "Multiple point of interest discovery and coverage with mobile wireless sensors," *Ad Hoc Networks*, vol. 11, no. 8, pp. 2288–2300, 2013.

[51] M. Li, W. Cheng, K. Liu, Y. He, X. Li, and X. Liao, "Sweep coverage with mobile sensors," *IEEE Transactions on Mobile Computing*, vol. 10, no. 11, pp. 1534–1545, 2011.

[52] X. Li, H. Frey, N. Santoro, and I. Stojmenovic, "Strictly localized sensor self-deployment for optimal focused coverage," *IEEE Transactions on Mobile Computing*, vol. 10, no. 11, pp. 1520–1533, 2011.

[53] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," *Wireless networks*, vol. 8, no. 5, pp. 481–494, 2002.

[54] A. Gallais, J. Carle, D. Simplot-Ryl, and I. Stojmenovic, "Localized sensor area coverage with low communication overhead," *IEEE Transactions on Mobile Computing*, vol. 7, no. 5, pp. 661–672, 2008.

[55] S. Alam and Z. J. Haas, "Coverage and connectivity in three-dimensional networks," in *Proceedings of the 12th annual international conference on Mobile computing and networking.* ACM, 2006, pp. 346–357.

[56] G. Tan, S. Jarvis, A.-M. Kermarrec *et al.*, "Connectivity-guaranteed and obstacle-adaptive deployment schemes for mobile sensor networks," *IEEE Transactions on Mobile Computing*, vol. 8, no. 6, pp. 836–848, 2009.

[57] I. Khoufi, S. Mahfoudh, P. Minet, and A. Laouiti, "Data gathering architecture for temporary worksites based on a uniform deployment of wireless sensors," *International Journal of Sensor Networks*, p. 19, 2014.

[58] 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).* IEEE, 2014, pp. 3040–3045.

[59] N. Boufares, I. Khoufi, P. Minet, L. Saidane, and Y. Ben Saied, "Three dimensional mobile wireless sensor networks redeployment based on virtual forces," in *International Conference on Wireless Communications and Mobile Computing (IWCMC)*, 2015.

[60] C.-Y. Chang, C.-T. Chang, Y.-C. Chen, and H.-R. Chang, "Obstacle-resistant deployment algorithms for wireless sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 6, pp. 2925–2941, 2009.

[61] Y.-C. Wang, C.-C. Hu, and Y.-C. Tseng, "Efficient placement and dispatch of sensors in a wireless sensor network," *IEEE Transactions on Mobile Computing*, vol. 7, no. 2, pp. 262–274, 2008.

[62] S. Lee and M. Younis, "Optimized relay node placement for connecting disjoint wireless sensor networks," *Computer Networks*, vol. 56, no. 12, pp. 2788–2804, 2012.

[63] F. Senel and M. Younis, "Optimized relay node placement for establishing connectivity in sensor networks," in *Global Communications Conference (GLOBECOM).* IEEE, 2012, pp. 512–517.

[64] H. M. Almasaeid and A. E. Kamal, "On the minimum k-connectivity repair in wireless sensor networks," in *IEEE International Conference on Communications, 2009. ICC'09.* IEEE, 2009, pp. 1–5.

[65] X. Han, X. Cao, E. L. Lloyd, and C.-C. Shen, "Fault-tolerant relay node placement in heterogeneous wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 9, no. 5, pp. 643–656, 2010.

[66] S. Misra, S. D. Hong, G. Xue, and J. Tang, "Constrained relay node placement in wireless sensor networks to meet connectivity and survivability requirements," in *Conference on Computer Communications, INFOCOM.* IEEE, 2008.

[67] D. Chen and P. K. Varshney, "Geographic routing in wireless ad hoc networks," in *Guide to Wireless Ad Hoc Networks.* Springer, 2009, pp. 151–188.

[68] I. Khoufi, P. Minet, A. Laouiti, and E. Livolant, "A simple method for the deployment of wireless sensors to ensure full coverage of an irregular area with obstacles," in *Proceedings of the 17th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems.* ACM, 2014, pp. 203–210.

[69] C.-Y. Chang, J.-P. Sheu, Y.-C. Chen, and S.-W. Chang, "An obstacle-free and power-efficient deployment algorithm for wireless sensor networks," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 39, no. 4, pp. 795–806, 2009.

[70] H.-Y. Shi, W.-L. Wang, N.-M. Kwok, and S.-Y. Chen, "Game theory for wireless sensor networks: a survey," *Sensors*, vol. 12, no. 7, pp. 9055–9097, 2012.

[71] "Gambit: Software tools for game theory," http://gambit.sourceforge.net/.

[72] B. M. Baker and M. Ayechew, "A genetic algorithm for the vehicle routing problem," *Computers & Operations Research*, vol. 30, no. 5, pp. 787–800, 2003.

[73] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[74] N. Kumar and R. K. Karambir, "A comparative analysis of pmx, cx and ox crossover operators for solving traveling salesman problem," *International journal of Latest Research in science and technology*, vol. 1, 2012.

[75] G. A. Croes, "A method for solving traveling-salesman problems," *Operations research*, vol. 6, no. 6, pp. 791–812, 1958.

[76] I. Khoufi, P. Minet, M. A. Koulali, and M. Erradi, "A game theory-based approach for robots deploying wireless sensor nodes," in *International Conference on Wireless Communications and Mobile Computing (IWCMC)*, 2015.

[77] R. Lougee-Heimer, "The common optimization interface for operations research: Promoting open-source software in the operations research community," *IBM Journal of Research and Development*, vol. 47, no. 1, pp. 57–66, 2003.

[78] I. Khoufi, E. Livolant, P. Minet, M. Hadded, and A. Laouiti, "Optimized trajectory of a robot deploying wireless sensor nodes," in *Wireless Days (WD), 2014 IFIP.* IEEE, 2014, pp. 1–6.