TESIS DE LA UNIVERSIDAD DE ZARAGOZA

2018 33

José Manuel Palacios Gasós

Multi-Robot Persistent Coverage in Complex Environments

Departamento
Informática e Ingeniería de Sistemas

Director/es
Sagües Blazquiz, Carlos
Montijano Muñoz, Eduardo

ISSN 2254-7606

Prensas de la Universidad
Universidad Zaragoza

ISSN 2254-7606

Universidad Zaragoza
1542

Tesis Doctoral

MULTI-ROBOT PERSISTENT COVERAGE IN COMPLEX ENVIRONMENTS

Autor

José Manuel Palacios Gasós

Director/es

Sagües Blazquiz, Carlos
Montijano Muñoz, Eduardo

UNIVERSIDAD DE ZARAGOZA

Informática e Ingeniería de Sistemas

2018

Ph.D Thesis

Multi-Robot Persistent Coverage in Complex Environments

José Manuel Palacios Gasós

Advisors: Eduardo Montijano Muñoz

Carlos Sagüés Blázquiz

Departamento de Informática e Ingeniería de Sistemas (DIIS)
Instituto de Investigación en Ingeniería de Aragón (I3A)
Escuela de Ingeniería y Arquitectura (EINA)

Universidad de Zaragoza

December 19, 2017

Multi-Robot Persistent Coverage in Complex Environments

José Manuel Palacios Gasós

Supervisors

Eduardo Montijano Muñoz	Universidad de Zaragoza, Spain
Carlos Sagüés Blázquiz	Universidad de Zaragoza, Spain

Dissertation Committee

José Ramón García	BSH Home Appliances, Spain
Andrea Gasparri	Roma Tre University, Italy
Gonzalo López	Universidad de Zaragoza, Spain
Carlos Franco	BSH Home Appliances, Spain
Miguel Aranda	Institut Pascal, France

External Examiners

Mac Schwager	Stanford University, USA
Javier Alonso-Mora	Delft University of Technology, NL

Funding

Scholarship C076/2014 and Grupo T04	Gobierno de Aragón, European Social Fund
RTC-2014-1847-6 and DPI-2015-69376-R	MINECO, Spain / UE
CUD2013-05 and CUD2016-17	Centro Universitario de la Defensa, Spain

To Carmen

To my parents and sisters

Acknowledgements

During these years, I had the opportunity to collaborate and learn from many outstanding professionals and incredible people. I want to thank from these short lines all of them, who definitely deserve a big part of the credit of this thesis.

In the first place, I want to thank my supervisors, Eduardo and Carlos, for their guidance and support. Thank you for being always available to discuss research or help me with whatever I needed. You are a role model and I hope we can keep working together in the future.

I also want to thank BSH Home Appliances Group in Zaragoza for the opportunity that they gave me to pursue my Ph.D. in the context of an exemplary collaboration with the university. I appreciate all the support, means and freedom to conduct my research. In particular, I want to specially thank all the people there: Sergio, David, Tomás and all the others for embracing me and being always willing to help. I also want to thank the DISAL lab at EPFL for hosting me for a few months and for their contribution to a fruitful collaboration. My labmates at the university, Alejandro, Raúl and the others also deserve a thank you for the good times that we spent together.

This thesis is the result of years of hard work but it gave me the chance to meet some extraordinary people daring to follow the same path: Alberto, Fernando, Edgar and Eduardo. Sharing all the time and experiences we became good friends and I am grateful for it.

Finally, I want to thank my parents, my sisters and brothers-in-law for always being there for me no matter what and for showing me their unconditional love and support. I want to dedicate an special thank you to my Dad, Manuel, for being far more than a dad with his technical support. You really know what this thesis means and I hope you are proud of what we made together. Definitely, I am. My deepest thank you is to my wife, Carmen. Thank you for your love and for always being by my side. I would not have been able to reach this milestone without you.

Abstract

Recent advances in mobile robotics and an increasing development of affordable autonomous mobile robots have motivated an extensive research in multi-robot systems. The complexity of these systems resides in the design of communication, coordination and control strategies to perform complex tasks that a single robot can not. A particularly interesting task is that of persistent coverage, that aims to maintain covered over time a given environment with a team of robotic agents. This problem is of interest in many applications such as vacuuming, cleaning a place where dust is continuously settling, lawn mowing or environmental monitoring. More recently, the apparition of useful unmanned aerial vehicles (UAVs) has encouraged the application of the coverage problem to surveillance and monitoring.

This thesis focuses on the problem of persistently covering a continuous environment in increasingly more difficult settings. At first, we propose a receding-horizon optimal solution for a centralized system in a convex environment using dynamic programming. Then we look for distributed solutions, which are more robust, scalable and efficient. To deal with the lack of global information, we present a communication-effective distributed estimation algorithm that allows the robots to have an accurate estimate of the coverage of the environment even when they can not exchange information with all the members of the team. Using this estimation, we propose two different solutions based on coverage goals, which are the points of the environment in which the coverage can be improved the most. The first method is a motion controller, that combines a gradient term with a term that drives the robots to the goals, and which performs well in convex environments. For environments with some obstacles, the second method plans open paths to the goals that are optimal in terms of coverage. Finally, for complex, non-convex environments we propose a distributed algorithm to find equitable partitions for the robots, i.e., with an amount of work proportional to their capabilities. To cover this region, each robot plans optimal, finite-horizon paths through a graph of sweep-like paths.

The final part of the thesis is devoted to discrete environment, in which only a finite set of points has to be covered. We propose a divide-and-conquer strategy to separate the problem to reduce its complexity into three smaller subproblem, which can be optimally solved. We first plan closed paths through the points, then calculate the optimal coverage times and actions to periodically satisfy the coverage required by the points, and finally join together the individual plans of the robots into a collision-free team plan that minimizes simultaneous motions. This solution is eventually used for a novel application that is domestic induction heating with mobile inductors. We adapt it to the particular setting of a domestic hob and demonstrate that it performs really well in a real prototype.

Contents

Chapter 1

Introduction

In recent years, the popularity of robots in our society has increased significantly thanks to the variety of task that they are capable of performing. The range of applications in which they are used has increasingly expanded from the traditional assembly-line robotic arms to the trendy household cleaning robots. In every case, robots are intended to carry out tasks that are tedious, dangerous or even impossible for humans. They are capable of assembling components tirelessly, deactivating explosives in adversarial environments, lifting heavy loads or operating with exceptional precision, as shown in Fig. 1.1. In fact, one of the most interesting capabilities of the robots is mobility, due to the possibilities that it offers for transportation, surveillance, agriculture, search and rescue or assistance. Research in mobile robotics is currently a very active field and addresses a diversity of subjects such as mapping, localization, path planning or motion control.

(a) Bosch Rooxter.[1]

(b) Talon 3B.[2]

(c) Da Vinci Surgical System.[3]

Figure 1.1: Examples of commercial robots for different task: (a) vaccuming, (b) demining and (c) surgery.

In many situations, a single robot is not capable of carrying out a task on its own. In other scenarios, a cooperative work between several robots produces a better result. In these cases, multi-robot teams become much more efficient and reliable and can complete the tasks

[1]Image obtained from thesinsamode.com/2017/07/bosch-presents-first-robot-vacuum-cleaner/.

[2]Image obtained from army-technology.com/features/featurewinning-the-war-on-landmine-casualties-4569412/.

[3]Image obtained from radianceiitb.org/blog/da-vinci-robot/.

in less time [1]. On the downside, they require more sophisticated control strategies and pose new challenges for researchers and developers.

To perform a task as a team, all the members require a common knowledge or, at least, enough information to play their part. This can be achieved either through a centralized scheme or a distributed one [2]. In the first option, a central node gathers the information coming from all the robots, makes the decisions and assigns to each member of the team a simpler task. In the latter, each robot is capable of making its own decision based on its own information and the information received from its neighbors. Centralized approaches are easier to design and implement and can make better decisions based on global information. However, they are difficult to scale and become unsuitable when the number of robots is large. On the other hand, distributed approaches, that rely only on local information, are harder to devise but are generally lighter in terms of computation. The main advantage of these approaches is that they are robust to single agent failures, since no central node is required, and they can adapt to changes in the communication network. Although distributed approaches attract more attention than centralized ones, especially for applications which are inherently distributed in space, time or functionality, centralized approaches usually offer optimality guarantees in a more natural way. Depending on the application, the size of the team, the scalability or the communication and computational constraints one alternative may be preferable to the other.

The novel research challenges that multi-robot systems introduce include among others cooperative map building and localization [3], formation control [4], collaborative manipulation [5] and coverage problems [6]. This thesis focuses on this last kind of problems, in particular, in persistent coverage problems.

1.1 The Coverage Problem in Multi-Robot Systems

The coverage problem aims to cover a given environment with a team of mobile robots. This problem can be considered as a general framework that includes a great variety of problems such as locational optimization [7], wireless sensor networks [8], the art gallery problem [9], exploration [10], mapping or sweeping [11].

The coverage problem in itself appears in the literature in three different approaches depending on the coverage capabilities of the robots and on the coverage objective: *static*, *dynamic* and *persistent coverage*. In this thesis we focus on the third problem and propose different solutions to it. However, the three of them are closely related, in many cases solutions can be extended from one to the others with slight adaptations and most of the tools are useful for the three of them. For instance, this classification could be extended with a fourth category, *persistent static coverage*, that shares some properties of static and persistent coverage. In the following subsections we elaborate on each subproblem to give an overview of the approaches proposed in the literature and the common tools that are used in the end to solve the persistent coverage problem.

1.1.1 Static coverage

The most traditional approach to coverage is static coverage, typically known as the deployment problem. It seeks to determine the optimal static positions of a group of robots to cover an environment, optimizing a certain utility function. Most solutions to this problem are distributed and based on classic Lloyd algorithms with Voronoi partitions, although different approaches are also investigated [12, 13]. In [6], robots compute their own Voronoi regions and move towards its center of mass. They demonstrate that the final set of centroidal Voronoi configurations maximizes the probability of detecting an event that takes place in the environment. Other Voronoi-based strategies are presented with anisotropic sensors [14] or in non-convex environments [15] for finite-size, heterogeneous robots [16]. The same notion of optimal sensing configuration is also used in [17] with an adaptive control law that learns the density function through sensor measurements. In [18], the robots follow the gradient of the energy to deploy minimizing the energy expenditure. Different partitioning schemes have also been used as in [19], where a gradient-descent is applied with a power diagram.

Static coverage has also been addressed in discrete spaces represented as graphs, using Voronoi tesellations via pairwise optimal gossiping [20] or via asynchronous greedy updates [21] and, alternatively, in a game theoretic framework [22].

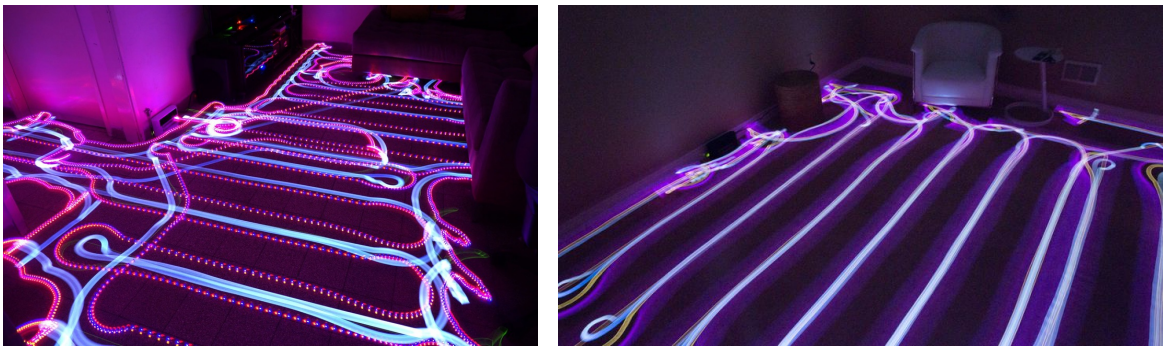
In this context of static coverage, some traditional problems have been revisited and extended taking advantage of mobile robots. In locational optimization [7, 23] the objective is to allocate resources in an area according to a fixed criterion, e.g., placing mail boxes in a city [24] to minimize distances from inhabitants. The main difference is that locational optimization solves off-line a limited problem while a robotic network relies on a generally distributed, on-line computation with reduced communication capabilities. Other revisited problems are coverage in wireless sensor networks [8], that is usually defined as a measure of how well and for how long the sensors are able to observe the physical space, and the art gallery problem [9], that aims to find how many (robotic) guards are necessary, and how many are sufficient to surveil the paintings and works in an art gallery with a finite number of walls.

Persistent static coverage

Static coverage may need to be adapted or repeated if the environment or the network changes, that is, the robots must change their positions to the new optimal ones when a change occurs. Therefore, static coverage can be extended to persistent static coverage in the sense that new sets of static positions have to be obtained over time. In [25] the robots deploy to cover an environment with a certain density function as in [6] and a single human is capable of changing this function to control the behavior of the team and get the robots relocated to optimize the coverage with the new density. In [26] a similar approach, that is aware of the energy remaining on each robot, is followed. When one of the robots leaves the coverage task for refueling, the others relocate to the optimum coverage and they do the same when the agent rejoins the coverage.

1.1.2 Dynamic coverage

The second approach is dynamic coverage, in which the robots must cover all the points of the environment at least once or until a desired coverage level is reached. After that, the task is considered complete. The goal of this approach is to minimize the time and energy consumption needed to complete the task. Some applications of this problem are exploration, mapping, search and rescue, demining, harvesting, car painting or snow removal. In Fig. 1.2 we show the paths followed by a single robot to complete the dynamic coverage of two different domestic environments. In this case, the task is to clean the floor and no repetition is required.



(a) (b)

Figure 1.2: Paths followed by a single robot to complete the coverage of two domestic environments.

Two different types of approaches to dynamic coverage appear in the literature. The first one consists in planning paths that allow the robots to cover the entire environment by following them. In [27] a review of path planning algorithms for complete coverage is presented and four categories are established based on the decomposition of the environment: heuristic, approximate, partial-approximate and exact cellular decompositions. In [28] a convex polygonal environment is covered following its boundary and its subsequent scaled inner polygons and an upper bound on the completion time is given. The authors of [29] propose the construction of spanning trees to create coverage paths that cover the whole terrain and minimize the time to complete coverage. A neural network is used in [30] to plan paths for nonholonomic robots that avoid collisions with both static and moving obstacles.

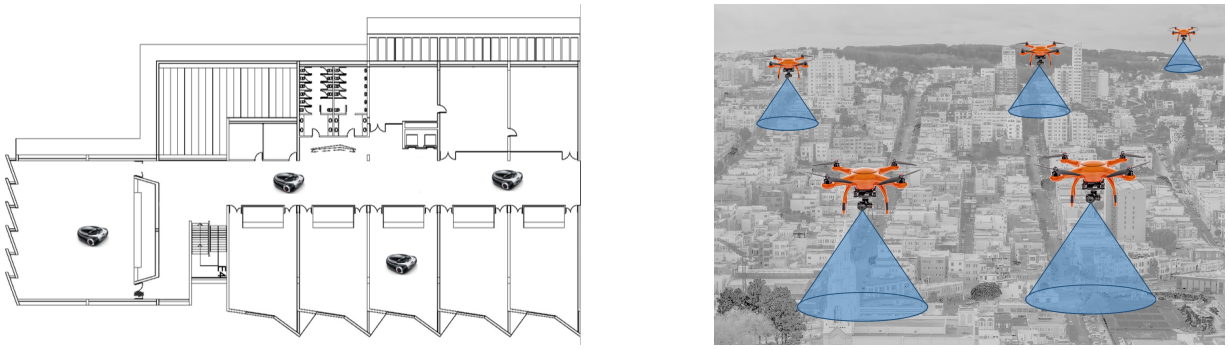
In the second type of solutions the key idea is to move the robots to the unexplored regions that provide more benefit. Hussein et al. [31] propose a control law based on the gradient of the coverage error that ensures that the error converges to zero for fully and partially connected networks. A similar approach is followed in [32] with anisotropic sensors and pairwise collision avoidance. In [33] a centralized coverage control law ensures the coverage task is done until the agents end up in local minima and then, a global trajectory tracking control law ensures that the agents are deployed to uncovered regions. In the work from Burgard et al. [34] a group of robots coordinately explore an unknown environment and try to minimize the overall exploration time by calculating an utility metric in the unexplored

areas and assigning targets to the robots. Over a receding horizon an integer programming planning is applied in [35] to maximize spatio-temporal coverage and comply with collision avoidance restrictions.

1.1.3 Persistent coverage

The last type of coverage problems is persistent coverage, in which this thesis focuses. The aim of persistent coverage is to maintain a desired coverage level over the environment with a group of mobile robots. Persistent coverage differs from static and dynamic coverage in that the coverage of the environment persistently decays, which requires the robots to continually move in order to maintain the desired level. Therefore, as opposed to dynamic coverage, in this case the task is never completed.

Many applications arise from this problem. Autonomous vacuum cleaners [36] or lawn mowers [11, 37], either alone or as members of a team, are ready to automate these tedious and time-consuming tasks (Fig. 1.3a). Cleaning a place where dust is continuously settling also includes oil spill confrontation [38]. Heating buildings, watering crops or monitoring the environment can also experience a considerable increase in efficiency if they are performed by autonomous robots [39]. In this sense, this problem is often addressed as *persistent surveillance* or *environmental monitoring* [40, 41]. In fact, the development of useful unmanned aerial vehicles (UAV) has encouraged the application of the coverage problem to surveillance and patrolling [42] (Fig. 1.3b). In addition, new applications in which the robots directly help people have arisen recently, for instance giving information in a touristic place [43] or edutainment in a pediatric hospital [44].



(a) Vacuuming of a building.

(b) Surveillance of a city.

Figure 1.3: Examples of persistent coverage applications with a team of mobile robots.

Three important aspects of persistent coverage formulations, which are actually greatly influenced by the application, define the settings of the problem that must be solved:

- **Coverage level and deterioration of the coverage:**

  From a practical point of view, the coverage level can be seen as the amount of dust while vacuuming, the accumulated heat or the temperature while heating a place, or

the time since a point was last visited while surveying an area. These magnitudes deteriorate over time due to physical phenomena as dust settling or cooling, or due to an increase of the observation uncertainty.

Different deterioration models are used in different approaches depending on the application. The most common formulation is a constant decay rate, that is, a deterioration proportional to the coverage level and, thus, exponential with time. This formulation is well suited for the settling of dust or the growth of lawn. Using this model, a saturation is applied to a maximum or a minimum level in some cases as in [45]. In surveillance applications, where the coverage level is usually equal to the time since the last visit to each point, its deterioration is linear with time. Apart from these two alternatives, other models can be used based on the uncertainty of measurements for monitoring applications or on heat transfer equations for heating applications. However, these models usually become complicated and do not provide additional advantages to the algorithms that solve the persistent coverage problem.

- **Importance function:**

  When covering an environment with a team of mobile robots, some points may be more important to cover than others. For instance, keeping a public building clean with mopping robots requires the entrance, the main corridors or the events room to be always cleaner than the basement and, therefore, they must be visited more frequently. This is expressed as a function of the environment that is called *importance function*. It has many interpretations such as the probability of an event taking place at a particular location as in [6] or a sensory function in [46].

- **Performance Metrics:**

  There are many variations in the way in which the persistent coverage objective is defined. In the broadest framework, the goal is to keep the coverage level as close as possible to a desired level over time and, as a consequence, a good metric can be the integral over time of the quadratic difference between the desired and the actual levels. The quadratic difference encodes that overcovering a point is as undesirable as undercovering it. For instance, it is necessary to square the difference in applications such as painting, since overcovering a point is a waste of resources, or heating, where a higher temperature than required is as bad a lower one. In the context of persistent surveillance [42] the objective usually is to minimize the maximum time that a point remains unvisited. Another alternative is to guarantee that every region is patrolled with a shorter period than the time needed by an intruder to cross a border.

Finally, one of the key and more challenging features of persistent coverage is that finding the optimal solution is NP-hard. Although there are many references in which this fact is proven [47, 48], it is fairly intuitive to see that it requires the solution of an instance of the Travelling Salesman Problem (TSP), which is a classical NP-hard problem. This property influences greatly the solutions proposed in the literature, that we analyze in detail in the following section.

1.2 State of the Art in Persistent Coverage

In this section we classify current state-of-the-art solutions to persistent coverage according to different criteria and pay special attention to the most important aspects of these solutions in the different categories that we provide.

1.2.1 Classification

A unique classification of solutions to persistent coverage is not possible due to the diversity of approaches to it. For this reason, we propose different categories to classify current solutions and to give the reader some understanding on how the scientific community is approaching the problem.

Computation

The first criterion, as for every multi-robot system, classifies the solutions to persistent coverage depending on how the information is exchanged and who makes the decisions:

- **Centralized solutions:** these approaches are usually needed to plan closed periodic paths for the robots and to be able to give some optimality measures [49].
- **Distributed solutions:** these are usually preferred for their adaptability, scalability, robustness and low communication bandwidth requirements [50] although in many cases it is difficult to guarantee optimality and the robots need a mechanism to retrieve or estimate the coverage of the entire environment

Environment

The second criterion that differentiates the proposed solutions is the nature of the environment at which persistent coverage is aimed:

- **Continuous environments:** these environments are usually a closed subset of $\mathbb{R}^2$, that is, they are considered planar even for applications with UAVs. We also include in this category those who, aiming to cover a continuous environment, discretize it in cells [51].
- **Discrete environments:** approaches in this category only require to maintain covered a finite set of interest points [52] although the robots can move in the continuous space.

Task Division

The third criterion distinguishes the different solutions depending on how the robots divide the coverage workload:

- **Partitioning:** the most basic approach to multi-robot coverage is to partition the environment in as many regions as robots and assign each one to a single robot, that

is responsible of covering it [53, 54]. Once the division has been made, each robot can act individually on its assigned region regardless of the other robots.

- **Cooperation:** although all the multi-robot solutions to persistent coverage are inherently cooperative, we use the term cooperation here to express that all the robots cover the entire environment together [33], i.e., they are not restricted to particular areas, as in the previous partition-based solutions.

Robot Motion

Eventually, in each of the aforementioned scenarios, several possibilities exist to define the motion of the robots:

- **Infinite horizon path planning:** the objective is to find, usually offline, one or more closed paths that the robots follow and periodically repeat to cover the entire environment [55] and maintain the coverage level. It is important to remark that these solutions are intrinsically periodic and the paths are required to be closed.

- **Finite horizon path planning:** at particular times the robots plan open paths from their current position. In addition to strictly receding horizon approaches [35], here we also include event-triggered planning to certain spatial goals [56].

- **Feedback control:** instead of a path planning algorithm, a motion controller is developed such that the robots decide at each time which is the best direction to move [57] according to the evolution of the coverage. It could be considered as a particular case of the previous one with a planning horizon equal to one. However, we separate because of the amount of solutions and techniques proposed for it.

In the following sections we provide detailed explanations on the important aspects of each category and reference the current solutions that can be included in them.

1.2.2 Centralized and Distributed Solutions

The differences between these two types of solutions are basically the same for persistent coverage as for any other multi-robot problem. Centralized solutions, which are usually used to plan periodic paths offline [58], often allow some kind of theoretical optimality guarantees at the expense of being computationally intensive and hardly scalable. When used as online strategies, they have the risk associated with a failure of the central node. On the contrary, distributed approaches are adaptable to failure of single agents, easily scalable and usually not computationally complex. However, it is generally more difficult to achieve global optimality with them. In between these two types, there are hybrid solutions such as [59] in which the robots make decisions distributedly but with some sort of coordination achieved through a central supervisor.

Coverage Estimation

A typical assumption in the approaches to the coverage problem is the exact knowledge of the coverage levels of the whole environment, that permits the calculation of the motion of the robots. In centralized solutions, such global information is available at the central node since it can communicate with all the robots of the team. Nevertheless, in distributed solutions it has to be retrieved locally from the information that only the neighboring robots communicate. One alternative to collect this information is to communicate each production or measurement of every robot through the network labeled with a robot identifier and a time stamp. Routing and communication protocols [60] allow each robot to receive information and propagate it through the network. Then, each robot is capable of retrieving the actual coverage map using all the received information. However, this knowledge can be challenging when dealing with large environments, changing robotic networks or when the team of robots has limited communication capabilities. Therefore, distributed estimation strategies where the information is synthesized instead of accumulated become more appropriate for this problem. In terms of estimation, in [61] the environment is parameterized by a set of basis functions, which are estimated by using a continuous time PI-consensus algorithm [62]. Additionally, the uncertainty of the estimation is reduced moving the robots so as to maximize their sensory information. A distributed interpolation scheme is used in [63] with a Kalman-like formulation and compression of the data, since the load of information to be transmitted between vehicles increases as more measurements are taken.

1.2.3 Solutions for Continuous and Discrete Environments

In continuous environments, an infinite amount of points require coverage. In many cases, these environments are restricted to be convex in order to simplify the solution but, in general, they can contain obstacles and be non-convex [35]. Some works discretize them in cells [51] to propose solutions based on graphs or on dynamic programming but, since they are designed to cover continuous environments, they are still considered as approaches for them. The strategy to control the coverage of these environments it to develop velocity controllers to spend more time covering the points where the coverage is bad or changes quickly. In [45] the moving speed of the robots and their initial locations on the path are optimized while in [64] the speed of each robot along its path is controlled to prevent the coverage from growing unbounded at any location.

As approaches for discrete environments we consider those which aim to maintain covered a finite amount of discrete points of interest such as [65]. Most of these works do not consider the times required to complete the coverage task at each point or assume that they are known. The former is acceptable in monitoring or surveillance applications [42], since the information gathering can be considered instantaneous, but the same does not hold in problems such as heating or watering, where the coverage action of the robots requires some time. The problem of calculating those times that the robots have to spend at each point to satisfy the coverage objective has not been deeply studied in the literature. In [66] they translate the problem of finding the coverage times of a single robot into a discrete-time dynamic system whose underlying continuous dynamics converge to a periodic cycle with a fixed period. A market-

based approach is presented in [67], where the robots locally calculate the cost of covering each point of interest and bid for it. Without a desired coverage level to accomplish, in [65] a single agent obtains a reward depending on the time that it covers each point and a path is calculated to obtain the maximum reward with a limited amount of fuel.

1.2.4 Partition-based and Cooperative Solutions

Partition-based Solutions

There are many works that address the persistent coverage problem with a divide-and-conquer strategy, in the sense that they partition the environment and assign each partition to a robot that becomes responsible of developing the coverage on it.

The most well-known partition approach for coverage in multi-robot systems is the one from Cortés et al. using centroidal Voronoi tesellations [6]. Although different partition strategies for coverage had been previously proposed [68], the Voronoi partition has been the most used, specially for static coverage [20, 69, 70]. Voronoi partitions are purely geometric and perform quite well in convex environments or environments with few obstacles. However, in complex environments, such as office-like ones, they may result disconnected. This entails that a robot needs to cross the regions of others robots to reach a part of its own, as shown in Fig. 1.4. Therefore, the coverage share-out is no longer maintained and a collision avoidance policy is needed. A workaround proposed in [71] is to repartition the non-reachable areas with other robots.

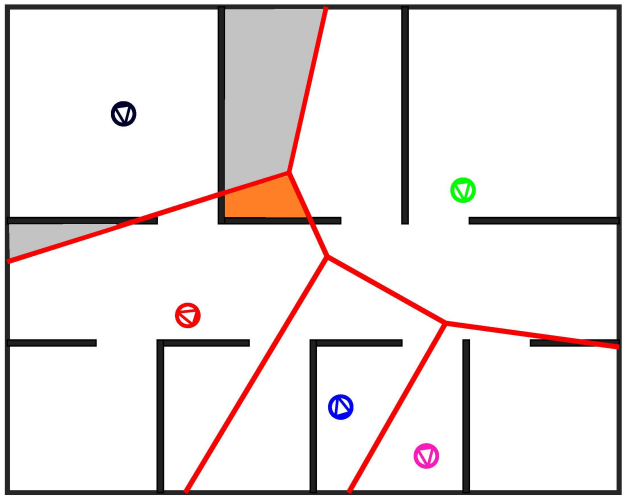


Figure 1.4: Voronoi partition with disconnected regions. The black robot has to cross the region of the red and green robots to reach the gray areas of its own region. The same happens to the red robot to reach the orange area of its own region.

In the case of persistent coverage, the most common approach is to initially find the partition and maintain it over time. This partition has to be equitable, i.e., each robot must be assigned to a region in which the workload is proportional to its capabilities with respect to rest of the team.

Equitable partitions for convex environments are usually calculated in a centralized fashion for teams of UAVs. In [72] a line in the optimal sweeping direction is moved along the environment to slice it each time the area on one of the sides is the same as any required partition. A polygon decomposition is applied in [53] with the restriction of including the robot position inside its partition and in [49] an algorithm handles sporadic communication between each robot and a central base to update the partition. In a distributed way, Pavone et al. [73] compute equitable partitions using power diagrams.

For ground vehicles, the vast majority of environments are non-convex and the assumption of convex environments is not reasonable. Therefore, different approaches have been proposed in these type of environments. A Voronoi-based load-balancing using the geodesic distance over a graph is presented in [74] and distributed vertex substitution is used in [21] with two-hop communication to guarantee convergence to the locally optimal configuration. A solution based on Lloyd's algorithm with power diagrams appears in [16] to deal with heterogeneous robots. Complete coverage and minimum coverage paths without any preparatory stage is guaranteed in [75] with an algorithm based on spanning trees. In [76] the feature of online obstacle and decay learning is added to a decentralized partitioning of the target area on the basis of the robot performances.

There is also an interesting open problem in which, instead of initially fixing static partitions, the partitions are updated over time to adapt to the progress of the coverage task or the energy reserves of the robots. In an application to persistent static coverage [26], a Voronoi partition is updated every time a robot needs to leave the environment for recharging.

Cooperative Solutions

As opposed to partition-based solutions, we refer as cooperative solutions to those in which the robots share the entire environment and every one is allowed to move to every point. Cooperation appears in the sense that the robots have to reach an agreement on where each individual goes to provide the optimal coverage.

In this context, some works plan a single closed path that covers the entire environment and is followed by all the robots of the team [77]. Then, the robots are distributed along the path to minimize the time between consecutive visit to every point [45]. An alternative is to plan a single path for each robot with the objective of covering the entire environment as in [48]. Also a single path for each robot can be planned but allowing them to overlap at the same points to ensure the optimal coverage of the regions of interest [46]. The last option presented in the literature is to perform a team-optimal assignment of spatial goals that the robots follow over the entire environment [57].

1.2.5 Path Planning and Feedback Control Solutions

The positions of the robots at every time instant are one of the two principal outcomes of any persistent coverage algorithm. These positions can be obtained in two ways: by controlling the motion direction online or by planning the paths of the robots and following them. Many path planning methods have been proposed in the literature and two interesting reviews of methods for coverage applications are [27] and [78]. Regarding persistent coverage,

two different options appear at the time of planning the robots' trajectories: planning periodic closed paths over an infinite time horizon or planning open paths over a finite horizon. Most solutions to persistent coverage have focused on the first planning alternative but in the following we review the approaches with the three strategies. Since in all cases they have to be aware of collisions, energy depletion and time constraints, we also include a subsection discussing them.

Infinite Horizon Path Planning

The first and most commonly selected option to obtain the positions of the robots consist in planning a priori closed paths that cover the entire environment and the robots periodically follow. The objective is to minimize some metric, preferably related to coverage, while periodically visiting all the points of the environment.

Some approaches obtain lawn-mower-like trajectories by finding straight paths and minimizing the number of turns. In [79] they compute a tour of minimum cost in the convex decomposition of a non-convex environment. A Zamboni pattern in the optimal sweep direction is modified in [72] to minimize turns and a sweeping pattern in such direction is followed in [53] for convex regions. A slightly different solution is proposed by Hokayem et al. [77] over-approximating the environment by a convex polygon and creating a spiral path with said polygon and its inner level sets. Using the same paths, [45] guarantees that full awareness is eventually reached in some interest points. These solutions are sufficient to provide the optimal coverage to environments where the decay and the importance are uniform.

Spanning tree coverage (STC) is also used for closed path planning under the assumption that the operation area does not get more narrow than two times the size of the robot. In [51] the minimum frequency of visits is obtained following a single STC path with all the robots in the team. A polynomial-time coverage algorithm is designed in [48] based on the single-agent STC introduced in [80]. In [29] they tackle the problem of constructing a STC that minimizes the time to complete coverage. In [81] a STC with backtracking is proposed and STC paths are found for each robot in regions of equal area in [75].

Apart from the previous ones, many other methods have been proposed to plan paths over an infinite time horizon. The authors in [82] and [83] aim to minimize the time between visits regions of interest that are represented by the vertices of a graph. In [84], Lin et al. formulate a parametric optimization to determine a set of elliptical trajectories for the robots, extending their previous work [58] to a two-dimensional space. Rapidly-exploring Random Cycles are used in [85] to sense a dynamic gaussian random field of the environment that evolves over time. The trajectories aim to minimize the covariance of the estimation of such field. In [86] a utility metric is maximized through a quadratic mixed integer program (MIP) to find the best tours for the agents when correlation between the points of interest is considered. A linear MIP is solved in [87] to obtain the optimal routes for a fleet of vehicles to periodically visit a set of targets while minimizing the energy consumption. Soltero et al. develop in [46] an online adaptive path planning algorithm based on gradient descent of a Voronoi-based cost function.

The vast majority of these solution do not allow that two robots cover the same area

at different points of their paths. These non-redundant paths have a worst-case coverage equal to the single robot case. On the other hand, the worst-case performance of redundant algorithms is a half of that time as shown in [81]. The advantages of these paths is that they are much easier to find than redundant ones. With respect to other solutions, they only have to be computed once and the robots just follow them. They also offer optimality guarantees for static and uniform environments. On the downside, they are limited in changing environments. If an obstacle blocks a single path or an agent fails, a replanning of the whole team is needed, typically resulting in a high computational cost and a centralized solution. They are also limited in environments where the decay or the importance is not constant for all the points. In those cases, it is not possible to reach the optimal coverage since all the points have the same visitation frequency regardless of their importance or coverage level.

Finite Horizon Path Planning

The second option to obtain the desired positions for the robots consists in planning online open paths that optimize the coverage along them over a finite time horizon. They lead the robots through the worst covered areas at that time or, at least, to the worst covered areas. This alternative has the potential to practically overcome all the limitations of closed paths at the only expense of changing global optimality guarantees for local ones. This is most likely the reason why this approach has received little attention in the context of persistent coverage. Nevertheless, many approaches to static coverage apply this tool. An integer program applied over a receding planning horizon is solved in [35] to find the paths that maximize spatio-temporal coverage. In [88] they solve approximately a dynamic program over a finite time horizon, which optimization criterion is the minimum uncertainty of a field estimate. The local path planning algorithm TangentBug is used in [56] to plan the trajectories to the Voronoi centroids in a non-convex environment.

Feedback Control

In order to obtain the positions of the robots over time an alternative to path planning is to use a controller that drives the robots according to the coverage level of the environment. This alternative allows to find the locally optimal directions of motion and can be easily distributed and scaled. On the downside, it does not guarantee global optimality and suffers from local minima. In [57] a gradient descent method is used to locally optimize the coverage and a centralized assignment of objectives leads the robots to undercovered areas and avoids local minima. In [89] a control policy decides to which cell move next to minimize the maximum time between visits to all cells. Hubel et al. [90] propose a control law based on the gradient of a cost function that penalizes the lack of coverage weighted according to the importance of each point. These methods behave well in the long term but may overcover areas locally and waste some time and energy.

It is important to differentiate this type of control from the low level control of the robots, that in the case of path planning is used to follow the paths. With the term feedback control, we refer to the higher level that decides the best direction of movement using the coverage of the environment as feedback.

Collision avoidance and other constraints

Each of the three previous methods to calculate or control the positions of the robots must be aware of the constraints imposed by the environment, the network and the application. These constraints include collisions with obstacles and other robots, connectivity, energy depletion or time requirements. Some of them have been addressed directly in persistent coverage; some others, more broadly in the context of general coverage problems; and others still represent open problems.

Most path planners provide closed trajectories that are collision-free with respect to static obstacles [82]. These paths are followed using low-level controllers that, at the same time, avoid collisions with other robots and moving obstacles while trying to follow the paths as close as possible [91]. A Dynamic Window Approach (DWA) finds at each time the optimal velocity in the space of discretized feasible velocities [92]. The resulting paths are safe but may no longer be the best possible for the same planning method. The problem of collision avoidance has also been considered in the planning of open paths [93]. It has been addressed by introducing time delays at the beginning of the paths [94] or solving a Mixed-Integer Linear Program (MILP) [95]. In [96] they also assign goals to interchangeable agents to minimize the maximum cost over all trajectories. In [97] robots decide the optimal assignment of objectives and activate a family of Lyapunov functions to guarantee safety. In [35] they compute trajectories that maximize spatio-temporal coverage while satisfying collision avoidance constraints. Motion controllers can also include terms to avoid collisions, as in [98], where a bounded potential repulsion is used.

Energy and time constraints such as autonomy and refueling [99] are also important in multi-robot persistent coverage. To find a closed path to visit all the points of the environment with a single UAV, heuristic solutions are presented with refueling depots [100] and with revisit constraints [52]. In [101] a MILP and a genetic algorithm are compared for the scheduling of mission trajectories with automated refueling stations. Another MILP is solved in an application to persistent people tracking [102] with automated logistics support. In [87], they seek the optimal routing strategy with a MILP that allows a fleet of vehicles to periodically visit a set of targets while trying to minimize the energy consumption. Similarly, in [103] they maximize the frequency of task completion allowing online calculation of task costs to avoid energy depletion. In [104] they schedule and plan the paths for a second team of recharging robots using both a MILP and a transformation to the TSP. A feedback controller for multiple quadrotors with charging constraints is generated in [105] to meet a complex temporal logic specification.

An additional problem that constrains persistent coverage is connectivity maintenance. It has not been addressed specifically but applicable solutions have been proposed [106, 107] and it has been deeply studied for multi-robot systems in general [108, 109].

1.2.6 Coverage Control

The coverage actions of the robots are the second principal outcome of any persistent coverage algorithm. The energy expended by these coverage actions performed by the robots in the environment is a common aspect that all the possible solutions introduced before share

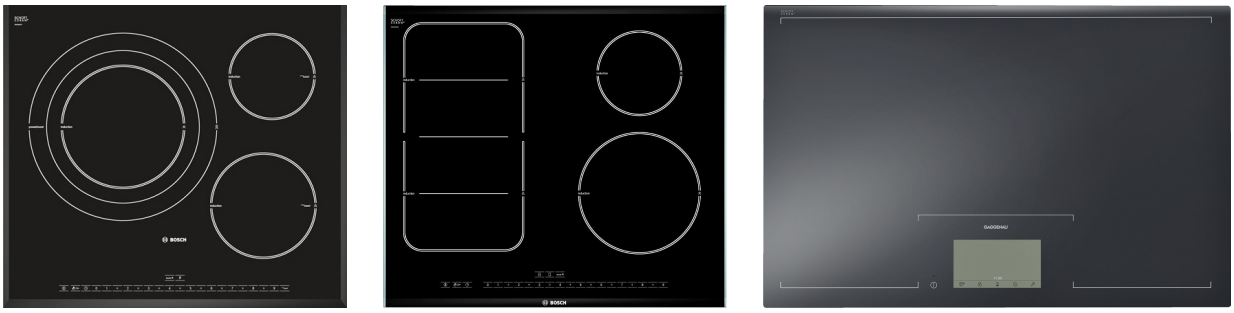
and that has received very little attention in the literature.

In applications such as surveillance or sensing, the range of the coverage actuators can be modified to save energy and improve the system efficiency. This issue has been widely studied in the context of wireless sensor networks for static coverage, where the range of the sensor is adapted to save energy while they are placed strategically in the environment optimizing the coverage [110, 111].

In applications in which the robot actions have an effect on the environment, the coverage action can be controlled and adapted to the needs of the environment at each time and point. The idea is to provide more coverage in the undercovered areas and less, or even nothing, in the already covered areas. This is essential in applications such as heating or watering, to avoid over-heating or flooding, respectively, or to save power or resources in the case of vacuuming, cleaning or painting. Only one approach to persistent coverage takes this into account, [59], where the coverage action and the coverage radius are controlled to reduce the consumption of energy and improve the coverage performance.

1.3 Mobile Induction Heating and Persistent Coverage

In recent years, there has been an increasing tendency in research and innovation to increase convenience and quality of life of people and to alleviate the workload, making everyday tasks easier. Induction technology for domestic cooking [112] is experiencing lately an important turn towards flexibility, to transform everyday cooking to an amusing and entertaining experience in itself and, at the same time, compatible with other tasks. Such transformation started with induction hobs that provide the user with increasingly more freedom to place any kind of pot in any place of the hob to cook. In the first place, inductors with concentric rings allowed different diameters of pots in the same place (Fig. 1.5a). The next step was to provide rectangular areas where different shapes and sizes were allowed (Fig. 1.5b) and the last one, that is already in the market, is a totally flexible hob in which any pot can be placed anywhere (Fig. 1.5c).



(a) Tradditional cooktop with concentric rings.

(b) Flexible induction.

(c) Totally active induction cooktop.

Figure 1.5: Evolution of commercial induction cooktops in the search for flexibility.

In the near future, the evolution tends to increasingly bigger hobs that do not sacrifice

space in the countertop by being integrated under a unique countertop surface. This surface is intended to cook, to prepare the food that has been or will be cooked, to have lunch or even to carry out other tasks like children doing homework while their parents are cooking. It can even be placed in the living room table as illustrated in Fig. 1.6. In this context, induction heating with mobile inductors appears as an appropriate solution to the challenge. The key idea is that the inductors are attached to robotic arms to be able to move under the countertop, instead of being fixed in a particular position. In this configuration, they can automatically position themselves under the pots that the user wants to be heated. Additionally, the ability to move allows the inductors to heat pots of different sizes and shapes, located in very different positions, or even to maintain more than one pot heated at the same time by repeatedly moving from one to another. This problem is an specific application of the persistent coverage problem with some particularities.

Figure 1.6: Illustration of the possibilities of induction in the near future. The living room table includes an induction hob and makes cooking an amusing family experience.

1.3.1 State of the Art in Mobile Induction

This work arises from a long-term research that BSH Electrodomésticos España is carrying out in collaboration with the University of Zaragoza. Some important results have been obtained in this research, specially in the design and construction of robotic arms capable of holding and moving the inductors under the countertop.

One of the most important results is the construction of a prototype of a domestic hob with three mobile inductors [113], that is shown in Fig. 1.7. The size of the hob is 726×482 mm. The inductors have a double coil that allows the activation of an inner circle of 12 cm of diameter or of an outer ring of 18 cm of diameter. Each inductor is attached to a robotic arm with two parallel rotational joints, each one actuated by a stepper motor, and the arms are clamped to the inner chassis of the hob. The movements of each inductor are controlled

by an Arduino board, capable of receiving orders from a computer, planning paths in the arm coordinates and command the actions of the motors.

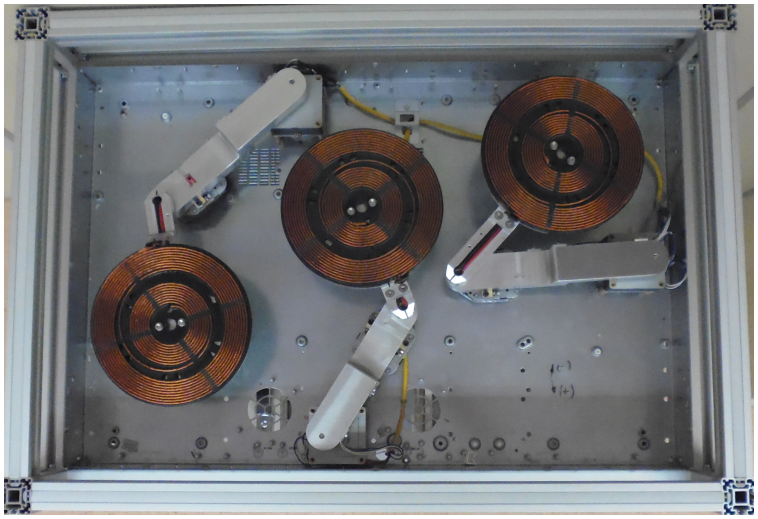


Figure 1.7: Top view of the prototype of mobile induction.

The decision on the diameters of the inductor coils was made thanks to the results in other research in this line [114]. It proves that an inductor of 18 cm of diameter can heat pots of up to 21 cm of diameter. In a similar way, an inductor of 12 cm can heat pots of up to 14 cm. These sizes fit with the size of the hob and with the vast majority of pots currently in the market.

On the other hand, since the positions, sizes and shapes of the pots are free, a pot detection system is needed to determine these properties and, consequently, locate the inductors under the pots. In this sense, two different systems have already been designed and tested. The first one [115] is based on computer vision techniques and uses a commercial color and depth camera situated above the hob. The second one [116] consists of a layer of capacitive sensors that can be incorporated under the countertop. Another issue that must be taken into account is that flexible cooktops should not have a fixed user interface since the main objective is to provide freedom to the user. To this end, a generic hand gesture recognition system was developed in [117].

All these innovations are important steps towards the final solution. However, the main remaining challenge is a strategy to plan the movements and the actions of the inductors which guarantees that the power requirements established by the user are satisfied. This strategy is the solution to a persistent coverage problem in a discrete environment shared between the robots where the probability of collisions is a decisive factor.

1.4 Objectives

In the context of the multi-robot persistent coverage problem, the first objective of this thesis is to improve existing methods and propose new algorithms to extend the state-of-

the-art solutions in continuous environments. This includes investigating the optimality and suitability of finite-horizon-planning and feedback-control solutions and applying them to increasingly more complex environments, from obstacle-free, convex ones to complex, non-convex ones, where the decay and the importance of the coverage are not uniform. An important focus of this first objective is on the applicability of these solutions to distributed systems, which have proven their advantages for this problem.

The second objective of this thesis is to develop an application of the persistent coverage problem to the open problem of domestic induction heating with mobile inductors. This requires a method for discrete environments, that is capable of calculating the coverage time of each robot at each point and that guarantee collision avoidance between robots in a constrained space.

1.5 Contributions

The contributions of this thesis focus on the two objectives: investigating solutions to the open problems in persistent coverage with special attention to the application to mobile induction heating. In particular, the first contribution is an optimal centralized solution based on finite horizon planning in a continuous environment. In the same type of scenario, we move to a distributed solution which requires a distributed estimation of the coverage. We propose several solutions in this context: the first one based on a feedback controller, the second one in finite horizon planning for environments with obstacles and the third one in an equitable partition of complex environments. After that, we present an optimal, centralized solution for general discrete environments that is particularized for the application of mobile induction in the final part.

Now we detail these main contributions and how they are structured in this dissertation:

- **A centralized, finite-horizon-path-planing solution for convex, continuous environments [118].** We consider a cost function that quantifies the quality of the coverage of a continuous environment in a finite prediction horizon. We transform the problem of planning the optimal open paths in such horizon into a discrete optimization problem with constraints that we solve with a novel algorithm based on Branch and Bound. This algorithm allows us to find the set of actions that minimizes the cost function and includes a procedure to split the sets of candidate solutions and the calculation of upper and lower bounds on the cost. Additionally, we propose a method to reduce the problem to several smaller subproblems. This method and its evaluation are presented in Chapter 2.

- **A distributed coverage estimation algorithm [119].** We propose an algorithm that allows every robot to estimate in a distributed manner the global coverage function using only local information, i.e., exchanging information only with neighbors. We pay special attention to the characterization of the algorithm, establishing bounds on the estimation error for every point of the environment. Additionally, we demonstrate that the algorithm guarantees a perfect estimation in a particular area for each robot.

The algorithm is described in Chapter 3 along with a test of its correctness and an evaluation of its performance.

- **A distributed, partition-based solution combining a locally optimal feedback controller with a selection of coverage goals [120].** We introduce a new function to determine the potential improvement of the coverage at each point of the environment. Upon this metric, we build a feedback control strategy that combines two actions. The first one maximizes the improvement. The second one drives the robots to the points of the highest improvement, which the robots are capable of finding distributedly, inside their partition. This contribution is introduced in Chapter 4.

- **A distributed, finite-horizon-optimal path planning solution with coverage control for continuous environments with obstacles [121].** We propose an algorithmic solution in which each robot locally finds the best paths to keep the desired coverage level over the whole environment. Using Fast Marching Methods, a set of optimal, finite-horizon paths are computed in terms of coverage quality, while keeping a safety distance to obstacles, and the one that mostly improves the coverage along the whole path is followed. We also propose a coverage action controller, locally computed and optimal, that makes the robots maintain the coverage level of the environment significantly close to the objective. This approach is presented in Chapter 5 with simulations and real experiments.

- **A distributed solution for complex, non-convex, continuous environments with equitable partitioning and graph-based, finite-horizon optimal paths [122].** The first part of our solution partitions the environment and assigns each partition to a single robot that becomes responsible of developing the coverage on it. It is based on the geodesic distance and divides general, non-convex environments in regions such that the work required to the robots inside them is equal. We also extend the algorithm to be aware of the energy, coverage capabilities or status of the robots and of the connectivity of the partitions. The second part of our solution is a planning method that allows each robot to find the optimal, finite-horizon path inside its partition in terms of the current coverage error. This path is planned through a graph that covers the entire partition in a sweep-like manner and also allows the robot to traverse the boundary of the partition. The entire solution is introduced in Chapter 6.

- **A centralized, periodic and cooperative strategy for discrete environments with optimal coverage times and actions [123,124].** In this approach, the robots have to visit periodically a discrete set of points of interest and spend some time on them carrying out the coverage task. We use a divide-and-conquer strategy and split the problem into three smaller subproblems to counteract its complexity. In the first place, we plan individual closed paths for the robots to cover all the points periodically. Secondly, we formulate a quadratically constrained linear program to find the optimal coverage times and actions that satisfy the coverage objective. Finally, we join together

the individual plans of the robots in a periodic team plan by obtaining a schedule with a MILP that guarantees collision avoidance. We introduce this strategy in Chapter 7.

- **An application to domestic mobile induction heating [124].** We particularize the previous solution to the application of domestic induction heating with mobile inductors. The problem is to persistently heat a finite set of pots with a team of mobile inductors. Since the motion of the inductors is very constrained due to the small size of the hob with respect to them, we develop a particular collision avoidance policy to guarantee the heating of all the pots. We show the performance of the entire solution with experiments on the real prototype of an induction hob with mobile inductors carried by robotic arms. This application is detailed in Chapter 8

In addition to the aforementioned contributions to the persistent coverage problem and its particular application to domestic heating, during this thesis many other contributions have been made in the evolution and modernization of home appliances. Specifically, these contributions are patents in human-machine interaction based on gestures [117,125–127] and in applying induction technology to domestic ovens [128].

All the contributions in this thesis have been published (or are in the process of evaluation) and have been presented and made available to the scientific community through several media.

Journal Publications

- José Manuel Palacios-Gasós, Eduardo Montijano, Carlos Sagüés and Sergio Llorente. "Distributed Coverage Estimation and Control for Multi-Robot Persistent Tasks". IEEE Transactions on Robotics, vol. 32, no. 6, pp. 1444-1460. December, 2016.

- José Manuel Palacios-Gasós, Eduardo Montijano, Carlos Sagüés and Sergio Llorente. "Multi-Agent Periodic Persistent Coverage with Collision Avoidance". IEEE Transactions on Control Systems Technology. 2017. In the second review round.

- José Manuel Palacios-Gasós, Eduardo Montijano and Carlos Sagüés. "Equitable persistent coverage of non-convex environments with graph-based optimal planning". In preparation.

- José Manuel Palacios-Gasós, Eduardo Montijano, Carlos Sagüés and Sergio Llorente. "Human-computer interaction based on hand gestures using RGB-D sensors". Sensors, vol. 13, no. 6, pp. 11842–11860. 2013.

Peer-Reviewed International Conferences

- José Manuel Palacios-Gasós, Eduardo Montijano, Carlos Sagüés and Sergio Llorente. "Distributed Coverage Estimation for Multi-Robot Persistent Tasks". European Control Conference, Linz, Austria. pp. 3681-3686. 2015.

- José Manuel Palacios-Gasós, Eduardo Montijano, Carlos Sagüés and Sergio Llorente. "Multi-Robot Persistent Coverage using Branch and Bound". American Control Conference, Boston, MA, USA. pp. 5697-5702. 2016.

- José Manuel Palacios-Gasós, Eduardo Montijano, Carlos Sagüés and Sergio Llorente. "Multi-Robot Persistent Coverage with Optimal Times". Conference on Decision and Control. Las Vegas, NV, USA. pp. 3511-3517. 2016.

- José Manuel Palacios-Gasós, Zeynab Talebpour, Eduardo Montijano, Carlos Sagüés and Alcherio Martinoli. "Optimal Path Planning and Coverage Control for Multi-Robot Persistent Coverage in Environments with Obstacles". IEEE International Conference on Robotics and Automation (ICRA). Singapore. pp. 1321-1327. 2017.

  Online video: https://www.youtube.com/watch?v=FjczAACcbaY.

Invited Talks

- José Manuel Palacios-Gasós. "Multi-Robot Persistent Coverage". Distributed Intelligent Systems and Algorithms Laboratory, EPFL, Laussane, Switzerland. April 2016.

- Eduardo Montijano Muñoz. "Research Overview in Perception, Planning and Cooperative Control of Multi-Robot Systems". Multi-Robot Systems Lab, Stanford University, San Francisco, CA, USA. December 2017.

- Eduardo Montijano Muñoz. "Research Overview in Perception, Planning and Cooperative Control of Multi-Robot Systems". Robotics and Perception Group, University of Zurich, Zurich, Switzerland. June 2017.

- Eduardo Montijano Muñoz. "Distributed Perception and Control of Multi-Robot Systems: Persistent Coverage". Department of Engineering, Roma Tre University, Roma, Italy. October 2017.

Patents

- Sergio Llorente, José Manuel Palacios and Carlos Sagüés. "Domestic appliance with gesture detection". Spanish Patent ES2564879 (B1). 2017.

- Alejandro Inogés, Sergio Llorente, Gonzalo López, Eduardo Montijano, José Manuel Palacios and Carlos Sagüés. "Domestic appliance". European Patent EP3184907 (A1). 2015.

- Alejandro Inogés, Sergio Llorente, Gonzalo López, Eduardo Montijano, José Manuel Palacios and Carlos Sagüés. "Domestic appliance comprising a gesture detection". European Patent EP3187785 (A1). 2015.

- Juan José Galindo, Sergio Llorente, Carlos Obón, José Manuel Palacios, Enrique J. Pérez, Edgar J. Ramírez, Fernando Sanz and Beatriz Villanueva. "Sistema de aparato de cocción". Spanish Patent ES2603780 (A1). 2015.

Workshops

- José Manuel Palacios-Gasós, Eduardo Montijano and Carlos Sagüés. "Goal-Based Distributed Approaches to Persistent Coverage". Workshop on Multi-Robot Perception-Driven Control and Planning, International Conference on Robotics and Automation, Singapore. June 3, 2017.

Chapter 2

Centralized, Finite-Horizon Path Planing for Convex, Continuous Environments

In this chapter we present a centralized solution to persistently cover a continuous environment. We consider a cost function that quantifies the quality of the coverage in a finite prediction horizon and transform the path planning problem into a discrete optimization problem with constraints. We introduce a novel branch-and-bound algorithm that allows us to find the optimal solution to the problem, i.e., the set of paths that minimizes the cost function. This algorithm includes a procedure to split the sets of candidate solutions and the calculation of upper and lower bounds on the cost. Additionally, we propose a method to reduce the problem to several smaller subproblems and alleviate its computational complexity. Finally, we carry out simulations to evaluate the performance of the system.

2.1 Introduction

In this chapter, we begin our proposal of new solutions to the persistent coverage problem starting from the most basic and approachable settings: a convex, continuous environment and a centralized system. Many solutions that plan infinite-horizon, closed paths for the robots have been proposed for persistent coverage. They are not optimal for environments with variable decay or importance if no redundancy is allowed. For this reason, we propose a solution to the problem with finite-horizon, open paths, that can adapt to the non-uniformity of the environment. In addition, a centralized system offers the guarantee optimality of such solution thanks to the global information that is gathered in the central node.

The first contribution of this chapter is a new approach to the problem based on planning open paths that are optimal in terms of coverage of the whole team in a finite prediction horizon. To this end, we propose a cost function that quantifies the quality of the coverage performed by the team given a set of inputs. The set of paths that minimizes such cost function is the one that gives the best coverage and, therefore, the cost function leads to an optimization problem that is expressed in discrete time and continuous space and

actions, that define the desired paths. By discretizing both the actions and the environment we transform the problem into a discrete optimization problem with restrictions, that is amenable to being solved using dynamic programming techniques [129].

Different well-known dynamic programming techniques, such as Branch and Bound [130], are capable of solving general shortest path problems. Branch-and-bound algorithms, which have successfully been used in formation control [95] and trajectory planning with collision avoidance [93], seek the optimal solution performing a systematic evaluation of increasingly smaller subsets of the solution set. It can be interpreted as a rooted tree of feasible solutions with the full set at the root and with each branch representing a subset of candidate solutions. The key idea of branch and bound is to calculate bounds on the attainable cost of each brand to eliminate it from further consideration. In order to build a branch-and-bound algorithm two procedures are needed: one to split a given set of candidate solution into smaller subsets and one to decide if a branch has to be pruned. The efficacy of the second procedure is key to the computational cost since it is strongly dependent on the number of explored nodes. If a branch is pruned, there is no need to explore all the nodes below it. Therefore, the nearer the branch is to the top of the tree, the less nodes explored and the lower computational cost.

In this context, the second contribution of this chapter is an algorithm based on branch and bound, that allows us to find the optimal solution to the optimization problem. To achieve this, we propose one method to split the branches of the tree of candidate solutions and another one to eliminate branches from further consideration. For the second one we introduce and prove an upper and a lower bound of the attainable cost of a branch. Additionally, we propose a method to reduce the problem to several smaller subproblems in order to reduce the computational cost while still finding the optimal solution.

The remainder of the chapter is structured as follows: Section 2.2 formulates this version of the persistent coverage problem. In Section 2.3 we introduce the finite-state version of the problem. The branch and bound solution is proposed in Section 2.4. In Section 2.5 we present an alternative to separate the problem into smaller subproblems. The performance of the algorithm is analyzed in simulations in Section 2.6. Finally, Section 2.7 gathers the conclusions of this work.

2.2 Problem Formulation

Let $\mathbf{Q} \subset \mathbb{R}^2$ be a known bounded environment which a team of $N \in \mathbb{N}$ mobile robots has to persistently cover and that is known by the robots. We assume that they are holonomic and, due to the discrete-time communications of the multi-robot team, their motion in discrete time is expressed as

$$\mathbf{p}_i(k) = \mathbf{p}_i(k-1) + \mathbf{u}_i(k-1), \tag{2.1}$$

where $\mathbf{p}_i(k) \in \mathbf{Q}$ is the position of robot i at time $k \geq 1$ with $i \in \{1, \ldots, N\}$ and $\mathbf{u}_i(k-1)$ is the motion control action. The maximum distance that a robot can move in one step is $u_i^{\max}$, i.e., $\|\mathbf{u}_i(k)\| \leq u_i^{\max}, \forall\, i \in N$. We also consider that there is a low-level motion controller that calculates the forces or torques of the specific robot and guarantees that it reaches the

position $\mathbf{p}_i(k)$ at time k and that the robot has an accurate measurement of its own position at every time.

The coverage of the environment is modeled with a time-varying field, $Z(\mathbf{q}, k)$, which we call *coverage function* or *coverage map* indistinctly. Recall that this field may represent a physical quantity, such as an accumulation of heat or a height of grass. The aim of the robotic team is to maintain a desired coverage level, $Z^*(\mathbf{q}) > 0, \forall\, \mathbf{q} \in \mathbf{Q}$, that we alternatively refer to as *objective map*. Each point has a different importance $\phi(\mathbf{q})$ or, equivalently, provides a different reward when being covered.

To reach this aim, the robots are able to generate an increase on the coverage level at each time instant. Depending on its position, $\mathbf{p}_i(k)$, each robot i produces an increment $\alpha_i(\mathbf{q}, \mathbf{p}_i(k)) \geq 0$ on the points within the coverage radius r_{cov} around $\mathbf{p}_i(k)$. This production can also be adjusted by a gain $0 \leq \rho_i(k) \leq \rho_i^{\max}$, that is the coverage action. The purpose of this gain is to keep the variable production level independent from the fixed shape of the production.

On the other hand, as time goes by, the coverage level of a given point deteriorates. This decrement can be modeled according to a time-independent decay gain $d(\mathbf{q})$, with $0 < d(\mathbf{q}) < 1$. It is important to note that the decay rate can be different from one point to another, although some works consider it constant over the environment. Therefore, the evolution model of the coverage level is represented by the following recurrence equation,

$$Z(\mathbf{q}, k) = d(\mathbf{q}) Z(\mathbf{q}, k-1) + \sum_{i=1}^{N} \rho_i(k-1) \alpha_i(\mathbf{q}, \mathbf{p}_i(k)). \tag{2.2}$$

This exponential deterioration of the coverage is suitable for applications such as heating a place or watering crops. It differs from alternative formulations as in [64], where the coverage level increases over time and is decreased by the robots.

For the sake of clarity, in the rest of the chapter we omit the dependency of $\mathbf{q}$ unless it is strictly necessary, i.e., $Z(\mathbf{q}, k) \equiv Z(k)$ or $Z^*(\mathbf{q}) \equiv Z^*$. The objective of any control policy that aims to make $Z(k)$ equal to Z^* is to find the inputs of the robots that produce the best coverage of the environment. Although this can be done considering one time step at a time, a better solution is to find the current actions while planning multiple steps ahead, i..e, planning a longer open path. We let $T > 0$ be the horizon where we want to design our control actions and, for any k,

$$\mathbf{u}_k = \begin{bmatrix} \mathbf{u}_{1,k} & \dots & \mathbf{u}_{1,k+t} & \dots & \mathbf{u}_{1,k+T-1} \\ \dots & \dots & \dots & \dots & \dots \\ \mathbf{u}_{N,k} & \dots & \mathbf{u}_{N,k+t} & \dots & \mathbf{u}_{N,k+T-1} \end{bmatrix},$$

$$\boldsymbol{\rho}_k = \begin{bmatrix} \rho_{1,k} & \dots & \rho_{1,k+t} & \dots & \rho_{1,k+T-1} \\ \dots & \dots & \dots & \dots & \dots \\ \rho_{N,k} & \dots & \rho_{N,k+t} & \dots & \rho_{N,k+T-1} \end{bmatrix},$$

the associated control inputs, where the temporal dependency is written as a sub-index for clarity. Note that these actions represent in a discretized way the optimal paths that the robots have to follow.

To compute these inputs, we define a cost function as the sum of the quadratic coverage errors in $\mathbf{Q}$ along T instants,

$$f\left(k, \mathbf{u}_k, \boldsymbol{\rho}_k\right) = \sum_{t=1}^{T} \int_{\mathbf{Q}} \phi \cdot \left(Z^* - Z(k+t)\right)^2 \mathrm{d}\mathbf{q}.$$

Note that it penalizes under-covered points as well as over-covered ones. This also gives more importance to covering the lowest covered points rather than the ones with a coverage level near the objective. Therefore, the persistent coverage problem is presented in this case as a constrained non-linear optimization problem,

$$\begin{aligned} \underset{\mathbf{u}_k, \boldsymbol{\rho}_k}{\text{minimize}} \quad & f\left(k, \mathbf{u}_k, \boldsymbol{\rho}_k\right) \\ \text{subject to} \quad & \|\mathbf{u}_i(k+t-1)\| \leq u_i^{\max}, \\ & 0 \leq \rho_i(k+t-1) \leq \rho_i^{\max}, \\ & \mathbf{p}_i(k+t) \in \mathbf{Q}, \\ & \|\mathbf{p}_i(k+t) - \mathbf{p}_j(k+t)\| \geq \delta, \\ & \qquad \forall\, i, j \in \{1, \ldots, N\} \text{ with } i \neq j, \\ & \qquad \forall\, t \in \{1, \ldots, T\}, \end{aligned} \tag{2.3}$$

where δ is a safety distance to avoid collisions between pairs of robots. In the case of (possibly moving) obstacles, that can be detected with sensors on board the robots, the restriction of the positions inside the environment in (2.3) becomes $\mathbf{p}_i(k+t) \in \mathbf{Q} \setminus \mathbf{Q}_{obs}$ where $\mathbf{Q}_{obs}$ are the points of the environment $\mathbf{Q}$ that are occupied by such obstacles.

2.3 Finite-State Version of the Problem

Since finding the optimal solution of (2.3) is very complex, we consider instead a discretized version of the whole setup. For all k we impose that

$$\begin{aligned} \mathbf{q}, \mathbf{p}_i(k) &\in \mathbf{Q}_d \subseteq \mathbf{Q}, \\ \mathbf{u}_i(k) &\in \mathbf{U}_{d,i}, \\ \rho_i(k) &\in P_{d,i}, \end{aligned}$$

where $\mathbf{Q}_d$ is a discretization of the environment and $\mathbf{U}_{d,i}$ and $P_{d,i}$ are finite sets of action values, such that

$$\begin{aligned} 0 \leq \max_{\mathbf{U}_{d,i}} \|\mathbf{u}_i(k)\| &\leq u_i^{\max}, \\ \mathbf{p}_i(k) + \mathbf{u}_i(k) &\in \mathbf{Q}_d, \\ 0 \leq \max_{\mathbf{P}_{d,i}} \rho_i(k) &\leq \rho_i^{\max}. \end{aligned}$$

With this discretization we formulate the discrete version of the optimization problem,

$$\begin{aligned} \underset{\mathbf{u}_k, \boldsymbol{\rho}_k}{\text{minimize}} \quad & f\left(k, \mathbf{u}_k, \boldsymbol{\rho}_k\right) = \sum_{t=1}^{T} \sum_{\mathbf{Q_d}} \left(Z^* - Z(k+t)\right)^2 \\ \text{subject to} \quad & \mathbf{u}_i(k+t-1) \in \mathbf{U}_{d,i}, \\ & \rho_i(k+t-1) \in P_{d,i}, \\ & \|\mathbf{p}_i(k+t) - \mathbf{p}_j(k+t)\| \geq \delta, \\ & \quad \forall\, i, j \in \{1, \ldots, N\} \text{ with } i \neq j \\ & \quad \forall\, t \in \{1, \ldots, T\}. \end{aligned} \tag{2.4}$$

There are two main reasons that make it interesting to initially consider (2.4) instead of (2.3). The first reason is that for this problem it is possible to obtain the exact optimal solution, whereas for the continuous version, with the existing optimization methods we can only expect to obtain a local minimum. The second reason is that this solution can also be used in a second stage to generate an accurate initial seed for (2.3), guaranteeing that, even if we still reach a local minimum, it will be close to the global one.

2.4 Finite-Horizon Path Planing based on Branch and Bound

In this section we propose a branch-and-bound algorithm to find the optimal solution of the optimization problem (2.4). Branch-and-bound algorithms [130] seek the optimal solution performing a systematic evaluation of increasingly smaller subsets of the solution set. The solution set is interpreted as a rooted tree of feasible solutions with the full set at the root and with each branch representing a subset of candidate solutions, as shown in Fig. 2.1. To build a branch-and-bound algorithm, two procedures are needed: one to split a given set of candidate solutions into smaller subsets and another one to prune branches using bounds on the cost. The rest of the section is devoted to explain in detail these two procedures.

2.4.1 Splitting Procedure

Each node that is not a leaf in the tree has some values of the action vectors which are already fixed and some others still unfixed. Without loss of generality, we separate the action vector in these two components:

$$\begin{aligned} \mathbf{u}_k &= \left[\begin{array}{cc} \mathbf{u}_k^{fixed} & \mathbf{u}_k^{non-fixed} \end{array}\right] \equiv \left[\begin{array}{cc} \mathbf{u}_k^{f} & \mathbf{u}_k^{nf} \end{array}\right], \\ \boldsymbol{\rho}_k &= \left[\begin{array}{cc} \boldsymbol{\rho}_k^{fixed} & \boldsymbol{\rho}_k^{non-fixed} \end{array}\right] \equiv \left[\begin{array}{cc} \boldsymbol{\rho}_k^{f} & \boldsymbol{\rho}_k^{nf} \end{array}\right]. \end{aligned}$$

For a particular node, we take one of the actions in the non-fixed set and generate as many new nodes as feasible values for that action. A representation of this idea can be seen

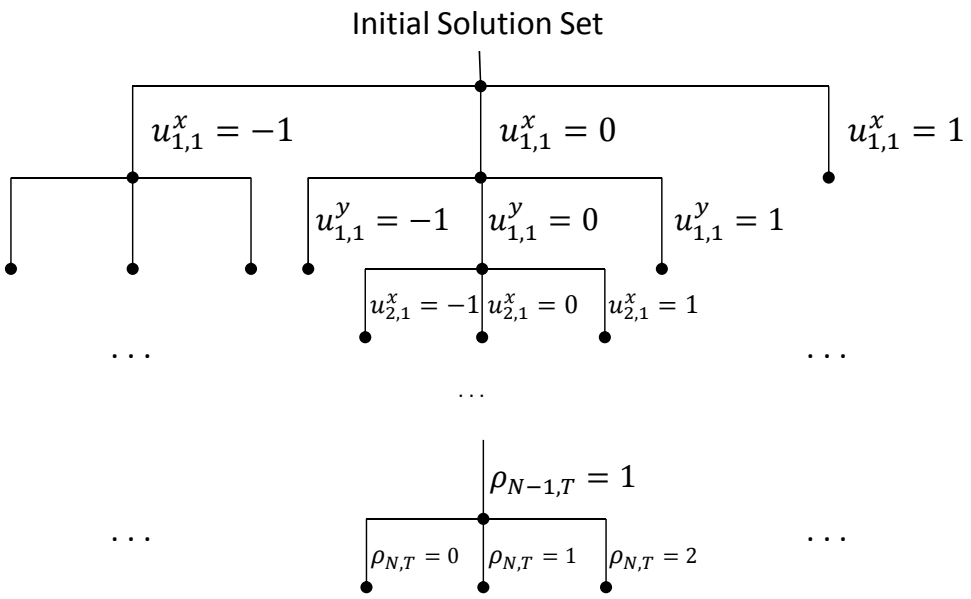


Figure 2.1: Representation of a branch-and-bound solution tree.

in Fig. 2.1. At each level, each node is divided by assigning all the possible values to the first action that has not been fixed yet. The order in which we select the value of such action is the following: first, the movements of the N robots for time k to time $k + T - 1$ and second, the productions of the N robots for each time instant. We choose this order since the movements in the first prediction instants are more likely to have considerable different costs and, thus, to allow the pruning of more branches. Note that any other order can be followed and that such order may influence the number of branches that are pruned.

2.4.2 Pruning Procedure

The second procedure aims to decide whether a branch can be pruned or needs further exploration. The efficacy of this procedure is key to the computational cost of the algorithm since the number of explored nodes strongly dependent on it. If a branch is pruned, there is no need to explore all the nodes below it. Therefore, the nearer the branch is to the top of the tree, the less nodes are explored.

The decision of pruning a branch or not is made by calculating an upper and a lower bound on the cost of this branch, comparing it with the best solution found so far. If the lower bound of a branch is higher than the current best upper bound, this branch can be pruned since it will not give a better solution than the stored one. Otherwise, the node is stored in a list to be explored. Additionally, if its upper bound is lower than the current best, then the value of the current best upper bound is updated with this one.

Upper Bound

For a given node, an upper bound on the cost function can be calculated as the cost of any feasible solution on the set that is being evaluated. To rapidly find a feasible solution, we assign to all the non-fixed actions of the current node, $\mathbf{u}_k^{nf}$ and $\boldsymbol{\rho}_k^{nf}$, the first feasible value of their sets. By feasible values, we refer to those that belong to $\mathbf{U}_{d,i}$, do not make a robot leave $\mathbf{Q}_d$ and do not violate the constraints on the collisions between robots.

Lower Bound

The lower bound on the cost function that we propose is

$$\underline{f}(k) = \sum_{t=1}^{T} \sum_{\mathbf{Q_d}} \big(Z^* - \underline{Z}(k+t)\big)^2. \tag{2.5}$$

Before introducing $\underline{Z}(k)$ we give an interpretation of this lower bound. The cost in the lower bound produced by the actions that are already fixed, $\mathbf{u}^f$ and $\boldsymbol{\rho}^f$, coincides with the actual cost. In particular, $f(k) = \underline{f}(k)$ if $\mathbf{u}^{nf}$ and $\boldsymbol{\rho}^{nf}$ are empty. Let also $\mathbf{p}_i^f(k+t)$ denote the set of robot positions obtained by fixed actions, $\mathbf{u}^f(k+t-1)$, which will allow the formulation of $\underline{Z}(k)$ and $\underline{Z}'(k)$.

For the non-fixed actions we introduce a *desirable production* $\underline{\alpha}_i\big(\underline{Z}'(k+t), \mathbf{p}_i^r(k+t)\big)$ that represents the optimal distribution of the non-fixed production in the area that can be reached with the non-fixed movements. In other words, we aim to share out the productions that are not fixed yet between all the points that could be covered by the robots. To find these points, let $\mathbf{p}_i^r(k+t)$ be the set of reachable positions at time $k+t$,

$$\mathbf{p}_i^r(k+t) = \{\mathbf{p} \mid \mathbf{p} = \mathbf{p}_i(k) + \sum_{\ell=0}^{t-1} \mathbf{u}_i^f(k+\ell) + \sum_{\ell=0}^{t-1} \mathbf{u}_i^{nf}(k+\ell),\ \mathbf{u}_i^{nf}(k+\ell) \in \mathbf{U}_{d,i}\},$$

which includes the already fixed actions and all the feasible values of the non-fixed ones, considering domain boundaries and collisions as well.

Now we are in the position to formulate $\underline{Z}(k)$ from (2.5):

$$\underline{Z}(k+t) = \underline{Z}'(k+t) + \sum_{i=1}^{N} \underline{\alpha}_i\big(\underline{Z}'(k+t), \mathbf{p}_i^r(k+t)\big), \tag{2.6}$$

$$\underline{Z}'(k+t) = d\underline{Z}(k+t-1) + \sum_{i=1}^{N} \rho_i^f(k+t-1)\alpha_i\big(\mathbf{p}_i^f(k+t)\big).$$

The formal definition of the desirable production function requires a profuse intermediate formulation that needs to be presented in advance. First we formalize the reachable area and the production that has to be distributed.

From the set of reachable positions, $\mathbf{p}_i^r(k+t)$, we can compute the *reachable coverage set*, as the set of points $\mathbf{q} \in \mathbf{Q}_d$ in which robot i could apply a positive production,

$$\boldsymbol{\Omega}_i^r(k+t) = \{\mathbf{q} \mid \alpha_i(\mathbf{q}, \mathbf{p}) > 0,\ \mathbf{p} \in \mathbf{p}_i^r(k+t)\}. \tag{2.7}$$

Note that the points covered from positions that only involve fixed actions are not considered, since that production is already included in $\underline{Z}'(k+t)$.

Now we define the sets of overlapped reachable coverage sets by different robots,

$$\mathbf{\Omega}_m^r(k+t) = \bigcup_{i \in V_m(k+t)} \mathbf{\Omega}_i^r(k+t),$$

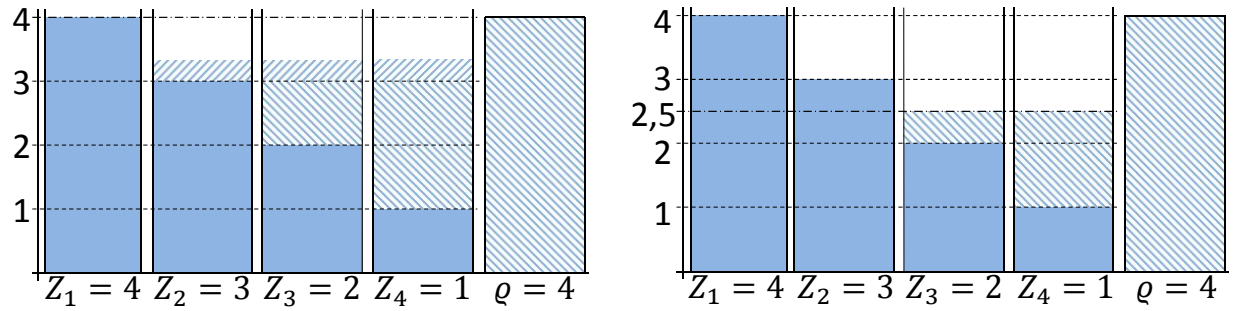
with $V_m(k+t)$ the subsets of robots that form a connected component in terms of overlapping.

The idea of the optimal production is to share out the maximum production of the overlapped robots fairly in their respective overlapped set, i.e., to share out

$$\varrho_m(k+t) = \sum\nolimits_{\mathbf{Q}_d} \sum\nolimits_{i \in V_m(k+t)} \rho_i^{max} \alpha_i(\mathbf{q}, \mathbf{p}) \tag{2.8}$$

among $\mathbf{\Omega}_m^r(k+t)$. We use $\alpha_i(\mathbf{q}, \mathbf{p})$ since the maximum coverage action that a robot can apply is constant.

The optimal way to perform such distribution in terms of the quadratic coverage error is to cover the lowest covered points until there is no more production to give or there are no points with a coverage value below the objective. An illustrative analogy is the problem of filling glasses with a bottle of water. The glasses can be seen as the overlapped points of a set of robots, $\mathbf{\Omega}_m^r(k+t)$, with the water level as the coverage level, $Z_1(k+t-1)$ to $Z_4(k+t-1)$ in Fig. 2.2. The amount of water in the bottle is the maximum production of this set of robots, $\varrho_m(k+t)$. The optimal water distribution is achieved in the following way: one starts filling the emptiest glass (glass 4) until it reaches the level of the second emptiest (glass 3 when $Z_4 = 2$). Then fills both fairly until they reach the level of the third glass with the lowest level (glass 2 in Fig. 2.2a) and so on until they are all full, the objective level is reached (Fig. 2.2b) or the bottle is empty (Fig. 2.2a).



(a) (b)

Figure 2.2: Example of the analogy of filling glasses with a bottle of water. a) $Z^* = 4$, $Z_m^+ = 3$ and $Z_m = 3.33$. b) $Z^* = 2.5$, $Z_m^+ = 3$ and $Z_m = 2.5$.

We now formulate the distribution and prove its optimality. We find the maximum coverage level in the overlapped set until which all the less-covered points can be covered:

$$Z_m^+(k+t) = \max_{\mathbf{q} \in \mathbf{\Omega}_m^r(k+t)} \underline{Z}'(k+t)$$

subject to

$$\sum\nolimits_{\mathbf{q}\in\mathbf{\Omega}_m^{Z^+}(k+t)} \big(Z_m^+(k+t) - \underline{Z}'(k+t)\big) \leq \varrho_m(k+t),$$

where

$$\mathbf{\Omega}_m^{Z^+}(k+t) = \big\{\mathbf{q} \in \mathbf{\Omega}_m^r(k+t) \,|\, \underline{Z}'(k+t) < Z_m^+(k+t)\big\}.$$

Note that $Z_m^+(k+t)$ does not depend on $\mathbf{q}$. In the example of Fig. 2.2a this quantity is $Z_m^+(k+t) = Z_2 = 3$ since, for $\varrho_m = 4$, we can fill Z_3 and Z_4, that are less covered than Z_2, until they reach this coverage value but we cannot make Z_2, Z_3 and Z_4 reach the value of Z_1. Additionally, to avoid over covering the environment and reach the optimum coverage, if $Z_m^+(k+t) > Z^*$, we only apply enough action in the points to reach the objective, i.e.,

$$Z_m(k+t) = \min\big(Z^*, Z_m^+(k+t)\big).$$

This is the case of Fig. 2.2b, where $Z_m^+(k+t)$ is still equal to 3 but $Z_m(k+t) = Z^* = 2.5$. After that, we calculate the total production required to make the points with coverage less or equal to $Z_m(k+t)$ reach this value,

$$\varrho_m^Z(k+t) = \sum\nolimits_{\mathbf{q}\in\mathbf{\Omega}_m^Z(k+t)} \big(Z_m - \underline{Z}'(k+t)\big),$$

with

$$\mathbf{\Omega}_m^Z(k+t) = \big\{\mathbf{q} \in \mathbf{\Omega}_m^r(k+t) \,|\, \underline{Z}'(k+t) \leq Z_m(k+t)\big\}.$$

In Fig. 2.2a, $\varrho_m^Z(k+t) = 3$, 1 unit for Z_3 and 2 units for Z_4, whereas in In Fig. 2.2b, $\varrho_m^Z(k+t) = 2$. Finally, the remaining production, $\varrho_m(k+t) - \varrho_m^Z(k+t)$, is equally divided among points with coverage value $Z_m(k+t)$. In Fig. 2.2a, where there is 1 unit left, it is divided between Z_2, Z_3 and Z_4.

Taking all of this into account, for any $\mathbf{q} \in \mathbf{\Omega}_m^Z$, the optimal coverage function introduced in Equation (2.6) is

$$\underline{\alpha}_i = Z_m - \underline{Z}' + \min\Big(\frac{\varrho_m - \varrho_m^Z}{|\mathbf{\Omega}_m^Z|}, Z^* - Z_m\Big), \tag{2.9}$$

where the temporal indices have been omitted for clarity. The first two terms in (2.9), make all the points in $\mathbf{\Omega}_m^Z$ reach Z_m and the second term is the one in charge of the equal distribution of the remaining production value. The function also considers the case where Z^* is reached, thanks to the minimum function, as in Fig. 2.2b. When this happens, $\sum_{\mathbf{Q}_d}\sum_{V_m} \underline{\alpha}_i \leq \varrho_m$. For the rest of the points in the environment, the coverage action is zero, $\underline{\alpha}_i = 0$. This means that the action does not increase the coverage of the points that are not reachable or points with a coverage level such that ϱ_m is not enough to bring the less-covered points to it, e.g., Z_1 in Fig 2.2. Eventually, this formulation allows us to state the optimality of the optimal production in the following theorem.

Theorem 2.4.1. *The optimal production from* (2.9) *represents the optimal way in which the total production can be distributed in terms of the quadratic coverage error, i.e.,*

$$\sum\nolimits_{\mathbf{Q}_d} \big(Z^* - \underline{Z}(k+1)\big)^2 \leq \sum\nolimits_{\mathbf{Q}_d} \big(Z^* - Z(k+1)\big)^2, \tag{2.10}$$

where $\underline{Z}(k+1)$ *is obtained from* (2.6) *and* $Z(k+1)$ *from* (2.2) *with* $\underline{Z}(k) = Z(k)$.

Proof. It is clear that

$$\sum_{\mathbf{q}\notin\mathbf{\Omega}_m^r(k+1)} \big(Z^* - \underline{Z}(k+1)\big)^2 = \sum_{\mathbf{q}\notin\mathbf{\Omega}_m^r(k+1)} \big(Z^* - Z(k+1)\big)^2$$

with $\alpha_i(k) = \underline{\alpha}_i(k) = 0$, $\forall\, \mathbf{q} \notin \mathbf{\Omega}_m^r(k+1)$, and $\underline{Z}(k) = Z(k)$.

Now we want to prove (2.10) in the rest of the points:

$$\sum_{\mathbf{\Omega}_m^r(k+1)} \big(Z^* - \underline{Z}(k+1)\big)^2 \leq \sum_{\mathbf{\Omega}_m^r(k+1)} \big(Z^* - Z(k+1)\big)^2. \tag{2.11}$$

The following proof for this equation is valid for each $m \in \{1, \ldots, M\}$ and, thus, for the sum of all of them since they are independent by definition. In the first place we develop the cost with the optimal production, left-hand side of (2.11). The definition of $\underline{\alpha}_i(k)$ in (2.9) guarantees that all the points in the set $\mathbf{\Omega}_m^Z(k+1)$ have the same $Z^* - \underline{Z}(k+1) = \underline{e}(k+1), \forall\, \mathbf{q} \in \mathbf{\Omega}_m^Z(k+1)$, where

$$\underline{e} = Z^* - \left(Z_m + \min\left(\frac{\varrho_m - \varrho_m^Z}{|\mathbf{\Omega}_m^Z|}, Z^* - Z_m\right)\right) \geq 0.$$

We omit the temporal dependencies when it is not necessary for clarity. Then, the left-hand side of (2.11) in this subset becomes

$$\sum\nolimits_{\mathbf{\Omega}_m^Z} \big(Z^* - \underline{Z}(k+1)\big)^2 = \sum\nolimits_{\mathbf{\Omega}_m^Z} \underline{e}^2 = |\mathbf{\Omega}_m^Z|\, \underline{e}^2. \tag{2.12}$$

Now we define the coverage error out of $\mathbf{\Omega}_m^Z$, i.e., in $\mathbf{\Omega}_m^{r\setminus Z} = \mathbf{\Omega}_m^r \setminus \mathbf{\Omega}_m^Z$. Since the coverage level in $\mathbf{\Omega}_m^{r\setminus Z}$ must be greater than in $\mathbf{\Omega}_m^Z$ by definition of $\mathbf{\Omega}_m^Z$, the coverage error must be lower and we can write it as the error in $\mathbf{\Omega}_m^Z$ minus a positive quantity, $\tilde{\underline{e}} \equiv \tilde{\underline{e}}(\mathbf{q}, k+1) \geq 0$ as

$$Z^* - \underline{Z}(k+1) = \underline{e} - \tilde{\underline{e}}(\mathbf{q}, k+1), \qquad \forall\, \mathbf{q} \in \mathbf{\Omega}_m^{r\setminus Z}.$$

Therefore, we have

$$\sum\nolimits_{\mathbf{\Omega}_m^{r\setminus Z}} \big(Z^* - \underline{Z}(k+1)\big)^2 = \sum\nolimits_{\mathbf{\Omega}_m^{r\setminus Z}} (\underline{e} - \tilde{\underline{e}})^2 = |\mathbf{\Omega}_m^{r\setminus Z}|\, \underline{e}^2 - 2\underline{e} \sum\nolimits_{\mathbf{\Omega}_m^{r\setminus Z}} \tilde{\underline{e}} + \sum\nolimits_{\mathbf{\Omega}_m^{r\setminus Z}} \tilde{\underline{e}}^2. \tag{2.13}$$

In the second place we develop the right-hand side of (2.11) with an arbitrary production function $\alpha_i(k+1)$, as a function of $\underline{e}$ and $\tilde{\underline{e}}$. In the points of $\mathbf{\Omega}_m^Z$, the coverage error may be greater or lower than $\underline{e}$ and it can be written as

$$Z^* - Z(k+1) = \underline{e} - e(\mathbf{q}, k+1) \equiv \underline{e} - e, \qquad \forall\, \mathbf{q} \in \mathbf{\Omega}_m^Z,$$

where e can be either positive or negative. This gives

$$\sum\nolimits_{\mathbf{\Omega}_m^Z} \big(Z^* - Z(k+1)\big)^2 = \sum\nolimits_{\mathbf{\Omega}_m^Z} (\underline{e} - e)^2 = |\mathbf{\Omega}_m^Z|\, \underline{e}^2 - 2\underline{e} \sum\nolimits_{\mathbf{\Omega}_m^{r\setminus Z}} e + \sum\nolimits_{\mathbf{\Omega}_m^Z} e^2. \tag{2.14}$$

And in the points of the set $\mathbf{\Omega}_m^{r\setminus Z}$ the coverage error is the one of the lower bound, $\underline{e} - \tilde{\underline{e}}$, minus a positive quantity, $e(\mathbf{q}, k+1) \equiv e \geq 0$, that represents the production $\alpha_i(k+1)$, $\forall\, i \in V_m(k+1)$, that can be given in these points:

$$Z^* - Z(k+1) = \underline{e} - \tilde{\underline{e}} - e, \qquad \forall\, \mathbf{q} \in \mathbf{\Omega}_m^{r\setminus Z}.$$

Thus, we have

$$\begin{aligned} &\sum\nolimits_{\mathbf{\Omega}_m^{r\setminus Z}} \left(Z^* - Z(k+1)\right)^2 = \sum\nolimits_{\mathbf{\Omega}_m^{r\setminus Z}} \left(\underline{e} - (\tilde{\underline{e}} + e)\right)^2 \\ &= |\mathbf{\Omega}_m^{r\setminus Z}|\, \underline{e}^2 - 2\underline{e} \sum\nolimits_{\mathbf{\Omega}_m^{r\setminus Z}} (\tilde{\underline{e}} + e) + \sum\nolimits_{\mathbf{\Omega}_m^{r\setminus Z}} (\tilde{\underline{e}} + e)^2. \end{aligned} \tag{2.15}$$

If we sum all the points of $\mathbf{\Omega}_m^r$ with the optimal production, (2.12) and (2.13) and with a generic production function, (2.14) and (2.15), and cancel equal terms, (2.11) becomes

$$\sum\nolimits_{\mathbf{\Omega}_m^{r\setminus Z}} \tilde{\underline{e}}^2 \leq -2\underline{e} \sum\nolimits_{\mathbf{\Omega}_m^r} e + \sum\nolimits_{\mathbf{\Omega}_m^Z} e^2 + \sum\nolimits_{\mathbf{\Omega}_m^{r\setminus Z}} (\tilde{\underline{e}} + e)^2.$$

If $\sum_{\mathbf{Q}_d} \sum_{V_m} \underline{\alpha}_i \geq \sum_{\mathbf{Q}_d} \sum_{V_m} \alpha_i$, then $\sum_{\mathbf{\Omega}_m^r} e \leq 0$. On the contrary, if $\sum_{\mathbf{Q}_d} \sum_{V_m} \underline{\alpha}_i < \sum_{\mathbf{Q}_d} \sum_{V_m} \alpha_i$, this means that $Z_m = Z^*$ and $\underline{e} = 0$. Therefore, the first term of the right-hand side is always zero and since $\tilde{\underline{e}} \geq 0$ and $e \geq 0$, $\forall\, \mathbf{q} \in \mathbf{\Omega}_m^{r\setminus Z}$, Equation (2.11) is proven. □

We have demonstrated that sharing out the maximum production of the robots among the less-covered points leads to a lower quadratic coverage error than applying the actual actions of the robots when the previous maps are equal. The last thing that must be considered to see that (2.5) is a lower bound on the cost is that, for the calculation of the bound, the production that is not used at a time $k+t-1$ can be used at time $k+t$ with the corresponding decay. We define

$$\Delta\varrho_m(k+t-1) = \varrho_m(k+t-1) - \sum\nolimits_{\mathbf{Q}_d} \sum\nolimits_{V_m} \underline{\alpha}_i(k+t-1).$$

Therefore, if instead of (2.8) we use

$$\varrho_m(k+t) = \sum_{\mathbf{Q}_d} \sum_{i \in V_m(k+t)} \rho_i^{max} \alpha_i(\mathbf{q}, \mathbf{p}_i(k)) + d\Delta\varrho_m(k+t-1),$$

we allow the distribution of at least the same coverage in the calculation of the lower bound than in the actual production of the robots. Intuitively it can be seen that this idea together with the optimal production leads to a lower bound on the cost since the lower bound distributed optimally the same coverage or more.

In the example this means that the water that is not used at a time to fill the glasses is kept and can be used at the following instant. In particular, in Fig. 2.2b there are 2 coverage units that have not been distributed. This means that, if at the following instant an empty glass appears, these 2 units will be available to fill it. This is a reasonable assumption for the calculation of the lower bound since having the maximum production to distribute up to the objective and saving the rest will always be better than covering at each time with a fixed production.

2.5 Problem Reduction

In this section we propose a separation of the problem in several smaller subproblems that allows a parallelization of the resolution and reduces the computational cost by reducing the number of nodes and the number of actions involved in each node. The key idea is that if two robots can cover different sets of points at time k, the optimal actions of both of them in k calculated separately are the same as if they are calculated together since the movement and production of one robot does not alter the set that the other can cover. This intuitive idea requires a more complex formulation in the case that we have N robots and T prediction instants.

Let us define the sets of robots whose reachable coverage sets, $\mathbf{\Omega}_i^r(k+t)$ from Equation (2.7), overlap at any time in the prediction horizon:

$$B_e(k) = \big\{ i, j \in \{1, \dots, N\} \mid \mathbf{\Omega}_i^r(k+t) \cap \mathbf{\Omega}_j^r(k+s) \neq \emptyset,\ t, s \in \{1, \dots, T\} \big\}.$$

with $e \in \{1, \dots, E\}$. The key idea of these sets is that, if a robot i can cover a point $\mathbf{q}$ at time $k+t$, it can modify the coverage of such point and, therefore, influence the actions of a different robot that can cover the same point $\mathbf{q}$ at other time $k+s$. Thus, we also define the regions

$$\mathbf{\Omega}_e(k) = \bigcup_{\substack{i \in B_e \\ t \in \{1,\dots,T\}}} \mathbf{\Omega}_i^r(k+t),$$

that include the reachable points of the robots of each $B_e(k)$.

Now we are in the position to present the reduction of the problem in the following proposition.

Proposition 2.5.1. *The solution of the optimization problem* (2.4) *can be obtained by solving E smaller subproblems,*

$$\min_{\mathbf{u},\boldsymbol{\rho}} f(k) = \min_{\mathbf{u}_1,\boldsymbol{\rho}_1} f_1(k) + \cdots + \min_{\mathbf{u}_E,\boldsymbol{\rho}_E} f_E(k) + \sum\nolimits_{t=1}^{T} \sum\nolimits_{\mathbf{q} \notin \bigcup \mathbf{\Omega}_e(k)} \big(Z^* - d^t Z(k)\big)^2,$$

with

$$f_e(k) = \sum\nolimits_{t=1}^{T} \sum\nolimits_{\mathbf{\Omega}_e(k)} \big(Z^* - Z(k+t)\big)^2.$$

These subproblems can be solved independently with the method introduced in Section 2.4.

Proof. The definition of $\mathbf{\Omega}_e(k)$ allows us to reformulate the discrete cost function from (2.4) as

$$\begin{aligned} f(k) &= \sum_{t=1}^{T} \sum_{\mathbf{\Omega}_1(k)} \big(Z^* - Z(k+t)\big)^2 + \sum_{t=1}^{T} \sum_{\mathbf{\Omega}_E(k)} \big(Z^* - Z(k+t)\big)^2 + \sum_{t=1}^{T} \sum_{\mathbf{q} \notin \bigcup \Omega_e(k)} \big(Z^* - d^t Z(k)\big)^2 \\ &= f_1(k) + \cdots + f_E(k) + \sum_{t=1}^{T} \sum_{\mathbf{q} \notin \bigcup \mathbf{\Omega}_e(k)} \big(Z^* - d^t Z(k)\big)^2. \end{aligned}$$

where the last term does not depend on the actions of the robots in the prediction horizon since it gathers all the points that are not reachable for any robot. □

This reduction is very significant since it allows us to divide the original problem (2.4) in problems with less dimensions that can be solved independently with the introduced method and, therefore, to obtain the same optimal solution with a much lower computational cost.

2.6 Simulations

In this section we present simulation results for the branch-and-bound approach to the persistent coverage problem. The simulations are implemented in Matlab. The environment $\mathbf{Q}$ is a square of 20×20 units with a decay rate $d = 0.99$ and the desired coverage level is $Z^* = 100$ for every point. The environment is covered with $N = 3$ robots whose discrete motion is $u_i^x(k) \in U_{d,i}^x = \{-1, 0, 1\}$ and $u_i^y(k) \in U_{d,i}^y = \{-1, 0, 1\}$. The coverage action is $\rho_i^x(k) \in P_{d,i} = \{0, 50\}$, the coverage radius $r_{cov} = 2$ units and the production function

$$\alpha_i(k) = \begin{cases} \dfrac{(r - r_{cov})^2}{r_{cov}^2}, & \text{if } r = \|\mathbf{p}_i(k) - \mathbf{q}\| \leq r_{cov}, \\ 0, & \text{otherwise.} \end{cases}$$

In the first place we present simulation results for the bounds of the branch-and-bound method. In Fig. 2.3 we represent in red the total number of nodes and in blue the number of explored nodes Our bounds allow the algorithm to find the optimal solution exploring only, on average, the 21% of the total. This means that, thanks to the accuracy of the bounds, the 79% of the branches are pruned.

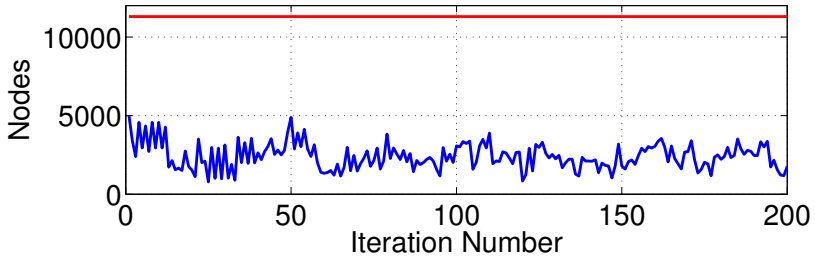


Figure 2.3: Number of explored nodes (blue) vs. total number of nodes (red).

Fig. 2.4 shows the computational cost of the evaluation of the lower bound of a node depending on the number of non-fixed actions. It can be seen that it grows more than linearly. The total execution time for these settings was around 4.5 seconds per iteration on average.

In the second place we focus on the reduction of the problem that has been presented in Section 2.5. The results obtained for the reduction of the complexity of the problem for this simulation are shown in Table 2.1. The entire problem with the actions of the three robots was solved only in the 5% of the cases; one subproblem of 2 robots and one of 1 robot, in the 39% of the cases; and three subproblems of 1 robot in the 56% of the cases. These numbers demonstrate that the proposed reduction of the problem was of utility in the 95% of the iterations and, as can be seen in the table, this implies an enormous reduction in the

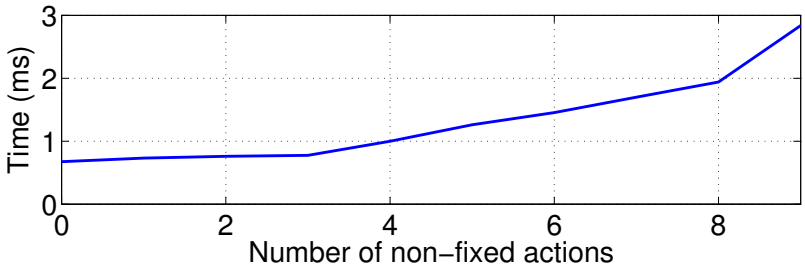


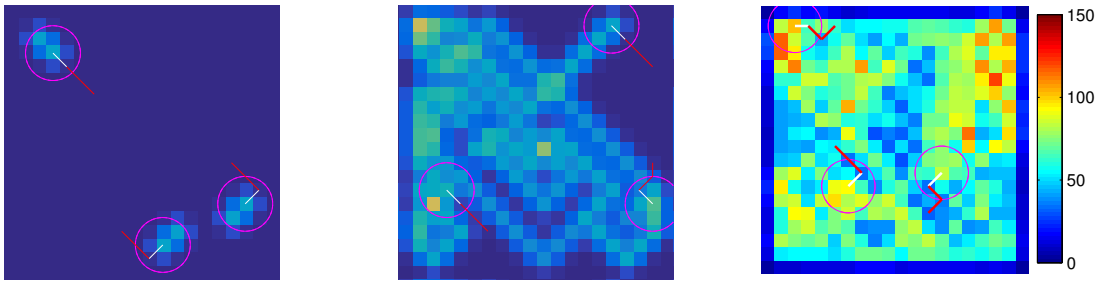
Figure 2.4: Computational cost of the lower bound.

number of nodes that can be explored, around three orders of magnitude for every robot that is separated.

Problems	$N = 3$	$N_1 = 2, N_2 = 1$	$N_1 = 1, N_2 = 1, N_3 = 1$
Ocurrences	5%	39%	56%
Number of nodes	$4 \cdot 10^9$	$4 \cdot 10^6 + 3280$	$3 \cdot 3280$

Table 2.1: Results on problem reduction.

The number of non-fixed actions is bounded by 3 when only one robot is involved in the problem and by 6 when only two are involved. Therefore, the computational cost of the calculation of the lower bound is considerably reduced when applying the reduction of the problem according to Fig. 2.4. In fact, the average iteration time for this case was only 0.64s, almost 7 times lower than without reduction.



(a) k = 1. (b) k = 50. (c) k = 100.

Figure 2.5: Coverage map of three different times and quadratic coverage error of the simulation. Magenta circumferences represent the coverage areas of the robots; white segments, the current motion action and red segments, the predicted actions in T.

Finally, we provide simulation results on the coverage obtained with our entire approach. In Fig. 2.5a - 2.5c the coverage maps of three different times are shown. The current motion actions of the robots are represented with white segments while the red segments represent

the actions calculated for the following instants in the horizon. Initially all the points are uncovered. As the robots move, the coverage level of the points progressively increases with some points reaching the objective while others are still under-covered. In the end, the coverage level of the points spread out around the value of the objective. Fig. 2.6 presents the evolution of the quadratic coverage error of the environment,

$$f\left(k, \mathbf{u}_k, \boldsymbol{\rho}_k\right) = \int_{\mathbf{Q}} \phi \cdot \left(Z^* - Z(k+t)\right)^2 \mathrm{d}\mathbf{q}.$$

It can be seen that it decreases rapidly when the robot start producing and that it tends to a steady-state value different from zero. This happens because near the robots some points are over-covered while the coverage on distant points decays and they become under-covered.

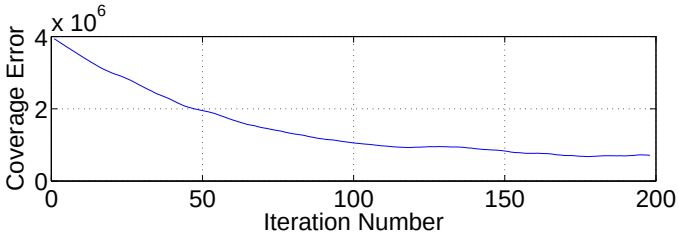


Figure 2.6: Quadratic coverage error.

2.7 Conclusions

In this chapter we have presented an optimal, centralized approach to the persistent coverage problem in a continuous environment. We have formulated the problem as a deterministic finite-state problem and proposed an algorithm based on branch and bound to obtain the optimal actions for the robots. To this end, we have introduced a splitting procedure for the sets of feasible solutions as well as an upper and a lower bound that allow us to prune the branches that do not lead to a better solution. Moreover, we have presented an alternative to reduce the problem to several smaller subproblems whose computational cost is considerably lower. Finally, we have carried out different simulations to validate the proposed contributions and to evaluate the performance of the system.

It has been shown that the computational cost of the branch-and-bound methods is high, since it grows exponentially with the number of variables in the solution set. Actually, the problem become intractable when such number is elevated. As an example, the iteration time for three robots and a time horizon of three steps was 4.5 seconds. Nevertheless, this method presents several theoretical advantages that support and make the exploration of this type of solution worth. The first advantage is that for a moderate number of optimization variables and a reasonable discretization of the environment this method guarantees the optimal open paths at a team level. The second advantage is that, even for a high number of optimization variables, it can be used to generate an accurate initial seed for other optimization methods

that only give the optimal solution if the initial guess is closer enough to it. This initial seed can be obtained from the branch-and-bound method using a looser discretization.

After presenting and discussing the advantages and disadvantages of an optimal centralized solution, in the following chapters we focus on distributed solutions that are computationally much less demanding and scalable, although they may be suboptimal, and progressively tackle more complicated environments.

Chapter 3

Distributed Coverage Estimation

In this chapter we propose an algorithm that allows every robot to estimate in a distributed scenario the global coverage function using only local information, i.e., exchanging information only with neighbors. We pay special attention to the characterization of the algorithm, establishing bounds on the estimation error for every point of the environment. Additionally, we demonstrate that the algorithm guarantees a perfect estimation in a particular area. Finally, we test its correctness and evaluate its performance by means of simulations.

3.1 Introduction

Distributed approaches in multi-robot systems present several advantages with respect to centralized ones in term of robustness, scalability and computational complexity. The main disadvantage, though, is the lack of information on each robot, since they only collect information from their neighbors and may not have global information of what is happening in the environment.

A typical assumption in the approaches to the coverage problem is the exact knowledge of the coverage levels of the whole environment, that permits the calculation of the motion of the robots. One alternative to collect this information is to communicate each production or measurement of every robot through the network labeled with a robot identifier and a time stamp. Then, each robot is capable of retrieving the actual coverage map using all the received information. However, this information exchange can be challenging when dealing with large environments, large and changing robotic networks or when the team of robots has limited communication capabilities.

In the present chapter we present an algorithm to estimate the coverage of the environment using only local information that allows the robots to exactly retrieve the coverage level in their surroundings and to have an estimation with a bounded error in the entire environment. This property implies that, when the motion is calculated based on the coverage level of the area close to the robot, our system behaves as well as a system with centralized information. The algorithm is based on distributed max-consensus combined with additive inputs given by the actions of the robots. It reduces the communication overhead due to the fact that robots only communicate their local estimation of the global map and their local

information of the points where several robots are adding their contributions. We use the term estimation as in deterministic problems of control theory such as state observers [131], although we do not consider delays, noise or uncertainties in the parameters. We provide bounds on the estimation error of the robots for any point of the environment and characterize the set of allowable actions that ensure that each robot maintains a perfect estimation within a certain area, that we call reachable area. Moreover, we characterize the area in which there is no estimation error for any possible action.

The remainder of the chapter is structured as follows. Section 3.2 presents a discrete-time formulation of the coverage estimation problem. The description of the algorithm for the local map update is in Section 3.3. The behavior of the algorithm is characterized in Section 3.4. The performance of the whole approach is analyzed in simulations in Section 3.5 and Section 3.6 gathers the conclusions of this work.

3.2 Problem Formulation

In this chapter the formulation of the problem is almost the same as in the previous one. For this reason, we build upon Section 2.2 and only introduce the differences with respect to it.

We again consider a continuous environment $\mathbf{Q} \subset \mathbb{R}^2$ where the evolution of the coverage is modeled with a constant *decay* gain $d(\mathbf{q})$ as in (2.2) and a coverage level $Z^*(\mathbf{q})$ is desired. This must be attained with $N \in \mathbb{N}$ holonomic robots which obey (2.1). In this case we consider that the maximum distance that a robot can move in one step is the same for all the team, $\|\mathbf{u}_i(k)\| \leq u^{\max}$. Depending on their position, $\mathbf{p}_i(k)$, each robot has a coverage area, $\mathbf{\Omega}_i\big(\mathbf{p}_i(k)\big) \subseteq \mathbf{Q}$, which is bounded by a circle of radius r_i^{cov} centered at $\mathbf{p}_i(k)$, although it is not necessarily circular, convex or equal for all the robots. In this area a robot increases the value of the coverage by $\alpha_i\big(\mathbf{q}, \mathbf{p}_i(k)\big) > 0$, which we call production, and consider constant over time. For the rest of the points of the environment, i.e., the ones that are not covered by robot i, we consider that $\alpha_i\big(\mathbf{q}, \mathbf{p}_i(k)\big) = 0$. From now on we will denote $\mathbf{\Omega}_i(k) \equiv \mathbf{\Omega}_i\big(\mathbf{p}_i(k)\big)$, $\alpha_i(k) \equiv \alpha_i\big(\mathbf{q}, \mathbf{p}_i(k)\big)$ and $\alpha(k) \equiv \sum_{i\in\{1,\dots,N\}} \alpha_i(k)$. Note that in this setting the coverage level of the environment is bounded by $0 \leq Z(k) \leq \max \alpha(k)/(1-d)$.

The robots form a network defined by a communication graph $G_{com}(k) = \big(V(k), E(k)\big)$. The vertices $V(k)$ of the graph are the positions $\mathbf{p}_i(k)$ of the robots. To define the edges $E(k)$, we let $r^{com} > 0$ be the *communication radius*, the maximum distance between two robots at which they can communicate. We assume it constant and equal for all the robots. With this radius, an edge $(i,j) \in E(k)$ if $\|\mathbf{p}_i(k) - \mathbf{p}_j(k)\| \leq r^{com}$. Additionally, $N_i(k) = \{j \in \{1,\dots,N\} \mid (i,j) \in E(k)\}$ are the *neighbors* of robot i at instant k.

Depending on the motion of the robots, the communication graph could end up being disconnected. In this thesis we assume that this is not the case, and we assume that it remains connected at all times even though the graph topology can change over time. This could be achieved, for example, by including in the motion of the robots connectivity constraints such as [132], [108]. Nevertheless, a more relaxed assumption such as periodic joint connectivity [106, 107] could be made, where the union of the disconnected communication graphs would become connected at most every T instants.

The communication in the network is considered synchronous and not affected by delays or noise, assuming this can be provided by a synchronizer [133]. Moreover, the presence of obstacles does not affect the communication capabilities of the network, i.e., two robots keep in communication even if there is an obstacle between them.

In this context, it is clear that any control policy that aims to make $Z(k)$ equal to Z^* will consider the coverage level in the design of the inputs of the robots. However, in equation (2.2) we observe that the value of $Z(k)$ depends on the coverage actions of all the robots. In our distributed scenario, where a robot may not be in direct communication with all the others and may only have partial information of the environment, this represents an obstacle and, for this reason, we introduce our distributed estimation algorithm in the following section.

3.3 Local Estimation of the Coverage

The local estimation of the coverage function of each robot is represented by $Z_i(\mathbf{q}, k)$, that we call local map. Each robot updates its local map using the information received from its current neighbors, which may include the local map and, in some cases, the coverage function of the neighboring robots. Before presenting our updating algorithm, let us introduce two assumptions referring to the communication radius and the initial estimation.

Assumption 3.3.1. *We assume that robots with overlapping coverage areas can communicate, i.e.,* $r^{com} > 2\, r^{cov}_{\max}$, *where* $r^{cov}_{\max} = \max_{i\in\{1,\dots,N\}} r^{cov}_i$.

Assumption 3.3.2. *We assume that at the initial time the robots know the actual coverage level, i.e., the estimation of every robot is correct:* $Z_i(0) = Z(0), \forall\, \mathbf{q} \in \mathbf{Q}$.

Now we present in Algorithm 1 our 2-step strategy to update the estimation along with an illustrating example in Figure 3.1. In this example we have the global coverage map $Z(k)$ from Figure 3.1a and we refer to the estimation of robot 1. The intuitive idea of the algorithm is the following: we extract in Step 1 the additions that the neighbors have already gathered, both from their productions and from other robots, and we take into account the overlappings between the coverage areas of the robots to correct the estimation in Step 2. In fact, the objective of this update is to keep the estimation as close to the actual value as possible without overestimating it.

In the first step, at each communication time, k, each robot generates its *map-to-communicate*, $Z^{com}_i(k)$ (Figure 3.1b in example), as the map of the previous time instant with its decay plus its current production,

$$Z^{com}_i(k) = dZ_i(k-1) + \alpha_i(k), \tag{3.1}$$

where we assume that the decay rate is known by the robots. Note that at time k each robot includes the production of the same time instant k. Therefore, there is no delay or offset in the estimation.

Each robot sends its map-to-communicate to its neighbors and receives their maps. With this information, the first step of the update is performed dividing the map into two parts:

Algorithm 1 *Coverge Estimation Update*

– *Step 1:*
Calculate map-to-communicate $Z_i^{com}(k)$, (3.1).
Communicate map to neighbors.
Update local map $Z_i^-(k)$, (3.2).
– *Step 2:*
Extract overlapped production $\beta_i\big(\mathbf{q}, \mathbf{p}_i(k)\big)$, (3.3).
Communicate regions to neighbors.
Update local map $Z_i(k)$, (3.4).

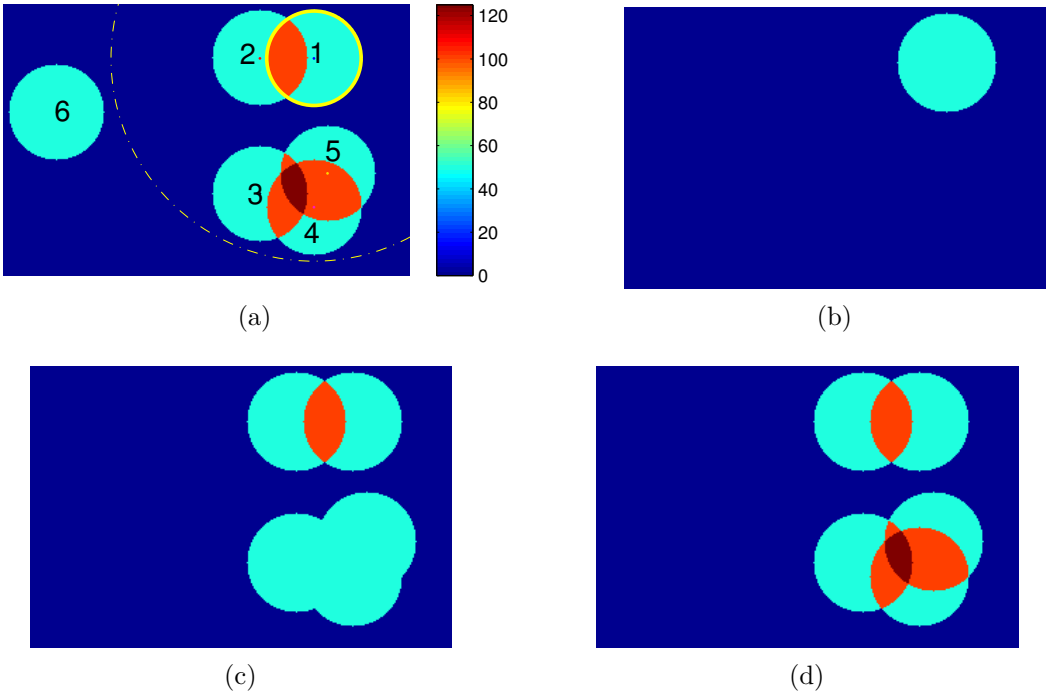


Figure 3.1: Example of the estimation algorithm at time k. (a) Global coverage map, $Z(k)$. The coverage area of robot 1 is represented with a yellow circumference and its communication area with a dash-dotted one. (b) Map-to-communicate of robot 1, $Z_1^{com}(k)$. (c) Map calculated by robot 1 in the first step of the update, $Z_1^-(k)$. (d) Estimation of the coverage map of robot 1, $Z_1(k)$. The orange areas coincide with the overlapped productions of the robots, $\beta_i(k)$.

the coverage area, $\mathbf{\Omega}_i(k)$, and the rest of the map. The robot updates each region separately according to

$$Z_i^-(k) = Z_i^{com}(k) + \sum_{j \in N_i(k)} \max\big(Z_j^{com}(k) - dZ_i(k-1), 0\big), \qquad \forall\, \mathbf{q} \in \mathbf{\Omega}_i(k), \tag{3.2a}$$

$$Z_i^-(k) = \max_{j \in N_i(k)} \big(dZ_i(k-1), Z_j^{com}(k)\big), \qquad \forall\, \mathbf{q} \notin \mathbf{\Omega}_i(k). \tag{3.2b}$$

For the update (3.2a) robot i adds its map-to-communicate from (3.1) to the contributions of its neighbors. The contribution of a neighbor j can be calculated as the difference $Z_j^{com}(k) - dZ_i(k-1)$, e.g., the orange area of Figure 3.1c. To prevent this difference from being negative we use the maximum function. This may happen when the estimation of robot j is smaller than the estimation of robot i as in the following situation. Let robot i be overlapped with a robot ℓ that is not a neighbor of robot j, i.e., $j, \ell \in N_i(k)$ and $j \notin N_\ell(k) \Leftrightarrow \ell \notin N_j(k)$. The estimation $Z_j^{com}(k)$ may be lower than $dZ_i(k-1)$ in the overlapping region $\mathbf{\Omega}_i(k) \cap \mathbf{\Omega}_\ell(k)$ since the production of robot ℓ at time k is not available to robot j. This would happen to the estimation of robot 2 in Figure 3.1a at the next iteration. It would be overlapped with robot 1 and in direct communication with robot 6, while 1 and 6 would not be neighbors.

In the second part of the local map update (3.2b), each robot updates its local map outside its coverage area, $\mathbf{q} \notin \mathbf{\Omega}_i(k)$, as the maximum of the received values and its own. Such value corresponds to the latest information of this robot or its neighbors. This update underestimates the coverage level when two or more neighbors are overlapping outside the coverage area of a robot as in Figure 3.1c. In that situation only the highest contribution is considered and not the addition of all of them.

To counteract this error, a second updating step is executed. At first, each robot extracts the region of its coverage area that is overlapped with the one of another robot,

$$\mathbf{\Omega}_i^o(k) = \big\{\mathbf{q} \in \mathbf{\Omega}_i(k) \cap \mathbf{\Omega}_j(k) \,|\, j \in N_i(k)\big\}.$$

This area can be determined as

$$\mathbf{\Omega}_i(k) \cap \mathbf{\Omega}_j(k) = \big\{\mathbf{q} \in \mathbf{\Omega}_i(k) \,|\, Z_j^{com}(k) - dZ_i(k-1) > 0\big\}, \qquad \forall\, j \in N_i(k).$$

Then, the robot sends its coverage function in the overlapped region to its neighbors:

$$\beta_i(k) \equiv \beta_i\big(\mathbf{q}, \mathbf{p}_i(k)\big) = \alpha_i\big(\mathbf{q}, \mathbf{p}_i(k)\big), \qquad \forall\, \mathbf{q} \in \mathbf{\Omega}_i^o(k). \tag{3.3}$$

This information is the only that a neighboring robot may not be able to extract from the maps-to-communicate in the first step due to the maximum function in (3.2b). Although sending a full map with the robot production at the first step of the algorithm makes it simpler, it is not a communication-effective solution since it is not necessary to send all the points. On the contrary, with our algorithm each robot locally decides which points are of interest for its neighbors and discards the rest.

After exchanging the overlapped productions with their neighbors, they perform the final update with the received ones:

$$Z_i(k) = \begin{cases} Z_i^-(k), & \forall\, \mathbf{q} \in \mathbf{\Omega}_i(k), \\ Z_i^-(k) - \max\limits_{j \in N_i(k)} \beta_j(k) + \sum\limits_{j \in N_i(k)} \beta_j(k), & \forall\, \mathbf{q} \notin \mathbf{\Omega}_i(k). \end{cases} \tag{3.4}$$

This final step adds the contributions that are not considered in the first and ends the estimation, as shown in Figure 3.1d.

The presented algorithm permits that each robot locally decides which points of its own production are of interest for its neighbors and discards the rest in a communication-effective solution. Another important property, that will be useful in the next section, is that it satisfies

$$Z_i(k) \geq dZ_i(k-1), \quad \forall\, \mathbf{q} \in \mathbf{Q}. \tag{3.5}$$

Also note that the presence of obstacles or a non-convex environment would not affect the estimation algorithm according to the communication policy. In any case, slightly different rules could be applied to estimate the coverage but, in that case, different information should be exchanged. For instance, another estimation strategy could exchange the coverage α_i and forward the received α_j. However, this alternative would require a trace of which productions have been included and forwarded, several communication rounds at each iteration and an increased communication expense.

3.4 Characterization of the Estimation

We characterize now the accuracy of the estimation using Algorithm 1. In the first place we determine the allowable actions of each robot in order to guarantee that the estimation is equal to the global map inside what we call the zero-error area and the reachable area of the robots. Next, we delimit the areas in which a correct estimation is guaranteed regardless of the movement of the robots. Finally, we establish bounds to the estimation error in all the points of the environment.

Let us begin defining the zero-error area of robot i,

$$\mathbf{\Omega}_i^z(k) = \left\{\mathbf{q} \in \mathbf{Q} \,|\, \|\mathbf{q} - \mathbf{p}_i(k)\| < r^{com} - r_{max}^{cov}\right\},$$

that contains the points in which our algorithm guarantees a correct estimation. These are the points of the environment that can be covered by another robot only if it is a neighbor of i at each time. In Figure 3.2 an illustrative example of this area is shown in red.

We also define the reachable area of robot i,

$$\mathbf{\Omega}_i^r(k) = \left\{\mathbf{q} \in \mathbf{Q} \,|\, \|\mathbf{q} - \mathbf{p}_i(k)\| < r^{com} - r_{max}^{cov} + u_i^{\max}(k)\right\},$$

that comprises the points of the environment that can be reached by the zero-error area of robot i with the action calculated at time k, where $u_i^{\max}(k)$ is the maximum action of robot i at such time.

Let us introduce now some useful results in the following lemmas to demonstrate the correctness of the estimation in these points in Theorem 3.4.4. First we prove that a correct estimation at a time $k-1$ in $\mathbf{\Omega}_i^r(k-1)$ leads to a bounded estimation in all the environment at time k and also to a correct estimation in $\mathbf{\Omega}_i^z(k)$, so that the update of the algorithm does not introduce an error in the current zero-error area.

Lemma 3.4.1. *For any iteration k, consider that at the previous iteration the estimation of the coverage map of a robot i is equal to the global coverage map inside its reachable area:*

$$Z_i(k-1) = Z(k-1), \qquad \forall\, \mathbf{q} \in \mathbf{\Omega}_i^r(k-1). \tag{3.6}$$

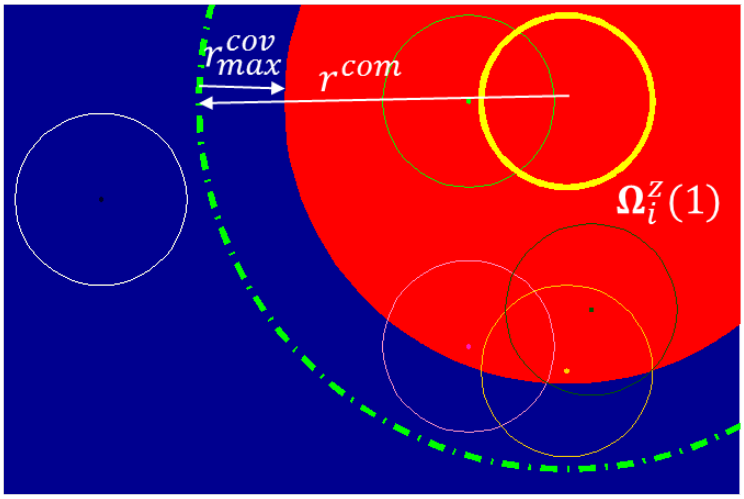


Figure 3.2: Example of zero-error area (red) for the robot depicted in yellow.

Then, the execution of Algorithm 1 leads to a bounded estimation in all the environment,

$$Z_i(k) \leq Z(k), \qquad \forall\, \mathbf{q} \in \mathbf{Q},$$

and does not introduce an error in the estimation inside the zero-error area,

$$Z_i(k) = Z(k), \qquad \forall\, \mathbf{q} \in \mathbf{\Omega}_i^z(k).$$

Proof. In this proof all the areas refer to time k unless it is explicitly stated, for instance, $\mathbf{\Omega}_i^z \equiv \mathbf{\Omega}_i^z(k)$ and also $N_i \equiv N_i(k)$. In the first place we prove the upper bound of the estimation. Following the principle of induction:

1) Consider $Z_i(0) \leq Z(0)$ from Assumption 3.3.2.

2) Assume as an induction hypothesis that

$$Z_i(k-1) \leq Z(k-1), \qquad \forall\, \mathbf{q} \in \mathbf{Q},\ i \in \{1, \ldots, N\}. \tag{3.7}$$

3) We split the environment into two areas in accordance to the estimation algorithm. If a point $\mathbf{q} \in \mathbf{\Omega}_i$, the update from (3.1), (3.2a), (3.4) and Assumption 3.3.1 is

$$Z_i(k) = dZ_i(k-1) + \alpha_i(k) + \sum_{j \in N_i} \max\Big(dZ_j(k-1) + \alpha_j(k) - dZ_i(k-1), 0\Big),$$

where, according to the hypothesis of this lemma (3.6) and the induction hypothesis,

$$0 \leq \max\Big(dZ_j(k-1) + \alpha_j(k) - dZ_i(k-1), 0\Big) \leq \alpha_j(k).$$

Therefore, we have

$$Z_i(k) \leq dZ_i(k-1) + \alpha_i(k) + \sum_{j \in N_i} \alpha_j(k) \leq Z(k).$$

In the second area, i.e., if $\mathbf{q} \notin \mathbf{\Omega}_i$, the update from (3.1),(3.2b) and (3.4) is

$$Z_i(k) = \max_{j \in N_i} \big(dZ_i(k-1), dZ_j(k-1) + \alpha_j(k)\big) - \max_{j \in N_i} \beta_j(k) + \sum_{j \in N_i} \beta_j(k), \tag{3.8}$$

where the first term

$$\max_{j \in N_i} \big(dZ_i(k-1), dZ_j(k-1) + \alpha_j(k)\big) \leq dZ(k-1) + \max_{j \in N_i} \alpha_j(k).$$

If such $\mathbf{q} \in \mathbf{\Omega}_j^o$, that is, $\beta_j(k) > 0$, then $\max_{j \in N_i} \alpha_j(k) = \max_{j \in N_i} \beta_j(k)$, $\sum_{j \in N_i} \beta_j(k) = \sum_{j \in N_i} \alpha_j(k)$ and (3.8) becomes

$$Z_i(k) \leq dZ(k-1) + \sum\nolimits_{j \in N_i} \alpha_j(k) \leq Z(k). \tag{3.9}$$

Finally, if such $\mathbf{q} \notin \mathbf{\Omega}_j^o$, there exists no $\beta_j(k)$, (3.8) becomes

$$Z_i(k) \leq dZ(k-1) + \max\nolimits_{j \in N_i} \alpha_j(k) \leq Z(k).$$

and the first result is proven.

For the second result of this lemma an analogous demonstration can be developed splitting the environment into $\mathbf{\Omega}_i$ and the rest of the zero-error area, $\mathbf{\Omega}_i^z \setminus \mathbf{\Omega}_i$, particularizing each region for $\mathbf{q} \in \mathbf{\Omega}_j$ and $\mathbf{q} \notin \mathbf{\Omega}_j$, and applying the hypothesis and the first result of this lemma. □

Next, we prove that the estimation at a particular time takes into account all the productions from at least $N-1$ instants before. To do so, we denote the diameter of the communication graph as $\operatorname{diam}\big(G_{com}(k)\big)$, that is bounded by $1 \leq \operatorname{diam}\big(G_{com}(k)\big) \leq N-1$. Thus, the maximum time that the production of a robot needs to propagate through the entire network is equal to the number of robots minus one.

Lemma 3.4.2. *Consider that for any iteration $k-N$ the estimation of the coverage map of all robots is equal to the global coverage map inside their reachable area,*

$$Z_i(k-N) = Z(k-N), \qquad \forall\, \mathbf{q} \in \mathbf{\Omega}_i^r(k-N).$$

Then, according to Algorithm 1, the production of the robots developed at time instant $k-N+1$, i.e., $\alpha(k-N+1)$, does not induce an error in the estimation of the coverage at time k, i.e., in $Z_i(k), i \in \{1, \ldots, N\}$.

Proof. Let $\mathbf{q} \in \mathbf{Q}$ be a point such that, for at least one robot, j_1, $\alpha_{j_1}(k-N+1) > 0$. In such case $\mathbf{q} \in \mathbf{\Omega}_{j_1}^r(k-N)$, by assumption of this lemma $Z_{j_1}(k-N) = Z(k-N)$ and from Lemma 3.4.1, $Z_{j_1}(k-N+1) = Z(k-N+1)$. In the following iterations, according to the property of the estimation from (3.5), it is clear that,

$$Z_{j_1}(k-N+1+\ell) \geq d^{\ell-1} Z(k-N+1) = d^{\ell} Z(k-N) + d^{\ell-1} \alpha(k-N+1), \tag{3.10}$$

with $\ell \in \{1, \ldots, N-1\}$, implying that any possible error in the estimation of j_1 is not due to the productions at time $k-N+1$.

In the next iteration, $k-N+2$, for any neighbor of j_1, say j_2, it can happen that $\mathbf{q} \in \mathbf{\Omega}^r_{j_2}(k-N+1)$ or not. If $\mathbf{q} \notin \mathbf{\Omega}_{j_2}(k-N+2)$, from (3.2b) and (3.4),

$$\begin{aligned} Z_{j_2}(k-N+2) &= \max_{m_2}(dZ_{j_2}(k-N+1), Z^{com}_{m_2}(k-N+2)) \\ &\quad + \sum\nolimits_{m_2} \beta_{m_2}(k-N+2) - \max_{m_2}\left(\beta_{m_2}(k-N+2)\right) \\ &\geq dZ_{j_1}(k-N+1) \geq d^2 Z(k-N) + d\alpha(k-N+1), \end{aligned}$$

with $m_2 \in N_{j_2}(k-N+2)$. On the other hand, if $\mathbf{q} \in \mathbf{\Omega}_{j_2}(k-N+2)$, the update from (3.2a) and (3.4) results in

$$\begin{aligned} Z_{j_2}(k-N+2) &\geq dZ_{j_2}(k-N+1) + \alpha_{j_2}(k-N+2) \\ &\quad + \max\left(dZ_{j_1}(k-N+1) + \alpha_{j_1}(k-N+2) - dZ_{j_2}(k-N+1), 0\right) \\ &\geq dZ_{j_1}(k-N+1) \geq d^2 Z(k-N) + d\alpha(k-N+1). \end{aligned}$$

Thus, j_2 has no estimation error at time $k-N+2$ caused by $\alpha(k-N+1)$. Additionally, using the same argument as in (3.10) it is clear that any error in j_2 in the following iterations is not due to $\alpha(k-N+1)$.

A similar reasoning can be developed at time $k-N+3$ for neighbors of j_1 or j_2 and, consequently, in at most $N-1$ iterations the same result can be obtained for all the robots of the network. Therefore, the estimation of any robot of the network at time k includes at least the productions of the robots at time $k-(N-1)$ and the proof is complete. □

The third result establishes bounds in the motion of each robot in order to assure that its zero-error area does not reach the production of other robots which were not its neighbors at the previous time.

Lemma 3.4.3. *Let $\mathbf{\Omega}^\rho_i(k)$ be a circle of center $\mathbf{p}_i(k)$ and radius $\rho_i(k) - r^{cov}_{\max}$. $\rho_i(k) = \min_{j,\ell} \|\mathbf{p}_i(k) - \mathbf{p}_j(\ell)\|$, with $j \notin N_i(k)$ and $\ell > 0, \ell \in \{k-N+1, ..., k\}$, is the minimum distance from the current position of robot i to any of the last $N-1$ positions of other robots j that are not neighbors of i at iteration k. If the motion of robot i at time k satisfies that*

$$\|\mathbf{u}_i(k)\| \leq u^{\max}_i(k) = \min\left(u^{\max}, \rho_i(k) - r^{com}\right), \tag{3.11}$$

then, $\alpha_j(\ell) = 0$, $\forall\, \mathbf{q} \in \mathbf{\Omega}^r_i(k)$, with $j \notin N_i(k)$ and $\ell > 0, \ell \in \{k-N+1, ..., k\}$.

Proof. Since $\rho_i(k)$ is the minimum distance from the current position of robot i to any of the last $N-1$ positions of other robots j that are not neighbors of i at iteration k, then $\alpha_j(\ell) = 0$ for all $\mathbf{q} \in \mathbf{\Omega}^\rho_i(k)$ and $\ell = \{k-N+1, \ldots, k\}$.

Therefore, we need to demonstrate that $\mathbf{\Omega}^r_i(k) \subseteq \mathbf{\Omega}^\rho_i(k)$. This happens if the radius of $\mathbf{\Omega}^r_i(k)$ is smaller than the radius of $\mathbf{\Omega}^\rho_i(k)$, that is,

$$r^{com} - r^{cov}_{\max} + u^{\max}_i(k) \leq \rho_i(k) - r^{cov}_{\max},$$

which is true if (3.11) holds. □

In practice, the restriction that the previous lemma establishes on $\|\mathbf{u}_i(k)\|$ may be stronger than needed, since the direction of the movement may direct the robot to a place with no possible error, while the module of the action restricts such movement. Therefore, this restriction should only be applied depending on the direction of the motion.

These results allow us to introduce the theorem that guarantees a perfect estimation of the coverage in the reachable area of the robots at every time. This theorem proves the hypothesis of Lemmas 3.4.2 and 3.4.1.

Theorem 3.4.4. *If Assumptions 3.3.1 and 3.3.2 hold and the motion of the robots satisfies* (3.11)*, then, according to Algorithm 1, the local map is equal to the global coverage map at time instant k inside the reachable area of the robot,*

$$Z_i(k) = Z(k), \qquad \forall\, \mathbf{q} \in \mathbf{\Omega}_i^r(k).$$

Proof. Following the principle of induction:

1) Consider $Z_i(0) = Z(0)$ from Assumption 3.3.2.

2) Assume as an induction hypothesis that

$$Z_i(k-1) = Z(k-1), \qquad \forall\, \mathbf{q} \in \mathbf{\Omega}_i^r(k-1).$$

3) Using Lemma 3.4.1 and the induction hypothesis we have $Z_i(k) = Z(k)$ in $\mathbf{\Omega}_i^z(k)$. Then for those areas of $\mathbf{\Omega}_i^r(k)$ that intersect with $\mathbf{\Omega}_i^z(k)$ the result is proven.

We now focus on demonstrating that the same holds for any point $\mathbf{q} \in \mathbf{\Omega}_i^r(k) \setminus \mathbf{\Omega}_i^z(k)$. Lemma 3.4.2 demonstrates that the production of the robots before or at time $k-N+1$ induces no error in $Z_i(k)$ and Lemma 3.4.3 guarantees that $\alpha_m(\ell) = 0$ in $\mathbf{\Omega}_i^r(k)$ for all $\ell \in \{k-N+1, ..., k\}$, $m \notin N_i(k)$. Therefore, only the production after $k-N+1$ of the current neighbors of robot i can cause an error on $Z_i(k)$ within $\mathbf{\Omega}_i^r(k) \setminus \mathbf{\Omega}_i^z(k)$. For any point of this region the updating law from (3.1), (3.2b) and (3.4) is (3.8).

If there exists some neighbor j such that $\mathbf{q} \in \mathbf{\Omega}_j^z(k)$, then $Z_j(k) = Z(k)$ and the first term of (3.8) becomes $dZ(k-1) - \max_{j\in N_i(k)} \alpha_j(k)$. Since $\alpha_i(k) = 0$ and $\max_{j\in N_i(k)} \alpha_j(k) = \max_{j\in N_i(k)} \beta_j(k)$, then $Z_i(k) = Z(k)$ in $\big(\mathbf{\Omega}_i^r(k) \setminus \mathbf{\Omega}_i^z(k)\big) \cup \mathbf{\Omega}_j^z(k)$.

On the other hand, if there is no neighbor j such that $\mathbf{q} \in \mathbf{\Omega}_j^z(k)$, i.e., $\alpha_j(k) = 0$ for all $j \in V$ in $\mathbf{q}$, then, (3.8) results in $Z_i(k) = \max_{j\in N_i(k)\cup i} dZ_j(k-1)$.

Repeating the same procedure for time $k-1$, if $\mathbf{q} \in \mathbf{\Omega}_j^z(k-1)$ for any $j \in \{N_i(k), i\}$, then $Z_j(k-1) = Z(k-1)$ which implies that $Z_i(k) = dZ(k-1) = Z(k)$. If $\mathbf{q} \notin \mathbf{\Omega}_j^z(k-1)$, then we can repeat recursively until $k-(N-1)$. If $\mathbf{q} \notin \mathbf{\Omega}_j^z(\ell)$ with $\ell \in \{k-N+1, ..., k-1\}$, then, according to Lemma 3.4.2,

$$Z_i(k) = d^N Z(k-N) + d^{N-1}\alpha\big(k-(N-1)\big) = Z(k),$$

completing the proof. □

We can now extend the area in which a robot has no estimation error at each time depending on the positions of the non-neighbors.

Corollary 3.4.5. *Considering the same as in Theorem 3.4.4, the local estimation of the coverage map of any robot i is equal to the global map inside $\mathbf{\Omega}_i^{\rho}(k)$,*

$$Z_i(k) = Z(k), \qquad \forall \mathbf{q} \in \mathbf{\Omega}_i^{\rho}(k).$$

Proof. The same proof as for Theorem 3.4.4 can be applied here using $\mathbf{\Omega}_i^{\rho}(k)$ instead of $\mathbf{\Omega}_i^{r}(k)$. □

The previous results guarantee zero estimation error in a region that depends on the positions of the robots that are not neighbors, $m \notin N_i(k)$, at each time k. However, in a distributed approach a robot does not know such positions. Regardless of the motion of the robot and the positions of the other robots there is a region, presented in the following theorem, in which zero error is guaranteed.

Theorem 3.4.6. *Let $r_z^* = r^{com} - r_{max}^{cov} - (N-1)u^{\max}$. Then, if Assumptions 3.3.1 and 3.3.2 hold, Algorithm 1 guarantees that*

$$Z_i(k) = Z(k), \qquad \forall\, \mathbf{q} \in \mathbf{Q} \,|\, \|\mathbf{p}_i(k) - \mathbf{q}\| < r_z^*.$$

Proof. According to Lemma 3.4.2, whose hypothesis is confirmed by Theorem 3.4.4, the production of the robots from $N-1$ or more instants before does not induce an error. Therefore, it is guaranteed that the robot does not have an estimation error in an area around it if it is not capable of reaching the production of non-neighbors in less than $N-1$ iterations. According to Assumption 3.3.1, in the worst case, the production of a non-neighbor is at a distance of $r^{com} - r_{max}^{cov}$. Additionally, a robot can move at most $(N-1)u^{\max}$ units in $N-1$ instants. Therefore, the radius of the area around the robot that cannot reach the previous productions of a non-neighbor is $r_z^* = r^{com} - r_{max}^{cov} - (N-1)u^{\max}$ and the proof is complete. □

Finally, we set bounds to the estimation error in all points.

Theorem 3.4.7. *Consider that Assumptions 3.3.1 and 3.3.2 hold. Also consider that the maximum value of the production function of every robot is*

$$P = \max_{i \in \{1,\ldots,N\}} \alpha_i(k). \tag{3.12}$$

Then, according to Algorithm 1, the error of the local estimation of the coverage map for all $k > 0$ is bounded by

$$0 \leq Z(k) - Z_i(k) \leq P(N-1)\frac{1-d^N}{1-d}. \tag{3.13}$$

Proof. The lower bound is proven in Lemma 3.4.1, whose hypothesis is confirmed by Theorem 3.4.4. We prove here the upper bound. As we mentioned, the error of the local estimation, $\varepsilon^{est} = Z(k) - Z_i(k)$, is caused by the lack of communication between all robots of the network. In fact, the more non-neighbors of robot i overlap between them, the higher this error is for robot i at each time instant. Consequently, we can formulate the maximum error produced at a particular time instant as $e = Pn_o$ where $0 < n_o \leq N-1$ is the number of overlapping robots outside the communication area of any other robot. In the worst case scenario, this error is repeated every time instant and added to the previous ones that have decayed over time. Therefore, we have

$$\varepsilon^{est} = e + de + d^2e + ... + d^{\delta-1}e = e\frac{1-d^{\delta}}{1-d},$$

where $\delta - 1$ represents the number of iterations until the robot receives the information and its value depends on n_o. However, as introduced for Lemma 3.4.2, the maximum is $N-1$ instants, i.e., $\delta - 1 \leq N-1$, and only the errors of the last $N-1$ iterations have to be added. Thus,

$$\varepsilon^{est} = Pn_o\frac{1-d^{\delta}}{1-d} \leq P(N-1)\frac{1-d^{N}}{1-d},$$

and the desired result is proven. □

The bound in (3.13) can also be modified to take into account disconnected networks under the standard assumption of periodic joint connectivity [134], which supposes the union of the disconnected communication graphs form a connected one at most every T instants.

Remark 3.4.8. *Consider the same as in Theorem 3.4.7 and also consider periodic joint connectivity of the network. Then, according to Algorithm 1, the error of the local estimation of the coverage map for all $k > 0$ is bounded by*

$$0 \leq Z(k) - Z_i(k) \leq P(N-1)\frac{1-d^{TN}}{1-d}.$$

3.5 Simulations

Eventually, we present simulation results for the proposed estimation algorithm. At the beginning, we show the behavior in an illustrative example and afterwards, we provide an exhaustive evaluation of the influence of the parameters using a Monte Carlo analysis.

The reference settings that we have used in these simulations are the following. The environment $\mathbf{Q}$ is a rectangle of 100×150 units with a decay rate $d = 0.995$ and the desired coverage level is $Z^* = 100$ for every point. The team consists of $N = 6$ robots, whose maximum motion is $u^{\max} = 5$ and whose production function is equal to

$$\alpha_i(k) = \begin{cases} \dfrac{P}{r_i^{cov2}}(r - r_i^{cov})^2, & \text{if } r = \|\mathbf{p}_i(k) - \mathbf{q}\| \leq r_i^{cov}, \\ 0, & \text{otherwise,} \end{cases}$$

with a maximum value $P = 25$ and a coverage radius $r_i^{cov} = 10$ units. The communication radius is $r^{com} = 120$, that clearly satisfies Assumption 3.3.1, and the motion controller will be explained in the next chapter. Note that in a real-world environment most parameters are fixed by the application and by the specific robots used. Thus, for the simulation the selection is arbitrary. Also note that, without loss of generality, we use the same coverage radius and the same coverage function for all the robots.

Illustrative Example

This section presents an illustrative example of the simulations with the reference settings. In Theorem 3.4.7, we state that the estimation error is bounded by (3.13). If we calculate the maximum estimation error of the network for this example, the maximum value reached is 25.1, as can be seen in Figure 3.3. This error is lower than the value of our upper bound that is 740.7. Although the theoretical bound is loose, it shows that the estimation error does not go to infinity and remains bounded. In practice, it can be seen that the estimation error is much lower and that the algorithm works even better than theoretically predicted. The minimum value of the estimation error is always 0 which also confirm the lower bound in (3.13).

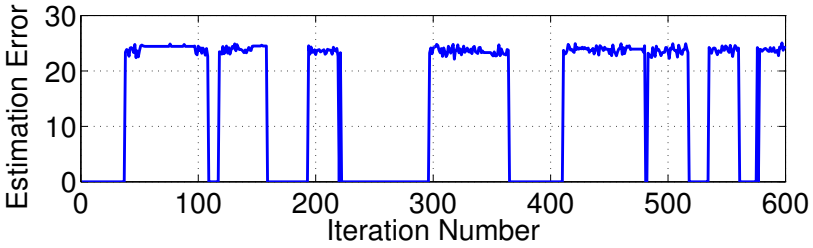


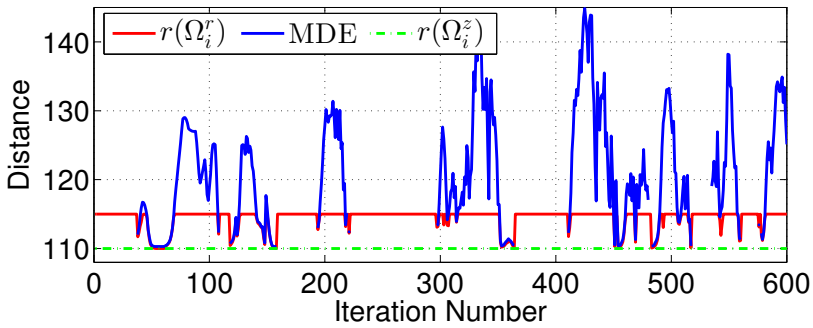
Figure 3.3: Evolution of the maximum estimation error of the network.

In Theorem 3.4.4, we state that the estimation is perfect inside the reachable area of each robot when the motion action satisfies (3.11) and in Theorem 3.4.6, the area in which there is no estimation error when no restriction is applied to the motion. To check that these statements hold, we use the minimum distance to an estimation error between all the robots (MDE). This distance is the shortest Euclidean distance from a robot to the points in which it has an estimation error, i.e., where $Z_i(k) \neq Z(k)$:

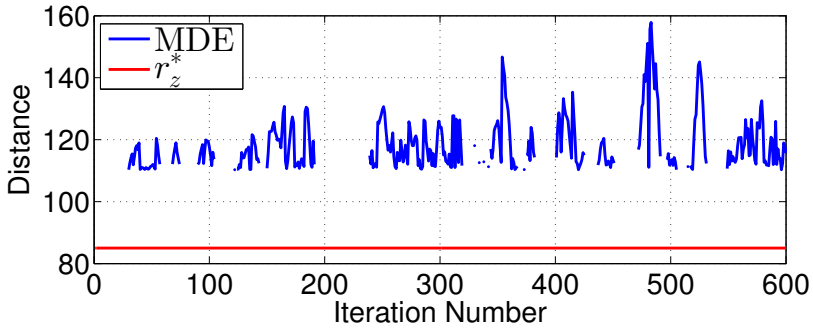
$$\text{MDE}(k) = \min_{i,\mathbf{q}} \big(\|\mathbf{q} - \mathbf{p}_i\| \mid Z_i(\mathbf{q}, k) \neq Z(\mathbf{q}, k) \big). \tag{3.14}$$

In Figure 3.4 we depict the MDE from (3.14) in each iteration. When the motion satisfies (3.11) (Figure 3.4a), the MDE is always greater than the radius of the reachable area (red line) and, thus, comply with Theorem 3.4.4. It is also greater than the radius of the zero-error area (green dash-dotted line) that has a constant value. On the other hand, when there is no restriction to the motion, Figure 3.4b shows that MDE is greater than $r_z^* = 85$ (red line) as asserted in Theorem 3.4.6. In fact, although it is not theoretically guaranteed, it can be seen that most of the time the estimation is correct in an area with the same

radius as the zero-error area, i.e., MDE greater than 110, and even in a bigger region. This is an advantage since the velocity restriction from Equation (3.11) can not be calculated in a distributed scenario and allows us to omit it in a real-world application. In addition, in both figures the MDE is not depicted in the iterations in which all the robots have a perfect estimation.



(a) Motion satisfies Equation (3.11).



(b) No restriction to the motion.

Figure 3.4: Evolution of the MDE.

If we test the restriction in terms of the motion of the robots, this example shows that, in practice, it is not very limiting. Figure 3.5 shows the maximum allowed movement in the calculated direction at each iteration for a robot. It can be seen that the desired movement is under the maximum allowed in most cases.

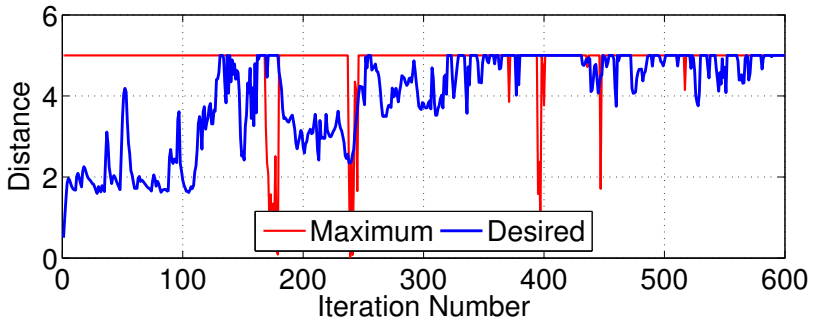


Figure 3.5: Movement of a robot. The blue line represents the desired movement of the robot and the red line, the maximum movement allowed by Lemma 3.4.3.

Local Estimation of the Coverage

In order to perform a thorough validation of our local map updating strategy from Algorithm 1, we developed a Monte Carlo analysis, which includes 20 runs of the simulations from 20 different initial positions of the robots for different combination of parameters. In particular, we evaluate the influence of varying the decay, d, the number of robots, N, the communication radius, r^{com}, the maximum value of the production, P, and the coverage radius, r^{cov}, with respect to the reference settings. To do so, we synthesized the two measurements introduced in the example: we calculate the maximum estimation error and the minimum MDE over all the runs and average both over time. We call them $\overline{\varepsilon}_{\max}$ and $\overline{\text{MDE}}$, respectively. In Table 3.1 the results of this study are summarized and the conclusions from them are:

d	**0.991**	**0.993**	**0.995**	**0.997**
$\overline{\varepsilon}_{\max}$	25.3	25.4	25.8	26.2
$\overline{\text{MDE}}$	111.3	111.2	111.2	111.3

N	**4**	**6**	**8**	**12**	**16**
$\overline{\varepsilon}_{\max}$	25.2	25.8	26.3	29.5	31.8
$\overline{\text{MDE}}$	112.6	111.2	110.8	110.5	110.4

$\mathbf{r^{com}}$	**80**	**100**	**120**	**140**	**160**
$\overline{\varepsilon}_{\max}$	42.6	32.7	25.8	22.4	4.1
$\overline{\text{MDE}}$	70.5	90.8	111.2	132.4	153.8

P	**15**	**20**	**25**	**30**	**35**
$\overline{\varepsilon}_{\max}$	15.1	20.3	25.8	31.7	36.6
$\overline{\text{MDE}}$	111.2	111.3	111.2	111.1	111.2

$\mathbf{r^{cov}}$	**5**	**10**	**15**	**20**
$\overline{\varepsilon}_{\max}$	25.1	25.8	28.0	31.4
$\overline{\text{MDE}}$	116.6	111.2	106.1	101.1

Table 3.1: Maximum estimation error, $\overline{\varepsilon}_{\max}$, and minimum distance to an estimation error, $\overline{\text{MDE}}$.

- The variation of the decay has a small influence on $\overline{\varepsilon}_{\max}$. This error is around the value of P which means two things: few overlappings take place outside the zero-error area of the robot and the production of the non-neighbors needs only one instant to

reach the robot. In these conditions, the decay does not play an important role. In addition, the decay does not influence the $\overline{\text{MDE}}$ since it only modifies the coverage level. Further tests showed that both conclusions are valid for even larger variations of the decay rate.

- An increase in the number of robots makes $\overline{\varepsilon}_{\max}$ increase because the number of overlappings rises. On the contrary, it makes the $\overline{\text{MDE}}$ decrease because it is more likely that a robot appears on the limit of the zero-error area. Nevertheless, the influence of this parameter is small in both measurements.

- The communication radius is the parameter with the biggest influence in the estimation. For this reason we represent the evolution of the two measurements in Figure 3.6. It can be seen that $\overline{\varepsilon}_{\max}$ decreases when the communication radius increases. This happens because a larger radius implies that more robots are in direct communication, reducing the error drastically. On the contrary, the $\overline{\text{MDE}}$ increases with the communication radius, since the production of the robots that are in direct communication does not induce an estimation error.

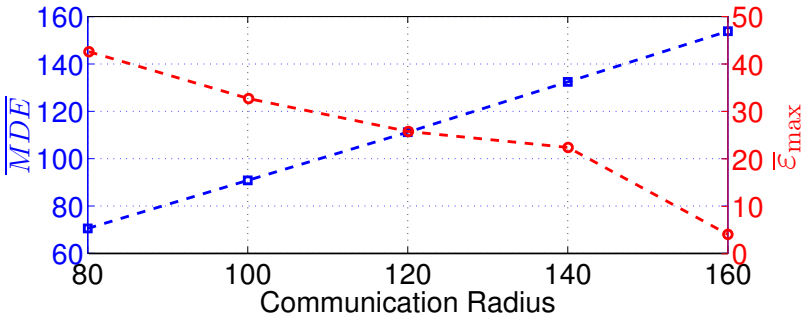


Figure 3.6: Evolution of the estimation with the communication radius.

- The influence of the maximum production is very important since it directly modifies the estimation error. However, as well as the decay, it has no effect in the distance to the error.

- The coverage radius makes $\overline{\varepsilon}_{\max}$ increase since the bigger the radius is, the more overlappings occur. The $\overline{\text{MDE}}$ decreases with this radius. If two robots are not neighbors, a larger coverage radius makes the $\overline{\text{MDE}}$ that they produce on each other shorter.

Although it cannot be shown graphically for all the trials, it should be noted that both the boundedness of the estimation error and the correct estimation in the characterized areas hold at all times for all the simulations that we have carried out.

These simulation results are in concordance with the theoretical results of the chapter: the local estimation is equal to the global map in the characterized areas and bounded in the rest of the domain.

3.6 Conclusions

In this chapter we have presented an algorithm to construct an estimation of the global coverage map with only local information received from neighboring robots. We have formulated the problem in discrete time so as to deal with the discrete nature of the communications in a distributed system. This algorithm has been proven to be accurate when calculating the estimation. We have proved that the estimation is equal to the global map inside the zero-error area of each robot when the motion of the robot satisfies a condition. Moreover, we have characterized the area in which zero error is guaranteed for any velocity. Additionally, we have established bounds on the estimation error for the rest of the environment. Finally, we have carried out different simulations to validate the proposed contributions.

The estimation algorithm represents the stepping stone for a distributed persistent coverage solution. Using our estimation procedure, any kind of strategy for the planning or control of the positions of the robots can be used used with formal guarantees of accuracy in the coverage value. For this reason, in the following chapter we develop a motion control law that builds upon the correctness of the estimation and the characterized regions.

Chapter 4

A Distributed, Partition-Based Solution with Feedback Control for Continuous Environments

In this chapter we propose a distributed feedback control law for the persistent coverage of a continuous environment. We introduce a new function to determine the potential improvement of the coverage at each point of the environment. The robots compute a Voronoi partition that is updated over time and calculate, using the values of this metric in their own regions, a control input that combines two actions. The first one maximizes the improvement. The second one drives the robots to the points with the highest improvement, which the robots are capable of finding locally inside their partition. The strategy is analyzed by means of simulations and discussed in detail at the end of the chapter.

4.1 Introduction

In a persistent coverage application, the motion of the robots must be controlled or planned according to the coverage of the environment, either the actual coverage when using motion controllers or online planning or the expected coverage when planning trajectories a priori. Thanks to the estimation algorithm presented in the previous chapter, the robots are able to accurately retrieve the coverage level in their surroundings in a distributed manner. From this point of view, a local controller based on this information appears as the best fit to control the motion of the robots. However, since these types of controllers suffer from local minima, an additional strategy has to be used to avoid getting stuck at them.

In this chapter, we propose a distributed strategy to control the motion of the robots that combines a gradient term and a goal-oriented term. Both terms are based on a new metric, called *Improvement Function*, that characterizes how profitable it is, in terms of coverage, to locate a robot at a specific position. The gradient term pushes the robots in the direction that improves the coverage the most, i.e., the locally-optimal direction. To avoid local minima and we also build an algorithm that allows each robot to locally determine the points with the highest improvement value. These points become goals that it has to reach

and the goal-oriented term drives the robot to them. The combination of both terms avoids revisiting well-covered areas and drives the robots towards uncovered areas.

In this chapter we consider the same formulation as in the previous one and, therefore, we omit the problem formulation section and refer the reader to Section 3.2. The remainder of the chapter is structured as follows. The notion of improvement function is proposed in 4.2 along with a distributed partition of the environment, preliminaries for Section 4.3 where we introduce the motion law to control the movement of the robots. The approach is discussed in Section 4.5 and its performance is analyzed in simulations in Section 4.4. Finally, Section 4.6 gathers the conclusions of this work.

4.2 Improvement Function and Environment Partition

In this section we introduce a new metric, the *Improvement Function*, that characterizes how profitable it is to locate a robot at a specific position in terms of the coverage level, and divide the environment using a distributed Voronoi partition, that provides each robot with a safe region where there is no interference with other robots. Using this metric and inside this region, each robot will be able to safely determine its own motion action.

Improvement Function

An approach to control the motion of the robots is to define goals to which the robots must move. These goals could be selected randomly according to a sampling-based motion planning algorithm [135]. However, for the distributed coverage problem, these goals must be points that provide the maximum improvement of the coverage according to the local information of the robots. To find them, we introduce a new metric that characterizes how profitable it is to locate a robot at a specific position, $\mathbf{p} \in \mathbf{Q}$, in terms of the coverage level estimated by the robots. We call this metric *Improvement Function*,

$$M_i(\mathbf{p}, k) = \frac{\displaystyle\int_{\mathbf{\Omega}_i(\mathbf{p})} \frac{Z^* - Z_i(k)}{Z^*}\, \Phi(\mathbf{q})\, \alpha_i(\mathbf{q}, \mathbf{p})\, \mathrm{d}\mathbf{q}}{\displaystyle\int_{\mathbf{\Omega}_i(\mathbf{p})} \Phi(\mathbf{q})\, \alpha_i(\mathbf{q}, \mathbf{p})\, \mathrm{d}\mathbf{q}}. \tag{4.1}$$

Recall that $\Phi(\mathbf{q}) \in (0, 1]$ is a function that represents the importance of covering each point. The *Improvement Function* computes the integral of the normalized coverage error $\big(Z^* - Z_i(k)\big)/Z^*$ weighted with the production of a robot located in $\mathbf{p}$, $\alpha_i(\mathbf{q}, \mathbf{p})$, and normalizes it to obtain a value independent of the production function. Note that $M_i(\mathbf{p}, k) \leq 1$ by definition and that it has a value between 0 and 1 when the coverage area around the point $\mathbf{p}$ is slightly covered and below zero when it is over-covered.

According to the previous definition, we will show that this function provides a crucial information for the control of the motion of the robots and even for the planning of trajectories for them. For instance, the point where this function reaches its maximum represents the point in which a robot can improve the coverage the most and, therefore, it could be assigned to the robot as the goal to follow.

We also introduce the temporal evolution of the improvement to simplify its online computation and update:

$$M_i(\mathbf{p}, k) = dM_i(\mathbf{p}, k-1) + A_i(\mathbf{p}) - \frac{1}{B_i(\mathbf{p})} \int_{\mathbf{\Omega}_i(\mathbf{p})} \alpha(k)\, \Phi(\mathbf{q})\, \alpha_i(\mathbf{q}, \mathbf{p})\, \mathrm{d}\mathbf{q}, \tag{4.2}$$

where

$$A_i(\mathbf{p}) = \frac{1}{B_i(\mathbf{p})} \int_{\mathbf{\Omega}_i(\mathbf{p})} \big(1 - d(\mathbf{q})\big)\, \Phi(\mathbf{q})\, \alpha_i(\mathbf{q}, \mathbf{p})\, \mathrm{d}\mathbf{q}$$

and

$$B_i(\mathbf{p}) = \int_{\mathbf{\Omega}_i(\mathbf{p})} \Phi(\mathbf{q})\, \alpha_i(\mathbf{q}, \mathbf{p})\, \mathrm{d}\mathbf{q}$$

are constants that can be calculated a priori. This temporary evolution calculates directly the improvement at each time as a function of its previous value and the coverage of the robots. It is interesting to highlight that the evolution of the improvement is the opposite of the evolution of the coverage. The improvement increases constantly with time due to the decay in $A_i(\mathbf{p})$ and decreases when the robots are producing an increase of the coverage level around each point.

Distributed Partition of the Environment

The selection of improvement maxima as goals in any part of the environment may lead to long and unnecessary shifts, redundancy or even collisions. Additionally, each robot may not be able to find the actual maximum improvement since (4.1) only uses its local estimation, $Z_i(k)$, that, although bounded, may have errors in regions far away from the robot. The solution to these problems is to assign particular regions to the robots in which they are responsible for the coverage. This can be achieved partitioning the environment in a distributed fashion. In particular, we choose a distributed Voronoi tesellation. It is a well-known technique that associates to each robot its closest points of the environment:

$$\mathbf{V}_i^*(k) = \{\mathbf{q} \in \mathbf{Q} \,|\, \|\mathbf{p}_i(k) - \mathbf{q}\| \leq \|\mathbf{p}_j(k) - \mathbf{q}\|\, \forall j \neq i\}.$$

The estimation of the coverage is always guaranteed to be exact within the zero-error area of the robots when their motion satisfies (3.11) and, since the size and shape of this region is constant, we reduce the Voronoi region to the intersection with this area, resulting in the r-disk Voronoi partition [2]:

$$\mathbf{V}_i(k) = \mathbf{V}_i^*(k) \cap \mathbf{\Omega}_i^z(k).$$

This region, though little conservative in some cases, presents several advantages:

1. It has been successfully applied in solutions to the static and dynamic coverage problems [6].

2. It prevents the robots from selecting as goals points in which the estimation is not exact, since inside $\mathbf{\Omega}_i^z(k)$ the estimation is always correct.

3. It overcomes the problem of long shifts, since each robot chooses goals only in its proximity.

4. It inherently prevents collisions in convex environments, since each robot has an exclusive region.

A small limitation of this partition occurs when the motion of the robots is unrestricted. In that case, a region of radius $\mathbf{r}_\mathbf{z}^*$ should be used to guarantee a correct estimation inside the r-disk Voronoi cell but depending on the parameters of the system, $\mathbf{r}_\mathbf{z}^*$ might be small and lead to a small partition. However, as simulation results in Section 3.5 show, the motion restriction is not often violated and the estimation algorithm is able to recover from it.

It should be noted that any other partition method such as power diagrams [16] or K-Means clustering [136] could be used. The partitions could also be combined with some task assignment procedure [137] to consider in the problem the optimization of some global metric, e.g., total distance walked by the robots, at the expense of some additional coordination. In fact, the selection of the distributed partition method is independent of the rest of the elements considered in the distributed strategy to coordinate the robots.

4.3 Feedback Controller

In this section we introduce the motion controller that combines the locally-optimal direction of movement, calculated as the gradient of the improvement function, with the direction to poorly covered areas that maximize the improvement of the coverage.

The idea behind this combination is twofold. The path to the goal may go through well- or over-covered areas, that can deteriorate the overall performance, and the gradient-based term helps to avoid these and move across areas where the coverage level is under the objective. Alternatively, the gradient term on his own may lead to local minima of the coverage and get stuck temporarily there, and it only relies on the coverage at the position of the robot, discarding the information in the rest of the partition. Therefore, the goal-directed term allows the robots to avoid local minima and to move to more profitable regions.

4.3.1 Local, gradient-based control

As mentioned before, the locally-optimal direction of movement is calculated as the gradient of the *Improvement Function* with respect to the robot position. To develop such gradient we omit the dependencies with $\mathbf{q}$ for simplicity and we use $\mathbf{p}_i \equiv \mathbf{p}_i(k)$ and $\mathbf{\Omega}_i \equiv \mathbf{\Omega}_i(\mathbf{p}_i)$. The first step is to derive the quotient in Equation (4.1) particularized for the position of the robot, $\mathbf{p}_i(k)$:

$$\nabla_{\mathbf{p}_i} M_i(\mathbf{p}_i, k) = \frac{\nabla_{\mathbf{p}_i} B_1 \cdot B_2 - B_1 \cdot \nabla_{\mathbf{p}_i} B_2}{B_2^2} \tag{4.3}$$

with

$$B_1(k) = \int_{\mathbf{\Omega}_i} \frac{Z^* - Z_i}{Z^*} \, \Phi \, \alpha_i(k) \, \mathrm{d}\mathbf{q}$$

and

$$B_2(k) = \int_{\mathbf{\Omega}_i} \Phi\, \alpha_i(k)\, \mathrm{d}\mathbf{q}.$$

If we define

$$\alpha_j^i(k) = \max\big(Z_j^{com}(k) - dZ_i(k-1), 0\big),\ \forall\, \mathbf{q} \in \mathbf{\Omega}_i(k), \tag{4.4}$$

the estimation algorithm from (3.1), (3.2a) and (3.4) gives

$$Z_i(k) = dZ_i(k-1) + \alpha_i(k) + \sum_{j \in N_i} \alpha_j^i(k), \forall\, \mathbf{q} \in \mathbf{\Omega}_i. \tag{4.5}$$

Applying the Leibniz integral rule to the term $\nabla_{\mathbf{p}_i} B_1$, we obtain

$$\begin{aligned}
\nabla_{\mathbf{p}_i} B_1 &= \int_{\mathbf{\Omega}_i} \frac{\Phi}{Z^*} \nabla_{\mathbf{p}_i} \left(Z^* \alpha_i(\mathbf{p}_i) - \Big(dZ_i(k-1) + \alpha_i(\mathbf{p}_i) + \sum_{j \in N_i} \alpha_j^i(k) \Big) \alpha_i(\mathbf{p}_i) \right) \mathrm{d}\mathbf{q} \\
&= \int_{\mathbf{\Omega}_i} \frac{\Phi}{Z^*} \Big(Z^* \nabla_{\mathbf{p}_i} \alpha_i(\mathbf{p}_i) - dZ_i(k-1) \nabla_{\mathbf{p}_i} \alpha_i(\mathbf{p}_i) - \\
&\qquad \sum_{j \in N_i} \alpha_j^i(k) \nabla_{\mathbf{p}_i} \alpha_i(\mathbf{p}_i) - 2\alpha_i(\mathbf{p}_i) \nabla_{\mathbf{p}_i} \alpha_i(\mathbf{p}_i) \Big)\, \mathrm{d}\mathbf{q} \\
&= \int_{\mathbf{\Omega}_i} \frac{\Phi}{Z^*} \nabla_{\mathbf{p}_i} \alpha_i(\mathbf{p}_i) \Big(Z^* - dZ_i(k-1) - \sum_{j \in N_i} \alpha_j^i(k) - 2\alpha_i(\mathbf{p}_i) \Big)\, \mathrm{d}\mathbf{q}
\end{aligned} \tag{4.6}$$

Substituting

$$A(k) = Z^* - dZ_i(k-1) - \sum_{j \in N_i} \alpha_j^i(k) - 2\alpha_i(k), \tag{4.7}$$

we have

$$\nabla_{\mathbf{p}_i} B_1 = \int_{\mathbf{\Omega}_i} \frac{\Phi}{Z^*} \nabla_{\mathbf{p}_i} \alpha_i(\mathbf{p}_i) A(k)\, \mathrm{d}\mathbf{q}.$$

On the other hand we have

$$\nabla_{\mathbf{p}_i} B_2 = \int_{\mathbf{\Omega}_i} \Phi \nabla_{\mathbf{p}_i} \alpha_i(\mathbf{p}_i)\, \mathrm{d}\mathbf{q}.$$

Introducing this equation and (4.6) in (4.3), the gradient of the *Improvement Function* results in:

$$\begin{aligned}
\mathbf{u}_i^{grad}(k) &= \nabla_{\mathbf{p}_i} M_i\big(\mathbf{p}_i(k), k\big) \\
&= \frac{\displaystyle\int_{\mathbf{\Omega}_i} \frac{\Phi}{Z^*} \nabla_{\mathbf{p}_i} \alpha_i(k) A(k)\, \mathrm{d}\mathbf{q}}{\displaystyle\int_{\mathbf{\Omega}_i} \Phi \alpha_i(k)\, \mathrm{d}\mathbf{q}} - \frac{\displaystyle\int_{\mathbf{\Omega}_i} \Phi \frac{Z^* - Z_i(k)}{Z^*} \alpha_i(k)\, \mathrm{d}\mathbf{q} \int_{\mathbf{\Omega}_i} \Phi \nabla_{\mathbf{p}_i} \alpha_i(k)\, \mathrm{d}\mathbf{q}}{\displaystyle\left(\int_{\mathbf{\Omega}_i} \Phi \alpha_i(k)\, \mathrm{d}\mathbf{q} \right)^2}.
\end{aligned} \tag{4.8}$$

This gradient represents how the *Improvement Function* changes inside the coverage area of robot i due to small variations of its position and it gives the direction of the motion of

the robot that maximizes the increase of the improvement, i.e., the direction in which the coverage would be most improved.

The use of this kind of gradient descent solution is usually applied to centralized systems [57]. Fortunately, our distributed estimation algorithm of Section 3.3 allows us to use the same gradient-based controller as a centralized system to obtain the same solution because each robot has an accurate estimation of the global map in the area required for the computation of the gradient.

Proposition 4.3.1. *The response of the gradient-based motion controller from* (4.8) *is the same for our distributed system as for a centralized system with*

$$B(k) = Z^* - dZ(k-1) - \sum_{j\in\{1,\dots,N\}} \alpha_j(k) - \alpha_i(k) \tag{4.9}$$

instead of $A(k)$ *and* $Z(k)$ *instead of* $Z_i(k)$.

Proof. In the first place we demonstrate that

$$\sum_{j\in N_i(k)} \alpha_j(k) = \sum_{j\in N_i(k)} \alpha_j^i(k), \quad \forall\, \mathbf{q} \in \mathbf{\Omega}_i(k). \tag{4.10}$$

According to the definition of the map-to-communicate from (3.1),

$$\alpha_j(k) = Z_j^{com}(k) - dZ_j(k-1). \tag{4.11}$$

Inside $\mathbf{\Omega}_i(k) \cap \mathbf{\Omega}_j^r(k)$, Theorem 3.4.4 states that $Z_j(k) = Z(k) = Z_i(k)$. Then,

$$\alpha_j^i(k) = Z_j^{com}(k) - dZ_i(k-1) = \alpha_j(k), \qquad \forall\, \mathbf{q} \in \mathbf{\Omega}_i(k) \cap \mathbf{\Omega}_j^r(k). \tag{4.12}$$

In $\mathbf{\Omega}_i(k) \setminus \mathbf{\Omega}_j^z(k)$, according to Theorem 3.4.7, $Z_j(k-1) \leq Z(k-1) = Z_i(k-1)$. Then, we have

$$Z_j^{com}(k) - dZ_i(k-1) \leq 0 \implies \alpha_j^i(k) = 0 = \alpha_j(k), \tag{4.13}$$

completing the first part of the proof.

If we now introduce 4.10 in 4.9 we have

$$B(k) = Z^* - dZ(k-1) - \sum_{j\in N_i(k)} \alpha_j^i(k) - \sum_{j\notin N_i(k)} \alpha_j(k) - \alpha_i(k). \tag{4.14}$$

Since the integrals in 4.8 are only computed inside $\mathbf{\Omega}_i(k)$ and in this region $\alpha_j(k) = 0$, $\forall\, j \notin N_i(k)$, $j \neq i$, then we have $\sum_{j\notin N_i(k)} \alpha_j(k) = \alpha_i(k)$. Therefore,

$$B(k) = Z^* - dZ(k-1) - \sum_{j\in N_i(k)} \alpha_j^i(k) - 2\alpha_i(k) = A(k). \tag{4.15}$$

Recalling that in $\mathbf{\Omega}_i(k)$ Theorem 3.4.4 proves that $Z_i(k) = Z(k)$ since $\mathbf{\Omega}_i(k) \subset \mathbf{\Omega}_i^r(k)$, the proof is complete. □

This proposition demonstrates that with our proposal any robot can calculate its own action using only local information and obtain the same results as a centralized system.

4.3.2 Goal-oriented control

The goal-oriented control aims to direct the robots to areas where the *Improvement Function* is maximized. The direction in which each robot moves toward its goal is

$$\mathbf{u}_i^{goal}(k) = \mathbf{g}_i^*(k) - \mathbf{p}_i(k), \tag{4.16}$$

where $\mathbf{g}_i^*(k)$ is the goal of robot i at time k.

In accordance with the *Improvement Function* (4.1), the goal should be the point $\mathbf{q}$ where it reaches its maximum value inside the Voronoi region:

$$\mathbf{g}_i^*(k) = \underset{\mathbf{q}\in\mathbf{V}_i(k)}{\operatorname{argmax}} M_i(\mathbf{q}, k). \tag{4.17}$$

However, this point may not exist or may not be unique, for instance when there are several points uncovered, or may change every iteration, leading to undesirable oscillations or cyclic behaviors. To cope with these problems we propose an algorithm that handles several goals. The key idea of this algorithm is to maintain a list of possible goals, $\mathbf{L}_i(k)$, that is updated every iteration and from which the current goal is selected.

In our solution, any goal $\mathbf{g}_i \in \mathbf{L}_i(k)$ must be inside the Voronoi partition,

$$\mathbf{g}_i \in \mathbf{V}_i(k), \tag{4.18}$$

and must be under-covered,

$$M_i(\mathbf{g}_i, k) > 0. \tag{4.19}$$

Therefore, the first step is to check if the goals from the list of the previous iteration, $\mathbf{g}_i \in \mathbf{L}_i(k-1)$, are still feasible, including if they have been reached:

$$\mathbf{g}_i \text{ is feasible} \iff M_i(\mathbf{g}_i, k) > 0 \wedge \mathbf{g}_i \in \mathbf{V}_i(k) \wedge \|\mathbf{g}_i - \mathbf{p}_i(k)\| > \mathcal{E}, \tag{4.20}$$

where $\mathcal{E}$ is the distance at which a goal is considered reached. Whenever a goal stops satisfying these criteria, it is removed from the current list $\mathbf{L}_i(k)$.

Next, we define a region in which new possible goals are searched as

$$\mathbf{R}_i(k) = \mathbf{V}_i(k) \setminus \boldsymbol{\Omega}_i\big(\mathbf{p}_i(k)\big) \setminus \boldsymbol{\Omega}_i(\mathbf{g}_i), \tag{4.21}$$

that includes all points of the Voronoi region that are not in the coverage area of the robot centered either at $\mathbf{p}_i(k)$ or at any $\mathbf{g}_i \in \mathbf{L}_i(k)$. Note that in the initialization step for $\mathbf{L}_i(1)$, the search region is $\mathbf{R}_i(1) = \mathbf{V}_i(1)$, and that each new possible goal found in the search region meets the distance requirement

$$\mathbf{g}_i^2 \notin \boldsymbol{\Omega}_i(\mathbf{g}_i^1), \tag{4.22}$$

with $\mathbf{g}_i^1$, $\mathbf{g}_i^2$ any pair of possible goals in $\mathbf{L}_i(k)$.

Each robot iteratively looks for new possible goals inside $\mathbf{R}_i(k)$, that are global or local maxima or quasi-maxima of the *Improvement Function*. The set of these points is

$$\mathrm{P}_i = \big\{\mathbf{p} \in \mathbf{R}_i(k) \,|\, M_i(\mathbf{p}, k) \geq M_i(\mathbf{p} + \epsilon, k)\big\}, \tag{4.23}$$

which can be found using state-of-the-art methods such as pattern search. Such $\mathbf{p} \in \mathrm{P}_i$ must also satisfy $M_i(\mathbf{p}, k) > 0$. Nevertheless, from this set only the nearest point to the robot is selected as a new possible goal:

$$\mathbf{g}_i = \arg\min_{\mathbf{p}\in \mathrm{P}_i}(\|\mathbf{p} - \mathbf{p}_i(k)\|). \tag{4.24}$$

This point $\mathbf{g}_i$ is the most interesting of the current search region both in terms of the *Improvement Function* and in terms of the time until it can be covered. Thus, it is appended to the current list of goals, $\mathbf{L}_i(k)$, and the points close to it are eliminated from the search region to keep satisfying (4.22):

$$\mathbf{R}_i(k) = \mathbf{R}_i(k) \setminus \mathbf{\Omega}_i(\mathbf{g}_i). \tag{4.25}$$

The search is repeated until $\mathbf{R}_i(k)$ is empty or there are no more points with a positive value of the *Improvement Function* and eventually, each robot has an updated list of possible goals, $\mathbf{L}_i(k)$.

The selection of the goal to follow from this list can be done in accordance with several criteria such as the value of the *Improvement Function*, $M_i(\mathbf{g}_i, k)$, the distance to the current position of the robot or the distance to the neighbors. This means that, for instance, the most profitable goal may be the one with the highest value of $M_i(\mathbf{q}, k)$ or the nearest to the robot and it is clear then that the definition of profitability of a goal depends on the application. Therefore, the selection of the current goal from the list can be generalized as

$$\mathbf{g}_i^*(k) = \underset{\mathbf{g}_i\in\mathbf{L}_i(k)}{\operatorname{argmin}} f(M_i(\mathbf{g}_i, k), \|\mathbf{g}_i - \mathbf{p}_i\|, \|\mathbf{g}_i - \mathbf{p}_j\|), \tag{4.26}$$

where $j \in N_i(k)$ and $f(\cdot)$ represents the profitability of an objective from the list, thus we call it *Profitability Function*. In particular, in this chapter we use

$$f(\mathbf{g}_i, k) = -M_i(\mathbf{g}_i, k). \tag{4.27}$$

However, to give a more general idea of this function in the simulation section 4.4 we provide a comparison of several different alternatives.

Finally, in Algorithm 2 we summarize this method to search for goals.

4.3.3 Motion Control Law

The overall motion control law that we propose gathers both previous actions, the local, gradient-based, $\mathbf{u}_i^{grad}(k)$, and the goal-oriented, $\mathbf{u}_i^{goal}(k)$, as follows:

$$\mathbf{u}_i(k) = W_L(k)u_i^{\max}(k)\frac{W_L(k)\mathbf{u}_i^{grad}(k) + W_G(k)\mathbf{u}_i^{goal}(k)}{\|W_L(k)\mathbf{u}_i^{grad}(k) + W_G(k)\mathbf{u}_i^{goal}(k)\|}. \tag{4.28}$$

$u_i^{\max}(k)$ is bounded by Lemma 3.4.3 or by $u^{\max}$ and, with a little abuse of notation, $\mathbf{u}_i^{grad}(k)$ and $\mathbf{u}_i^{goal}(k)$ are the normalizations of the directions introduced in (4.8) and (4.16), respectively. The weight $W_L(k)$ of the gradient-based action is also used as a gain of the maximum velocity of the robot and is defined according to the *Improvement Function* from (4.1):

$$W_L(k) = 1 - \max\big(M_i\big(\mathbf{p}_i(k), k\big), 0\big), \tag{4.29}$$

Algorithm 2 *Selection of goals.*

Require:

1. Previous list of goals, $\mathbf{L}_i(k-1)$.
2. Voronoi region of the robot, $\mathbf{V}_i(k)$.
3. *Improvement Function*, $M_i(\mathbf{q}, k)$, $\forall\, \mathbf{q} \in \mathbf{V}_i(k)$.

$\mathbf{L}_i(k) = \mathbf{L}_i(k-1)$
For all $\mathbf{g}_i \in \mathbf{L}_i(k-1)$
  If $\mathbf{g}_i$ is not feasible (4.20),
    Remove $\mathbf{g}_i$ from $\mathbf{L}_i(k)$.
Define search region $\mathbf{R}_i(k)$ (4.21).
While $\mathbf{R}_i(k) \neq \emptyset$ and $\exists\, \mathbf{p} \in \mathbf{R}_i(k) \,|\, M_i(\mathbf{p}, k) > 0$
  Search for a new possible goal $\mathbf{g}_i \in \mathbf{R}_i(k)$ (4.23)-(4.24).
  Append $\mathbf{g}_i$ to $\mathbf{L}_i(k)$.
  Eliminate points near $\mathbf{g}_i$ from $\mathbf{R}_i(k)$ (4.25).
Select current goal $\mathbf{g}_i^* \in \mathbf{L}_i(k)$ (4.26).

where $0 \leq W_L(k) \leq 1$. When the coverage level is near to the objective in the coverage area of the robot, the importance of the local action tends to 1, to avoid revisiting areas already well covered. When the coverage level is poor, the local direction becomes less important since coverage is still needed. This weight (4.29) can also be seen as a velocity gain. It has the effect of slowing the robot down to keep covering its area when it has a low coverage level and speeding it up to quickly move to less covered areas when its coverage level is near the objective.

The importance of the goal-oriented action is

$$W_G(k) = M_i\big(\mathbf{g}_i(k), k\big), \tag{4.30}$$

where $0 \leq W_G(k) \leq 1$ and $\mathbf{g}_i(k)$ is the goal of robot i, described in Section 4.3.2. This weight increases the importance of the goal when the coverage area of the robot virtually located there is poorly covered and, vice versa, the goal becomes less important when it is well covered.

The motion control law that we propose avoids getting stuck thanks to the weights W_L and W_G. They have the effect that, when the coverage area of the robot is poorly covered, the robot moves slowly and directed to the goal and, when the area is well covered, it moves fast in the direction of the gradient. In the highly improbable case that both weighted actions cancel out, the robot does not move but keeps covering, making the weight of the gradient W_G change and, thus, breaking the equality.

4.4 Simulations

We present simulation results for the proposed solution to the persistent coverage problem. At the beginning, we show the behavior of the entire system using an illustrative example and afterwards, we provide an exhaustive evaluation of different alternatives to the

Profitability Function, the control of the motion of the robots and the whole approach to the problem using a Monte Carlo analysis.

The reference settings that we have used in these simulations are the same as in the previous chapter. In addition to those settings, the *Profitability Function* that we use is (4.27) and we select the number of robots that is able to produce more coverage than the total decay of the environment when it is at the desired level, i.e., we use the minimum number of robots that satisfy

$$\int_{\mathbf{Q}} \sum_{i\in\{1,\dots,N\}} \alpha_i(k)\mathrm{d}\mathbf{q} > \int_{\mathbf{Q}} (1-d)Z^*\mathrm{d}\mathbf{q}. \tag{4.31}$$

Although this is a sufficient condition for the robots to be able to reach the desired coverage level, it is not a requirement of the algorithm. If less robots were deployed they would try to keep the average coverage level as high as possible. Also note that, without loss of generality, we use the same coverage radius and the same coverage function for all the robots although they could be different [57] without modifying the global behavior of the system. In addition, we define the quadratic coverage error,

$$\tilde{\varepsilon}(k) = \int_{\mathbf{Q}} \left(Z^* - Z(k)\right)^2 \mathrm{d}\mathbf{q}, \tag{4.32}$$

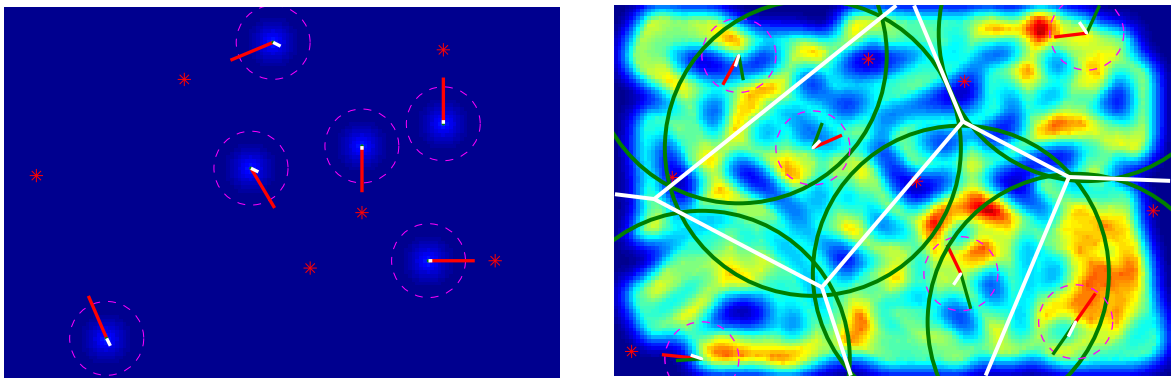
that is an interesting metric used in some of the following results. The simulations are implemented in Matlab, run on a laptop with an Intel Core i7 and the average iteration time for these settings is around 80ms.

4.4.1 Illustrative Example

This section presents an illustrative example of the simulations with the reference settings. In Figure 4.1 we depict the coverage map with the coverage areas of the robots and their motion control actions (4.28) for three different instants. One can see that the performance of the system is quite good. In fact, the average coverage level reached is 96.4 with a standard deviation of 21.3 that is, at most, 30 in the transient state. This supports that, although it is not theoretically guaranteed, not only most of the points reach the desired level at some time but also an average level near the objective is maintained over time.

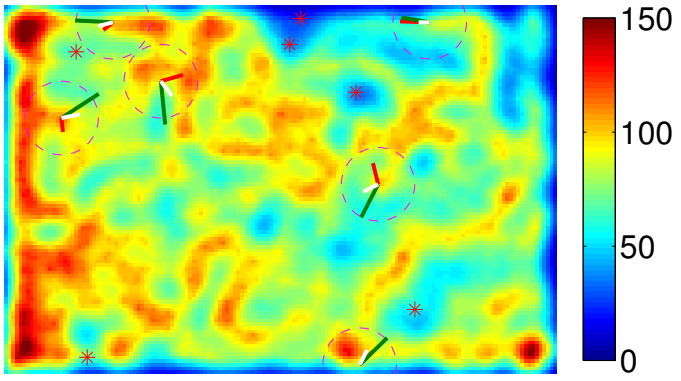
Figure 4.1a shows that in the beginning there is almost no movement since the environment is uncovered and the weight W_L, that is the velocity gain, is small. At this time, the goal-oriented term, $W_G(k)\mathbf{u}_i^{goal}(k)$, prevails. When the environment starts to be covered, the combination of both terms drives the robots (Figure 4.1b). In the end, when all the environment is covered, the term that has more influence is the gradient-based, $W_L(k)\mathbf{u}_i^{grad}(k)$, since there are few uncovered areas that are worth visiting (Figure 4.1c). In Figure 4.2 the weight of each action is depicted for one of the robots. It confirms that the goal-oriented term is more important in the transient while it is the gradient in the steady state and that the velocity gain W_L progressively increases with the coverage of the environment.

The trajectories for the first 200 instants of this simulation are shown in Figure 4.3. At start, the robots focus on their particular regions for three reasons: (i) they follow the goals from the initial list, that belong to the same region and all are poorly covered; (ii) the goal



(a) k = 1.

(b) k = 150.



(c) k = 300.

Figure 4.1: Example of simulation: coverage map with the coverage areas of the robots (magenta dashed circumferences) and their motion control actions for three different instants. Red lines represent the goal-oriented term, $W_G(k)\mathbf{u}_i^{goal}(k)$, magnified by 2.5 and point to the objectives that are represent by red asterisks. Green lines represent the gradient-based term, $W_L(k)\mathbf{u}_i^{grad}(k)$, magnified by 2.5. The resulting action is represented in white. Green circumferences in (b) represent the zero-error areas of the robots and the thickest white lines, the Voronoi partition of the environment.

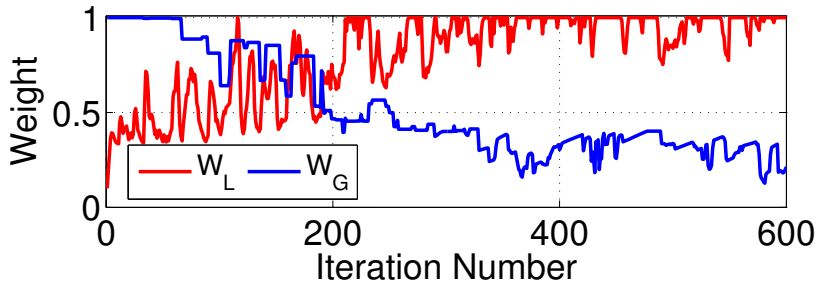


Figure 4.2: Evolution of the weights of the actions W_L and W_G for one of the robots.

term is more important in the control law and (iii) they move slowly since the velocity gain,

$W_L(k)$, is small due to the low coverage of the environment. As the algorithm proceed, the gradient term dominates the motion and the velocity is higher, making the covered regions of the robots overlap more and more. It is worth mentioning that for these settings, and in particular for this number of robots and communication radius, the network did not become disconnected in any of the carried-out simulations.

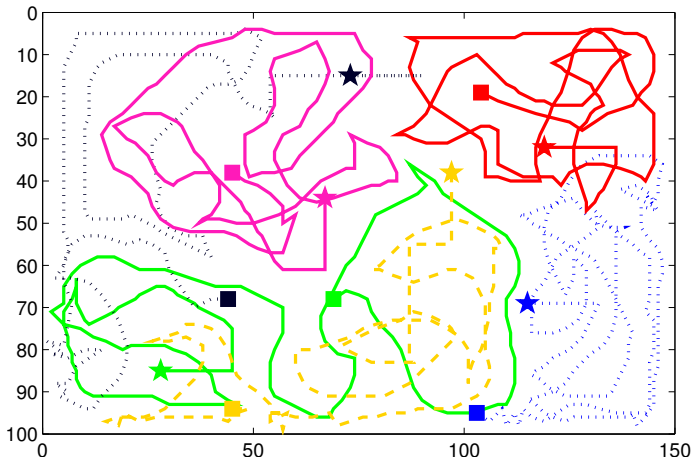


Figure 4.3: Trajectories followed by the 6 robots during the first 200 iterations. Initial positions are represented with stars and final positions with squares.

4.4.2 Profitability Function

As mentioned in Section 4.3.2, the *Profitability Function*, $f(\cdot)$, that allows the selection of the current goal from the list can be defined in a different manner depending on the application. In this section we compare several different alternatives to see the influence that they have in the global behavior of the system. The first one selects the point with the best value of the *Improvement Function*,

$$f_1(\mathbf{g}, k) = -M_i(\mathbf{g}, k). \tag{4.33}$$

The second one is based on the minimum distance to the objectives and is aimed at reducing the overall motion,

$$f_2(\mathbf{g}, k) = \|\mathbf{g} - \mathbf{p}_i(k)\|. \tag{4.34}$$

In the third place we weight the previous two, varying the importance of the *Improvement Function* and the distance:

$$f_3(\mathbf{g}, k) = -0.7\, M_i\big(\mathbf{g}, k\big) + 0.3\, \|\mathbf{g} - \mathbf{p}_i(k)\|, \tag{4.35}$$

$$f_4(\mathbf{g}, k) = -0.5\, M_i\big(\mathbf{g}, k\big) + 0.5\, \|\mathbf{g} - \mathbf{p}_i(k)\|, \tag{4.36}$$

$$f_5(\mathbf{g}, k) = -0.3\, M_i\big(\mathbf{g}, k\big) + 0.7\, \|\mathbf{g} - \mathbf{p}_i(k)\|. \tag{4.37}$$

We also include in the comparison other selection methods that may not work properly, to have a better idea of the performance of all of them. The sixth alternative is to select the

goal with the lowest value of the *Improvement Function*, i.e., the point of the list that causes the smallest improvement of the coverage,

$$f_6(\mathbf{g}, k) = M_i(\mathbf{g}, k). \tag{4.38}$$

Additionally, we use a random selection of the goal from the list that we refer to as $f_7(\mathbf{g}, k)$. Finally, we also compare these methods with a random selection of the goal in the Voronoi region of the robot. Instead of constructing a list of goals we directly select a random point of the region as the goal to follow. We refer to this as $f_8(\mathbf{g}, k)$.

From the simulations of these eight alternatives with the reference settings we obtained the following results. Firstly we find that the standard deviation of the coverage level averaged over the runs is very significant to compare the options as can be seen in Figure 4.4. The average coverage level is the same in all the cases since the production of the robot does not change, but the standard deviation gives an idea of how homogeneous is the coverage of the environment. The figure shows that the lowest deviation is achieved with f_1, f_3, f_4 and f_5, i.e., with the *Profitability Functions* that look for the goal with the maximum value of the improvement. These selections lead to a more homogeneous coverage of the environment than the others in the transient state.

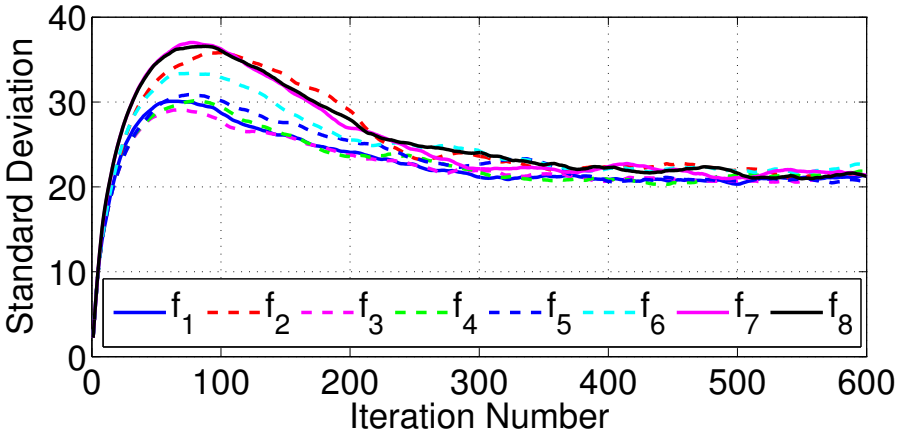
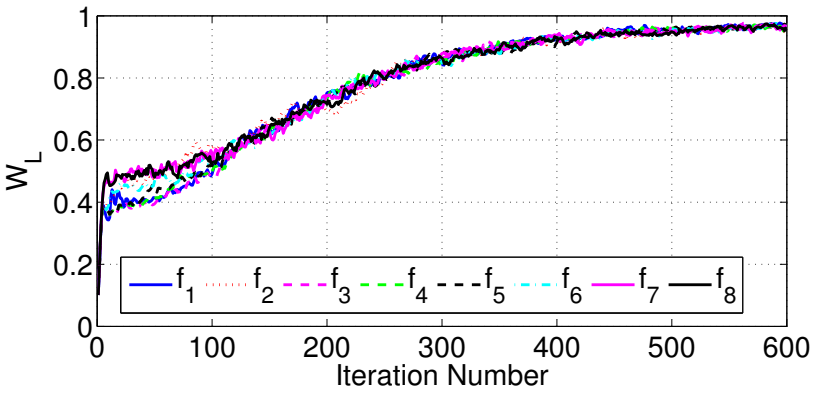


Figure 4.4: Average standard deviation of the coverage level for the different *Profitability Functions* and additional selection methods.

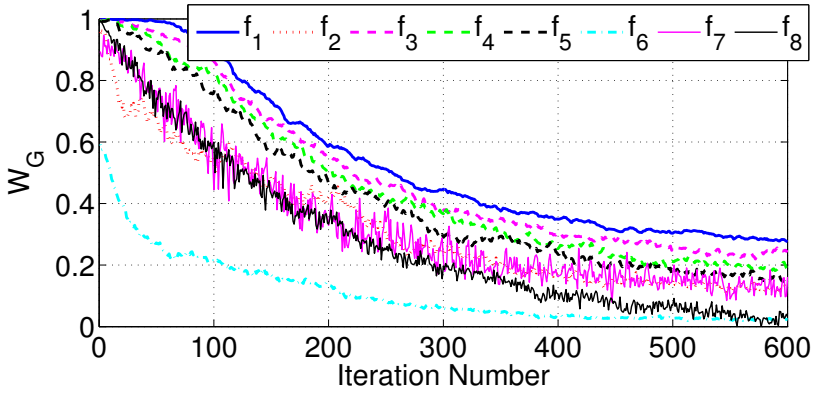
The Profitability Functions that look for the worst covered areas select goals that produce high values of W_G and not only direct the robots to the most profitable areas but also give more importance to the goal-oriented term in the control law. On the contrary, if the improvement of the selected goals is really low, as for f_6 or f_8, W_G tends to zero. In that case, the gradient term dominates the motion and the results tend to be similar to the ones obtained only with the gradient term. This is the reason why the performance differences between so different Profitability Functions are not larger and why all of them converge to the same standard deviation value in the steady state, because eventually the gradient term dominates the motion.

Figure 4.5 shows the average weights for the eight Profitability Functions. It can be seen in Figure 4.5a that the gradient weight is very similar for all of them except for those based

upon the *Improvement Function* in the first 100 iterations. In the end the weight of the gradient is almost 1 in all cases, supporting that all the alternatives converge to the same value of the standard deviation. However, the most important result is the influence of the Profitability Function in W_G, shown in Figure 4.5b. The chosen function clearly determines which term is more important in the control action in the transient: the goal-oriented term dominates in the beginning for f_1, f_3, f_4 and f_5 but for f_6 the gradient rules the motion from the start. This is the reason why f_1 or f_3 give much better results than the other alternatives in the transient state.



(a) W_L.



(b) W_G.

Figure 4.5: Evolution of the weights for the different Profitability Functions.

Eventually, Figure 4.5a also gives an idea of the distances traveled by the robots since W_L acts as a velocity and the total traveled distance is proportional to the integral of W_L over time. Intuitively, f_2 should lead to the shortest distance but in practice it does not because the nearest goals may not be the most profitable and then the robots move faster and travel more distance. The alternatives that lead to shorter distances are f_1 and f_3 but the differences are under the 1.2% since the gradient weight is very similar for all of them.

Finally we compare the performance of the alternatives using the quadratic coverage error of the system, $\tilde{\varepsilon}(k)$. Upon this metric we compute the relative coverage error as the quadratic coverage error of each alternative relative to the best one. In this case we calculate $\big(\tilde{\varepsilon}(k) - \tilde{\varepsilon}_{f_1}(k)\big)/\tilde{\varepsilon}_{f_1}(k)$ where $\tilde{\varepsilon}_{f_1}(k)$ is the quadratic coverage error obtained with f_1, so as

to obtain a representative measure in percentage for each alternative (Figure 4.6). It shows that the *Profitability Functions* that are not build upon the *Improvement Function* perform up to a 20% worse.

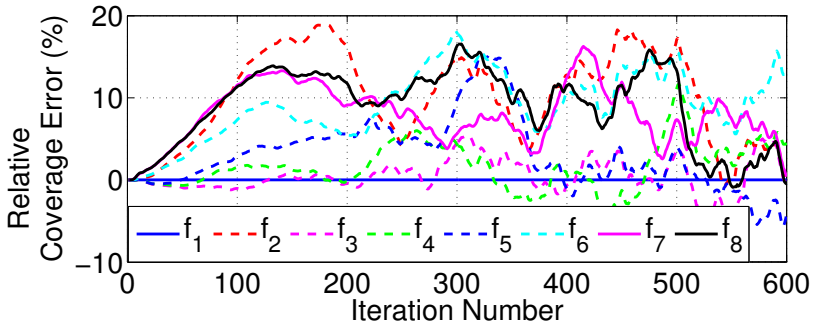


Figure 4.6: Relative quadratic coverage error with respect to f_1 for the different *Profitability Functions* and additional selection methods.

If we average these relative errors over time (Table 4.1), we can have a single measure of the differences that the selection methods cause on the performance of the system.

f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
0%	10.1%	0.5%	1.7%	3.1%	9.2%	8.0%	9.1%

Table 4.1: Relative coverage error with respect to f_1 averaged over time.

The conclusion from this results is that, although the *Profitability Function* is important conceptually, using a criterion that is based on the maximum *Improvement Function* of the coverage, such as f_1 or f_3, leads to a coverage around a 10% better than using any other selection that does not take it into account.

4.4.3 Motion Control

In the third place we provide results regarding the control of the motion of the robots. To evaluate the influence of the restriction from Equation (3.11) in terms of the motion of the robots, we ran a Monte Carlo simulation with the reference settings and concluded that, in practice, this restriction in the motion is not very limiting. In 20 runs of 600 iterations, it was only applied in the 6.4% of the cases and caused on them an average reduction of the motion of the 54.0%.

Next, we compare our motion control law with the performance of the gradient-term and the goal-oriented term separately. In Figure 4.7 it can be seen that the average standard deviation of the coverage level is greater when only one of the terms of the control law is used. This means that our motion control law produces a more homogeneous coverage in the environment. It is also remarkable that, in the steady-state, the control using only the goal-oriented term has a much higher deviation. This happens when the environment is

already well covered because the path to the objective produces over-covered regions. On the contrary, the behavior of the gradient-term separately is very similar to the entire law because this term prevails over the goal-oriented.

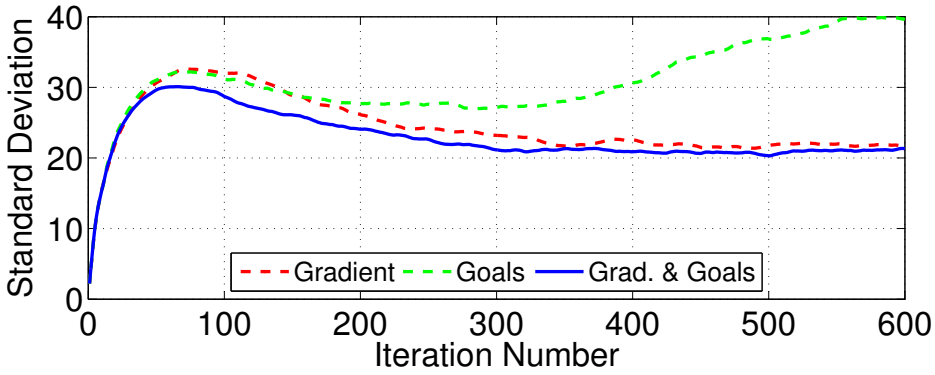


Figure 4.7: Average standard deviation of the coverage level for the different motion control laws.

We also represent in Figure 4.8 the quadratic coverage error of each term separately relative to the quadratic coverage error obtained with our motion control law, $\big(\tilde{\varepsilon}(k) - \tilde{\varepsilon}_{u_i}(k)\big)/\tilde{\varepsilon}_{u_i}(k)$. It shows that the goal-oriented term alone also produces a great increment of the coverage error at the end of the simulations while the gradient term does not.

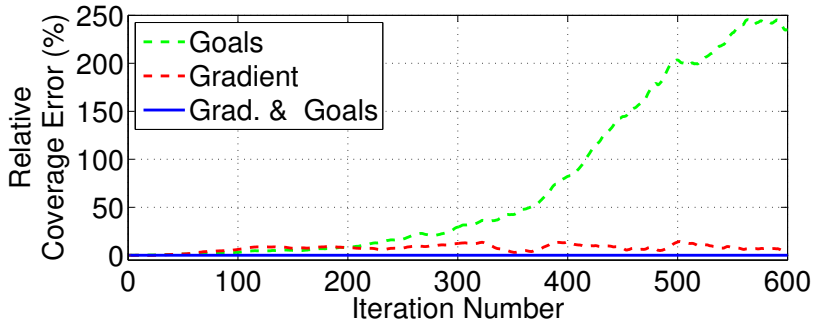


Figure 4.8: Relative quadratic coverage error with respect to (4.28) for the different motion control laws.

Table 4.2 demonstrates that, on average, the coverage with our motion control law performs a 7.5% better than only the gradient-based term and a 73.4% better than only the goal-oriented term.

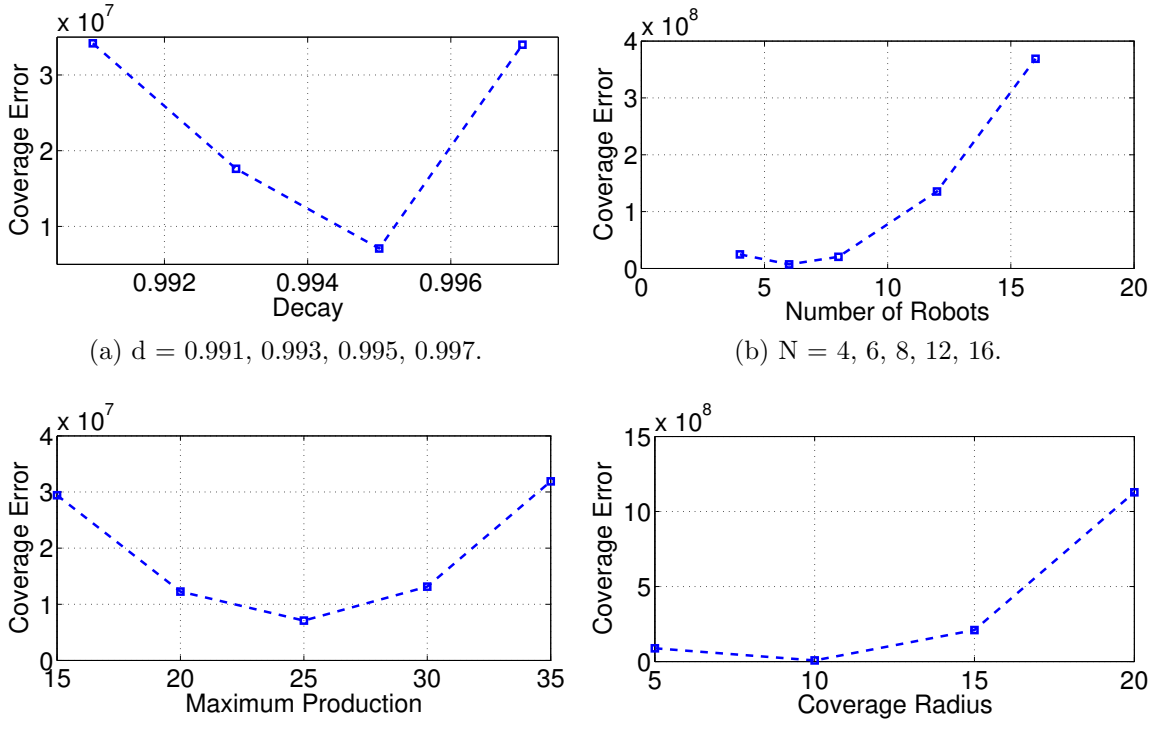
Gradient and Goals	Gradient	Goals
0%	7.5%	73.4%

Table 4.2: Relative coverage error with respect to (4.28) averaged over time.

4.4.4 Global Performance Analysis

In this final part we provide simulation results on the performance of our entire approach to the persistent coverage problem, that comprises the estimation algorithm and the motion control. Specifically, we study the influence of the parameters of the system in the quadratic coverage error of the steady-state, calculated for $k = 600$, and in the standard deviation of the coverage level.

In Figure 4.9 we present the evolution of the quadratic coverage error of the steady-state averaged over the runs for different values of the parameters. It can be seen that, in terms of this coverage error, there is an optimum value for each parameter when all the others are fixed and, when a different value is applied, the coverage error of the steady-state is higher. The reason is that higher values of the parameters increase the production of the system or decrease the deterioration of the coverage, leading to a coverage level of the environment greater than the objective. Similarly, lower values lead to an under-covered environment. In both cases, the quadratic coverage error is higher than in a well-covered environment according to its definition.



(a) d = 0.991, 0.993, 0.995, 0.997.

(b) N = 4, 6, 8, 12, 16.

(c) P = 15, 20, 25, 30, 35.

(d) $r^{cov} = 5, 10, 15, 20$.

Figure 4.9: Evolution of the average quadratic coverage error of the steady-state for different values of the parameters.

d	**0.991**	**0.993**	**0.995**	**0.997**	
$\overline{Z}$	56.9	72.0	96.4	140.5	
N	**4**	**6**	**8**	**12**	**16**
$\overline{Z}$	65.1	96.4	127.8	190.7	253.6
P	**15**	**20**	**25**	**30**	**35**
$\overline{Z}$	58.6	77.5	96.4	115.4	134.6
$\mathbf{r^{cov}}$	**5**	**10**	**15**	**20**	
$\overline{Z}$	25.0	96.4	212.1	368.4	

Table 4.3: Average coverage level of the steady-state.

These results are supported by the specific values of the average coverage level of the steady-state, gathered in Table 4.3. Since the robots are capable of increasing the coverage level, the average level of the environment increases with the number of robots, the maximum value of the production, P, and the coverage radius. Moreover, since higher decays produce lower decreases on the coverage level, the evolution with respect to this parameter is the same. An uncertainty on the value of the decay might lead to under- or over-covering the points of the environment, i.e., an inaccurate estimation of the rate would lead to a certain value in Figure 4.9a but the actual value might be different. This can be solved by correcting the estimation with the information from sensors mounted on the robots.

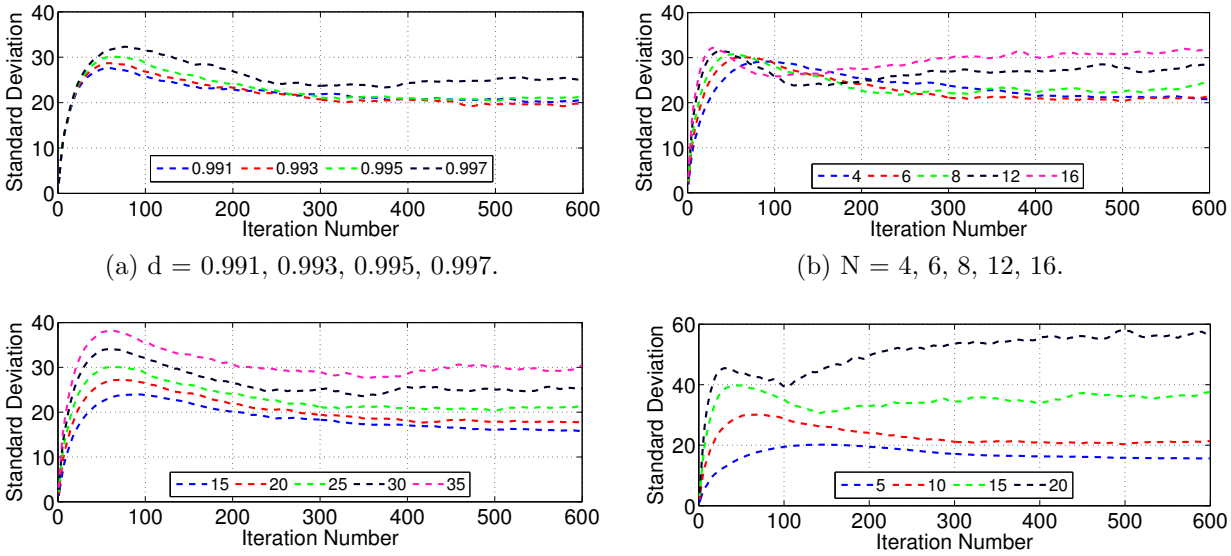
Note that we do not show the evolution with the communication radius since it is negligible due to the size of the Voronoi region with respect to the zero-error area and to the selection of the goals near the robot, as can be seen in Figure 4.1b.

We also evaluate the influence of all the parameters in the standard deviation of the coverage level. Figure 4.10 shows the average standard deviation for each value of the parameters. We can draw the following conclusions:

- Low values of the decay present a similar behavior, especially in the steady state. Only the greatest value shows a greater standard deviation for the transient and the stationary (Figure 4.10a).

- The influence of the number of robot in the transient is rather complex as can be seen in Figure 4.10b. However, in the end the more robots are used, the higher the standard deviation since the coverage action is not controlled.

- An increase in the maximum production makes the standard deviation of the coverage level increase since the increment of the coverage level at each time is higher and, therefore, the value of the most covered points with respect to the least covered ones (Figure 4.10c).



(a) d = 0.991, 0.993, 0.995, 0.997.

(b) N = 4, 6, 8, 12, 16.

(c) P = 15, 20, 25, 30, 35.

(d) r^{cov} = 5, 10, 15, 20.

Figure 4.10: Evolution of the average standard deviation of the coverage level for different values of the parameters.

- The deviation of the coverage level responds to the variation of the coverage radius (Figure 4.10d) in the same way as to P and this parameter is again the one that produces the greatest variation.

Although in most cases the lowest value of each parameter leads to the lowest deviation, it is essential that the environment reaches the objective and, thus, the average coverage level of the environment from Table 4.3 must be taken into account. It can be seen that the coverage level increases with all the parameters and that the optimal value according to this criterion is the same as for the coverage error.

4.5 Discussion

There are several aspects of the feedback control law that are worth highlighting and deserve a more detailed discussion:

1. The control law has been designed to be integrated with the estimation algorithm of the previous chapter. Together they form a complete distributed approach to the persistent coverage problem. Nevertheless, they can be used separately with other estimation or motion techniques.

2. There is no guarantee that the trajectories obtained with the proposed motion control law are optimal and, therefore, the coverage strategy may also be suboptimal. This is the price for obtaining a distributed, computationally cheap and not communication intensive solution. In any case, the simulations have shown the good performance of the system and the kindness of our approach.

3. It is known that gradient methods may lead to local optima and, therefore, the robots could end up stuck in one of these minima. However, there are several features of our approach that prevent this situation: i) we follow goals as well as the gradient and include the variable weights in the motion control law (4.28); ii) the coverage level outside the coverage area of the robots decreases when it is not covered and even if a robot reaches a local minimum, the area around it will decay and will require the robot to cover it; iii) if both weighted actions cancel out, the robot does not move, but keeps covering, making the weight of the gradient W_G change and, thus, breaking the equality. This does not imply that all the points are theoretically guaranteed to reach the objective, even though it should happen in practice.

4. Although the control law does not guarantee collision-free motion, the Voronoi partition prevents collisions as mentioned before, specially in convex environments. Additionally, state-of-the-art collision avoidance methods, such as potential fields, can be incorporated to the system and used in the case that the gradient term dominates the motion.

5. The discrete-time formulation of the problem requires the frequency of the estimation to be sufficiently high with respect to the maximum velocity and the coverage radius of the robots to avoid discretization problems. Since the actual production is continuous, the estimation must be run fast enough not to miss sensible information between two consecutive estimation instants. For instance, if a robot can move at 1 m/s with a coverage radius of 0.25 m, the frequency should be at least 10 Hz. On the contrary, if the coverage radius is 25 m, the frequency could be 1 Hz since the information loss due to the temporal discretization would be negligible.

6. The goal selection and motion control do not need to be run as fast as the estimation. In fact, there must be a trade-off between the quality of the solution and the computational cost in the sense that the more frequent the calculation is carried out, the better the final coverage is obtained and the higher computational cost is incurred. Regarding the order, the goal selection and motion step should follow the estimation step in order to calculate the actions with the latest information. In the event that a different sequence is used, the only implication is that the actions may lead to a worse coverage since they are calculated with non up-to-date information.

4.6 Conclusions

In this chapter we have introduced a motion control law that drives the robots to poorly covered areas while moving in the direction of the maximum improvement. The first part is achieved by following objectives, that are selected inside the Voronoi regions to avoid conflicts between robots. The second part is achieved using the local gradient of the newly presented *Improvement Function*. An important effort has been made to validate the proposed contributions and to evaluate the performance of the system. We have illustrated the simulations with an example, in which we have shown the main features of our approach, and

we have performed a thorough validation of the proposed algorithms and a detailed study of the influence of the parameters in its performance.

The proposed motion control law, though based on heuristics, addresses an NP-hard problem. With respect to previous approaches to persistent coverage based on motion controllers, it is fully distributed, accounts for the quality of a distributed estimation and shows good performance even though the coverage action is considered fixed. Despite all these advantages, a feedback control is in the end a naive planning with a temporal horizon of one step. The question that arises naturally is if it is possible to obtain better results by considering more natural paths to the goals. We explore this idea in the following chapter that, in addition, allows us to consider environments with some obstacles and even really complex environments.

Chapter 5

A Distributed, Finite-Horizon-Optimal Path Planning Solution for Continuous Environments with Obstacles

In this chapter we introduce a distributed solution for the coverage of a continuous environment with obstacles. We propose an algorithmic solution in which each robot locally finds the finite-horizon paths which are optimal in terms of coverage quality, to a set of potential goals inside its own Voronoi region. These paths are computed using a Fast Marching Method and keep a safety distance to obstacles. Between all the potential paths, the one that mostly improves the coverage along the whole path is followed. In addition, we propose a coverage action controller, locally computed and optimal, that makes the robots maintain the coverage level of the environment significantly closer to the objective. Simulations and real experiments support our solution.

5.1 Introduction

Persistent coverage applications with ground vehicles rarely occur in convex environments and the robots usually work in complex environments and surrounded by obstacles. The approach of the previous chapter may suffer in these environments since both terms may direct the robots to obstacles and even adding a term that repulses the robots from obstacles may not be enough to reach the destinations.

In addition, the accuracy of the estimation, the notion of improvement function and the definition of goals open the door to a different strategy that takes a step towards optimality. Still with the idea of following coverage goals, we can plan paths to them in order to maximize the coverage provided until they are reached instead of just controlling their motion actions.

For these reasons, in this chapter we propose a distributed algorithm that is capable of planning optimal paths in terms of improvement to many possible goals at the same time and actively select the one with the best accumulated improvement. The selection of goals

can be compared to discretized approaches. Nevertheless, our active search of the points that require the most coverage at each time allows us to cover continuous environments. Our path planning strategy is based on Fast Marching Methods (FMM), that have proved to be successful in real domestic spaces with high complexity [138]. We introduce in this method two cost functions that account for the coverage of the environment and for the obstacles and, consequently, the planned paths are optimal in terms of coverage and keep a safety distance to obstacles. This on board, fast computation of paths also allows the avoidance of unexpected obstacles as opposed to approaches that plan a priori, and the optimality of the paths outscores the use of local motion controllers. The paths are followed using a Dynamic Window Approach (DWA) [91]. Moreover, we introduce a coverage action controller that allows the robots to provide the optimal coverage at each point to maintain the desired coverage level. This is essential in applications such as heating or watering, to avoid over-heating or flooding, respectively, or to save power in the case of vacuuming or cleaning. In fact, it provides a significant improvement with respect to the approach in the previous chapter where the coverage action was fixed. We show this in simulations and demonstrate the effectiveness of this new approach with real experiments.

The remainder of the chapter is structured as follows: Section 5.2 presents the particular aspects of the formulation for this approach and an overview of the solution. The path planning algorithm is developed in Section 5.3 along with the goal selection method. Section 5.4 introduces the coverage controller and the navigation algorithm. The performance of the whole approach is analyzed in simulations and with real experiments in Section 5.5. Finally, Section 5.6 gathers the conclusions of this work.

5.2 Solution Overview

In order to introduce this approach to the problem, we build upon the formulation from the previous chapters and give an overview of the new solution.

The formulation of the problem is basically the same as in the previous chapter. In this chapter the environment is not necessarily convex and each robot has a map that may include walls and obstacles. We assume for simplicity of the formulation that the shape of the robots is a circle of radius r and we define $\mathbf{Q}_f(k)$ as the points in which the robot can be located without colliding with the obstacles. This is obtained by expanding the obstacles by the radius of the robot. Although it may change over time, we omit this dependency in the rest of the chapter for simplicity.

With respect to the production of the robots, $\alpha_i\big(\mathbf{q}, \mathbf{p}_i(k)\big)$, the main difference that we introduce in this chapter is that it can be controlled by a certain gain, $0 \leq \rho_i(k) \leq \rho_i^{\max}$, that we call coverage action.

The intuitive idea of the approach that we present here is the following. As in the previous chapter, the robots divide the environment in particular regions using an r-disk Voronoi partition. Each robot also creates a list of candidate goals that includes the points of its region in which the coverage level can be improved the most. At each iteration the partition and the list are updated, the latter using the improvement function $M_i(k)$. The goals that are no longer feasible or safe are removed and new goals are included. In fact, a

goal is considered safe if it is not close to any other robot: $\|\mathbf{g}_i - \mathbf{p}_j(k)\| > \mathcal{E}'$, with $\mathbf{g}_i \in \mathbf{L}_i(k)$, $j \neq i, j \in \{1, \ldots, N\}$ and $\mathcal{E}'$, a safety distance. If the current goal of a robot stops being feasible, profitable or safe, a new one must be selected from the list. The main contribution of this algorithm is in how to select such new goal and how to reach it. To do so, the robot first calculates the paths to all the candidates from the list, selects the one whose path optimizes the coverage and then follows the path calculating the optimal coverage action and controlling the movement.

An overview of this procedure that each robot locally performs can be seen in Alg. 3.

Algorithm 3 *Solution Overview*

```
Create initial partition and goal list.
while Robot in operation do
    Update Z(k), (2.2), and M_i(k), (4.2).
    Update partition and goal list (Algorithm 2).
    if Current goal not valid (4.20) then
        Plan paths to goals (Section 5.3).
        Select new current goal (Section 5.3.4).
    end if
    Calculate coverage action (Section 5.4.1).
    Control movement (Section 5.4.2).
end while
```

5.3 Coverage-Optimal, Finite-Horizon Path Planning

In order to reach the coverage goals the robots have to plan the paths from their location. These paths not only must avoid all the obstacles but also must go through the lowest covered areas. The idea is that the robots go to the goals providing the best improvement of the coverage along their paths. To do so, we use an FMM-based planning method that allows us to take into account these two variables: obstacles and improvement of the coverage. This method is suitable for online computation and with the same calculation permits us to find the paths to several points. This represent an interesting advantage, since each robot can analize several goals at the same time and select the best one based on the quality of their associated paths.

5.3.1 FMM-based Planning

Given a robot location, FMM-based path planning methods [138] calculate a potential field of the environment, $v_i(\mathbf{q})$, $\forall\, \mathbf{q} \in \mathbf{Q}_f$, that provides the optimal path if the gradient descent of the field is followed from any goal. This means that calculating the field once, the paths to different points can be calculated. The potential field represents the minimal time that a wave front needs to propagate from the robot location to any other feasible point

of the environment. To obtain this field, FMM approximates the solution to the nonlinear Eikonal equation

$$|\nabla v_i(\mathbf{q})| = F_i(\mathbf{q}),\ \forall\, \mathbf{q} \in \mathbf{Q}_f$$

given a boundary condition $v_i(\mathbf{q}) = 0,\ \forall\, \mathbf{q} \in \Gamma$, where $\Gamma \subseteq \mathbf{Q}_f$ is the initial wave front, that in this case is the position of the robot $\mathbf{p}_i(k)$. As an input, we have to provide the FMM planner with the initial point of the path, i.e., the position of the robot, and the speed of propagation of the wave front as a function of the points of the environment, $F_i(\mathbf{q}, k) > 0$. This speed, which is not directly related with the speed of the robot, can be seen as the inverse of a cost of the robot visiting each point of the environment. When the FMM calculates the potential field, that is smooth and free of local minima, the optimal path $\gamma_i(\tau)$ to any goal is calculated by following the direction of the gradient descent of the field from the goal position:

$$\gamma_i(\tau) = \gamma_i(\tau - \mathrm{d}\tau) - \mathrm{d}\tau \nabla v_i(\gamma_i(\tau - \mathrm{d}\tau)), \tag{5.1}$$

where the path is parametrized by $\tau \in [0, L]$ and L is the length of the path. Although this path is calculated regardless of the dynamics of the robot, it does not have abrupt turns thanks to the smoothness of the FMM potential field. On the other hand, it is optimal in the sense that it minimizes

$$\int_{\gamma_i(0)}^{\gamma_i(L)} \left[F_i\Big(\gamma_i(\tau), k\Big) \right]^{-1} \mathrm{d}\tau.$$

We consider as speed function

$$F_i(\mathbf{q}, k) = F_i^o(\mathbf{q}, k) + F_i^M(\mathbf{q}, k),$$

that is the result of merging together a function associated to obstacles and another associated to the improvement of the coverage. The idea is that the path optimizes the coverage from the robot position to the goal and avoids the obstacles.

5.3.2 Coverage Improvement Speed Function

In order to find the path towards the goal that optimizes the coverage, we introduce a sigmoid speed function that includes the improvement of the coverage as follows:

$$F_i^M(\mathbf{q}, k) = \frac{1}{1 + e^{-\beta_M M_i(\mathbf{q}, k)}}.$$

where β_M is a parameter to modify the shape of the sigmoid. This function assigns a high speed for the wave front to the points in which the *Improvement Function* is greater than zero, i.e., where the robot can improve the coverage, and a low speed to the points with negative improvement. The effect of this function is that the FMM planner calculates the path of maximum accumulated improvement to the goal.

Since the speed function is based on the *Improvement Function* (4.1) that changes over time, its online computation benefits from the temporal update of the improvement proposed in (4.2).

5.3.3 Obstacle Speed Function

The FMM-based planning already avoids collision with obstacles since the potential field is only calculated for $\mathbf{q} \in \mathbf{Q}_f$, i.e., the collision-free positions. Nevertheless, it is also desirable to keep a safety distance to the obstacles when possible. To do so, we assign a higher cost to the points around the obstacles than to points in free areas so that the wave front propagates faster in free areas than near obstacles. Therefore, the resulting path of the robot goes through free spaces rather than close to obstacles. The speed function that we propose, adapted from [92], is

$$F_i^o(\mathbf{q}, k) = \begin{cases} 0, & \delta(\mathbf{q}, k) < C, \\ 1 - \max(-\left(\frac{\delta(\mathbf{q}, k)}{C}\right)^2 + 2\frac{\delta(\mathbf{q}, k)}{C}, 0), & \text{otherwise}, \end{cases}$$

with C, the safety distance and $\delta(\mathbf{q}, k)$, the euclidean distance between the point $\mathbf{q}$ and the closest point of the obstacles.

We do not develop further the avoidance of moving obstacles because of the difficulty of predicting their future positions. However, the navigation algorithm that we introduce in Section 5.4 handles the obstacles that may appear in the path of the robots and avoids collisions with them.

5.3.4 Selection of the Current Goal

The point of the list with the best improvement may not be the best choice for the coverage of the entire environment. It is the one that needs the most coverage locally but the path that the robot has to follow to reach it may not be profitable at all. For this reason, we propose a new strategy to select the goal from the list.

The first step is to plan the paths to the goals of the list, $\gamma_{\mathbf{g}_i}(\tau)$, $\forall\, \mathbf{g}_i \in \mathbf{L}_i(k)$, according to (5.1). To do so, the potential field $v_i(\mathbf{q})$ only has to be calculated once from the current position of the robot for all the goals.

Then, the total improvement of the path per length unit is calculated as

$$M_{\gamma_i}(\mathbf{g}_i, k) = \frac{1}{L(\mathbf{g}_i)} \int_{\gamma_{\mathbf{g}_i}(0)}^{\gamma_{\mathbf{g}_i}(L(\mathbf{g}_i))} M_i\big(\gamma_{\mathbf{g}_i}(\tau), k\big) \mathrm{d}\tau,$$

where $L(\mathbf{g}_i) \equiv L(\gamma_{\mathbf{g}_i}(\tau))$ is the length of $\gamma_{\mathbf{g}_i}(\tau)$.

Finally, we select as current goal for robot i the one with the highest value of $M_{\gamma_i}(\mathbf{g}_i, k)$, that is,

$$\mathbf{g}_i^* = \underset{\mathbf{g}_i \in \mathbf{L}_i(k)}{\operatorname{argmax}}\, M_{\gamma_i}(\mathbf{g}_i, k).$$

In the case that various goals satisfy this equation, we select as current goal the one with the longest L, since it provides the longest plan in time for the robot and therefore the greatest overall improvement. Note that, for every goal, the path is optimal to reach it. However, since the selection is made from a finite set of goals, the path to the selected goal may not correspond to the global optimum among all the possible paths in the partition.

5.4 Coverage Action Control and Navigation

The last step to complete our persistent coverage approach is the control of the coverage action of the robots and the navigation towards the goals following the planned paths.

5.4.1 Coverage Action Control

We propose the following law to control the coverage action. It is based on the difference between the coverage level of the environment and the coverage objective, both in the coverage area of the robot:

$$\rho_i(k) = \begin{cases} \rho_i^{\max}, & \text{if } \rho_i^*(k) \geq \rho_i^{\max}, \\ \rho_i^*(k), & \text{if } 0 < \rho_i^*(k) < \rho_i^{\max}, \\ 0, & \text{if } \rho_i^*(k) \leq 0, \end{cases} \tag{5.2}$$

with

$$\rho_i^*(k) = \frac{\displaystyle\int_{\mathbf{\Omega}_i(\mathbf{p}_i(k))} \Phi\big(Z^* - dZ(k-1)\big)\alpha_i(k)\,\mathrm{d}\mathbf{q}}{\displaystyle\int_{\mathbf{\Omega}_i(\mathbf{p}_i(k))} \Phi\,\alpha_i(k)^2\,\mathrm{d}\mathbf{q}}.$$

This definition leads us to the following proposition:

Proposition 5.4.1. *The coverage controller from* (5.2) *is optimal in terms of the quadratic coverage error,*

$$\tilde{\varepsilon}(k) = \int_{\mathbf{Q}} \Phi\big(Z^* - Z(k)\big)^2\,\mathrm{d}\mathbf{q}.$$

Proof. We want to find the coverage action that optimizes the quadratic coverage error, i.e.,

$$\rho_i^*(k) = \underset{\rho_i(k)}{\operatorname{argmin}}\,\tilde{\varepsilon}(k).$$

Introducing (2.2) and recalling that $\alpha_i(k) = 0,\ \forall\,\mathbf{q} \notin \mathbf{\Omega}_i$, this error can be formulated as

$$\tilde{\varepsilon}(k) = \int_{\mathbf{\Omega}_i} \Phi\big(Z^* - dZ(k-1) - \rho_i(k)\alpha_i(k)\big)^2\mathrm{d}\mathbf{q}.$$

Now, we derive it with respect to $\rho_i(k)$ to find its optima:

$$\begin{aligned} \frac{\partial\tilde{\varepsilon}(k)}{\partial\rho_i(k)} &= \int_{\mathbf{\Omega}_i} \Phi\,\frac{\partial}{\partial\rho_i(k)}\big(Z^* - dZ(k-1) - \rho_i(k)\alpha_i(k)\big)^2\mathrm{d}\mathbf{q} \\ &= \int_{\mathbf{\Omega}_i} 2\,\Phi\big(Z^* - dZ(k-1) - \rho_i(k)\alpha_i(k)\big)\big(-\alpha_i(k)\big)\mathrm{d}\mathbf{q} \\ &= -2\left[\int_{\mathbf{\Omega}_i} \Phi\big(Z^* - dZ(k-1)\big)\alpha_i(k)\mathrm{d}\mathbf{q} - \int_{\mathbf{\Omega}_i} \Phi\,\rho_i(k)\alpha_i(k)^2\mathrm{d}\mathbf{q}\right]. \end{aligned} \tag{5.3}$$

If we make this derivative equal to zero, we obtain

$$\frac{\partial \tilde{\varepsilon}(k)}{\partial \rho_i(k)} = 0 \Rightarrow \rho_i^*(k) = \frac{\displaystyle\int_{\mathbf{\Omega}_i} \Phi\big(Z^* - dZ(k-1)\big)\alpha_i(k)\mathrm{d}\mathbf{q}}{\displaystyle\int_{\mathbf{\Omega}_i} \Phi\,\alpha_i(k)^2\mathrm{d}\mathbf{q}},$$

that is the optimum value of $\rho_i(k)$ since

$$\frac{\partial^2 \tilde{\varepsilon}(k)}{\partial \rho_i(k)^2} = \int_{\mathbf{\Omega}_i} \Phi\,\alpha_i(k)^2\mathrm{d}\mathbf{q} > 0.$$

Finally, we can directly apply the restrictions on the coverage action $0 \leq \rho_i(k) \leq \rho_i^{\max}$. Since the value of $\rho_i^*(k)$ is a minimum and the function $f\big(\rho_i(k), \mathbf{q}\big)$ is a sum of quadratic functions, the value of $\rho_i(k)$ in the restricted interval that gives the minimum $\tilde{\varepsilon}(k)$ is always the closest to $\rho_i^*(k)$. □

The main feature of the coverage action controller is that it allows the robots to maintain the coverage level as close as possible to the objective without overcovering it unboundedly. In comparison to our previous approach this is a significant advantage, as we will show it in simulations.

5.4.2 Robot Navigation

The navigation and guidance algorithm has to calculate the velocity commands for the robot to follow the path respecting its admissible actions. At the same time, it has to avoid the unmapped obstacles that may appear during the motion and can be detected by the sensors on board the robots. To this end, we make use of a Dynamic Window Approach (DWA) [91], that computes the velocity command as a function of the previous command, the desired path, the motion and acceleration restrictions of the robot, and its current pose and sensor readings.

The implications of this algorithm are the following. When there are no unmapped obstacles in the surroundings of the robot, it follows the planned path at its maximum allowed speed and provides the optimal coverage to the environment. If an obstacle appears, it is avoided while trying to minimize the deviation from the path. The same happens when the path can not be precisely followed due to the robot dynamics. In both cases the coverage along the path is not optimal, due to such deviation. However, thanks to the coverage action controller, the coverage provided at each point is still optimal.

5.5 Simulations

In this section we introduce the robotic platform, present the results of the simulation, including a comparison with the previous chapter and show some experimental results.

5.5.1 Robotic Platform and Simulations Settings

For the validation of this algorithm we used a team of robots (Fig. 5.1) developed within the MOnarCH Project[1] [44]. The robotic platform is omnidirectional thanks to four Mecanum wheels, actuated by four independent motors. The maximum linear velocity of the robots is $u_i^{\max} = 2.5m/s$; the maximum acceleration, $\dot{u}_i^{\max} = 1m/s^2$; and the maximum angular velocity, $\omega_i^{\max} = 600°/s$. Among other sensors, the robots are equipped with two laser range finders that are used for mapping and localization with off-the-shelf ROS packages.

Figure 5.1: Team of MOnarCH robots used for the experiments.

Since these robots were not specifically built to develop coverage and they are not equipped with actuators such as vacuuming or heating systems, we simulate the coverage as follows. The coverage area of each robot, $\mathbf{\Omega}_i\big(\mathbf{p}_i(k)\big)$, is a circle of radius equal to the actual radius of the robot, $r_i^{cov} = 0.325m$. The production is defined as constant inside this area with a value of $\alpha_i\big(\mathbf{q}, \mathbf{p}_i(k)\big) = 10$. The coverage of the environments decreases over time with a decay rate $d(\mathbf{q}) = 0.9995$ and the coverage objective is $Z^*(\mathbf{q}) = 100$. Recall that the aim of the team is to keep the coverage of all the points as close as possible to this value.

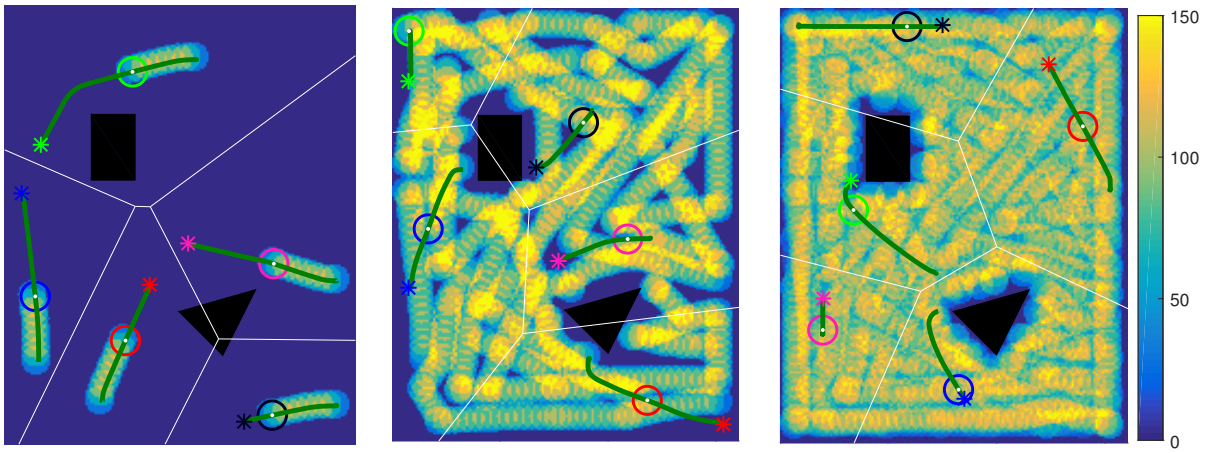
5.5.2 Simulation Results

In the first place we introduce simulation results to evaluate the effectiveness of the approach and the scalability of the team. Moreover, we compare the current approach with the one from the previous chapter.

In Fig. 5.2 we show an example of the persistent coverage of a 10 x 8 m^2, non-convex environment with five robots. At the beginning (Fig. 5.2a), the robots start covering the environment following almost straight paths to the goals and only avoiding obstacles. When the coverage of the environment increases (Fig. 5.2b), the planned paths go through the least covered areas to the goals. In the end, when all the point of the environment are well covered

[1]Multi-Robot Cognitive Systems Operating in Hospitals. FP7-ICT-2011-9-601033. Website: http://monarch-fp7.eu/

(Fig. 5.2c), the robots keep moving to maintain the coverage level as close to the objective as possible.



(a) k = 10. (b) k = 150. (c) k = 500.

Figure 5.2: Example of simulation: coverage level of the environment with the coverage areas of the robots in different colors for three different time instants. Colored asterisks represent the goals of the robots; green lines, the planned paths of the robots; and white lines, the Voronoi partition of the environment. Black areas are mapped obstacles.

The mean coverage level of the environment increases quickly when the robots start moving and reaches a steady state in around 200 iterations as shown in Fig. 5.3. The mean coverage level in the steady-state value is less than 4% away from the objective. Additionally, the standard deviation is around 15 coverage units while it only reaches a maximum value of 48 in the transient.

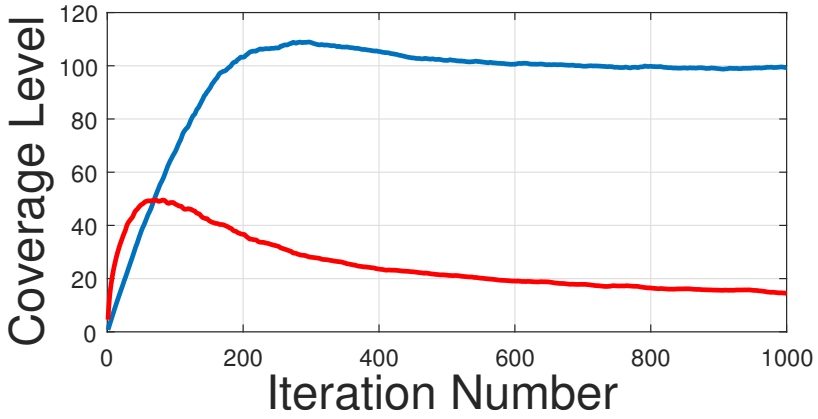
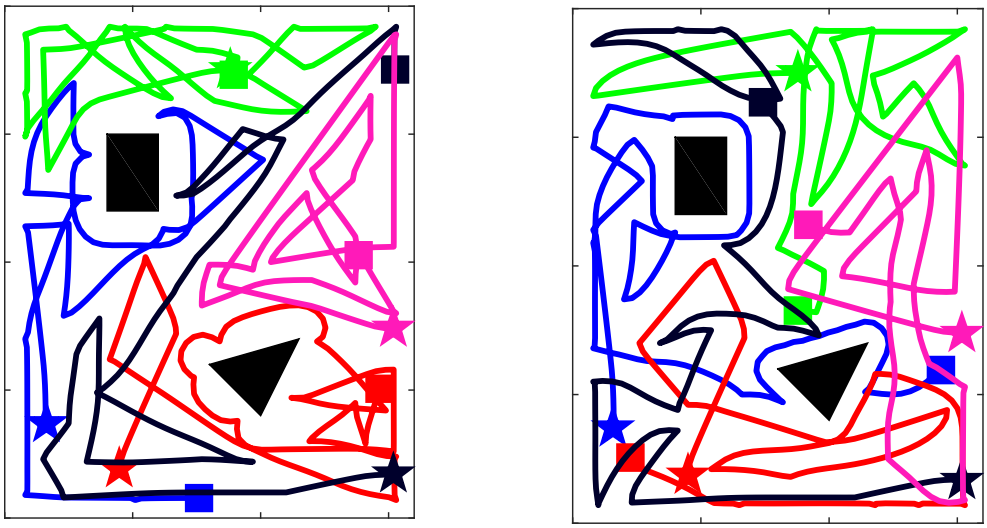


Figure 5.3: Evolution of the mean coverage level (blue) and its standard deviation (red) for the simulation example.

Fig. 5.4a shows the paths followed by the robots in the first 200 iterations with an iteration period of 125 ms. The green and magenta robots cover specific regions while the



(a) (b)

Figure 5.4: Paths followed by the robots during the first 200 iterations (a) using the obstacle speed function and (b) with no obstacle speed function. Stars represent the initial points and squares, the final ones.

others exchange regions due to the presence of obstacles in the environment. The paths keep a safety distance to these obstacles to avoid collisions. In contrast, they drive the robots much closer to the obstacles if the obstacle speed function is not included in the planning, as can be seen in Fig. 5.4b. The average minimum distance to an obstacle in these two cases is 0.94 and 0.72 m, respectively, and the minimum distance is lower than 0.5 m in 21 and 86 out of 200 iterations, respectively. This supports the use of the obstacle speed function to reduce the risk of collisions due to noise in mapping and/or localization.

As mentioned before, in Fig. 5.5 we test the scalability of our approach and compare it with our previous approach. These simulations have been carried out in a 15×15 m^2 environment to be able to increase the number of robots. Fig. 5.5a shows that a team of 2 or 5 robots is not capable of reaching on average the objective of $Z^*(\mathbf{q}) = 100$ and produces a higher standard deviation (Fig. 5.5c) than a team of 20 or 50 robots. These teams reach the objective rapidly with a low standard deviation. One of the main differences with our previous approach is that the coverage action was not controlled as it is in this one, i.e., it was fixed. Consequently, the mean coverage level increases with the number of robots (Fig. 5.5b) independently from the objective and only a team of 10 robots maintains the objective level over time. Additionally, the deviation of the coverage level with the previous approach is considerably higher (Fig. 5.5d).

The total computational cost of the simulation has been measured and represented in Fig. 5.6. Although this time increases more than linearly with the number of robots, with 50 of them it takes less than 35 ms per robot to calculate their control actions. This shows that our approach is scalable and can be applied distributedly, i.e., each robot can calculate its actions independently. Additionally, we show that this new approach is faster than the

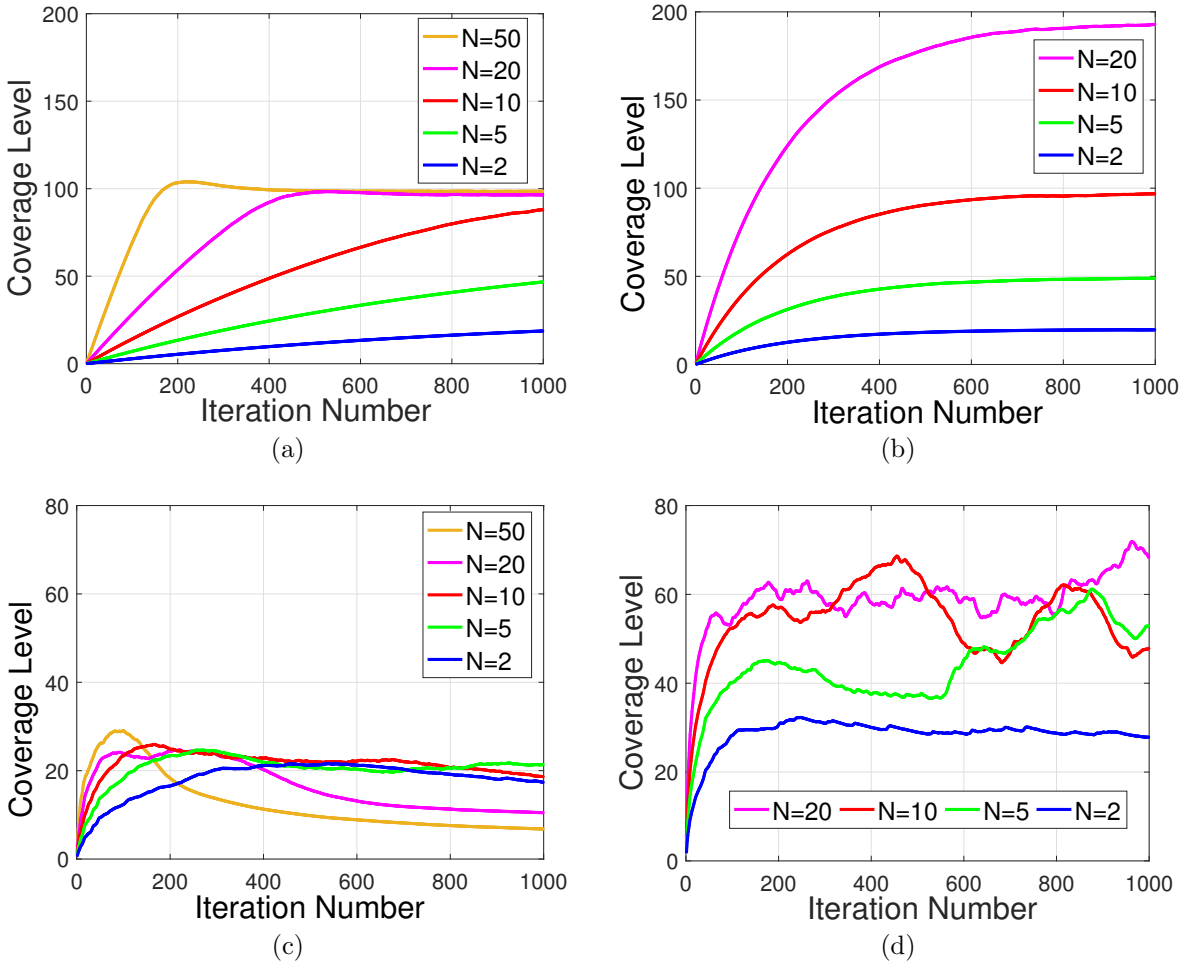


Figure 5.5: Evolution of the coverage with the number of robots. (a) Mean coverage level of this approach. (b) Mean coverage level with our pevious approach. (c) Standard deviation of the coverage level with this approach. (d) Standard deviation with our previous approach.

previous one, since once the path is planned, the calculation of the motion action is fast compared with the gradient calculated every time in 4.28.

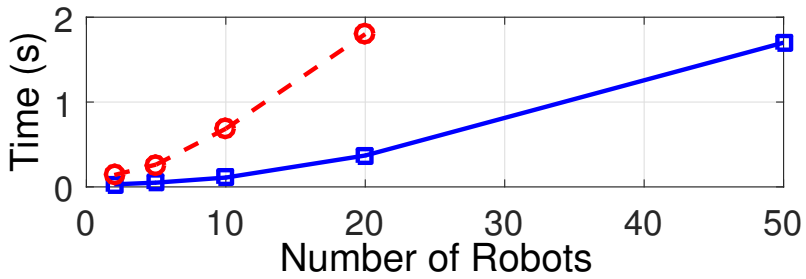


Figure 5.6: Evolution of the iteration time with the number of robots. Blue square markers represent the times with this approach and red circular markers, with our previous approach.

5.5.3 Experimental Results

The experimental validation of this work was developed in the robotics lab of the Distributed Intelligent Systems and Algorithms Laboratory at EPFL. The environment was an almost rectangular region of approximately 15 m^2 where $N = 2$ robots were deployed. In Fig. 5.7 we show a snapshot of one of the experiments and the complete experiment can be seen here.

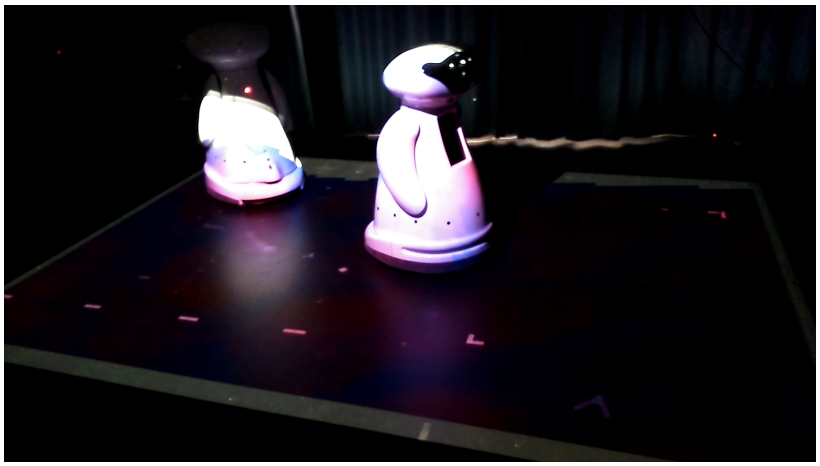


Figure 5.7: Snapshot of one of the experiments. The coverage level of a part of the environment is projected on the ground. Shades of blue represent undercovered points and shades of red, points whose level is around the objective.

In Fig. 5.8 the mean coverage level of the environment and its standard deviation are depicted. The evolution is similar to the simulated one and it proves the validity of the approach with real robots.

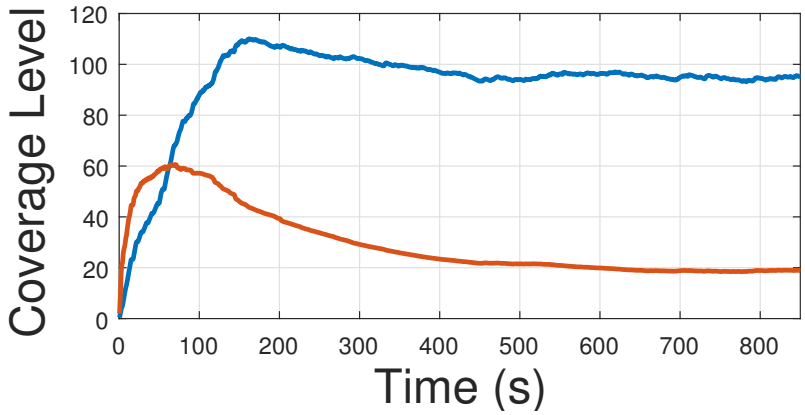


Figure 5.8: Evolution of the mean coverage level (blue) and its standard deviation (red).

Finally, we show the paths followed by the robots. These paths are more overlapped than the simulated ones due to the limited accuracy of the localization and motion of the robots.

Nevertheless, they cover the entire environment with almost no overlapping of the regions covered by each robot.

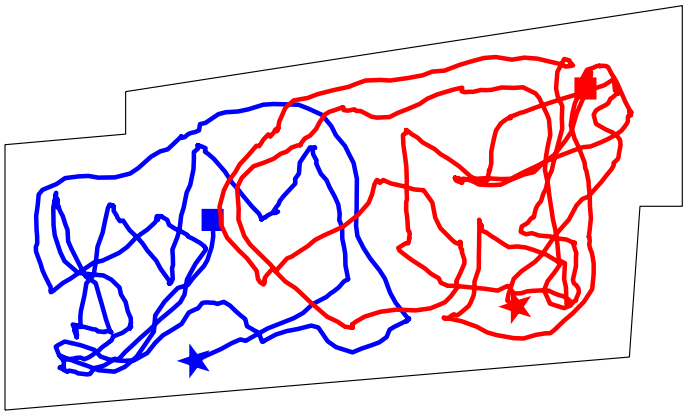


Figure 5.9: Paths followed by the robots during the first 200 iterations. Stars represent the initial points and squares, the final ones.

5.6 Conclusions

In this chapter we have introduced a method to plan open paths based on FMM. For this method, we have proposed two speed functions: one to optimize coverage and another one to keep a safety distance to obstacles. We have also presented a method to find and select the coverage goals which the robots have to reach based on the total improvement of the path. To keep the robot following their paths we use a DWA. Additionally, we have proposed a law to control the coverage action of the robots at every time.

One of the main advantages of this approach is the coverage controller. It provides a significant improvement to reach and maintain the coverage objective as we have shown in simulations. In separates the coverage from the motion in the sense that keeping the robot still or moving it fast does not necessarily deteriorate the coverage level.

The other main advantage of this approach is the planning of finite horizon paths. It leverages advantages and overcome limitations of the typical solutions based on feedback controllers or on the planning of infinite horizon paths. Whit respect to the latter, it can adapt to environments where the decay or the importance is not uniform, to changes in the environment including obstacles and coverage, and also to changes on the robotic team. With respect to controllers, it provides optimal coverage paths and does not get stuck in local minima or due to obstacles. In addition, it does not require sudden changes in the actions since the resulting paths are smooth.

The only disadvantage of this proposal resides in that the Voronoi partition of each robot may not be self-connected, i.e., it can be split into two or more components in the presence of complex obstacles or walls like in an office environment. This results in that a robot path may cross the partition of another robot with all its implication. Additionally, the partitions

are exclusively geometric and do not take into account how much coverage is required inside them. For these reasons and being aware that this is a problem inherent to the Voronoi partitions, in the following chapters we propose a more appropriate partitioning method for persistent coverage along with a motion strategy inside them.

Chapter 6

A Distributed Solution for Complex, Non-Convex Environments with Equitable Partitioning and Graph-based, Finite-Horizon-Optimal Paths

In this chapter we present a distributed solution for complex, non-convex environment. As a first step, our solution finds a partition of the environment where the amount of work for each robot, weighted by the importance of each point, is equal. This is achieved using a power diagram and finding an equitable partition through a provably correct distributed control law on the power weights. Compared to other existing partitioning methods, the contribution of our solution is the ability to work in complex environments. Additionally, we propose two extensions for this strategy. The first one is an update of the partition that is aware of the energy consumption of the robots, their different coverage capabilities or even the failure of one or more members of the team. The second extension controls the movement of the generators of the partitions with the objective of keeping the partitions self-connected. In the second step, each robot computes a graph that gathers sweep-like paths and covers its entire partition. At each planning time, the current coverage error at the graph vertices is assigned as weights of the corresponding edges. Then, our solution is capable of efficiently finding the optimal open coverage path through the graph. Simulation results are presented to support our proposal.

6.1 Introduction

The intention of this chapter is to introduce a solution to the problem for all kind of continuous environments, in general, complex, non-convex ones. This solutions can be included in the group of partition-based solutions. They address the problem with a divide-and-conquer strategy in the sense that they partition the environment and assign each partition

to a single robot that becomes responsible of developing the coverage on it [54]. The advantages of this type are that, once the environment is partitioned, each robot can work independently, with no risk of collision and with no need of communication in the normal operation of the system.

Our partitioning method builds upon the one presented in [73] that was limited to convex environments. Our contribution here is threefold. The main contribution is the extension of the method to general, non-convex environments, by formulating the algorithm in terms of the geodesic distance. This formulation allows the partition to be consistent with the shape of the environment. The second contribution is an energy-aware partition update that is intended to repartition the environment when one or more robots need to recharge or fail. The third contribution is a strategy for the generators of the partitions to follow the centroid of the biggest connected component to reduce the disconnections of the partitions. Together, our partitioning method guarantees convergence to the equitable partition while reducing the disconnections, is aware of the energy of the robots, does not require a discretization of the environment and does not require that the initial position of the robot belong to its final partition.

The second part of our solution is a planning method that allows each robot to find the optimal, finite-horizon path inside its partition in terms of the current coverage error. The use of the current coverage error is motivated by the complexity that implies the temporal evolution of the coverage, due to the decay and the actions of the robots. It permits to eliminate this temporal component from the planning and still obtain optimal paths. The paths are planned through a graph that covers the entire partition in a sweep-like manner and also allows the robots to traverse the boundary of their partitions. These paths are found in a finite horizon in order to handle that the decay of the coverage and the importance of the points are not uniform inside the partitions. While following the paths, the robots apply the coverage controller from (5.2)

Since the formulation of the problem for this chapter is the same as for the previous one, the remainder of the chapter is structured as follows. In Section 6.2 we present the equitable partitioning method along with an energy-aware updating strategy and an extension to reduce disconnections in the partitions. In Section 6.3 the strategy to plan paths and individually cover each partition is presented. Finally, we present simulation results in Section 6.4 and conclusions in Section 6.5.

6.2 Equitable Partitioning

The first step of our strategy to solve the persistent coverage problem is to divide the environment in as many equitable regions as robots. The goal is that each partition requires an amount of work proportional to the capabilities of its assigned robot with respect to the rest of the team and to the importance of its points. To this end, we introduce the notions of power diagram and geodesic distance, define the importance-weighted workload of the agents in terms of the coverage problem and detail our distributed partitioning algorithms along with an energy-aware extension.

6.2.1 Power Diagrams and Geodesic Distance

Let us first define a set of points, $\mathcal{G} = \{\mathbf{g}_1, \ldots, \mathbf{g}_N\}$, with $\mathbf{g}_i, \mathbf{g}_j \in \mathcal{Q}, \mathbf{g}_i \neq \mathbf{g}_j$ if $i \neq j, i, j \in \{1, \ldots, N\}$, which will act as the generators of the partitions. Our algorithm to find the equitable partitions is based on power diagrams [139], that are a generalization of the Voronoi diagrams. The main advantage, apart from generality, is that the equitable power diagram is always guaranteed to exist [73]. Instead of assigning the points to each partition according to the euclidean distance to the generator, the power diagrams uses the squared distance minus a certain weight. If we let $\mathbf{w} = \{w_1, \ldots, w_N\}$ be the set of power weights associated to the partitions, we can formally define the partitions obtained form the power diagram as

$$\mathcal{P}_i(\mathbf{w}) = \{\mathbf{q} \in \mathcal{Q}_f \,|\, d(\mathbf{q}, \mathbf{g}_i)^2 - w_i \leq d(\mathbf{q}, \mathbf{g}_j)^2 - w_j\}.$$

To be able to apply this partitioning to a non-convex environment, we make use of the geodesic distance, $d_g(\mathbf{q}, \mathbf{g})$, that is the length of the shortest path between two points of the environment. Assuming that the environment is composed of polygonal obstacles, the shortest path between every two points is the concatenation of a set of straight lines connecting the two points and some vertices of the obstacles, $\{\mathbf{q}, \mathbf{h}^1_{\mathbf{q},\mathbf{g}}, \mathbf{h}^2_{\mathbf{q},\mathbf{g}}, \ldots, \mathbf{h}^{l_{\mathbf{q},\mathbf{g}}}_{\mathbf{q},\mathbf{g}}, \mathbf{g}\}$, where $l_{\mathbf{q},\mathbf{g}}$ is the number of obstacle vertices that define such path. Therefore, the geodesic distance can be decomposed in the summation of the Euclidean distances between each pair of consecutive points:

$$d_g(\mathbf{q}, \mathbf{g}) = \|\mathbf{q} - \mathbf{h}^1_{\mathbf{q},\mathbf{g}}\| + \|\mathbf{h}^1_{\mathbf{q},\mathbf{g}} - \mathbf{h}^2_{\mathbf{q},\mathbf{g}}\| + \cdots + \|\mathbf{h}^{l_{\mathbf{q},\mathbf{g}}}_{\mathbf{q},\mathbf{g}} - \mathbf{g}\|. \tag{6.1}$$

Using this metric, the boundary between two partitions is

$$\Delta_{ij} = \{\mathbf{q} \in \mathcal{Q}_f \,|\, d_g(\mathbf{q}, \mathbf{g}_i)^2 - w_i = d_g(\mathbf{q}, \mathbf{g}_j)^2 - w_j\}. \tag{6.2}$$

According to this definition, the neighbors of a generator are

$$N_i = \{j \in \{1, \ldots, N\} \setminus i \,|\, \Delta_{ij} \neq \emptyset\}.$$

We assume that each robot can communicate with robots of neighboring regions.

6.2.2 Importance-Weighted Workload

The partitions that are assigned to the robots have to be equitable in terms of the work that has to be carried out inside them. At the same time, they have to take into account the importance of the coverage of each point. Therefore, it is essential to define the workload in terms of the variables of the problem. The definition of the work at each point of the environment that we consider is

$$\lambda(\mathbf{q}) = \phi\,(1 - d)\,Z^*.$$

Strictly speaking, $(1 - d)\,Z^*$ is the actual workload of each each point. It represents the coverage that decays at each time if the point has reached the desired level, i.e., in the

steady state. Nevertheless, we weight it with the importance to allow the robots to spend more time covering the most important points, at the expense of reducing the attention to less important points. This may lead to partitions in which $(1-d)\,Z^*$ is not equitable but the importance of the work of each partition is. From now on, we refer to this importance-weighted workload simply as workload.

The workload inside each partition can be calculated as

$$\lambda_{\mathcal{P}_i(\mathbf{w})} = \int_{\mathcal{P}_i(\mathbf{w})} \lambda(\mathbf{q})\, d\mathbf{q},$$

Both the importance and the decay take values between 0 and 1. On the contrary, the desired coverage level is expressed in general in coverage units and therefore, we normalize the entire workload by $\int_{\mathcal{Q}} \phi\,(1-d)\,Z^*\, d\mathbf{q}$ to obtain it in percentages.

6.2.3 Distributed Algorithm for Equitable Partitioning

The power diagram can be seen as a relation between the weights and the regions that belong to their corresponding generators. Therefore, we minimize a cost function whose minima correspond to sets of weights that lead to an equitable partition:

$$H(\mathbf{w}) = \sum_{i=1}^{N} \frac{1}{\lambda_{\mathcal{P}_i(\mathbf{w})}}.$$

One can see that this function reaches its minimum when the workload of all the partitions is the same. From now on, we omit the dependencies of H and $\mathcal{P}_i$ with $\mathbf{w}$ for the sake of clarity.

To minimize this function distributively, each robot is in charge of a generator, that we locate at the robot's position for simplicity, and its associated weight, initially set to zero. The following law controls the latter to reach an equitable partition by communicating only with its neighbors.

Theorem 6.2.1. *The power diagram generated by $\mathcal{G}$ and $\mathbf{w}$ converges to an equitable power diagram under the distributed control law*

$$\dot{w}_i = -k_w \frac{\partial H}{\partial w_i}, \tag{6.3}$$

with k_w, a positive gain, and

$$\frac{\partial H}{\partial w_i} = \sum_{j \in N_i} \left(\frac{1}{\lambda_{\mathcal{P}_j}^2} - \frac{1}{\lambda_{\mathcal{P}_i}^2} \right) \int_{\Delta_{ij}} \|n'_{ij}(\mathbf{q})\|\, \lambda(\mathbf{q})\, d\mathbf{q}, \tag{6.4}$$

where $n'_{ij}(\mathbf{q})$ is the outward normal to the boundary between the partitions i and j at point $\mathbf{q} \in \Delta_{ij}$.

Proof. In the first place we develop the gradient of the cost function with respect to a single weight w_i:

$$\frac{\partial H}{\partial w_i} = -\frac{1}{\lambda_{\mathcal{P}_i}^2}\frac{\partial \lambda_{\mathcal{P}_i}}{\partial w_i} - \sum_{j\in N_i}\frac{1}{\lambda_{\mathcal{P}_j}^2}\frac{\partial \lambda_{\mathcal{P}_j}}{\partial w_i}. \tag{6.5}$$

It only depends on the neighboring partitions since a variation on w_i only affects them. The derivative of the workload in the partition of robot i is

$$\frac{\partial \lambda_{\mathcal{P}_i}}{\partial w_i} = \frac{\partial}{\partial w_i}\int_{\mathcal{P}_i}\lambda(\mathbf{q})\,d\mathbf{q}.$$

It can be transformed using the following result associated with the divergence theorem:

$$\frac{\partial}{\partial w_i}\int_{\mathcal{P}_i}\lambda(\mathbf{q})\,d\mathbf{q} = \int_{\partial\mathcal{P}_i}\left(\frac{\partial \mathbf{q}}{\partial w_i}\cdot n_{\partial\mathcal{P}_i}(\mathbf{q})\right)\lambda(\mathbf{q})\,d\mathbf{q},$$

where $n_{\partial\mathcal{P}_i}(\mathbf{q})$ represents the unit outward normal to the boundary of the partition, $\partial\mathcal{P}_i$, at point $\mathbf{q}$. It results in

$$\frac{\partial \lambda_{\mathcal{P}_i}}{\partial w_i} = \sum_{j\in N_i}\int_{\Delta_{ij}}\left(\frac{\partial \mathbf{q}}{\partial w_i}\cdot n_{ij}(\mathbf{q})\right)\lambda(\mathbf{q})\,d\mathbf{q} + \sum_{j\in N_i}\int_{\Delta_i^{\mathcal{Q}_f}}\left(\frac{\partial \mathbf{q}}{\partial w_i}\cdot n_{ij}(\mathbf{q})\right)\lambda(\mathbf{q})\,d\mathbf{q}, \tag{6.6}$$

where $\Delta_i^{\mathcal{Q}_f}$ is the boundary between the partition and the environment and $n_{ij}(\mathbf{q})$ is the unit normal outward this boundary or the boundary between partitions i and j, Δ_{ij}. The second term is always zero either because $\Delta_i^{\mathcal{Q}_f} = \emptyset$ or because a variation on the weight does not affect the boundary between the partition and the environment, i.e., $\partial\mathbf{q}/\partial w_i = 0$. Applying the same procedure to $\partial\lambda_{\mathcal{P}_j}/\partial w_i$ and introducing in (6.5), we have

$$\frac{\partial H}{\partial w_i} = -\frac{1}{\lambda_{\mathcal{P}_i}^2}\sum_{j\in N_i}\int_{\Delta_{ij}}\left(\frac{\partial \mathbf{q}}{\partial w_i}\cdot n_{ij}(\mathbf{q})\right)\lambda(\mathbf{q})\,d\mathbf{q} - \sum_{j\in N_i}\frac{1}{\lambda_{\mathcal{P}_j}^2}\int_{\Delta_{ij}}\left(\frac{\partial \mathbf{q}}{\partial w_i}\cdot n_{ji}(\mathbf{q})\right)\lambda(q)\,d\mathbf{q},$$

Noting that $n_{ji}(\mathbf{q}) = -n_{ij}(\mathbf{q})$, we only have to calculate the scalar product $\partial\mathbf{q}/\partial w_i \cdot n_{ij}(\mathbf{q})$. The first term can be obtained by deriving the condition of the boundary points (6.2) with respect to the power weight,

$$\frac{\partial}{\partial w_i}\big(d_g(\mathbf{q},\mathbf{g}_i)^2 - w_i\big) = \frac{\partial}{\partial w_i}\big(d_g(\mathbf{q},\mathbf{g}_j)^2 - w_j\big),$$

that leads to

$$2d_g(\mathbf{q},\mathbf{g}_i)\frac{\partial d_g(\mathbf{q},\mathbf{g}_i)}{\partial \mathbf{q}}\cdot\frac{\partial \mathbf{q}}{\partial w_i} - 1 = 2d_g(\mathbf{q},\mathbf{g}_j)\frac{\partial d_g(\mathbf{q},\mathbf{g}_j)}{\partial \mathbf{q}}\cdot\frac{\partial \mathbf{q}}{\partial w_i}. \tag{6.7}$$

The partial derivative of the geodesic distance with respect to the boundary point $\mathbf{q}$ only affects the first term on the right hand side of (6.1):

$$\frac{\partial d_g(\mathbf{q},\mathbf{g}_i)}{\partial \mathbf{q}} = \frac{-(\mathbf{q} - h^1_{\mathbf{q},\mathbf{g}_i})}{\|\mathbf{q} - h^1_{\mathbf{q},\mathbf{g}_i}\|}. \tag{6.8}$$

Analogously for the other generator we have

$$\frac{\partial d_g(\mathbf{q}, \mathbf{g}_j)}{\partial \mathbf{q}} = \frac{-(\mathbf{q} - h^1_{\mathbf{q},\mathbf{g}_j})}{\|\mathbf{q} - h^1_{\mathbf{q},\mathbf{g}_j}\|}. \tag{6.9}$$

Note that these derivatives have two components, x and y, and that they appear in a scalar product in (6.7). Therefore, introducing (6.8) and (6.9) in (6.7), we have

$$\begin{aligned} 2d_g(\mathbf{q}, \mathbf{g}_i) \left(\frac{-(\mathbf{q} - h^1_{\mathbf{q},\mathbf{g}_i})_x}{\|\mathbf{q} - h^1_{\mathbf{q},\mathbf{g}_i}\|} \frac{\partial \mathbf{q}_x}{\partial w_i} + \frac{-(\mathbf{q} - h^1_{\mathbf{q},\mathbf{g}_i})_y}{\|\mathbf{q} - h^1_{\mathbf{q},\mathbf{g}_i}\|} \frac{\partial \mathbf{q}_y}{\partial w_i} \right) - 1 \\ = 2d_g(\mathbf{q}, \mathbf{g}_j) \left(\frac{-(\mathbf{q} - h^1_{\mathbf{q},\mathbf{g}_j})_x}{\|\mathbf{q} - h^1_{\mathbf{q},\mathbf{g}_j}\|} \frac{\partial \mathbf{q}_x}{\partial w_i} + \frac{-(\mathbf{q} - h^1_{\mathbf{q},\mathbf{g}_j})_y}{\|\mathbf{q} - h^1_{\mathbf{q},\mathbf{g}_j}\|} \frac{\partial \mathbf{q}_y}{\partial w_i} \right) \end{aligned} \tag{6.10}$$

and, regrouping the terms,

$$\begin{aligned} \Bigg[2d_g(\mathbf{q}, \mathbf{g}_i) \frac{-(\mathbf{q} - h^1_{\mathbf{q},\mathbf{g}_i})_x}{\|\mathbf{q} - h^1_{\mathbf{q},\mathbf{g}_i}\|} + 2d_g(\mathbf{q}, \mathbf{g}_j) \frac{(\mathbf{q} - h^1_{\mathbf{q},\mathbf{g}_j})_x}{\|\mathbf{q} - h^1_{\mathbf{q},\mathbf{g}_j}\|}, \\ 2d_g(\mathbf{q}, \mathbf{g}_i) \frac{-(\mathbf{q} - h^1_{\mathbf{q},\mathbf{g}_i})_y}{\|\mathbf{q} - h^1_{\mathbf{q},\mathbf{g}_i}\|} + 2d_g(\mathbf{q}, \mathbf{g}_j) \frac{(\mathbf{q} - h^1_{\mathbf{q},\mathbf{g}_j})_y}{\|\mathbf{q} - h^1_{\mathbf{q},\mathbf{g}_j}\|} \Bigg] \cdot \frac{\partial \mathbf{q}}{\partial w_i} = 1. \end{aligned} \tag{6.11}$$

To obtain the normal to the boundary between two partitions, $n_{ij}(\mathbf{q})$, we make use of the property of the gradient of a function, that is always perpendicular to the level curves of the function. To this end, we transform the equation that defines the boundary points (6.2) into a function,

$$f(\mathbf{q}) = d_g(\mathbf{q}, \mathbf{g}_i)^2 - w_i - d_g(\mathbf{q}, \mathbf{g}_j)^2 + w_j,$$

Then, we calculate the gradient at the points that satisfy $f(\mathbf{q}) = 0$, that is, $\mathbf{q} \in \Delta_{ij}$, for the x component,

$$\begin{aligned} \frac{\partial f(\mathbf{q})}{\partial \mathbf{q}_x} &= 2d_g(\mathbf{q}, \mathbf{g}_i) \frac{\partial d_g(\mathbf{q}, \mathbf{g}_i)}{\partial \mathbf{q}_x} - 2d_g(\mathbf{q}, \mathbf{g}_j) \frac{\partial d_g(\mathbf{q}, \mathbf{g}_j)}{\partial \mathbf{q}_x} \\ &= 2d_g(\mathbf{q}, \mathbf{g}_i) \frac{\partial \|\mathbf{q} - h^1_{\mathbf{q},\mathbf{g}_i}\|}{\partial \mathbf{q}_x} - 2d_g(\mathbf{q}, \mathbf{g}_j) \frac{\partial \|\mathbf{q} - h^1_{\mathbf{q},\mathbf{g}_j}\|}{\partial \mathbf{q}_x} \\ &= -2d_g(\mathbf{q}, \mathbf{g}_i) \frac{(\mathbf{q} - h^1_{\mathbf{q},\mathbf{g}_i})_x}{\|\mathbf{q} - h^1_{\mathbf{q},\mathbf{g}_i}\|} + 2d_g(\mathbf{q}, \mathbf{g}_j) \frac{(\mathbf{q} - h^1_{\mathbf{q},\mathbf{g}_j})_x}{\|\mathbf{q} - h^1_{\mathbf{q},\mathbf{g}_j}\|}, \end{aligned}$$

and, similarly, for the y component. Therefore, the normal to the boundary is

$$n'_{ij}(\mathbf{q}) = \left(\frac{\partial f(\mathbf{q})}{\partial \mathbf{q}_x}, \frac{\partial f(\mathbf{q})}{\partial \mathbf{q}_y} \right)$$

and the unit normal is

$$n_{ij}(\mathbf{q}) = \frac{n'_{ij}(\mathbf{q})}{\|n'_{ij}(\mathbf{q})\|}.$$

Introducing the last three equations in (6.11), we obtain

$$\frac{\partial \mathbf{q}}{\partial w_i} \cdot n_{ij} = \|n'_{ij}(\mathbf{q})\|,$$

and replacing in (6.6) we obtain the complete formulation of the gradient stated in (6.4). This gradient is distributed since each robot can update its own weight by only communicating with its neighbors. Finally, the proof of the convergence is equivalent to Theorem 3.7 in [73], completing the proof of this theorem. □

Remark 6.2.2. *The main differences of our proof with respect to the one presented in [73] reside in the derivation of the geodesic distance and in the calculation of the normal to the boundary points. These two key modifications generalize the algorithm to any kind of non-convex environments, yet still giving the same solution for convex ones.*

6.2.4 Energy-Aware Partition and Update

The algorithm to find the equitable partition that we have introduced simply divides the environment in regions with the same workload. However, the robots may have different capabilities and the workload assigned to each of them must be proportional to its capabilities with respect to the rest of the team. We can extend the algorithm to take this into account by including them in the cost function:

$$H = \sum_{i=1}^{N} \frac{\int_{\mathbf{\Omega}_i(\mathbf{p})} \rho_i^{\max} \alpha_i(\mathbf{q}, \mathbf{p}) \, \mathrm{d}\mathbf{q}}{\lambda_{\mathcal{P}_i(\mathbf{w})}},$$

where $\mathbf{p}$ is a virtual position to integrate the maximum coverage that each agent can provide.

In the same way as the capabilities, the energy level and the energy expenditure of the robots may also be different. If an evolution model for the expenditure is available, i.e., how the energy level decreases with the motion and coverage of the robot, it can be used to predict how fast each robot is running out of battery and include it in the initial partition. Nevertheless, for the general case, the initial partition can be updated depending on the actual energy level of the robots using

$$H = \sum_{i=1}^{N} \frac{e_i \int_{\mathbf{\Omega}_i(\mathbf{p})} \rho_i^{\max} \alpha_i(\mathbf{q}, \mathbf{p}) \, \mathrm{d}\mathbf{q}}{\lambda_{\mathcal{P}_i(\mathbf{w})}},$$

where $e_i \in [0, 1]$ represents the energy level of robot i.

Since persistent coverage is an infinite task, in the sense that it can never be considered complete, there is a situation in which updating the partitions can improve the performance of the team considerably: when one ore more robots stop covering to recharge or refuel. During this time, that can be significant, they do not provide coverage in their partitions and the optimal strategy for the whole team is to update the partition for the remaining robots and keep covering the entire environment. This requires again communication between the robots and is also applicable in the case of failure of one or more robots.

6.2.5 Partition Connectivity Maintenance

The partitions obtained with (6.3) are proven to be equitable. However, for non-convex environments, especially for complex ones, they might be disconnected, in the sense that in order to reach some points of the partition, a robot needs to go through the partitions of other robots. In this section we extend our partitioning algorithm and control the positions of the generators, that are still a degree of freedom, to reduce these disconnections.

Let us begin by defining the connected components, $\mathcal{P}_i^\ell(\mathbf{w})$, $\ell \in \{1, \dots, L_i\}$, $L_i \in \mathbb{Z}_+$, that form a partition:

$$\mathcal{P}_i = \bigcup_{\ell=1}^{L_i} \mathcal{P}_i^\ell.$$

Between these connected components we select the one with the highest workload:

$$\mathcal{P}_i' = \underset{\mathcal{P}_i^\ell}{\operatorname{argmax}}\, \lambda_{\mathcal{P}_i^\ell}.$$

Note that for connected partitions, $\mathcal{P}_i'$ coincides with $\mathcal{P}_i$. Then, we calculate the center of mass of $\mathcal{P}_i'$ as

$$\mathbf{g}_i^* = \frac{1}{\lambda_{\mathcal{P}_i'}} \int_{\mathcal{P}_i'} \mathbf{q}\, \lambda(\mathbf{q})\, d\mathbf{q}.$$

This point is the best point of the environment in which the generator $\mathbf{g}_i$ could be located to favor the connectivity of its partition. However, it cannot be moved straightaway in the direction $\overline{\mathbf{g}_i\mathbf{g}_i^*}$ since there are two restrictions that must be taken into account. The first one is that the motion of the generator must not hinder the evolution of the weights towards the equitable partition. Therefore, such motion can only be executed if it contributes to the minimization of (6.2.3) or, at least, it is not detrimental. The second restriction is that in a non-convex environment it has to follow the geodesic path from its current position to the center of mass to avoid sudden changes in the shape, size and workload of the partitions. If we call γ_i the shortest path from $\mathbf{g}_i$ to $\mathbf{g}_i^*$, we want the generator to move in the direction $\gamma_i(\mathbf{g}_i)$, that is the direction of the path at $\mathbf{g}_i$. In this context, the algorithm that we propose is presented in the following theorem.

Theorem 6.2.3. *The power diagram generated by $\mathcal{G}$ and $\mathbf{w}$ converges to an equitable power diagram and reduces the disconnections of the partitions under the distributed control law*

$$\dot{w}_i = -k_w \frac{\partial H}{\partial w_i}, \tag{6.12a}$$

$$\dot{\mathbf{g}}_i = k_g\, f_{sat}\left(\overline{\mathbf{g}_i\mathbf{g}_i^*} \cdot \frac{-\partial H}{\partial \mathbf{g}_i}\right) \gamma_i(\mathbf{g}_i), \tag{6.12b}$$

with k_w, k_g, positive gains,

$$f_{sat}(x) = \begin{cases} 0, & \forall\, x \le 0, \\ \exp\left(\dfrac{-1}{(k_{sat}\, x)^2}\right), & \forall\, x > 0, \end{cases} \tag{6.13}$$

a saturation function with $k_{sat} > 0$, and

$$\frac{\partial H}{\partial \mathbf{g}_{i,k}} = \sum_{j \in N_i} \left(\frac{1}{\lambda^2_{\mathcal{P}_j}} - \frac{1}{\lambda^2_{\mathcal{P}_i}} \right) \int_{\Delta_{ij}} \|n'_{ij}(\mathbf{q})\| \frac{(\mathbf{h}^{l_{\mathbf{q},\mathbf{g}_i}}_{\mathbf{q},\mathbf{g}_i} - \mathbf{g}_i)_k}{\|\mathbf{h}^{l_{\mathbf{q},\mathbf{g}_i}}_{\mathbf{q},\mathbf{g}_i} - \mathbf{g}_i\|} \lambda(\mathbf{q})\, d\mathbf{q}, \tag{6.14}$$

where $n'_{ij}(\mathbf{q})$ is the outward normal to the boundary between the partitions i and j at point $\mathbf{q}$.

Proof. The convergence to an equitable power diagram is guaranteed under the first part of the control law, (6.12a), in Theorem 6.2.1. In this proof we only show that the second part of the control law, (6.12b), does not counteract this convergence.

The direction of motion of the generators that maximizes the improvement of the cost function is the gradient of the cost function with respect to $\mathbf{g}_i$, i.e., $\partial H / \partial \mathbf{g}_i$. This gradient is obtained in the same way as with respect to w_i in the proof of Theorem 6.2.1 with only two particularities. The derivatives of the geodesic distances with respect to $\mathbf{g}_{i,x}$ are

$$\frac{\partial d_g(\mathbf{q}, \mathbf{g}_i)}{\partial \mathbf{g}_{i,x}} = \frac{\partial \|\mathbf{q} - \mathbf{h}^1_{\mathbf{q},\mathbf{g}_i}\|}{\partial \mathbf{g}_{i,x}} + \cdots + \frac{\partial \|\mathbf{h}^{l_{\mathbf{q},\mathbf{g}_i}}_{\mathbf{q},\mathbf{g}} - \mathbf{g}_i\|}{\partial \mathbf{g}_{i,x}} = \frac{-(\mathbf{q} - \mathbf{h}^1_{\mathbf{q},\mathbf{g}_i})}{\|\mathbf{q} - \mathbf{h}^1_{\mathbf{q},\mathbf{g}_i}\|} \cdot \frac{\partial \mathbf{q}}{\partial \mathbf{g}_{i,x}} - \frac{(\mathbf{h}^{l_{\mathbf{q},\mathbf{g}_i}}_{\mathbf{q},\mathbf{g}} - \mathbf{g}_i)_x}{\|\mathbf{h}^{l_{\mathbf{q},\mathbf{g}_i}}_{\mathbf{q},\mathbf{g}} - \mathbf{g}_i\|},$$

$$\frac{\partial d_g(\mathbf{q}, \mathbf{g}_j)}{\partial \mathbf{g}_{i,x}} = \frac{\partial \|\mathbf{q} - \mathbf{h}^1_{\mathbf{q},\mathbf{g}_j}\|}{\partial \mathbf{g}_{i,x}} = \frac{-(\mathbf{q} - \mathbf{h}^1_{\mathbf{q},\mathbf{g}_j})}{\|\mathbf{q} - \mathbf{h}^1_{\mathbf{q},\mathbf{g}_j}\|} \cdot \frac{\partial \mathbf{q}}{\partial \mathbf{g}_{i,x}},$$

and equivalent for $\mathbf{g}_{i,y}$. Note that in the first step of the first derivative all the intermediate terms represented with the dots are equal to zero since they do not depend on $\mathbf{g}_{i,x}$ or $\mathbf{q}$. The second particularity is that in the end we obtain

$$\frac{\partial \mathbf{q}}{\partial \mathbf{g}_{i,x}} \cdot n_{ij} = \|n'_{ij}(\mathbf{q})\| \frac{(\mathbf{h}^{l_{\mathbf{q},\mathbf{g}_i}}_{\mathbf{q},\mathbf{g}} - \mathbf{g}_i)_x}{\|\mathbf{h}^{l_{\mathbf{q},\mathbf{g}_i}}_{\mathbf{q},\mathbf{g}} - \mathbf{g}_i\|},$$

which leads to (6.14).

The saturation function in (6.12b) compares the direction of the gradient with the direction from the generator to the centroid of the biggest connected component. If both directions are approximately aligned, i.e., the movement of the generator to the centroid favors the convergence to the equitable partition, the generator is able to follow the geodesic path to the centroid according to $\gamma_i(\mathbf{g}_i)$. Otherwise, if the movement to the centroid hinders the convergence, $f_{sat} = 0$ and the generator does not move. Additionally, since the movement is to the centroid of the connected component with the biggest workload, it favors that the partition remains connected. □

Although the proposed control law converges to the equitable power diagram and tends to give connected partitions, the latter is not theoretically guaranteed. In the simulation section we evaluate this and show that the partitions rarely result disconnected. In any case, there are procedures to reassign the disconnected components to other robots such as [71] and finally obtain connected partitions at the expense of making them less equitable.

6.3 Finite-Horizon, Optimal Coverage Planning based on a Path Graph

The second part of our persistent coverage solution is in charge of planning online optimal coverage paths for the robots, $\Gamma_i(k)$. The objective of these paths should be to minimize

$$f(K) = \sum_{k=1}^{K} \int_{\mathcal{Q}} \Phi \left(Z^* - Z(k)\right)^2 d\mathbf{q}, \tag{6.15}$$

where $\Phi \equiv \Phi(\mathbf{q}) \in (0, 1]$ is the importance of covering each point. The squared term implies that overcovering a point is as undesirable as undercovering it although the coverage controller from (5.2) guarantees that the coverage does not increase unboundedly. However, solving this problem is NP-hard, and the very few attempts to solve it have demonstrated that the inclusion of a few robots and time steps in the planning horizon (< 5) already overwhelms current processors [118].

Since we divide the environment in equitable disjoint partitions, we can also separate the cost function according to the following proposition and reduce the complexity of the problem.

Proposition 6.3.1. *The paths that optimize* (6.15) *subject to an equitable partition resulting from Theorem 6.2.1 or Theorem 6.2.3 are the paths that optimize individually the cost function inside each partition, i.e.,*

$$\min_{\Gamma_i} f(K) = \min_{\Gamma_1} f_1(K) + \ \ldots \ + \min_{\Gamma_N} f_N(K),$$

with

$$f_i(K) = \sum_{k=1}^{K} \int_{\mathcal{P}_i} \Phi \left(Z^* - Z(k)\right)^2 d\mathbf{q}, \tag{6.16}$$

Proof. It is straightforward to see that

$$f(K) = f_1(K) + \ \ldots \ + f_N(K).$$

Since the path of each robot can only be planned inside its partition, the contribution of such path to the minimization of $f(K)$ is restricted to $\mathcal{P}_i$, that is represented by $f_i(K)$, completing this proof. □

The immediate consequence of this proposition is that each robot can plan locally its own optimal path and the cost function can be computed distributively, since each robot only needs the information of its own coverage, i.e., inside its own partition. Nevertheless, the problem remains significantly complex due to the dependency of the coverage of a point with the actions of the robot in the previous instants. To counteract this complexity, we define the current coverage error at time k at each point as

$$e(\mathbf{q}, k) = \Phi \left(Z^* - Z(k)\right)^2. \tag{6.17}$$

This metric allows us to efficiently plan the coverage paths as a function of the coverage error of the environment and to simplify the temporal complexity of the coverage while planning.

The coverage solution that we propose is to initially define a graph of sweep-like paths and at each time finding the optimal path through the graph in terms of the current coverage error from (6.17).

6.3.1 Graph Construction based on Sweep-like Paths

In order to define the paths that the robots can follow without colliding with the obstacles of the environment, we define the reduced partitions as

$$\mathcal{P}'_i = \mathcal{P}_i \cap \mathcal{Q}_f.$$

These reduced partitions, that we will refer to in this section simply as partitions, result from intersecting the initial partitions with the points of the environment in which the robot can be located without colliding with an obstacle.

Inside the partitions we separate the construction of the path graph in two parts: grid and boundary. The aim of the first part is to define paths that allow the robot to perform a complete sweep of the partition. To this end, we intersect a grid of points with the partition:

$$\mathcal{V}_i^g = \{\mathbf{v} \in \mathcal{P}'_i \,|\, \mathbf{v} = (x_i^0 \pm k\, d_i, y_i^0 \pm \ell\, d_i), k, \ell \in \mathbb{Z}\},$$

where $\mathbf{v}_i^0 = (x_i^0, y_i^0) \in \mathcal{P}'_i$ is the left-most, top point of the partition and d_i is the optimal separation between consecutive sweeping lines. These vertices are connected by an edge if they share the $x-$ or y-coordinate and are separated in the other by d_i. Formally,

$$\mathcal{E}_i^g = \{(\mathbf{v}_1, \mathbf{v}_2) \,|\, \|\mathbf{v}_1 - \mathbf{v}_2\| = d_i,\, \forall\, \mathbf{v}_1, \mathbf{v}_2 \in \mathcal{V}_i^g\}.$$

Figure 6.2a shows an example of the vertices and edges that correspond to the grid part of the path graph.

Remark 6.3.2. *The optimal separation between two horizontal or two vertical sweeping lines, d_i, depends on the shapes of the coverage area and the production function. For a circular area and a constant production, $d_i = 2\, r_i^{cov}$. For non-circular areas it is two times the maximum width of the area in the perpendicular direction to the movement of the robot. For non-constant areas it can be obtained by minimizing*

$$\int_{-r_i^{cov}}^{r_i^{cov}} \int_{0}^{d_i} \frac{1}{\big(\alpha_i((x,y),(0,0)) + \alpha_i((x,y),(0,d_i))\big)^2} dx\, dy, \tag{6.18}$$

as illustrated in Fig. 6.1.

The idea behind (6.18) is that, if the coverage area of the robot overlaps in two sweeps, the sum of the coverage provided in the two sweeps is as similar as possible for all the points in the perpendicular direction. For this reason, the inverse of the sum of the coverage of

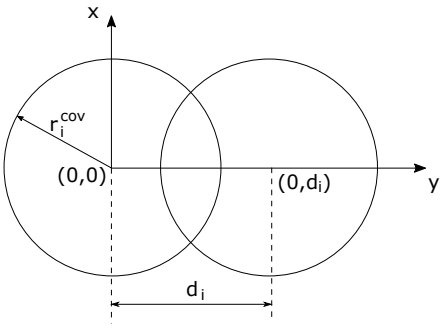


Figure 6.1: Schematic of the overlap of the coverage areas for two consecutive sweeps in the x direction.

both sweeps is integrated over the area between them. This is represented in Fig. 6.1 where the robot sweeps in the x direction at $x = 0$ and $x = d_i$.

The second part of the graph is intended to cover the entire boundary of the partition. We represent the boundary as $\partial\mathcal{P}_i'$ and locate a finite number of vertices over it as follows:

$$\mathcal{V}_i^b = \{\mathbf{v} \in \partial\mathcal{P}_i' \,|\, \mathbf{v}_x = x_i^0 \pm k\,\frac{d_i}{2} \vee \mathbf{v}_y = y_i^0 \pm \ell\,\frac{d_i}{2}, k, \ell \in \mathbb{Z}\},$$

They are located at the intersections of a square grid of size $d_i/2$ with the boundary, to provide a sufficiently fine discretization of it. These vertices are linked by the edges

$$\mathcal{E}_i^b = \{(\mathbf{v}_1, \mathbf{v}_2) \,|\, \mathbf{v}_1 \text{ and } \mathbf{v}_2 \text{ are consecutive over } \partial\mathcal{P}_i'\}.$$

These edges allow the robots to follow the entire boundary of the partition as shown in Fig. 6.2b.

Finally, we merge the two parts by including an additional set of links,

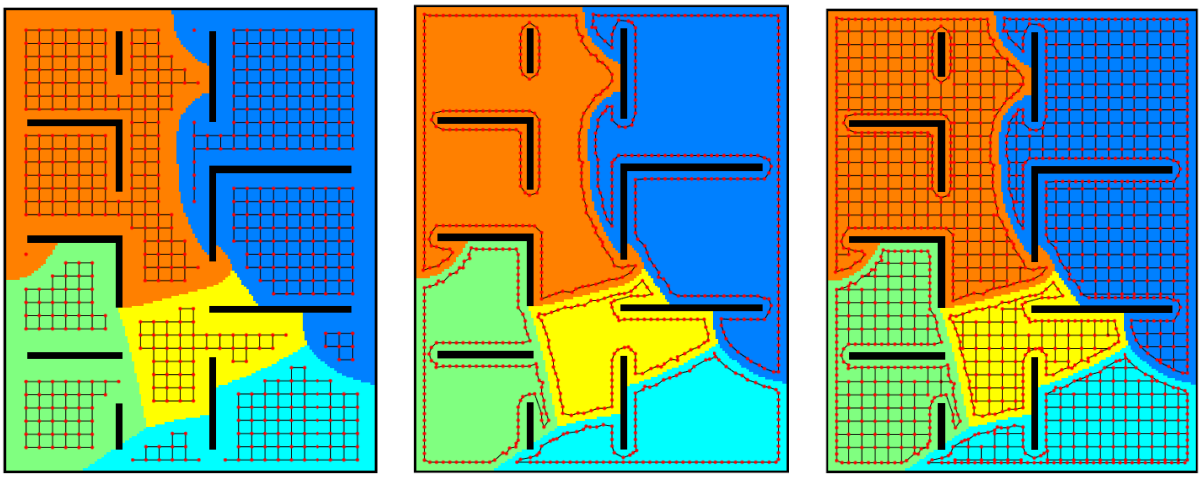
$$\mathcal{E}_i^{gb} = \{(\mathbf{v}_1, \mathbf{v}_2) \,|\, |\mathbf{v}_1^x - \mathbf{v}_2^x| < d_i \wedge |\mathbf{v}_1^y - \mathbf{v}_2^y| = 0 \vee |\mathbf{v}_1^x - \mathbf{v}_2^x| = 0 \wedge |\mathbf{v}_1^y - \mathbf{v}_2^y| < d_i, \\ \forall\, \mathbf{v}_1 \in \mathcal{V}_i^g, \mathbf{v}_2 \in \mathcal{V}_i^b\}.$$

As can be seen in Fig. 6.2c, these edges link the vertices of both parts that are closer than d_i either in the x- or in the y-axis and the resulting directed path graph can be expressed as

$$\mathcal{G} = (\mathcal{V}_i, \mathcal{E}_i) \equiv (\mathcal{V}_i^g \cup \mathcal{V}_i^b, \mathcal{E}_i^g \cup \mathcal{E}_i^b \cup \mathcal{E}_i^{gb}).$$

6.3.2 Optimal Path Planning

In order to maintain the coverage as close as possible to the desired level, we propose a planning approach in which each robot minimizes the current coverage error $e(\mathbf{q}, k)$ inside its own partition. Optimizing the current error might be suboptimal in terms of (6.16). However, it reduces drastically the complexity of the planning problem by eliminating the



(a) Grid. (b) Boundary. (c) Complete graph.

Figure 6.2: Example of graph construction.

temporal dependency. In fact, thanks to the path graph introduced before and the optimal separation between the nodes, this problem reduces to finding the set of ordered nodes that provides the biggest reduction of such error to the motion of the robots. Our solution can be found very efficiently with a well-known method and still guarantees that the path is optimal in terms of the current error. In fact, the resulting path can cover the entire partition if that is the optimum.

In the first place, let us transform the graph in a weighted digraph by defining the weights of the edges:

$$\omega_i(\mathbf{v}_1, \mathbf{v}_2, k) = \max\big(e(\mathbf{v}_2, k), 0\big), \quad \forall\,(\mathbf{v}_1, \mathbf{v}_2) \in \mathcal{E}_i.$$

Therefore, traversing each edge adds to the path the current error at the head of the edge. Note that these weights are fixed for the k of the planning instant although they vary over time with the coverage.

In the second place, we define the error per number of visited vertices of the path as

$$g_i\big(\Gamma_i(\mathbf{v})\big) = \frac{\sum_{\ell=1}^{L_v} \omega_i(\mathbf{v}_{\ell-1}, \mathbf{v}_\ell, k)}{L_v},$$

where L_v is the number of traversed vertices to reach $\mathbf{v}$ through its optimal path.

In this context, we pose the problem of planning the optimal path as

$$\underset{\Gamma_i(\mathbf{v})}{\text{maximize}} \quad g_i\big(\Gamma_i(\mathbf{v})\big),$$

that is, finding the path to the vertex of the graph that leads the robot to the maximum reduction of the error along it.

The transformation of the coverage path planning into finding the optimal path through a graph, allows the problem to be solved rapid and efficiently with proven methods to find the shortest paths in graphs. In particular, we adapt the Bellman-Ford algorithm [140]

in two ways. Instead of the shortest distance, we look for the highest accumulated error per vertex and, therefore, we initialize our metric to $-\infty$ for all the nodes of the graph. Additionally, since the aim is to maximize this metric, the path to a vertex is updated and the vertex appended to the priority queue if the previously stored value of the metric is lower than the new one. These modifications do not alter the optimality of the method, that is still guaranteed to find the optimal paths, and also maintain the worst-case complexity of $\mathcal{O}(|\mathcal{V}_i|\,|\mathcal{E}_i|)$.

6.4 Simulation Results

In this section we present simulation results for the partitioning algorithms and the complete approach to the persistent coverage problem. For the partitioning, we consider five different, non-convex environments of 10x8m and increasing complexity. We call them *Open Rooms, Rooms, Spiral*, *Maze* and *Random*, respectively. The last one is an environment with a 25% of randomly occupied cells. They can be seen in Fig. 6.4.1. These simulations have been carried out with $N = 5$ robots of size $r_i = 0.1m$, a circular coverage area Ω_i of radius $r_i^{cov} = 0.2m$, a production $\alpha_i = 1$, $\forall\, \mathbf{q} \in \Omega_i$, and $\rho_i^{\max} = 20$. The objective, decay and importance of the coverage are

$$
\begin{aligned}
Z^* &= 80 + 20\,\frac{\mathbf{q}_y}{|\mathcal{Q}|_y}, \\
d &= 0.999 - 0.005\,e^{\left(-\frac{\left(\mathbf{q}_y - \frac{|\mathcal{Q}|_y}{2}\right)^2}{8\,|\mathcal{Q}|_y} - \frac{\left(\mathbf{q}_x - \frac{|\mathcal{Q}|_x}{2}\right)^2}{8\,|\mathcal{Q}|_x}\right)}, \\
\phi &= 0.5 + 0.5\,\frac{\mathbf{q}_y}{|\mathcal{Q}|_y},
\end{aligned}
$$

where $\mathbf{q}_x$ and $\mathbf{q}_y$ are the x- and y-coordinates of point $\mathbf{q}$ and $|\mathcal{Q}|_x$ and $|\mathcal{Q}|_y$ are the x- and y-sizes of the environment $\mathcal{Q}$. The resulting work function $\lambda(\mathbf{q})$ can be seen in Fig. 6.3 for the *Rooms* environment, normalized by $\int_{\mathcal{Q}} \lambda(\mathbf{q})\, d\mathbf{q}$ and multiplied by 100. The upper part of the environment requires less work because of its lower importance and objective and the central and lower parts require more work due to the peak of the decay in the center and the higher importance and objective at the bottom.

The positive constant of the saturation function (6.13) is set to $k_{sat} = 3$ and, in order to speed up the convergence of the partitions, the gains of the control law are different for the different environments and are changed online depending on the value of $\partial H/\partial w_i$. In particular, k_g takes the values from Table 6.1 and k_w is assigned according to

$$
k_w = \begin{cases} k_w^1, & \text{if } |\partial H/\partial w_i| > \alpha_1, \\ k_w^2, & \text{if } \alpha_1 \leq |\partial H/\partial w_i| < \alpha_2, \\ k_w^3, & \text{if } \alpha_2 \leq |\partial H/\partial w_i| < \alpha_3, \\ k_w^4, & \text{if } |\partial H/\partial w_i| \leq \alpha_3, \end{cases} \tag{6.19}
$$

with the values of the constants presented in Table 6.2.

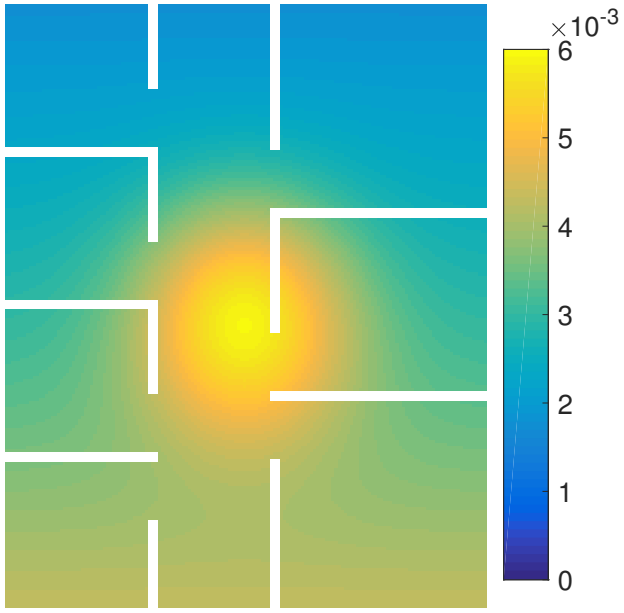


Figure 6.3: Normalized work function $\lambda(\mathbf{q})$ over the *Open Rooms* environment. White areas represent obstacles.

Table 6.1: Values for k_g depending on the environment type

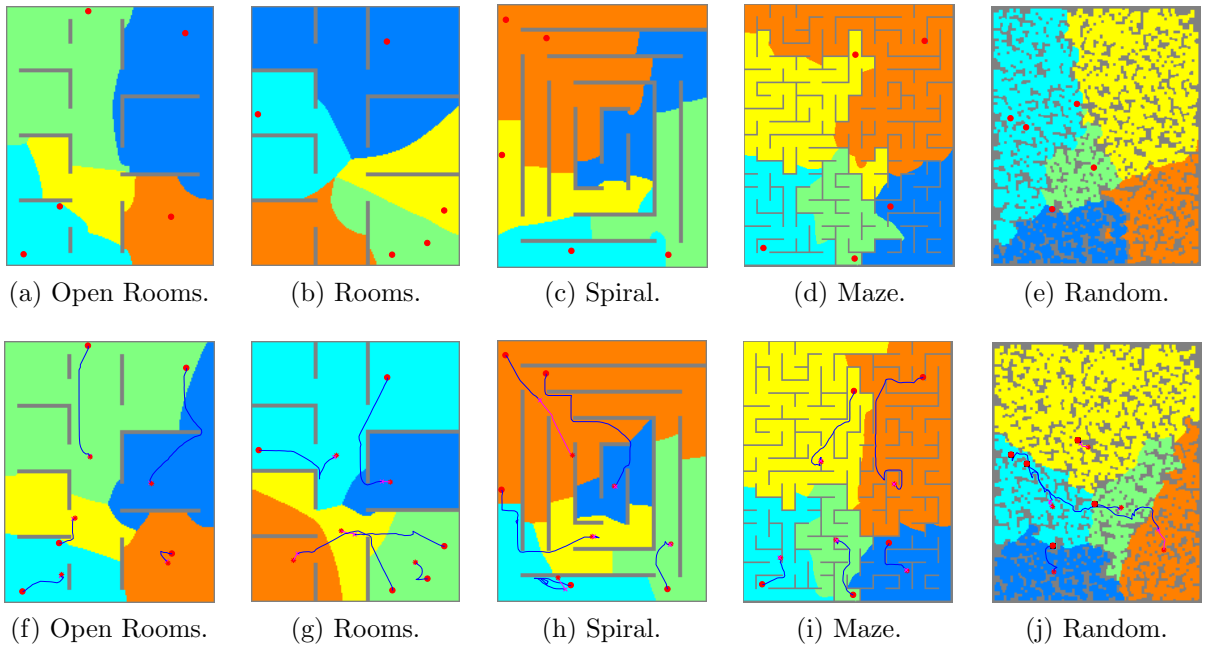
Open Rooms	*Rooms*	*Spiral*	*Maze*	*Random*
0.1	0.1	0.01	$5 \cdot 10^{-4}$	0.05

Table 6.2: Values for k_w depending on the environment

	Open Rooms	*Rooms*	*Spiral*	*Maze*	*Random*
k_w^1	0.01	0.02	0.01	0.01	0.01
k_w^2	0.2	0.4	0.1	0.1	0.1
k_w^3	1	2	0.5	1	1
k_w^4	10	20	2.5	10	10
α_1	$5 \cdot 10^{-3}$	$5 \cdot 10^{-3}$	0.01	0.01	0.01
α_2	$3 \cdot 10^{-4}$	$2 \cdot 10^{-4}$	10^{-3}	10^{-3}	10^{-3}
α_3	10^{-5}	10^{-5}	10^{-4}	10^{-4}	10^{-4}

6.4.1 Partitioning Example

In the first place we present an example of the partitioning algorithms in the five different environments. In the top row of Fig. 6.4.1 we show the resulting equitable partitions



(a) Open Rooms. (b) Rooms. (c) Spiral. (d) Maze. (e) Random.

(f) Open Rooms. (g) Rooms. (h) Spiral. (i) Maze. (j) Random.

Figure 6.4: Example of partitioning in the five different environments under control law (6.3) (top row) and under control law (6.12) (bottom row). The resulting partitions of the robots are shown in different colors. The initial locations of the generators are represented with red crosses; the final location with magenta crosses; and the locations of the centroid with black crosses. Blue lines represent the path of the generators until the convergence and magenta lines, the remaining geodesic paths from the generators to the centroids.

under (6.3), i.e., when only the weights are modified and the positions of the generators remain fixed. Although in some cases such as Fig. 6.4a or 6.4b the resulting partitions are connected, it can be seen that for more complex scenarios such as the *Spiral* (Fig. 6.4c) or the *Maze* (Fig. 6.4d) they are not. These disconnections are avoided under (6.12) as can be seen in the bottom row of Fig. 6.4.1. In this case, moving the generators to the centroids of the connected components with the highest workload at the same time as the weights are changing, allows the robots to find equitable and connected partitions.

Now we focus in the particular example of the *Maze* environment to assess the evolution and convergence under both (6.3) and (6.12). In Fig. 6.5 we represent the evolution of the workload, the weight and the gradient of the cost function with respect to the weight for each robot in a different color under (6.3). It is interesting to see that initially the evolution of the weights and the workload is slow although the gradient of the cost function is high. At iteration 295, k_w is increased from 0.01 to 0.1, according to (6.19) and Table 6.2. Thanks to this, the weights vary faster and the evolution of the workload is sped up. The same happens at iteration 375 and allows the team to finally converge to the equitable partition in 641 iterations. Convergence is assumed when the maximum difference between the workload of the partitions is lower that a 5%.

Under (6.12), in the beginning the weights and workloads also evolve slowly as shown

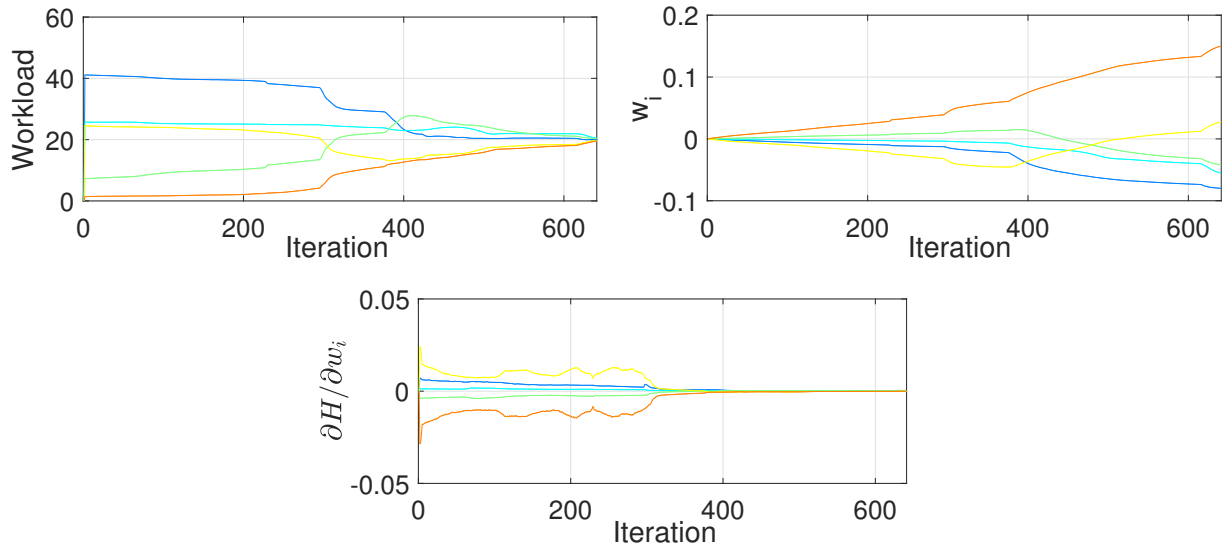


Figure 6.5: Evolution of the workload, weight and gradient of the cost function with respect to the weight for the *Maze* environment under control law (6.3). Different colors represent the variables for the different robots.

in Fig. 6.6. This evolution is sped up at iterations 290 and 506 when the value of k_w increases. Apart from that, the value of f_{sat} in Fig. 6.6d shows that the generators do not move in the 375 initial iterations. This means that up to this point their movement would not have improved the partitioning. At iteration 375 the generators start to move and the movement makes the workload change faster than before. This also makes the weights adapt to the new positions of the generators and, for this reason, they also adjust rapidly. Then, the movements stop between iterations 431 and 520, where the weights almost make the partitions converge. In the final part, the positions of the generators and the weights are adjusted to finally reach the equitable partition.

The evolution with (6.12) is not as smooth as with (6.3) but the all the resulting partitions are connected and, in this case, the locations of the generators coincide with their respective centroids. In the following section we carry out a parametric analysis to be able to generalize the results.

6.4.2 Parametric Analysis of the Partitioning Algorithms

We analyze now the evolution of the performance of the partitioning algorithm when the parameters of the problem change. In particular, we performed a Monte Carlo analysis of 10 runs of the algorithm for different initial positions of the generators and different work functions in the five different environments. In order to represent the results concisely, we refer to the different environments by their initials as follows: *Open Rooms* as OR, *Rooms* as R, *Spiral* as S, *Maze* as M and *Random* as Ra. The 10 initial positions of the generators were selected randomly and the variation of the work function was done through the decay function. In (6.4) the decay is set to be a 2D gaussian centered at $|\mathcal{Q}|_x/2$ and $|\mathcal{Q}|_y/2$, i.e., in

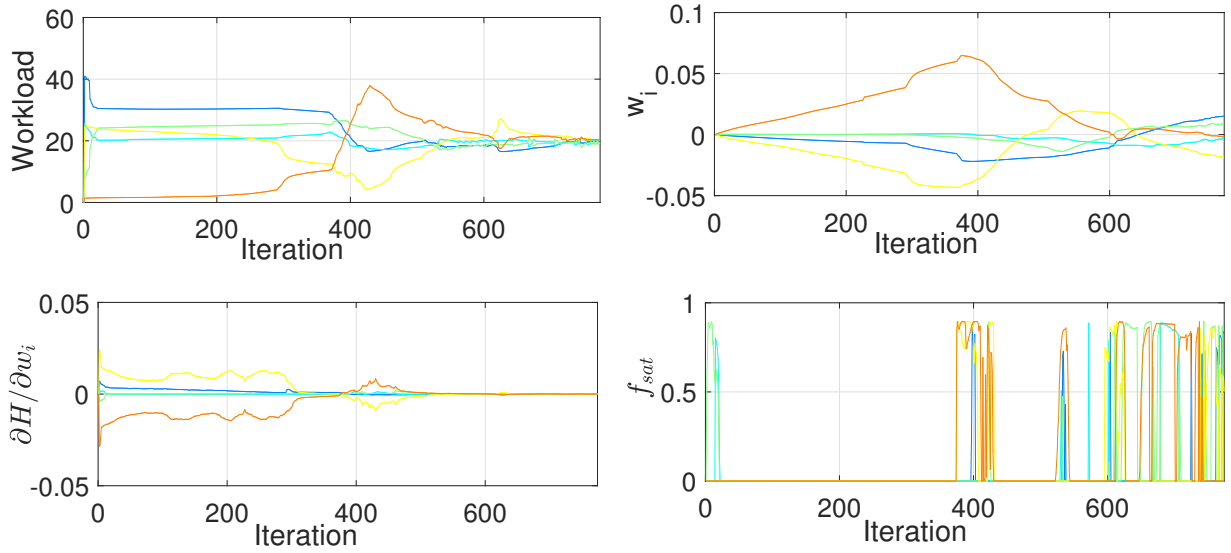
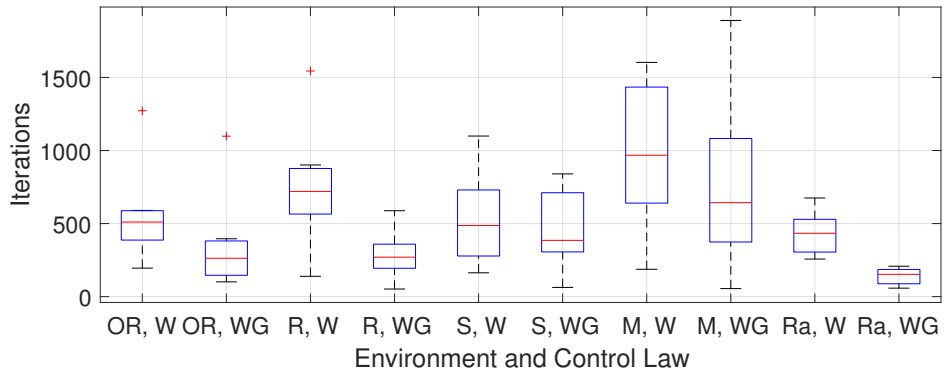


Figure 6.6: Evolution of the workload, weight, gradient of the cost function with respect to the weight, and saturation function for the *Maze* environment under control law (6.12). Different colors represent the variables for the different robots.

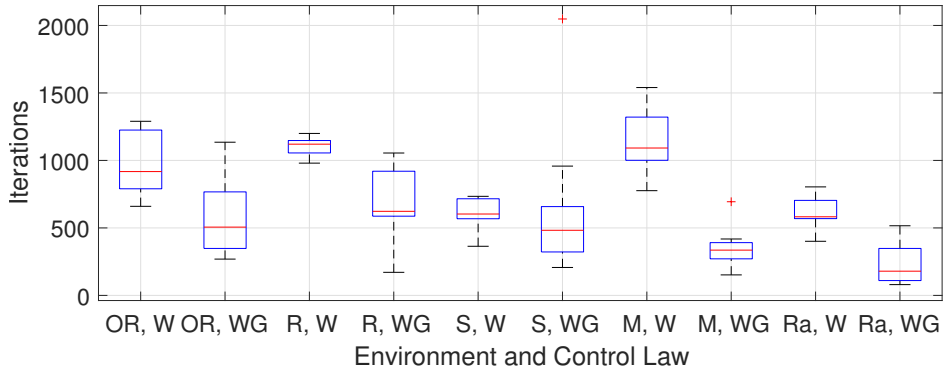
the middle of the environment. Instead of this center, we chose a random point for each trial. We execute this analysis for (6.3) and (6.12) and refer to them as W and WG respectively, representing that only the power weights or the power weights and the generator positions are modified.

In the first place we show in Fig. 6.7 two boxplots of the number of iterations required for the algorithm to converge to the equitable partition. Fig. 6.7a shows the results for different initial positions of the generators and Fig. 6.7b, for different decay functions. The most important feature of this results is that, for the same environment, WG converges much faster than W. Between the different environments, the speed of convergence depends on the complexity of the environment and on the tuning of the gains. The results show that under W, the *Rooms* and the *Maze* environments required more time to converge since their non-convexities are bigger and more complicated. Nevertheless, this does not hold under WG because the movement of the generators influences greatly the speed of convergence. Another interesting feature is on the *Random* environment. Although it seems to be the most complicated, the convergence is really fast because every two points of the environment are connected through many different paths.

In the second place we pay attention to the connectivity of the partitions. Table 6.3 gathers the number of partitions that resulted disconnected out of the 50 regions corresponding to each cell. As expected, this number is drastically reduced with the WG control law. In fact, for the easiest environments, *Open Rooms* and *Rooms*, it converges to connected partitions in all cases. However, for the most complicated, in some cases the partitions result disconnected. It is particularly surprising the number of disconnected partitions in the *Random* environment for which a great rate of connectivity was expected.



(a) Different initial positions.



(b) Different decay functions.

Figure 6.7: Boxplot of the number of iterations required to converge to the equitable partition.

Table 6.3: Number of disconnected partitions

	Different Positions		Different Decays	
	W	WG	W	WG
Open Rooms	5	0	10	0
Rooms	25	0	18	0
Spiral	17	3	13	7
Maze	27	9	26	10
Random	18	6	28	13

To evaluate the nature and the quality of the still non-connected partitions in Fig. 6.8 we represent in percentage the relative workload of the biggest connected component with respect to the workload of the entire partition. One can see that under WG, this value is on average above the 93% in the three environments with disconnected partitions. This means that almost all the work of the partition is on the biggest component, whose centroid

was followed by the generator. Therefore, a simple reassignment of the smaller disconnected regions could be done to reach total connectivity with only a small deviation from the equitable partition.

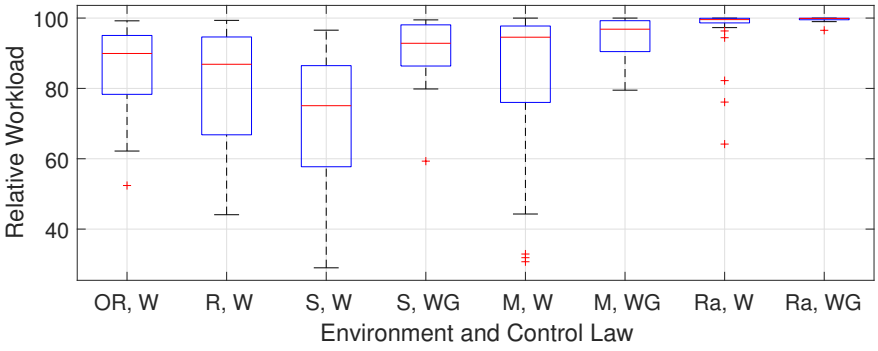
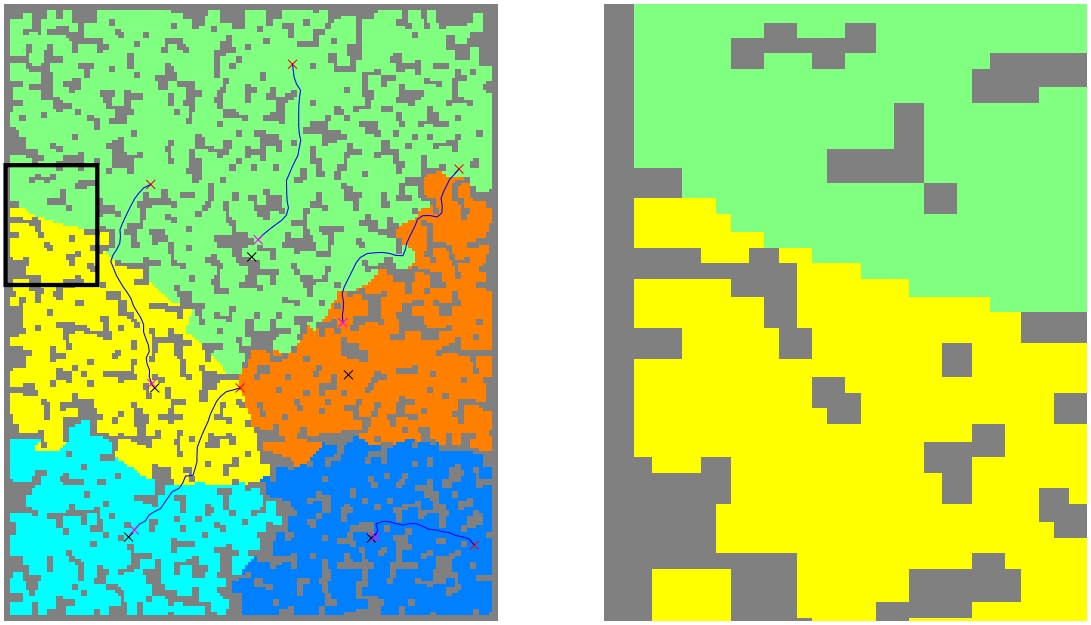


Figure 6.8: Boxplot of the relative workload of the biggest connected component with respect to the workload of the entire partition.



(a) Converged equitable partition.

(b) Zoom of the disconnected component of the yellow partition.

Figure 6.9: Example of convergence to the equitable partition with one disconnected region. The rectangle in (a) bounds the area zoomed in (b).

From the two previous results it follows that in the *Random* environment there are many disconnected partitions but more than the 99.5% of the work is on the biggest component. This happens due to the discretization of the environment for the computation, as can be seen in Fig. 6.9. In Fig. 6.9a we show an example of a converged partition. Apparently all

the partitions are connected but if we enlarge the area within the rectangle, Fig. 6.9b, we can see that a small region of the yellow partitions is disconnected from the rest. However, this small region only represents the 0.11% of the workload of the partition.

6.4.3 Coverage Planning Results

In this part of the simulations we analyze the coverage planning solution and the performance of the entire approach with an example. Fig. 6.10 shows the partition of the *Rooms* environment and the path graphs of the robots. According to the work map from Fig. 6.3, the robot assigned to the blue partition has to cover a bigger area than the other four. These other four share the peak of the work map and each of them is additionally in charge of a big room on the right-hand side or one and a half small rooms on the left-hand side.

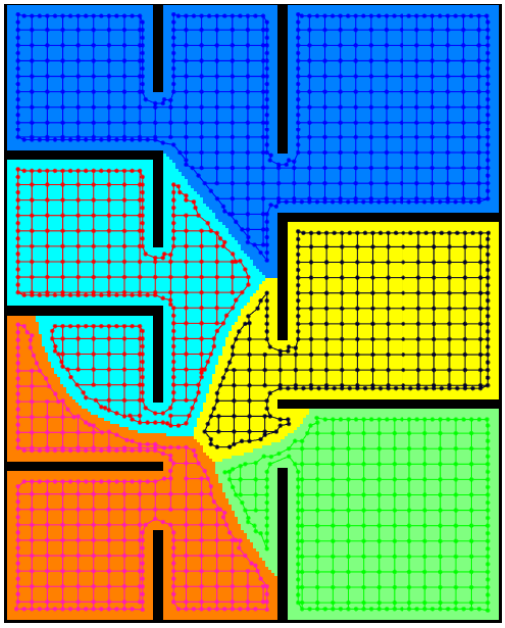


Figure 6.10: Example of partitions and path graphs for the *Rooms* environment.

In order to evaluate the quality of the paths and the provided coverage we normalize the coverage error at a single point at time k,

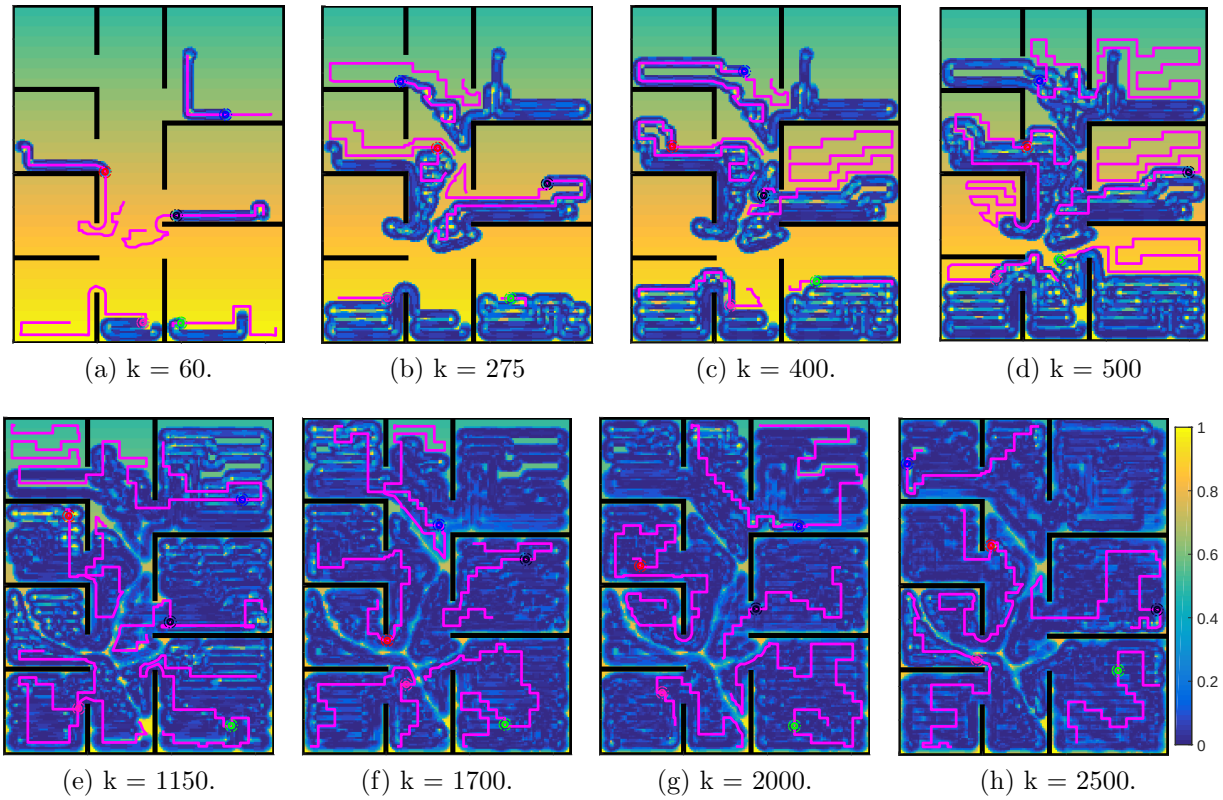
$$\varepsilon(k) \equiv \varepsilon(\mathbf{q}, k) = \frac{e(k)}{Z^{*2}},$$

and calculate the mean normalized error over the environment as

$$\bar{\varepsilon}(k) = \frac{\int_{\mathcal{Q}_f} \varepsilon(k) d\mathbf{q}}{|\mathcal{Q}_f|}.$$

In Fig. 6.11 we show the coverage error $e(\mathbf{q}, k)$ of the environment for eight different time instants. In Fig. 6.11a the robots are starting to cover the environment and, therefore, the

coverage error is initially proportional to the importance of the points and their objective. The paths of the top row, Fig. 6.11a-6.11d, represented in magenta, cover the partitions from bottom to top in a sweep-like manner. This demonstrates that the paths go through the points with the highest coverage error In Fig. 6.11e, almost all the robots have already covered their partitions for the first time. The blue robot needs a little more time since it has the biggest partition. The paths of the robots in Fig. 6.11e and 6.11f, fill the gaps that they have previously left uncovered. At this point, the coverage error is already small in the majority of the points. Eventually, in the steady state, Fig. 6.11g-6.11h, the robots keep moving to maintain the coverage as close as possible to the objective, paying special attention to the zones where more work is required.



(a) k = 60. (b) k = 275 (c) k = 400. (d) k = 500

(e) k = 1150. (f) k = 1700. (g) k = 2000. (h) k = 2500.

Figure 6.11: Example of evolution of the coverage. The background color map represents the value of $\varepsilon(k)$. The robots are depicted in different colors with their coverage areas as dash dotted circumferences. Magenta lines show the current paths of the robots.

The performance of our coverage strategy is assessed in Fig. 6.12a. We depict the mean coverage error and its standard deviation for the simulation. One can see that, at the beginning, when the environment is totally uncovered, the error is around 0.7 due to the importance function. After that, it decreases rapidly and converge to a small steady-state value. In fact, on average, the points have under a 7% of error. The standard deviation increases at the beginning because the undercovered points keep deteriorating and some others become slightly overcovered when the robots go over them. However, it also decreases

to a small steady-state value around 0.09.

A similar tendency persists inside the partitions of the robots as shown in Fig. 6.12b, where $\bar{\varepsilon}(k)$ is calculated only within $\mathcal{P}_i$. Nevertheless, it is interesting to see the differences between the biggest partition, the blue one, and the smaller ones, the magenta and green ones. The error in the beginning in the magenta and green ones is higher because the importance of the points inside them is higher, as opposed to the blue one. On the other hand, the reduction of the error in those partitions is faster because they are smaller and the robots need less time to cover them completely. On the contrary, the blue robot needs more time to cover its region and, therefore, its error decreases slower.

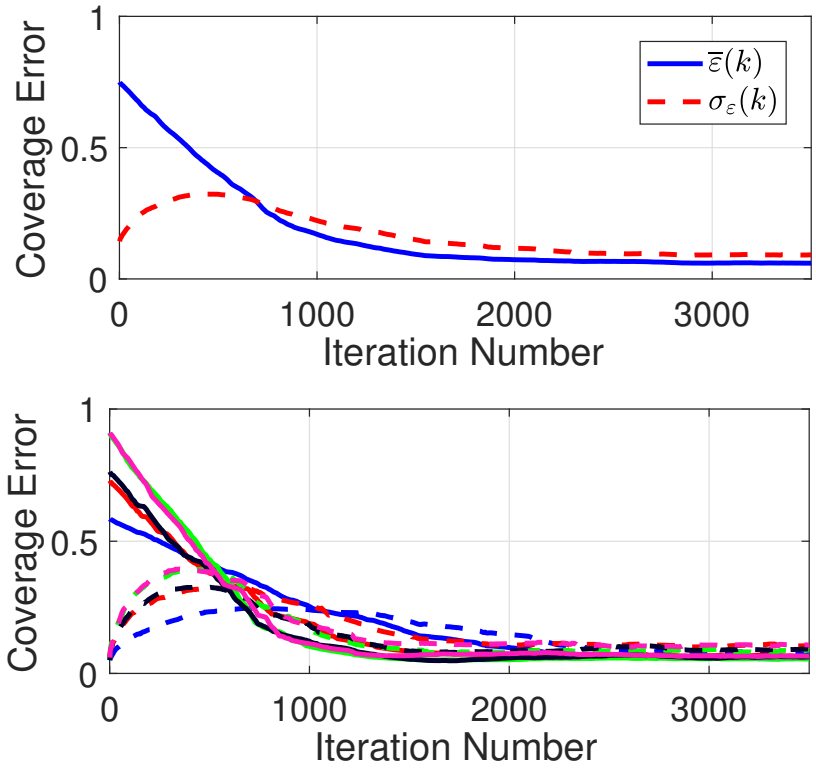


Figure 6.12: Evolution of the mean coverage error (solid lines) and its standard deviation (dashed lines). (Top) Complete environment. (Bottom) Separated by partitions.

Although the coverage error is the metric that defines the optimality of the coverage, it is difficult to visualize. A more intuitive representation is shown in Fig. 6.13. The mean coverage level of the environment increases rapidly towards the mean coverage objective. However, its steady-state value is smaller due to the different importance of the points. This happens because the robots pay more attention to the most important points at the expense of leaving the least important worse covered. For this reason, the mean value lays between the mean coverage objective and the mean coverage objective weighted by the importance.

Finally, we show in Fig. 6.14 the number of times that the robots covered each point of the environment. The resemblance with the work map make it clear that the robots visit more frequently the most important points and those with a lower decay, because the

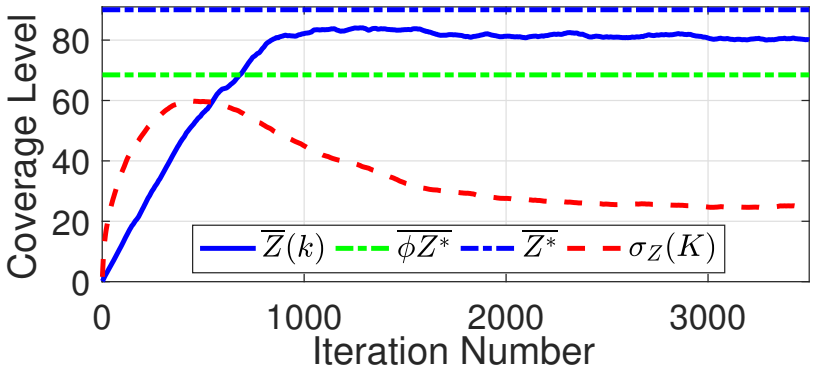


Figure 6.13: Evolution of the mean coverage level of the environment (blue line) with its standard deviation (red dashed line). The blue dash-dotted line represents the mean coverage objective and the green dash-dotted one, the mean coverage objective weighted by the importance of each point.

coverage deteriorates faster on them.

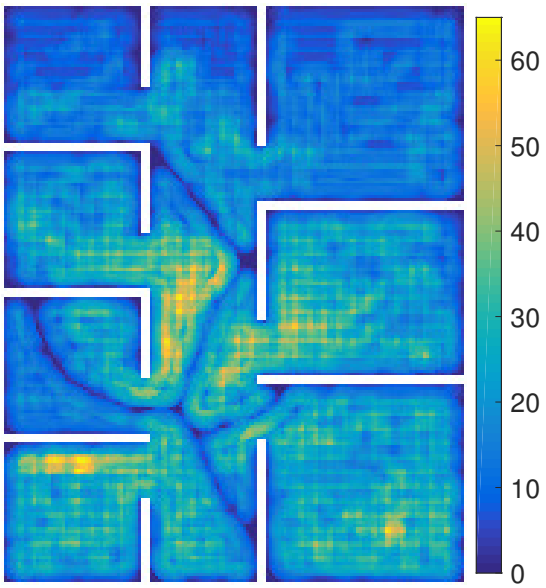


Figure 6.14: Number of times that the robots have covered each point of the environment.

6.5 Conclusions

In this chapter we have proposed a partitioning approach based on power diagrams along with a graph-based coverage planning strategy to solve the persistent coverage problem in complex, non-convex environments. We have introduced in the first place an algorithm to find the power weights that correspond an equitable partition of the environment. This

strategy can be applied to any kind of environment thanks to the geodesic distance derivation that we introduce. Moreover, this distance allows the partitions to take into account the shape of the environment. We have also presented two extensions for this strategy. The first one is an energy-aware update of the partitions. According to the battery level of the robots, the partitions can be modified to keep them equitable. This results specially appealing when one or more robots stop covering to recharge or they simply fail. The second extension drives the generator of each partition to the centroid of the connected component with the highest workload. The objective of this strategy is to reduce the disconnections of the partitions to the extent possible. In the second place, we have presented a planning algorithm that allows each robot to locally find the optimal path in terms of the current coverage error. This algorithm is based on the construction of a path graph that covers the entire partition of the robot with sweep-like paths. We assign the coverage errors of the head nodes to the edge weights and, therefore, the problem reduces to finding the optimal path through a graph, that can be achieved efficiently with state-of-the-art methods. Finally, we include simulation results to support our contributions and show how our solution can be applied to a real world scenario.

So far in this thesis, we have addressed the persistent coverage problem in all kind of continuous environments with different settings. Most of our solutions and, more importantly, they have demonstrated to be applicable to real world scenarios and perform very well. The next step, gathered in the second objective of this thesis, is to explore solutions for discrete environment which are applicable to domestic induction heating.

Chapter 7

Cooperative, Periodic Coverage for Discrete Environments

In this chapter, the robots have to visit periodically a discrete set of points of interest and spend some time on them carrying out the coverage task. We use a divide-and-conquer strategy and split the problem into three smaller subproblems to counteract its complexity. In the first place, we plan individual infinite-horizon paths for the robots to cover all the points periodically. Secondly, we formulate a quadratically constrained linear program to find the optimal coverage times and actions that satisfy the coverage objective. The third step is to join together the individual plans of the robots in a periodic team plan by obtaining a schedule with a MILP that guarantees collision avoidance. Eventually, we present thorough simulation results to assess the performance of our solution.

7.1 Introduction

One of the main contributions of this thesis is to particularize the persistent coverage problem to the application of domestic induction heating with mobile inductors. This application introduces some particularities to the setting in which the problem has to be solved and also some restrictions to the movement of the robots. The most important change with respect to the previous chapters is that the environment is now discrete, since only a finite set of points has to be covered. This environment is limited to the appliance itself and, therefore, a distributed solution is no longer necessary since all the computations are carried out in the same processing unit. On the other hand, the space is cluttered due to the relative size of the inductors and collision avoidance becomes a priority. Also the limited length of the robotic arms to which they are attached prevents them from reaching all the points of the hob or, equivalently, reduces their reachability.

In discrete environments, where only a finite set of points requires coverage, approaches to persistent coverage [87] are related to Task Assignment [141] and Vehicle Routing Problems (VRP) as addressed in operational research [142]. However, most of the these approaches assume that the times required to complete the coverage task at each point are known or neglect them. This is not acceptable in problems such as heating, where the coverage action

of the robot is not instantaneous.

In this chapter we propose a general solution to persistently cover a finite set of points with a team of robots, which must spend some time at each point to improve its coverage. We find a periodic strategy for the team, composed of individual periodic paths, and coverage times and actions at each point, that guarantees the satisfaction of the coverage objective for all the points, optimizes the actions of the robots and avoids collisions between them. To do so we use a divide-and-conquer strategy and separate the problem into three subproblems:

(i) Plan individual, closed paths for the robots that cover all the points of the environment.

(ii) Calculate the coverage time and coverage action of each robot at each point of its path to satisfy the coverage objective.

(iii) Scheduling the start of the robot paths to obtain a team plan in which collisions are avoided.

We do not elaborate specifically on the first subproblem since it reduces to solve an instance of the Traveling Salesman Problem (TSP) for which there are state-of-the-art solutions. Therefore, the most important contributions of this chapter are:

- A strategy to calculate the time and the coverage that each robot has to provide at each point, depending on the different production rates and the demands of the points, by solving a quadratic program. In this scenario we provide sufficient conditions for the existence of a solution to the problem.
- An iterative algorithm that refines the original solution, removing the points with coverage times equal to zero from the paths of the robots, thus letting the robots spend more time working at other points.
- A modified version of the cost function that intrinsically accounts for the length of the trajectories, yet still requiring to solve the same quadratic program.
- A procedure to schedule the robot paths that provides collision avoidance guarantees and minimizes simultaneous movements of the robots.

The remainder of the chapter is structured as follows. In Section 7.2 we introduce the formulation of this particular version of the problem. We calculate the optimal coverage times and actions in Section 7.3 with predefined paths and propose an iterative algorithm in Section 7.4 to shorten them. We face the problem with non-predefined paths in Section 7.5 and the team plan scheduling problem is solved in Section 7.6. Finally, we present simulation results in Section 7.7.

7.2 Problem Formulation and Solution Overview

Let $\mathcal{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_Q\}$ be a finite set of Q points of interest that must be covered. At each point, we define the coverage level with a scalar field, $Z(\mathbf{q}, t) \geq 0$. The objective is to maintain a desired level $Z^*(\mathbf{q}, t) \geq 0$ by providing a certain coverage action $P^*(\mathbf{q}, t) \geq 0$

over time, not necessarily the same for all the points. This is equivalent to model that the coverage level deteriorates over time due to a physical phenomenon or a measurement degradation with a constant decay rate. This is achieved using a team of $N \in \mathbb{N}$ mobile robots $\mathcal{I} = \{i_1, \ldots, i_I\}$ of radius r_i, $i \in \mathcal{I}$. They are capable of increasing the level at the points in which they are located according to a *production function*, $P_i(\mathbf{q}, t)$:

$$P_i(\mathbf{q}, t) = \begin{cases} 0, & \text{if } \mathbf{p}_i(t) \neq \mathbf{q}, \\ \rho_i(\mathbf{q}), & \text{if } \mathbf{p}_i(t) = \mathbf{q}, \end{cases} \tag{7.1}$$

where $\mathbf{p}_i(t)$ is the position of the robot at time $t \geq 0$ and $0 \leq \rho_i(\mathbf{q}) \leq \rho_i^{\max}(\mathbf{q})$ is the *coverage action* of the robot, which can also be controlled. The maximum production $\rho_i^{\max}(\mathbf{q})$ can be different for each robot-point pair.

Each robot may not be capable of reaching all the points due to physical constraints. The subset $\mathcal{Q}_i \subseteq \mathcal{Q}$ represents the *reachable set* of points for robot i, with the team satisfying $\cup_{i \in \mathcal{I}} \mathcal{Q}_i = \mathcal{Q}$. Reciprocally, each point $\mathbf{q} \in \mathcal{Q}$ can be covered by a set of visiting robots, $\mathcal{I}_\mathbf{q} \subseteq \mathcal{I}$. In this context, the coverage provided at each point can be calculated as

$$Z(\mathbf{q}, t) = \int_0^t \sum\nolimits_{i \in \mathcal{I}_\mathbf{q}} P_i(\mathbf{q}, \tau) d\tau, \tag{7.2}$$

assuming $Z(\mathbf{q}, 0) = 0$ as initial condition, and the desired coverage as

$$Z^*(\mathbf{q}, t) = \int_0^t P^*(\mathbf{q}, \tau) d\tau. \tag{7.3}$$

A solution that maintains all the time $Z(\mathbf{q}, \tau) = Z^*(\mathbf{q}, \tau)$ requires at least as many robots as points and does not exist if the number of robots is lower than the number of points. Since this is usually the case, an optimization metric is defined to try to maintain $Z(\mathbf{q}, \tau)$ as close as possible to $Z^*(\mathbf{q}, \tau)$ over time. Some optimality criteria, as in the previous chapter, could be to minimize over time the quadratic difference between the required and the provided coverage, i.e., minimize $\int_0^t \sum_{\mathbf{q} \in \mathcal{Q}} \big(Z^*(\mathbf{q}, \tau) - Z(\mathbf{q}, \tau)\big)^2 d\tau$, or minimize the maximum difference between $Z(\mathbf{q}, t)$ and $Z^*(\mathbf{q}, t)$ over time, i.e., minimize $\max \big(Z^*(\mathbf{q}, t) - Z(\mathbf{q}, t)\big)$. However, recall that trying to minimize globally a function associated with these metrics results in a problem that has been proven to be NP-hard. For these reasons, we seek to guarantee that the desired coverage is provided periodically with period T. A periodic approach, though inherently suboptimal with respect to such metrics and still NP-hard, allows us to guarantee that all the points receive on average the coverage that they require, with a repetitive strategy that can be calculated in advance. In this scenario, the periodic objective requires that

(i) at some time $t \leq T$ the coverage reaches $Z(\mathbf{q}, t) = Z^*(\mathbf{q}, t)$, and

(ii) from that time on, $Z(\mathbf{q}, t + kT) = Z^*(\mathbf{q}, t + kT)$, $\forall\, k \in \mathbb{Z}$.

We can formulate the second condition equivalently as

$$Z(\mathbf{q}, t + kT) - Z\big(\mathbf{q}, t + (k-1)\,T\big) = Z^*(\mathbf{q}, t + kT) - Z^*\big(\mathbf{q}, t + (k-1)\,T\big), \tag{7.4}$$

and we can assume that $P^*(\mathbf{q}) \equiv P^*(\mathbf{q}, t)$ is constant over time, or at least periodic, since the rate of change of $P^*(\mathbf{q})$ is usually much bigger than the period. Thus, the problem can be consider stationary between changes. Then, the right term is equal to $P^*(\mathbf{q}, \tau)\, T$ according to (7.3) and, introducing (7.2), the objective becomes

$$\int_{t+(k-1)T}^{t+kT} \sum\nolimits_{i \in \mathcal{I}_\mathbf{q}} P_i(\mathbf{q}, \tau) d\tau = P^*(\mathbf{q})\, T. \tag{7.5}$$

In order to satisfy this objective, the robots follow a periodic path, Γ_i, between reachable points, i.e., an ordered subset of $\mathcal{Q}_i$. Along these paths, the robots spend some time, $\theta_{i,\Gamma_i(j)}$, covering each point. With these times already normalized by the period, the periodic coverage objective can be stated as

$$\sum\nolimits_{i \in \mathcal{I}_\mathbf{q}} \rho_{i,\Gamma_i(j)}\, \theta_{i,\Gamma_i(j)} = P^*(\mathbf{q}), \tag{7.6}$$

for each point $\mathbf{q}$ of the environment. The equation is also normalized by the period and represents an average over it. With a little abuse of notation we represent $\mathbf{q}$ as the j-th point in the path of robot i, i.e., $\Gamma_i(j) \equiv \mathbf{q}$, even when j may be different for each robot i. In the following we refer to it only with subindex j for simplicity. Thus, the objective is

$$\sum\nolimits_{i \in \mathcal{I}_\mathbf{q}} \rho_{i,j}\, \theta_{i,j} = P^*(\mathbf{q}). \tag{7.7}$$

Each point is visited only once per period by the robot and each visit has associated three different times:

- *arrival time*, $a_{i,j}$, the instant in which the robot arrives and starts covering a point;
- *coverage time*, $\theta_{i,j}$, the duration of the coverage of the point; and
- *departure time*, $d_{i,j}$, the instant in which the robot stops covering the point and leaves towards the next one.

The total moving time, i.e., the time that robot i would need to traverse Γ_i without making any stops, is θ_i^m. All these times are also normalized by the period T and they belong to the interval $[0, 1]$. Note that the periodic plans can be repeated as long as all $\mathcal{Q}_i$, $\mathcal{I}_\mathbf{q}$ and $P^*(\mathbf{q})$ remain constant. Besides, they are independent of the value of the period and such value can be chosen differently depending on the application.

The solution that we propose to this problem follows a divide-and-conquer strategy, as shown in Alg. 4. We separate the problem into three subproblems (Steps 1, 2 and 4) to reduce its complexity and make it tractable. Even though this does not guarantee global optimality, it allows us to solve efficiently the three subproblems, each of them in an optimal manner.

For the first step, we use an order-first split-second approach [143] and solve a Traveling Salesman Problem (TSP) [144] to find such paths. For the second step, we pose a quadratically constrained linear program (QCLP) with the restrictions of satisfying the periodic

Algorithm 4 Solution Overview

1: Plan initial paths.
2: Calculate optimal coverage times and actions.
3: Shorten paths and recalculate times and actions.
4: Schedule the path starts to avoid collisions.

coverage objective from (7.7). We also propose an algorithm to shorten these paths and a variation of the same QCLP that takes into account the lengths of the paths. In the last step, we calculate a schedule that allows us to include the individual periodic plans of the robots in a periodic team plan in which collision avoidance is guaranteed. We formulate this scheduling as a MILP to minimize the time in which two or more robots are moving at the same time. In the following sections we explain in detail the two main steps of our solution, namely Steps 2 with its extensions and 4.

7.3 Optimal Times and Powers

The periodic coverage objective requires the calculation of the coverage times and coverage actions over the paths that the robots travel. The problem of calculating these times and productions reduces to find the times $\theta_{i,j}$ and productions $\rho_{i,j}$ that comply with (7.7) and with the periodicity of the system. To this end, we consider a cost function on the times and the productions, $f(\theta_{i,j}, \rho_{i,j})$, and the resulting QCLP that represents the persistent coverage problem is

$$
\begin{aligned}
\underset{\theta_{i,j},\,\rho_{i,j}}{\text{minimize}} \quad & f(\theta_{i,j}, \rho_{i,j}) \\
\text{subject to} \quad & \sum_{i \in \mathcal{I}_{\mathbf{q}}} \rho_{i,j}\, \theta_{i,j} = P^*(\mathbf{q}), && \forall\, \mathbf{q} \in \mathcal{Q}, && \text{(7.8a)} \\
& \sum_{\mathbf{q} \in \mathcal{Q}_i} \theta_{i,j} \leq 1 - \theta_i^m, && \forall\, i \in \mathcal{I}, && \text{(7.8b)} \\
& \sum_{i \in \mathcal{I}_{\mathbf{q}}} \theta_{i,j} \leq 1, && \forall\, \mathbf{q} \in \mathcal{Q}. && \text{(7.8c)}
\end{aligned}
$$

Eq. (7.8a) is the quadratic restriction on the coverage objective. The set of equations (7.8b) imposes that, for each robot, the time spent covering the points plus the time to move along the path must be lower than the period, and the set (7.8c) represents that each point cannot be covered for more time than the period. This restriction is only needed in the case that it is not allowed that two robots cover the same point at the same time, because otherwise it would be impossible to avoid collisions between robots. The selection of the cost function is entirely dependent on the performance expected from the particular application of the persistent coverage problem. For this reason, in Section 7.7 we explore and analyze different linear cost functions that are appropriate for many applications.

7.3.1 Existence of Solution

This section is devoted to analyze in which conditions there exists a feasible solution for (7.8). In particular, we find sufficient conditions to ensure a feasible solution in two particular cases, one where all the robots go through all the points and a more realistic second scenario, where the points visited by different robots are not the same. These conditions allow us to know a priori if the robots are capable of satisfying the coverage actions required by the points.

We define the minimum production that each point can receive when all robots provide their maximum,

$$\rho^{\min}(\mathbf{q}) = \min_{i \in \mathcal{I}_{\mathbf{q}}} \rho_i^{\max}(\mathbf{q}),$$

to take advantage of the implications of the following remark.

Remark 7.3.1. *If a feasible solution exists for* (7.8)*, a feasible solution also exists for the same problem with* $\rho_i(\mathbf{q}) = \rho_i^{\max}(\mathbf{q})$*.*

In the first place, we focus on a scenario in which all the robots visit all the points each period, i.e., $\mathcal{Q}_i = \mathcal{Q}$, $\forall\, i \in \mathcal{I}$, and $\mathcal{I}_{\mathbf{q}} = \mathcal{I}$, $\forall\, \mathbf{q} \in \mathcal{Q}$, and spend some time covering them. We present in the following theorem the conditions for this case while the intuition is the following: the maximum time that the robots need to cover a point is given by the minimum production that any robot can give at the point divided by the required production at that point, $P^*(\mathbf{q})$. Since the minimum production at each point can be applied by any robot, the robots become interchangeable at all the points. Therefore, if the sum of the maximum times for all the points is lower than the total available time of the robots, there exists a feasible solution.

Theorem 7.3.2 (Sufficient Conditions for Full Reachability)**.** *Assume that all the robots visit all the points in each period,* $\mathcal{Q}_i = \mathcal{Q}$, $\forall\, i \in \mathcal{I}$*, and* $\mathcal{I}_{\mathbf{q}} = \mathcal{I}$, $\forall\, \mathbf{q} \in \mathcal{Q}$*. If*

$$\rho^{\min}(\mathbf{q}) \geq P^*(\mathbf{q}), \qquad \forall\, \mathbf{q} \in \mathcal{Q},$$

and

$$\sum_{\mathbf{q} \in \mathcal{Q}} \frac{P^*(\mathbf{q})}{\rho^{\min}(\mathbf{q})} \leq N - \sum_{i \in \mathcal{I}} \theta_i^m, \tag{7.9}$$

then, there exists a feasible solution for (7.8)*.*

Proof. It is clear that

$$\sum_{i \in \mathcal{I}} \rho_{i,j} \cdot \theta_{i,j} \geq \rho^{\min}(\mathbf{q}) \sum_{i \in \mathcal{I}} \theta_{i,j}, \qquad \forall\, \mathbf{q} \in \mathcal{Q}.$$

Then, the restriction on the coverage (7.8a) is satisfied if

$$\rho^{\min}(\mathbf{q}) \sum_{i \in \mathcal{I}} \theta_{i,j} \geq P^*(\mathbf{q}), \qquad \forall\, \mathbf{q} \in \mathcal{Q},$$

that is, if $\rho^{\min}(\mathbf{q}) \geq P^*(\mathbf{q})$, $\forall\, \mathbf{q} \in \mathcal{Q}$. Therefore, we can choose $\theta_{i,j}$ such that

$$1 \geq \sum_{i \in \mathcal{I}} \theta_{i,j} \geq \frac{P^*(\mathbf{q})}{\rho^{\min}(\mathbf{q})}, \qquad \forall\, \mathbf{q} \in \mathcal{Q}, \tag{7.10}$$

and, consequently, restrictions (7.8a) and (7.8c) are satisfied.

Since all the robots can visit all the points, the coverage task of the robot with the minimum production rate, $\rho^{\min}(\mathbf{q})$, can be exchanged with any other robot of the team. Therefore, the restrictions on the period of the robots (7.8b) can be gathered as the sum of all of them:

$$\sum_{i \in \mathcal{I}} \sum_{\mathbf{q} \in \mathcal{Q}} \theta_{i,j} = \sum\nolimits_{i \in \mathcal{I}} (1 - \theta_i^m),$$

or, equivalently,

$$\sum_{\mathbf{q} \in \mathcal{Q}} \sum_{i \in \mathcal{I}} \theta_{i,j} = N - \sum_{i \in \mathcal{I}} \theta_i^m.$$

If we introduce the right hand-side inequality of (7.10) we have

$$\sum_{\mathbf{q} \in \mathcal{Q}} \frac{P^*(\mathbf{q})}{\rho^{\min}(\mathbf{q})} \leq N - \sum\nolimits_{i \in \mathcal{I}} \theta_i^m,$$

and according to (7.9) the proof is complete. □

In some application the robots may not be able to reach all the points of the environment or may not be able to perform the coverage task of some points [145]. In that case, for some $i \in \mathcal{I}$, $\mathcal{Q}_i \neq \mathcal{Q}$. In the following theorem we relax the assumption of all the robots having to visit all the points and assure the existence of a solution when the ratio available production/required production is sufficiently high.

Theorem 7.3.3 (Sufficient Conditions for Partial Reachability)**.** *If there exists a subset* $\mathcal{I}' \subseteq \mathcal{I}$ *such that*

$$\rho_i^{\max}(\mathbf{q}) \geq P^*(\mathbf{q}), \qquad \forall\, \mathbf{q} \in \mathcal{Q}_i,\, i \in \mathcal{I}', \tag{7.11}$$

$$\sum_{\mathbf{q} \in \mathcal{Q}_i} \frac{P^*(\mathbf{q})}{\rho_i^{\max}(\mathbf{q})} \leq 1 - \theta_i^m, \qquad \forall\, i \in \mathcal{I}', \tag{7.12}$$

$$\bigcup_{i \in \mathcal{I}'} \mathcal{Q}_i = \mathcal{Q}, \tag{7.13}$$

then, there exists a feasible solution for (7.8)*.*

Proof. First, we prove that if a robot i satisfies (7.11) and (7.12), it can cover all $\mathbf{q} \in \mathcal{Q}_i$. Particularizing the restriction on the coverage (7.8a) for a single robot i covering each point $\mathbf{q} \in \mathcal{Q}_i$, we have

$$\rho_{i,j} \cdot \theta_{i,j} = P^*(\mathbf{q}), \qquad \forall\, \mathbf{q} \in \mathcal{Q}_i.$$

Introducing the restriction on the period of the points (7.8c) and formulating as a sufficient condition, it becomes

$$1 \geq \theta_{i,j} \geq \frac{P^*(\mathbf{q})}{\rho_{i,j}}, \qquad \forall\, \mathbf{q} \in \mathcal{Q}_i. \tag{7.14}$$

Then, if (7.11) holds, we can choose $\theta_{i,j}$ such that (7.8a) and (7.8c) are satisfied.

On the other hand, substituting (7.14) in the restriction on the period of robot i (7.8b), we have

$$\sum_{\mathbf{q}\in\mathcal{Q}_i} \theta_{i,j} = 1 - \theta_i^m \geq \sum_{\mathbf{q}\in\mathcal{Q}_i} \frac{P^*(\mathbf{q})}{\rho_{i,j}}$$

and the first part of the proof is complete.

Similarly, for all the robots which satisfy (7.11) and (7.12), i.e., which can belong to $\mathcal{I}'$, their reachable points $\mathbf{q} \in \mathcal{Q}_i$ can be covered. Therefore, if (7.13) holds, all the points of the environment can be covered, completing the proof. □

The main reason for which a feasible solution may not exist is that the points require more coverage than the team is able to provide, since initially the paths of the robots go over all of their reachable points. In that case, the solution is to maximize the coverage provided to the points or to minimize the error between the coverage provided and the coverage objective.

7.4 Optimal Times and Powers with Shortened Paths

A particularity of the solutions obtained for (7.8) is that some of the times $\theta_{i,j}$ may be equal to zero. This happens especially in those robot-point pairs in which giving a production equal to zero is better than spending some time in terms of cost. This implies that those robots are not required to cover such points. Thus, equivalent paths can be followed, but only through points with coverage times greater than zero, reducing the moving times θ_i^m and, therefore, the total time required for the coverage of the environment.

We exploit this with a strategy that builds from (7.8) but allows us to find a solution with lower cost and shortened paths. This strategy is summarized in Algorithm 5.

Algorithm 5 Iterative Solution to Shorten the Paths

Require: Initial paths Γ_i and moving times θ_i^m.

1: Pose and solve the initial problem (7.8).
2: **while** any $\theta_{i,j} = 0$ **do**
3: Eliminate j from Γ_i.
4: Plan a shorter path with the points in Γ_i.
5: Recalculate θ_i^m of Γ_i.
6: Repose and resolve (7.8).
7: **end while**
8: **return** Optimal $\theta_{i,j}$, $\rho_{i,j}$ and final paths Γ_i.

In the first place, we pose the program (7.8), with twice decision variables as points in the initial paths of the robots, and find an initial solution for $\theta_{i,j}$ and $\rho_{i,j}$. Such initial solution

may include several points whose times or actions for one or more robots are equal to zero. These points are erased from the paths of the corresponding robots,

$$\Gamma_i = \Gamma_i \setminus j, \quad \forall\, j \text{ such that } \theta_{i,j} = 0 \text{ or } \rho_{i,j} = 0.$$

At this point, a shorter trajectory is computed between the remaining points. This reduces the moving times, θ_i^m, and therefore, the robots have more time available to cover the points of their current paths.

Finally, the recalculated moving times allow us to repose the program and find a new solution for $\theta_{i,j}$ and $\rho_{i,j}$. If this solution has any time or action equal to zero, the process is repeated and, when all the times and actions are different from zero, they form the optimal solution to the persistent coverage problem with shortened paths. In fact, in the following proposition we study the maximum number of iterations that the previous algorithm can be executed.

Proposition 7.4.1. *The maximum number of iterations that Algorithm 5 needs to be completed is* $(I-1)\,Q$.

Proof. The total number of times that are initially assigned is, in the worst case, $I \cdot Q$ for a full reachability. If all the times are greater than zero the algorithm has finished. Assume that this is not the case and that at each iteration only one new decision variable is set to zero, which represents the worst case. Since a feasible solution contains at least one coverage time greater than zero for each point, then the minimum number of non-zero decision variables must be Q and consequently the maximum number of iterations of the algorithm is equal to $(I-1)\,Q$. □

The result of the algorithm is that, at each iteration, the robot paths are optimized using the results of the previous optimal coverage times and productions. Then, the times and productions are optimized again for the new refined paths. This makes the combination of both to iteratively improve the global solution achievable with this periodic approach.

7.5 Optimal Times and Powers with Non-Predefined Paths

In the previous section we have seen that the moving times, θ_r^m, also affect the optimal coverage. This is particularly relevant in cases where they represent a substantial fraction of time from each period, i.e., in situations where moving from one point to another takes a lot of time compared to covering each point. Therefore, if the paths can be modified to reduce these times, the robots have more time to spend covering other points. Since including these times in the problem is highly complex, we instead present a modified version of the original problem (7.8) that indirectly includes the paths to reduce the distance traveled by the robots, as if the paths where not predefined. The idea is to use a linear cost function that includes coefficients, $\omega_{i,j}$, associated to the times $\theta_{i,j}$:

$$f(\theta_{i,j}) = \sum\nolimits_{\mathbf{q}\in\mathbf{Q}} \sum\nolimits_{i\in\mathbf{I}} \omega_{i,j}\, \theta_{i,j}. \tag{7.15}$$

These weights account for the moving times of the robots in the sense that they modify the cost of covering each point and allow us to favor that each robot covers only close points, thus reducing the traveled distance. It is important to note that, if the problem (7.8) has a feasible solution, it is also feasible for any value of the coefficients, i.e., the coefficients do not modify the existence of solution. Additionally, although the optimal solution for (7.15) will most likely be suboptimal for (7.8), the length of the paths, represented by the moving times θ_r^m, can be drastically reduced by defining appropriate coefficients.

7.5.1 Coefficient Calculation

To calculate the coefficients we present a strategy in Algorithm 6. The intuition behind this strategy is the following. Firstly, we divide the points in as many clusters as the number of robots and assign a cluster to each robot according to some distance function from the centroid of the clusters to the robots. Next, we check if all the points can be reached by their assigned robots, reassigning the ones that cannot be reached to the robot whose assigned cluster is the closest to the point. Finally, we calculate the coefficient of each pair point-robot depending on the final assignment. Note that this procedure is only used for the calculation of the coefficients of the cost function and it is not definitive in the sense that each robot may finally visit any point of its $\mathcal{Q}_i$ regardless of this assignment, i.e., our objective remains in (7.15).

Algorithm 6 Coefficient Calculation

1: Divide $\mathcal{Q}$ in N clusters.
2: Find the optimal robot-cluster assignment $\mathcal{A}^*$ (7.16), (7.17).
3: Find non-reachable points $\mathbf{q}_{i,r_i}^{nr}$ in each cluster.
4: Assign $\mathbf{q}_{i,r_i}^{nr}$ to the closest cluster $\mathcal{K}_j$ such that $\mathbf{q}_{i,r_i}^{nr} \in \mathcal{Q}_{r_j}$ (7.18).
5: Remove $\mathbf{q}_{i,r_i}^{nr}$ from previous cluster $\mathcal{K}_i$ (7.19).
6: Calculate coefficients (7.20).

The first step of the algorithm is to divide the set of points that must be covered in N clusters. For this we use a k-means solution, considering that there are state-of-the-art algorithms that minimize the within-cluster sum of squares and, as a consequence, the distance that a robot would need to travel to visit all the points in a cluster.

Then, we find the assignment,

$$\mathcal{A} = \big\{(i, \mathcal{K}_i) \,\forall\, i \in \mathcal{I} \,|\, i \neq j \iff \mathcal{K}_i \neq \mathcal{K}_j\big\}, \tag{7.16}$$

that associates each robot i with its own cluster $\mathcal{K}_i$. The optimal assignment minimizes the total distance from the cluster centroids to the assigned robots,

$$\mathcal{A}^* = \underset{\mathcal{A}}{\operatorname{argmin}} \sum_{i \in \mathcal{I}} g(i, \mathcal{K}) \,\|\, \mathbf{c}_{\mathcal{K}} - \mathbf{p}_i \|, \tag{7.17}$$

where $\mathbf{c}_{\mathcal{K}}$ is the centroid of cluster $\mathcal{K}$ and

$$g(i, \mathcal{K}) = \begin{cases} 1, & \text{if } \mathcal{K} \cap \mathcal{Q}_i \neq \emptyset, \\ G, & \text{otherwise,} \end{cases}$$

with $G > 1$, is a function that penalizes the pairs in which the cluster cannot be reached by the robot. This assignment is obtained in polynomial time using the Hungarian algorithm [146].

The third step is to find the points of each cluster whose assigned robot cannot reach:

$$\mathcal{Q}_i^{nr} = \{\mathbf{q} \in \mathcal{K}_i \,|\, \mathbf{q} \notin \mathcal{Q}_i\}.$$

Finally, these points are assigned to the closest cluster between the ones whose assigned robots can reach them,

$$\mathcal{K}_j = \mathcal{K}_j \cup \mathbf{q} \iff \|\mathbf{c}_{\mathcal{K}_j} - \mathbf{q}\| \leq \|\mathbf{c}_{\mathcal{K}_\ell} - \mathbf{q}\| \qquad \forall\, j, \ell \in \mathcal{I} \,|\, \mathbf{q} \in \mathcal{Q}_i^{nr}, \mathcal{Q}_j, \mathcal{Q}_\ell, \tag{7.18}$$

and removed from the previous ones

$$\mathcal{K}_i = \mathcal{K}_i \setminus \mathbf{q}. \tag{7.19}$$

According to the previous assignment we calculate the coefficients from (7.15), with $H > 1$, as

$$\omega_{i,j} = \begin{cases} 1, & \text{if } \mathbf{q} \in \mathcal{K}_i, \\ H, & \text{otherwise.} \end{cases} \tag{7.20}$$

7.5.2 Optimal Coverage Solution

Once the coefficients in (7.15) are completely known, the only thing left to solve the problem (7.8) are the moving times since here we assume that the paths are not initially set. To compute these times, we calculate an upper bound for each robot as a TSP through all the points in its $\mathcal{Q}_i$. With these values the QCLP is solved, obtaining the optimal solution for (7.15). As mentioned before, the coefficients of the cost function will favor that each robot has only positive coverage times for the points on its assigned cluster and the rest of them equal to zero. Nevertheless, different outcomes can be obtained, i.e., if some robot cannot cover all the points inside its cluster or if there is another robot that can apply the required consumption in less time, even with the extra weight of the coefficient. Finally, if we obtain some coverage time equal to zero, the approach can be iteratively refined using the same procedure as in Algorithm 5.

7.5.3 Problem Reduction

The previous strategy favors that robots cover some points preferentially but considers any other feasible solution. Now we propose a sufficient condition that allows us to separate the optimization problem in several smaller subproblems and reduce the computational cost. The key idea is that, if every robot can cover its assigned points, we can calculate the optimal times and productions independently for each robot since there are no points visited by more than one robot.

Proposition 7.5.1 (Sufficient Condition for Problem Reduction)**.** *Consider the persistent coverage problem from* (7.8)*. If every robot $i \in \mathcal{I}$ satisfies that*

$$\rho_i^{\max}(\mathbf{q}) \geq P^*(\mathbf{q}), \qquad \forall\, \mathbf{q} \in \mathcal{K}_i,$$
$$\sum_{\mathbf{q}\in\mathcal{K}_i} \frac{P^*(\mathbf{q})}{\rho_i^{\max}(\mathbf{q})} \leq 1 - \theta_i^m,$$

then, there exists a feasible solution for the problem in which each robot only covers its assigned points. Moreover, the problem can be solved by solving I subproblems of one robot with $\mathcal{Q} = \mathcal{K}_i$.

Proof. The proof for the first statement is the same as for Theorem 7.3.3 and for the second is trivial. □

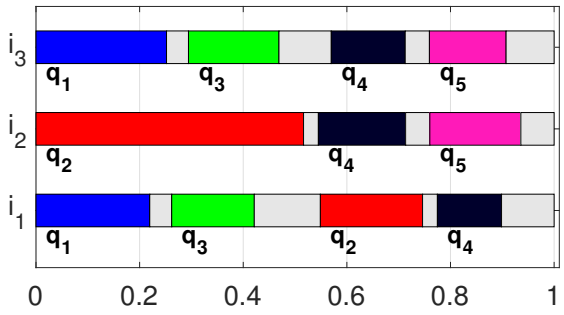
7.6 Team Plan Scheduling

The periodic paths and the optimal coverage times and actions obtained in the previous sections guarantee that the periodic persistent coverage objective is satisfied. However, this solution does not take into account the physical restrictions of the team. In particular, if all the robots start their periodic plans with no previous agreement, they may try to cover the same point at the same time resulting in a collision or they may also collide in their path from one point to the following. To solve this problem, the robots could be equipped with a collision avoidance system. However, handling a potential collision and, thus, modifying the original path, would vary the total moving time of the robot and violate the periodicity constraints or the periodic objective.

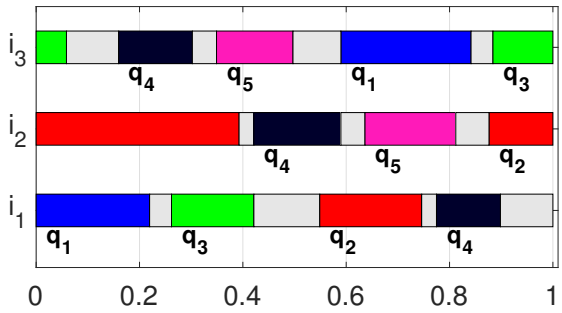
We devote this section to the calculation of a scheduling for the start of the periodic paths of the robots. It avoids collisions while covering a point and in the movements between points. The intuitive idea of the scheduling is to shift the individual plans of the robots in time to obtain a collision-free plan for the entire team, that is, to find a time, $0 \leq \varphi_i < 1$, for each robot $i \in \{2, \ldots, I\}$ such that the execution of its periodic plan shifted by this time leads to no collision with the others. We refer the initial times to the beginning of the path of robot $i = 1$, i.e., we fix $\varphi_1 = 0$, and that the paths remain the same through the scheduling.

In Fig. 7.1 we show an example of scheduling for $I = 3$ robots of size $r_i = 5$ and $Q = 5$ points. Fig. 7.1a shows the individual plan of each robot, which includes the path (order in which points are visited) and the coverage times (width of the colored rectangles). The beginning and end of the colored rectangles correspond to the arrival and departure times, $a_{i,j}$ and $d_{i,j}$, respectively, and the gray rectangles represents the time needed to move from one point to the following. It can be seen that some coverages of the same point are overlapped in time and, therefore, the execution of these individual plans leads to collisions. For instance, it happens when i_1 and i_3 try to cover $\mathbf{q}_1$ or $\mathbf{q}_3$, or when i_2 and i_3 try to cover $\mathbf{q}_4$ or $\mathbf{q}_5$. In Fig. 7.1b, where the team plan after the scheduling is depicted, it can be seen that the individual plans of robots i_2 and i_3 have been shifted a 88% and a 59% of the period respectively and that in the resulting plan there are no overlaps between coverages of the

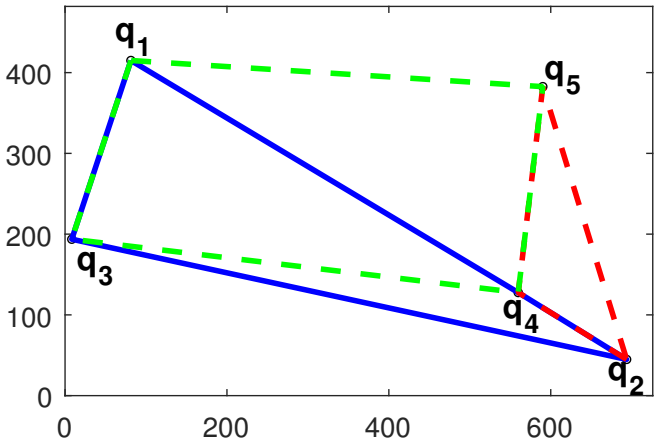
same point. Fig. 7.1c shows the locations of the points and the paths followed by the robot. It can be seen that no collisions occurs after the scheduling.



(a) Individual plans before scheduling.



(b) Team plan after scheduling.



(c) Robot paths.

Figure 7.1: Example of scheduling. (a)-(b) Each row represents the plan of a robot. The points $\mathbf{q}_i$ that each robot covers are represented in different colors. The coverage times are depicted by the width of the colored rectangles and the gray rectangles represent the time needed to move between points.

The calculation of the optimal schedule is done by solving a MILP with constraints. In the following subsections we develop the formulation, the restrictions and the cost function, and finally pose the problem and discuss its suitability.

7.6.1 Transformation of Individual Times to Team Plan

The previous step to calculate the schedule that avoids collisions is to refer the individual times of the robots to the team plan. The arrival times can be expressed as follows:

$$A_{i,j} = \varphi_i + a_{i,j}, \qquad \text{if } \varphi_i + a_{i,j} \leq 1, \tag{7.21a}$$

$$A_{i,j} = \varphi_i + a_{i,j} - 1, \qquad \text{if } \varphi_i + a_{i,j} > 1, \tag{7.21b}$$

for all $j \in \{2, \ldots, |\Gamma_i|\}$, $i \in \{2, \ldots, I\}$. Note that for $j = 1$, (7.21) does not apply since $a_{i,1} = 0$ and $A_{i,1} = \varphi_i$, and recall that for robot $i = 1$ we set $\varphi_1 = 0$ and, therefore, its corresponding times do not need transformation.

To be able to include these times in the MILP formulation, we introduce the binary variables, $c^a_{i,j}$, such that,

$$\begin{aligned} c^a_{i,j}, &= 0 \qquad \text{if } \varphi_i + a_{i,j} \leq 1, \\ c^a_{i,j}, &= 1 \qquad \text{otherwise.} \end{aligned}$$

These variables represent if the arrival times in the team plan are bigger than 1 and they have to be shifted to the left side of the team plan by subtracting 1, e.g., the arrival time of i_2 to $\mathbf{q}_4$ in Fig. 7.1b.

Thus, we can express the arrival times as

$$A_{i,j} = \varphi_i + a_{i,j} - c^a_{i,j} \tag{7.22}$$

subject to

$$\varphi_i + a_{i,j} \leq 1 + R\, c^a_{i,j}, \tag{7.23a}$$
$$-(\varphi_i + a_{i,j}) \leq -1 + R\,(1 - c^a_{i,j}), \tag{7.23b}$$

obtained with the big number method [147]. This method activates or deactivates the constraints depending on the value of the binary variable, with R being a sufficiently big number. In this case, $c^a_{i,j} = 0$ if (7.21a) has to be used and $c^a_{i,j} = 1$ if (7.21b) is needed.

The same formulation can be developed for the departure times:

$$D_{i,j} = \varphi_i + d_{i,j}, \qquad \text{if } \varphi_i + d_{i,j} \leq 1, \tag{7.24a}$$
$$D_{i,j} = \varphi_i + d_{i,j} - 1, \qquad \text{if } \varphi_i + d_{i,j} > 1, \tag{7.24b}$$

for all $j \in \{1, \ldots, |\Gamma_i|\}$, $i \in \{2, \ldots, I\}$. Introducing the binary variables $c^d_{i,j}$, we have

$$D_{i,j} = \varphi_i + d_{i,j} - c^d_{i,j} \tag{7.25}$$

subject to

$$\varphi_i + d_{i,j} \leq 1 + R\, c^d_{i,j}, \tag{7.26a}$$
$$-(\varphi_i + d_{i,j}) \leq -1 + R\ (1 - c^d_{i,j}). \tag{7.26b}$$

Since the binary variables $c^a_{i,j}$ and $c^d_{i,j}$ are variables of the problem, we include the following restrictions on their values to guarantee that the order of the paths is followed:

$$c^a_{i,j} - c^d_{i,j} \leq 0, \quad \forall\, j \in \{2, \ldots, |\Gamma_i|\}, \tag{7.27a}$$
$$c^d_{i,j} - c^a_{i,j+1} \leq 0, \quad \forall\, j \in \{1, \ldots, |\Gamma_i| - 1\}. \tag{7.27b}$$

7.6.2 Collision Avoidance During Coverage

A thorough consideration of collisions requires the inclusion of spatio-temporal restrictions in the problem, which implies a discretization of the environment and an analysis of what happens at each point [94]. Nevertheless, for large environments or with many robots or points to cover, the problem becomes computationally unaffordable or even intractable. For this reason, we include the collision avoidance as planning constraints. Although it is overprotective, it is much lighter in terms of computational cost.

The first situation in which a collision may occur is when two or more robots try to cover the same point at the same time. From the planning perspective, this happens if the coverages of such point by different robots are overlapped in the team plan. We avoid this type of collisions with two groups of constraints. The first group is

$$A_{i_1,j_1} - A_{i_2,j_2} \leq -\varepsilon + R\,(1 - c^{cc}_{i_1.j_1,i_2,j_2}), \tag{7.28a}$$
$$A_{i_1,j_1} - D_{i_1,j_1} \leq -\varepsilon + R\,(1 - c^{cc}_{i_1.j_1,i_2,j_2}), \tag{7.28b}$$
$$D_{i_1,j_1} - A_{i_2,j_2} \leq -\varepsilon + R\,(1 - c^{cc}_{i_1.j_1,i_2,j_2}), \tag{7.28c}$$
$$D_{i_2,j_2} - A_{i_1,j_1} \leq R\,(1 - c^{cc}_{i_1.j_1,i_2,j_2}) + R(1 - c^{d}_{i_2,j_2} + c^{a}_{i_2,j_2}), \tag{7.28d}$$

for all j_1, j_2 such that $\Gamma_{i_1}(j_1) = \Gamma_{i_2}(j_2)$, where the constant ε is the minimum separation between departures and arrivals to the same point. The first restriction, Eq. (7.28a), activates this group if $A_{i_1,j_1} \leq A_{i_2,j_2}$, that is, if the coverage of robot i_1 starts before the coverage of robot i_2. When this happens, the binary variable $c^{cc}_{i_1.j_1,i_2,j_2} = 1$. Since i_1 starts covering earlier, we have to assure that its coverage is not split by the end of the period, that is, its departure time is not translated to the first part of the period, as for the coverage of $\mathbf{q}_3$ by i_3 in Fig. 7.1b. Eq. (7.28b) guarantees this. In the third place, Eq. (7.28c) assures that the coverage of robot i_2 starts after the coverage of i_1 has ended, i.e., $D_{i_1,j_1} \leq A_{i_2,j_2}$. The last restriction guarantees that, if the coverage of i_2 has to be split, i.e., $c^{d}_{i_2,j_2} - c^{a}_{i_1,j_1} = 1$, it finishes before the coverage of i_1 starts.

The second group of restrictions is

$$A_{i_2,j_2} - A_{i_1,j_1} \leq -\varepsilon + R\,c^{cc}_{i_1.j_1,i_2,j_2}, \tag{7.29a}$$
$$A_{i_2,j_2} - D_{i_2,j_2} \leq -\varepsilon + R\,c^{cc}_{i_1.j_1,i_2,j_2}, \tag{7.29b}$$
$$D_{i_2,j_2} - A_{i_1,j_1} \leq -\varepsilon + R\,c^{cc}_{i_1.j_1,i_2,j_2}, \tag{7.29c}$$
$$D_{i_1,j_1} - A_{i_2,j_2} \leq R\,c^{cc}_{i_1.j_1,i_2,j_2} + R\,(1 - c^{d}_{i_1,j_1} + c^{a}_{i_1,j_1}). \tag{7.29d}$$

Opposite to the first one, this second group is activated when $A_{i_2,j_2} \leq A_{i_1,j_1}$ or, equivalently, when $c^{cc}_{i_1.j_1,i_2,j_2} = 0$.

7.6.3 Collision Avoidance During Motion

The second situation in which a collision between a pair of robots may occur is when both of them are moving or one is moving and the other is covering a point. This can be prevented in the same way that the collisions during coverage, by avoiding the overlap of the

movements and coverages that may cause a conflict. In fact, we propose the same restrictions as in (7.28)-(7.29) for the two cases: the movements of two robots, and a movement of a robot and a coverage of a point by another robot.

We define as $\delta_{i,j}$ and $\alpha_{i,j}$ the initial and final times of the movement between $\Gamma_i(j)$ and $\Gamma_i(j+1)$, respectively. They are defined opposite to the coverage times since $\delta_{i,j} = d_{i,j}$ is the departure of the movement and $\alpha_{i,j} = a_{i,j+1}, j \in \{1, \ldots, |\Gamma_i| - 1\}$, is the arrival of the movement. For $j = |\Gamma_i|$, we have $\alpha_{i,|\Gamma_i|} = 1$. Similarly to the coverage times, we can express the departure and arrival times of the movements in the team plan as a function of the times in the robot period and the binary variables $c^a_{i,j}$ and $c^d_{i,j}$ as follows:

$$\Delta_{i,j} = \varphi_i + \delta_{i,j} - c^d_{i,j}, \tag{7.30a}$$

$$\Lambda_{i,j} = \varphi_i + \alpha_{i,j} - c^a_{i,j+1}. \tag{7.30b}$$

For $j = |\Gamma_i|$, we define the binary variable c^a_i that represents if the final time of the last movement of each robot is greater than one or not. It requires the following restrictions:

$$\varphi_i + \alpha_{i,|\Gamma_i|} \leq 1 + R\, c^a_i, \tag{7.31a}$$

$$-(\varphi_i + \alpha_{i,|\Gamma_i|}) \leq -1 + R\,(1 - c^a_i), \tag{7.31b}$$

$$c^d_{i,|\Gamma_i|} - c^a_i \leq 0. \tag{7.31c}$$

In order to decide if two movements can lead to a collision, we calculate the minimum distance between the two trajectories of the movements, $d^{mm}_{i_1.j_1,i_2,j_2}$ and determine that a collision is possible if such distance is lower than the sum of the sizes of the robots, i.e., $d^{mm}_{i_1.j_1,i_2,j_2} < r_{i_1} + r_{i_2}$. If the pair of movements may result into conflict, the following two sets of constraints are included in the problem. These sets are the same as (7.28)-(7.29), respectively, but for the departure and arrival times of the movements. The first set,

$$\Delta_{i_1,j_1} - \Delta_{i_2,j_2} \leq -\varepsilon + R\,(1 - c^{mm}_{i_1.j_1,i_2,j_2}), \tag{7.32a}$$

$$\Delta_{i_1,j_1} - \Lambda_{i_1,j_1} \leq R\,(1 - c^{mm}_{i_1.j_1,i_2,j_2}), \tag{7.32b}$$

$$\Lambda_{i_1,j_1} - \Delta_{i_2,j_2} \leq -\varepsilon + R\,(1 - c^{mm}_{i_1.j_1,i_2,j_2}), \tag{7.32c}$$

$$\Lambda_{i_2,j_2} - \Delta_{i_1,j_1} \leq -\varepsilon + R\,(1 - c^{mm}_{i_1.j_1,i_2,j_2}) + R\,(1 - c^d_{i_2,j_2} + c^a_{i_2,j_2+1}), \tag{7.32d}$$

is active when the movement of robot i_1 starts before the movement of i_2, i.e., $\Delta_{i_1,j_1} \leq \Delta_{i_2,j_2}$ in Eq. (7.32a). This is represented by the binary variable $c^{mm}_{i_1.j_1,i_2,j_2} = 1$. The second constraint represents that the movement of robot i_1 cannot be split by the end of the period; the third one, that the movement of i_2 must start after the movement of i_1 has ended; and the last one that, if the movement of i_2 is split by the end of the period, it must finish before i_1 starts moving.

Equivalently, the second set of constraints activates when the movement of robot i_2 starts before the movement of i_1:

$$\Delta_{i_2,j_2} - \Delta_{i_1,j_1} \leq -\varepsilon + R\, c^{mm}_{i_1.j_1,i_2,j_2}, \tag{7.33a}$$

$$\Delta_{i_2,j_2} - \Lambda_{i_2,j_2} \leq R\, c^{mm}_{i_1.j_1,i_2,j_2}, \tag{7.33b}$$

$$\Lambda_{i_2,j_2} - \Delta_{i_1,j_1} \leq -\varepsilon + R\, c^{mm}_{i_1.j_1,i_2,j_2}, \tag{7.33c}$$

$$\Lambda_{i_1,j_1} - \Delta_{i_2,j_2} \leq -\varepsilon + R\, c^{mm}_{i_1.j_1,i_2,j_2} + R\,(1 - c^d_{i_1,j_1} + c^a_{i_1,j_1+1}), \tag{7.33d}$$

and the meaning is the same as (7.32) replacing i_1 by i_2 and vice versa.

The second type of collisions during the motion of a robot i_1 is with another robot i_2 that is covering a point. In that case, we calculate the minimum distance between the trajectory of the movement of i_1 and the point where i_2 is covering, $d^{mc}_{i_1.j_1,i_2,j_2}$. If a collision may happen, i.e., $d^{mc}_{i_1.j_1,i_2,j_2} < r_{i_1} + r_{i_2}$, we include the same two sets of constraints as before with the departure and arrival times of the movement of i_1 and the arrival and departure times of i_2 to the point that it must cover. The first set,

$$\Delta_{i_1,j_1} - A_{i_2,j_2} \leq -\varepsilon + R\,(1 - c^{mc}_{i_1.j_1,i_2,j_2}), \tag{7.34a}$$
$$\Delta_{i_1,j_1} - \Lambda_{i_1,j_1} \leq R\,(1 - c^{mc}_{i_1.j_1,i_2,j_2}), \tag{7.34b}$$
$$\Lambda_{i_1,j_1} - A_{i_2,j_2} \leq -\varepsilon + R\,(1 - c^{mc}_{i_1.j_1,i_2,j_2}), \tag{7.34c}$$
$$D_{i_2,j_2} - \Delta_{i_1,j_1} \leq -\varepsilon + R\,(1 - c^{mc}_{i_1.j_1,i_2,j_2}) + R\,(1 + c^{d}_{i_2,j_2} - c^{a}_{i_2,j_2}), \tag{7.34d}$$

is active if the movement starts before the coverage and the second,

$$A_{i_2,j_2} - \Delta_{i_1,j_1} \leq -\varepsilon + R\,c^{mc}_{i_1.j_1,i_2,j_2}, \tag{7.35a}$$
$$A_{i_2,j_2} - D_{i_2,j_2} \leq R\,c^{mc}_{i_1.j_1,i_2,j_2}, \tag{7.35b}$$
$$D_{i_2,j_2} - \Delta_{i_1,j_1} \leq -\varepsilon + R\,c^{mc}_{i_1.j_1,i_2,j_2}, \tag{7.35c}$$
$$\Lambda_{i_1,j_1} - A_{i_2,j_2} \leq -\varepsilon + R\,c^{mc}_{i_1.j_1,i_2,j_2} + R\,(1 - c^{d}_{i_1,j_1} + c^{a}_{i_1,j_1+1}), \tag{7.35d}$$

if the opposite happens. The interpretation of these constraint sets is the same as for (7.28)-(7.29) and (7.32)-(7.33).

7.6.4 Cost Function

The proposed restrictions guarantee that in the resulting schedule no collisions between robots occur at any time. In some applications, finding such solution may be enough and the problem can be posed as a Constraint Satisfaction Problem. Nevertheless, in many other applications, it is desirable to find a solution that is not only feasible but also optimizes some kind of metric. In particular, we aim to minimize the time in which two or more robots are moving simultaneously:

$$f_{schedule} = \sum_{i_1=1}^{I-1} \sum_{i_2=i_1+1}^{I} \max\left(0, \min(\Lambda_{i_{k_1},j_{k_1}} - \Delta_{i_{k_2},j_{k_2}})\right), \tag{7.36}$$

with $k_1, k_2 = 1, 2$. $f_{schedule}$ represents the sum of the times in which each pair of movements of different robots are overlapped. This function is motivated by the mobile induction application that we introduce in Chapter 8 and is intended to minimize the changes in power requested from the electric grid.

The problem of minimizing (7.36) can be transformed to the standard MILP formulation as follows. First, we introduce the auxiliary variables z_{i_1,j_1,i_2,j_2} to be greater or equal to the maximum inside (7.36):

$$f_{schedule} = \sum_{i_1=1}^{I-1} \sum_{i_2=i_1+1}^{I} z_{i_1,j_1,i_2,j_2}, \tag{7.37}$$

such that

$$z_{i_1,j_1,i_2,j_2} \geq 0, \tag{7.38a}$$
$$z_{i_1,j_1,i_2,j_2} \geq x_{i_1,j_1,i_2,j_2}. \tag{7.38b}$$

Second, we introduce the auxiliary variables x_{i_1,j_1,i_2,j_2} to be greater or equal to the minimum inside (7.36), that is,

$$f_{schedule} = \sum_{i_1=1}^{I-1} \sum_{i_2=i_1+1}^{I} \max(0, x_{i_1,j_1,i_2,j_2}), \tag{7.39}$$

with

$$x_{i_1,j_1,i_2,j_2} \geq \min(\Lambda_{i_1,j_1} - \Delta_{i_1,j_1}, \Lambda_{i_1,j_1} - \Delta_{i_2,j_2}, \Lambda_{i_2,j_2} - \Delta_{i_1,j_1}, \Lambda_{i_2,j_2} - \Delta_{i_2,j_2}). \tag{7.40}$$

This can be reduced to one of the following constraints

$$x_{i_1,j_1,i_2,j_2} \geq \Lambda_{i_1,j_1} - \Delta_{i_1,j_1} - R\, e^1_{i_1,j_1,i_2,j_2}, \tag{7.41a}$$
$$x_{i_1,j_1,i_2,j_2} \geq \Lambda_{i_1,j_1} - \Delta_{i_2,j_2} - R\, e^2_{i_1,j_1,i_2,j_2}, \tag{7.41b}$$
$$x_{i_1,j_1,i_2,j_2} \geq \Lambda_{i_2,j_2} - \Delta_{i_1,j_1} - R\, e^3_{i_1,j_1,i_2,j_2}, \tag{7.41c}$$
$$x_{i_1,j_1,i_2,j_2} \geq \Lambda_{i_2,j_2} - \Delta_{i_2,j_2} - R\, e^4_{i_1,j_1,i_2,j_2}, \tag{7.41d}$$
$$e^1_{i_1,j_1,i_2,j_2} + e^2_{i_1,j_1,i_2,j_2} + e^3_{i_1,j_1,i_2,j_2} + e^4_{i_1,j_1,i_2,j_2} = 3, \tag{7.41e}$$

where $e^k_{i_1,j_1,i_2,j_2}, k \in \{1,\ldots,4\}$, are binary variables to activate the constraint between (7.41a) and (7.41d) that gives the minimum value of x_{i_1,j_1,i_2,j_2}, thanks to (7.41e). Since the objective (7.37) is to minimize the sum of z_{i_1,j_1,i_2,j_2}, it can be seen that x_{i_1,j_1,i_2,j_2} will also be minimized by activating such constraint.

Constraints (7.41a)-(7.41d) capture the duration of the overlap between each pair of movements. In the case that one of the movements is completely overlapped with the other, the duration of the overlap is equal to the duration of such movement. Therefore, we can express (7.41a) and (7.41d) as

$$x_{1,2} \geq \Lambda_1 - \Delta_1 - R\, e^1_{1,2}, \tag{7.42}$$
$$x_{1,2} \geq \Lambda_2 - \Delta_2 - R\, e^4_{1,2}. \tag{7.43}$$

Note that from now on, we substitute subindices i_1, j_1 and i_2, j_2 by 1 and 2 for simplicity.

On the other hand, if the movements are only partially overlapped, the duration of such overlap is only bounded by (7.41b) or (7.41c). Since they depend on $c^a_{i,j}$ and $c^d_{i,j}$ through (7.30), the difference $\Lambda_{i_{k_1},j_{k_1}} - \Delta_{i_{k_2},j_{k_2}}$ may not represent such duration if any of the movements is split between the end and the beginning of the team plan. This happens if $c^d_k \equiv c^d_{i_k,j_k} = 0$ and $c^a_i \equiv c^a_{i_k,j_k+1} = 1$. In the following we formulate restrictions for all the cases in which none, one or both movements are split, that are conditionally activated, applying the big number method, depending on c^d_k and c^a_k. In particular, if $c^a_k - c^d_k = 0$, the

movement is not split, and its corresponding constraints activated, and if $1 - c_k^a + c_k^d = 0$, the movement is split.

We set these two restrictions for the case in which none of the movements is split:

$$x_{1,2} \geq \Lambda_1 - \Delta_2 - R\, e_{1,2}^1 - R\,(c_1^a - c_1^d) - R\,(c_2^a - c_2^d), \tag{7.44a}$$
$$x_{1,2} \geq \Lambda_2 - \Delta_1 - R\, e_{1,2}^4 - R\,(c_1^a - c_1^d) - R\,(c_2^a - c_2^d). \tag{7.44b}$$

Similarly, when both are split, the constraints are

$$x_{1,2} \geq \Lambda_1 - (\Delta_2 - 1) - R\, e_{1,2}^1 - R(1 - c_1^a - c_1^d) - R\,(1 - c_2^a - c_2^d), \tag{7.45a}$$
$$x_{1,2} \geq \Lambda_2 - (\Delta_1 - 1) - R\, e_{1,2}^4 - R(1 - c_1^a - c_1^d) - R\,(1 - c_2^a - c_2^d). \tag{7.45b}$$

In this case we only subtract 1 to the initial times to overcome the split.

We proceed in a similar way when only one of the movements is split although in these cases it is slightly more complicated. In the first place we focus in the case in which the movement of robot 1 is split. We check which part of the movement may overlap with the movement of robot i_2. This is done by introducing a binary variable $b_{1,2}^1$ such that $b_{1,2}^1 = 1$ if $\Delta_1 \leq \Lambda_2$, meaning that the second part is overlapped, and $b_{1,2}^1 = 0$, otherwise, meaning that the candidate for overlap is the first part. This variable is obtained from the following constraints:

$$\Delta_1 - \Lambda_2 \leq R\,(1 - c_1^a - c_1^d) + R\,(c_2^a - c_2^d) + R\,(1 - b_{1,2}^1), \tag{7.46a}$$
$$\Lambda_2 - \Delta_1 \leq R\,(1 - c_1^a - c_1^d) + R\,(c_2^a - c_2^d) + R\, b_{1,2}^1. \tag{7.46b}$$

Depending on the value of $b_{1,2}^1$ we transform the constraints differently. When $b_{1,2}^1 = 1$ we add one to the final time of the movement of i_1. This can be seen as moving the part of the movement that is at the beginning of the plan to the right side. Thus, the constraints are

$$x_{1,2} \geq (\Lambda_1 + 1) - \Delta_2 - R\, e_{1,2}^1 - R\,(1 - c_1^a - c_1^d) - R\,(c_2^a - c_2^d) - R\,(1 - b_{1,2}^1), \tag{7.47a}$$
$$x_{1,2} \geq \Lambda_2 - \Delta_1 - R\, e_{1,2}^4 - R\,(1 - c_1^a - c_1^d) - R\,(c_2^a - c_2^d) - R\,(1 - b_{1,2}^1). \tag{7.47b}$$

On the contrary, when $b_{1,2}^1 = 0$, we subtract one to the initial time of the movement to compare the movement of robot i_2 with the left part of the movement of i_1. The constraints result in

$$x_{1,2} \geq \Lambda_1 - \Delta_2 - R\, e_{1,2}^1 - R\,(1 - c_1^a - c_1^d) - R\,(c_2^a - c_2^d) - R\, b_{1,2}^1, \tag{7.48a}$$
$$x_{1,2} \geq \Lambda_2 - (\Delta_1 - 1) - R\, e_{1,2}^4 - R\,(1 - c_1^a - c_1^d) - R\,(c_2^a - c_2^d) - R\, b_{1,2}^1. \tag{7.48b}$$

The same procedure can be followed for the case in which the movement of i_2 is split. We include the binary variable $b_{1,2}^2$ with

$$\Delta_2 - \Lambda_1 \leq R\,(c_1^a - c_1^d) + R\,(1 - c_2^a - c_2^d) + R\,(1 - b_{1,2}^2), \tag{7.49a}$$
$$\Lambda_1 - \Delta_2 \leq R\,(c_1^a - c_1^d) + R\,(1 - c_2^a - c_2^d) + R\, b_{1,2}^2. \tag{7.49b}$$

When $b^2_{1,2} = 1$, the constraints are

$$x_{1,2} \geq \Lambda_1 - \Delta_2 - R\, e^1_{1,2} - R\,(c^a_1 - c^d_1) - R\,(1 - c^a_2 - c^d_2) - R\,(1 - b^2_{1,2}), \tag{7.50a}$$
$$x_{1,2} \geq (\Lambda_2 + 1) - \Delta_1 - R\, e^4_{1,2} - R\,(c^a_1 - c^d_1) - R\,(1 - c^a_2 - c^d_2) - R\,(1 - b^2_{1,2}), \tag{7.50b}$$

and, otherwise,

$$x_{1,2} \geq \Lambda_1 - (\Delta_2 - 1) - R\, e^1_{1,2} - R\,(c^a_1 - c^d_1) - R\,(1 - c^a_2 - c^d_2) - R\, b^2_{1,2}, \tag{7.51a}$$
$$x_{1,2} \geq \Lambda_2 - \Delta_1 - R\, e^4_{1,2} - R\,(c^a_1 - c^d_1) - R\,(1 - c^a_2 - c^d_2) - R\, b^2_{1,2}. \tag{7.51b}$$

7.6.5 Optimal Schedule

Finally, we are in position to formulate the complete problem of finding the optimal schedule for the team of robots.

Problem 7.6.1. *The optimal periodic schedule, in which each robot executes its own periodic plan and which guarantees that no collisions occur and that the time in which two or more movements overlap is minimum, is the solution of the following MILP:*

$$\min f_{schedule}$$

subject to the restrictions introduced between (7.23) *and* (7.51).

The variables that define the team schedule are φ_i, that represent the time that each individual plan has to be shifted to produce the team plan. Recall that only $I - 1$ of these variables are needed. Additionally, we have included the variables that are summarized in Table 7.1. In particular, x_{i_1,j_1,i_2,j_2} and z_{i_1,j_1,i_2,j_2} are real-valued while the others are binary.

It is important to emphasize that, although the number of variables and restrictions is high, the problem can be solved efficiently using standard state-of-the-art solvers [148]. Particularly, for a size of 3 robots and 6 points, that can be appropriate for the the application in which we focus, the problem is solved in less than $15ms$ on an average laptop. We provide more details on the computational time in the simulations, Section 7.7. In addition, the solution to the problem is independent of the value of the period since all the times are obtained as a fraction of such period. In fact, the period can be calculated separately depending on the desired performance of the system. For instance, it can be set to the maximum time that a point can remain uncovered to ensure that it is covered more frequently. Another alternative is to calculate the period depending on the maximum time allocated to move.

On the downside, there are several reasons for which a feasible solution may not exist. For instance, it may happen if two agents have a common path but in opposite directions, if the space in which the agents can move is limited, or if an agent has to go through a point of the environment that is never left unoccupied by another agent. They can be avoided by modifying the paths of the agents in at least three ways: (i) invert the direction of the

Name	Number of variables
$z_{i_1,j_1,i_2,j_2}, x_{i_1,j_1,i_2,j_2}$	$\sum_{i_1=1}^{I-1} \sum_{i_2=i_1+1}^{I} \lvert\mathcal{Q}_{i_1}\rvert\lvert\mathcal{Q}_{i_2}\rvert$
$e^k_{\lvert\mathcal{Q}_{i_1}\rvert}, k \in \{1,\dots,4\}$	$4\sum_{i_1=1}^{I-1} \sum_{i_2=i_1+1}^{I} \lvert\mathcal{Q}_{i_1}\rvert\lvert\mathcal{Q}_{i_2}\rvert$
$c^a_{i,j}$	$\sum_{i=2}^{I} \lvert\mathcal{Q}_i\rvert - 1$
$c^d_{i,j}$	$\sum_{i=2}^{I} \lvert\mathcal{Q}_i\rvert$
c^a_i	$I-1$
$c^{cc}_{i_1,j_1,i_2,j_2}$	Number of possible coverage overlaps
$c^{mm}_{i_1,j_1,i_2,j_2}, c^{mc}_{i_1,j_1,i_2,j_2}$	Number of possible collisions

Table 7.1: Name and number of additional variables included to formulate the problem as a MILP.

movement, (ii) change the points that are assigned to each agent, or (iii) use a different cost function for the path optimization (7.8). The best solution in each case depends on the application.

Finally, recall that this schedule minimizes the simultaneous motion of agent while satisfying the collision avoidance restrictions. The problem could be limited to satisfy them but it is worth to optimize another criteria at the same time. Actually, the restrictions might be included in the problem of the times and production, but this would exponentially increase the complexity of the problem and would still not guarantee the global optimum to be found.

7.7 Simulations

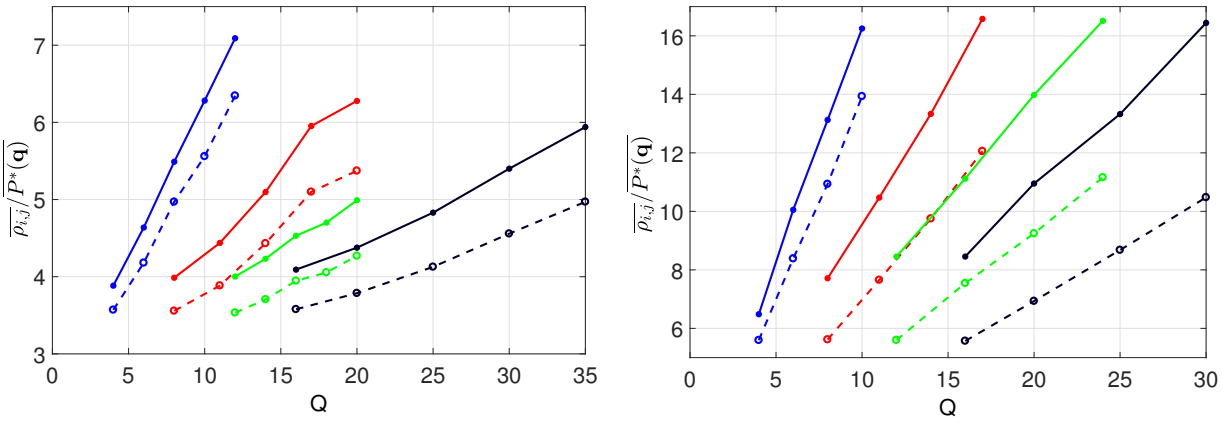
In this section we present simulation results for the proposed approach to the persistent coverage of a discrete environment. We focus on the existence of solution for the calculation of the coverage times and actions from Section 7.3.1; on the influence of the paths in the final solution (Sections 7.3-7.5); on the influence of the cost function used to calculate the optimal times and action; and on the computational cost of the different parts of the solution.

The simulations are implemented in Matlab, most of them have been carried out through a Monte-Carlo analysis and the results are aggregated for different values of the parameters, specifically, they are averaged results.

Existence of Solution

In the first place we present simulation results on the accuracy of the sufficient conditions on the existence of solution. The environment is a rectangle of 1×1.5 units in which $Q = 20$

interest points are deployed randomly and covered by $I = 5$ robots. We have simulated different values for the productions of the robots, which are fixed to the maximum, and the desired coverage, including a random component, and the result are shown in Figure 7.2. The solid lines represent how much production of the robots is required on average per desired coverage unit to satisfy the conditions and the dashed lines, for the problem to have a solution. Therefore, the difference between the solid and dashed lines of the same color represents how accurate is our sufficient condition with respect to the existence of solution. It can be seen that they are close to each other and that this accuracy increases when the number of robots and points is lower.



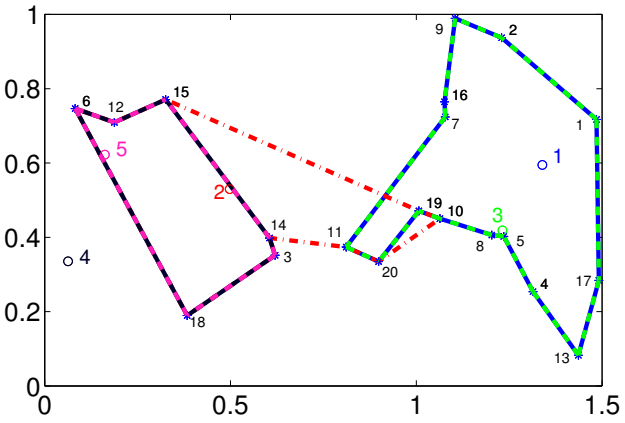
(a) Total reachability.

(b) Partial reachability.

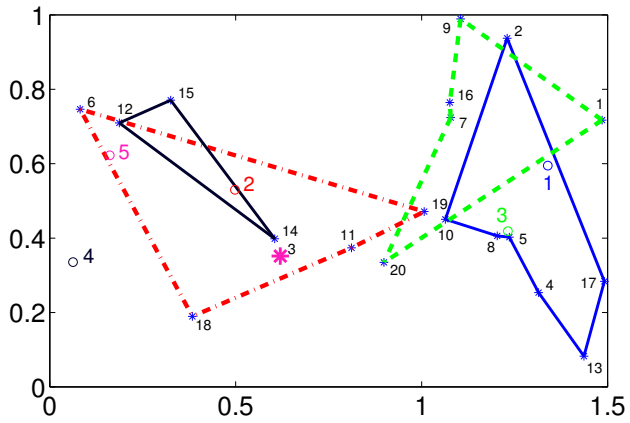
Figure 7.2: Sufficient conditions. Solid lines represent the quotient between the average consumption and the average accumulation when the sufficient condition is satisfied and dashed lines when the problem has a solution. Different results are depicted in blue, red, green and black for $I = 4, 8, 12, 16$ robots, respectively.

Influence of Paths

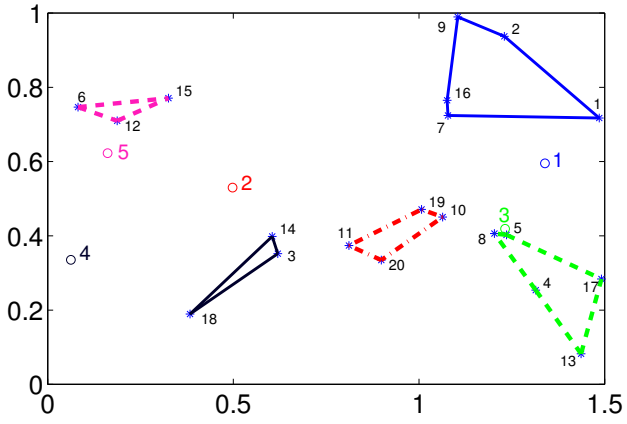
In the second place, we show an example of the evolution of the resulting trajectories for the initial solution with predefined paths, Fig 7.3a, for the iterated solution to shorten the paths, Fig. 7.3b and the entire solution with non-predefined paths, Fig. 7.3c. Table 7.2 reports the numerical results of this example where the cost function is $\sum_{\mathbf{q}\in\mathcal{Q},r\in\mathcal{R}} \theta_{\mathbf{q},r}$ and the productions are again fixed to their maximum value. It can be seen that the initial paths of different robots are overlapped and a long traveled distance is required. However, there are many points that do not need to be covered by more than one robot since the iterated solution provides much shorter paths and shorter moving times while keeping the values of the times. In the third case, the solution with variable paths obtains slightly longer coverage times but drastically reduces the moving times and the distance. For this reason, we use the sum of the coverage times and the move times as a metric for performance comparison, as in the third row of Table 7.2.



(a) Initial solution with predefined paths.



(b) Solution with shortened paths.



(c) Solution for non-predefined paths.

Figure 7.3: Example of the evolution of the resulting paths for the different solutions with $N_r = 5$ and $N_q = 20$. The blue asterisks represent the interest points and the paths are represented in different colors and line types for each robot.

Paths	Predefined	Shortened	Non-Predefined
$\sum_{\mathbf{q}\in\mathcal{Q},r\in\mathcal{R}} \theta_{\mathbf{q},r}$	1.572	1.572	1.601
$\sum_{r\in\mathcal{R}} \theta_r^m$	0.629	0.402	0.210
$\sum_{\mathbf{q}\in\mathcal{Q},r\in\mathcal{R}} \theta_{\mathbf{q},r} + \sum_{r\in\mathcal{R}} \theta_r^m$	2.201	1.974	1.811
Distance	11.358	7.270	3.793

Table 7.2: Results on problem reduction.

Now we focus on the iterative solution. In practice, the final coverage times do not differ from the initial ones (first row of Table 7.2) but, as can be seen in Fig. 7.4, the improvement of the total traveled distance is important. The greatest improvement takes place in the first iteration and the improvement of the second and subsequent iterations is smaller, specially when there are few robots in the environment. This means that a fast solution found in the first iteration can be enough and avoid more computations.

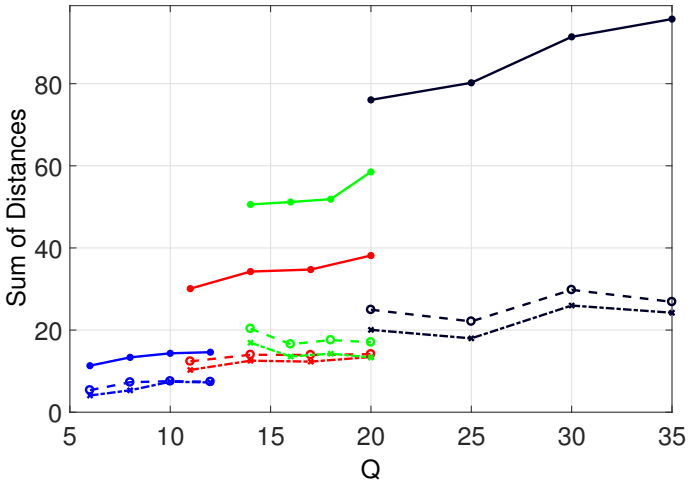


Figure 7.4: Evolution of the distance for the iterative solution. Solid lines represent the values for the initial solution; dashed lines, for the first iteration; and dash-dotted lines, for the second iteration. Different results are depicted in blue, red, green and black for $I = 4, 8, 12, 16$ robots, respectively.

We also compare the entire coverage performance of the iterated solution and the solution with non-predefined paths. We calculate the relative improvement of the solution with non-predefined paths with respect to the iterated solution in two situations: when the moving times represent a substantial part of the period (dash-dotted lines) and the opposite (dashed lines). As mentioned before, the metric that we use is the sum of the coverage times plus the move times. Thus, we calculate the relative difference on this metric calculated for both solutions. Fig. 7.5 shows that, in the first case, the solution with non-predefined paths performs between a 15 and a 40% better than the iterated. In the second case, the improvement is less significant, being more pronounced when the ratio robots/points is low.

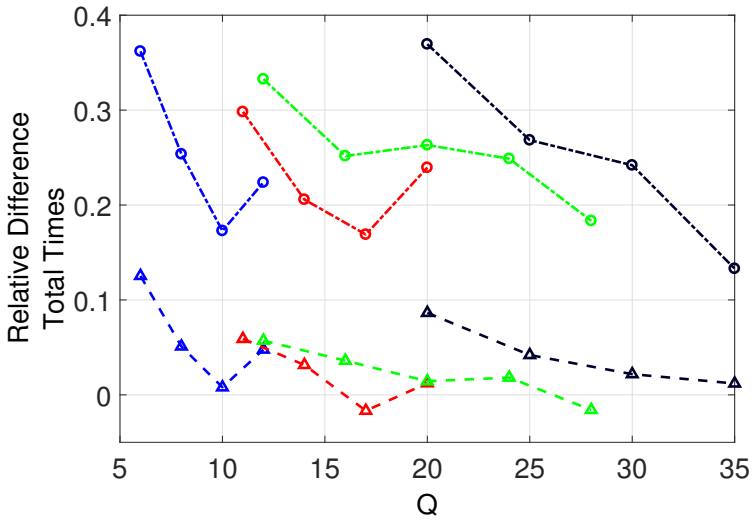


Figure 7.5: Relative difference in the sum of times $\theta_{\mathbf{q},r}$ and θ_r^m between the iterated solution and the solution with non-predefined paths. Dash-dotted lines represent the results when motion takes much more time than coverage and dashed lines the opposite. Different results are depicted in blue, red, green and black for $I = 4, 8, 12, 16$ robots, respectively.

Influence of the Cost Function

In this section we evaluate the results of our solution for different cost functions used in the calculation of the optimal times and productions. This function has an important influence in the final paths of the robots and, therefore, in the team plan and the quality of the coverage provided. In particular, we focus on the case of linear functions because they simplify the problem and are appropriate for the application of mobile induction heating, that we introduce in Chapter 8. The first function that we evaluate is

$$f_1(\theta_{\mathbf{q},r}) = \sum_{i\in\mathcal{I}} \sum_{j\in\Gamma_i} \left(-\theta_{i,j} + \frac{1}{\rho_i^{\max}} \rho_{i,j} \right), \tag{7.52}$$

in which the goal is to maximize the assigned times and minimize the normalized productions to provide a coverage as homogeneous in time as possible. The second alternative only tries to minimize the productions in order to reduce the maximum actions of the robots:

$$f_2(\theta_{\mathbf{q},r}) = \sum_{i\in\mathcal{I}} \sum_{j\in\Gamma_i} \frac{1}{\rho_i^{\max}} \rho_{i,j}, \tag{7.53}$$

In the third function we include a weight to the times with respect to f_1 that is the inverse of the normalized distance from the robot to the point,

$$f_3(\theta_{\mathbf{q},r}) = \sum_{i\in\mathcal{I}} \sum_{j\in\Gamma_i} \left(-\frac{1}{d_{\mathbf{p}_i^0,\mathbf{q}}} \theta_{i,j} + \frac{1}{\rho_i^{\max}} \rho_{i,j} \right), \tag{7.54}$$

and in the fourth one we also weight the productions with the relative importance of the required coverage $P^*(\mathbf{q})$ with respect to the maximum production of the robot $\rho_i^{\max}$.

$$f_4(\theta_{\mathbf{q},r}) = \sum_{i\in\mathcal{I}} \sum_{j\in\Gamma_i} \left(-\frac{1}{d_{\mathbf{p}_i^0,\mathbf{q}}}\,\theta_{i,j} + \frac{P^*(\mathbf{q})}{\rho_i^{\max 2}}\,\rho_{i,j} \right), \tag{7.55}$$

In the last two alternatives the times are weighted using the clustering procedure introduced in Section 7.5, without the productions in

$$f_5(\theta_{\mathbf{q},r}) = \sum_{i\in\mathcal{I}} \sum_{j\in\Gamma_i} -\omega_{i,j}\,\theta_{i,j}, \tag{7.56}$$

and with weighted productions in

$$f_6(\theta_{\mathbf{q},r}) = \sum_{i\in\mathcal{I}} \sum_{j\in\Gamma_i} \left(-\omega_{i,j}\,\theta_{i,j} + \frac{P^*(\mathbf{q})}{\rho_i^{\max 2}}\,\rho_{i,j} \right). \tag{7.57}$$

The evaluation consisted in 10 runs of the algorithm with different positions of the points for different numbers of robots and points, all of them for the six cost functions. The required coverage was randomly selected between 400 and 2200 units while the maximum production of the robots was 5000 units. The number of robots and points was selected in a way that the team had enough production to cover the points which results in at most twice more points than robots.

Since the cost function influences the points that each robot finally visits, we first evaluate in how many cases the optimal solution was found, both for the calculation of times and productions and for the team plan scheduling. In Fig. 7.6 it can be seen that a solution was found in all the 190 runs for the times and production. However, the first two cost functions only allowed a solution for the scheduling in the 75% and 68% of the cases while the others allowed more than 95% of solutions. Recall that the existence of solution is discussed in Section 7.6.5.

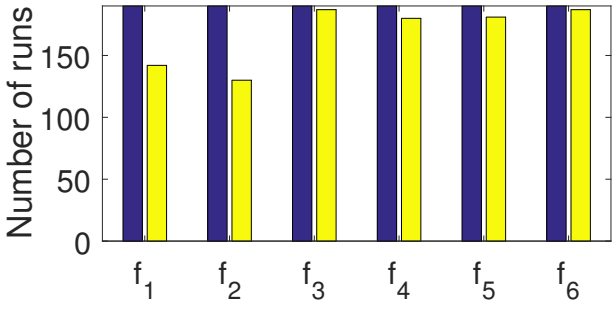


Figure 7.6: Number of solutions found for the times and productions (dark blue) and for the scheduling (yellow).

In Table 7.3 we compare six different metrics obtained from the simulations to give an idea of the behavior with each function. The first metric is the average number of iterations

	f_1	f_2	f_3	f_4	f_5	f_6
Iterations	1.11	**1.10**	1.27	1.12	1.36	1.13
Movements	**1.43**	1.44	1.44	1.44	1.45	1.44
Total Coverage	0.95	0.95	0.94	0.95	**0.97**	**0.97**
Uncovered Time	0.30	0.30	0.32	0.31	**0.29**	**0.29**
Max Production	0.73	0.73	0.93	0.69	0.81	**0.68**
Homogeneity	3.03	**2.93**	3.18	3.16	3.45	3.24

Table 7.3: Comparison of metrics between cost functions.

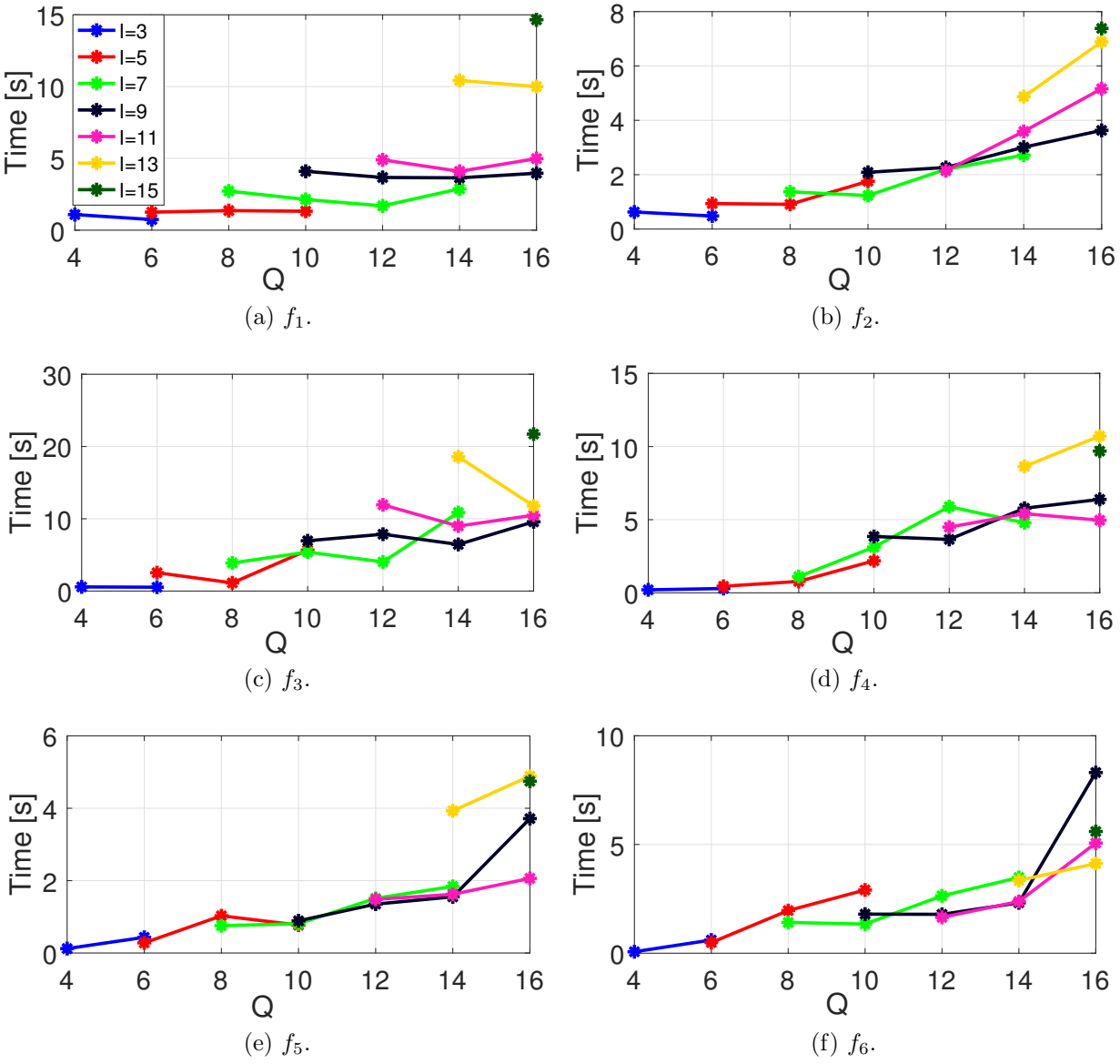
of Algorithm 5 to find the final solution. With f_3 and f_5 it is slightly bigger, which means that these cost functions allow the paths to be progressively shortened more than the others. The second metric is the average number of movements per robot during each period, which is really similar for all the functions. The third row includes the average time that each robot dedicates to cover the points in a period. The differences are again small with f_5 and f_6 producing a 2% better results. In the fourth place we calculate the maximum time that a point remains uncovered every period. The same functions f_5 and f_6 give better results even though there is not much difference between all of them. The fifth metric shows the maximum production required to the robots and normalized by the maximum production available. In this case, f_3 performs much worse than the others while f_4 and f_6 require the smaller maximum productions. Finally we compute a metric that determines the homogeneity of the coverage. It is defined as

$$h = \sum_{\mathbf{q}\in\mathcal{Q}} \sum_{i\in\mathcal{I}} \left(\theta_{i,j} - \frac{P^*(\mathbf{q})}{\rho_i^{\max}} \right)^2 , \tag{7.58}$$

and the results show that, on average, the cost functions f_1 and f_2 provide the most homogeneous coverage. As a conclusion, one can see that all the alternatives have advantages and disadvantages and the selection must be made depending on the application.

Computational Time

Finally we pay attention to the computational times of the two main steps of our solution. In Fig. 7.7 we show the computation cost incurred to obtain the coverage times and productions solving problem (7.8). Although in all the cases the increase of the time is more than linear with the number of robots and points, for a team of 13 or 15 robots, the computation requires around 20 seconds in the worst case, i.e., with f_3. This time is negligible for most persistent coverage applications in which the robots are required to carry out the task indefinitely. In addition, the difference between cost functions is significant. The worst-case computation required with f_3 is around 5 times lower with f_5 and in some cases, such as with f_1, the computation increases only with the number of robots.



(a) f_1. (b) f_2.

(c) f_3. (d) f_4.

(e) f_5. (f) f_6.

Figure 7.7: Mean calculation time of the coverage times and productions solving problem (7.8). Each figure represents the results with a different cost function and different colors represent different number of robots.

The results on the computation time of the scheduling are very similar for all the cost function and, therefore, we only show in Fig. 7.8 the times for f_1. These times are in the order of milliseconds for small teams and only increase to approximately one second for the biggest teams and numbers of points. According to these results, the computational complexity resides in problem (7.8) rather than in Problem 7.6.1.

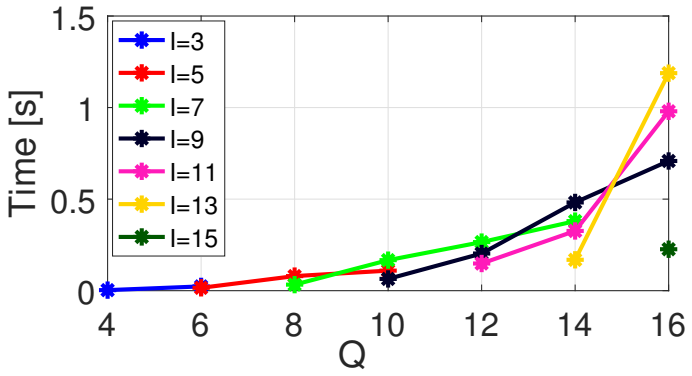


Figure 7.8: Mean calculation time of the scheduling solving Problem 7.6.1.

7.8 Conclusions

In this chapter we have introduced a periodic solution to the persistent coverage of a discrete environment. We have proposed a divide-and-conquer solution to divide the entire problem in three smaller subproblems. We obtain the optimal coverage times and actions to satisfy the coverage objective of the environment by solving a QCLP. We have presented guarantees on the existence of solution for this program, an algorithm to iterate its solution and shorten the robots paths, and an alternative cost function to include the lengths of the paths in the problem. We have also formulated a MILP to find a periodic team schedule in which no collisions occur.

The complete problem is hard to solve at once or even intractable and the computational complexity typically grows exponentially with the number of robots and points. Our divide-and-conquer strategy counteracts this growth and allows us to find the optimal solution for each subproblem independently. In fact, simulation results support our proposal and demonstrate that it is tractable for significant team and environment sizes. Moreover, although the final solution to the entire problem may not be the optimal, this is a conceptually simple and piecewise optimal way of solving the problem. In fact, this is the first attempt to calculate simultaneously the paths and the coverage times and actions, that allows the collaboration of several robots at the same point and guarantees collision avoidance.

The solution presented in this paper is valid for general discrete environments but it has been designed keeping in mind its applicability to the problem of domestic induction heating with mobile inductors. In the following chapter we particularize it to this problem to fulfill the second objective of this thesis.

Chapter 8

Application to Mobile Induction Heating

In this chapter we particularize the previous solution to the application of domestic induction heating with mobile induction. The problem is to persistently heat a finite set of pots with a team of mobile inductors. Since the motion of the inductors is very constrained due to the small size of the hob with respect to them, we develop a particular collision avoidance policy to guarantee the heating of all the pots. We show the performance of the entire solution with experiments on a real prototype of an induction hob with mobile inductors carried by robotic arms.

8.1 Introduction

This chapter of the thesis is devoted to the application of a persistent coverage solution to the problem of domestic heating with mobile inductors. In particular, we apply and particularize the solution from the previous chapter, that was already developed for a discrete environment and with some choices taken keeping in mind this application.

The most important feature of this proposal is that it addresses an open problem which offers a significant opportunity to increase the flexibility of domestic hobs at a moderate increase of cost. Moreover, it can change the way in which daily cooking is confronted and transform it to a new family experience that can be combined with many other tasks at the same time. This can be done by offering really big cooking areas in the countertop or even in other kind of furniture that no one could think of it as a hob (Fig. 1.6). The mobile inductors provide the flexibility to reach and cover such big areas while being cheaper than current flexible solutions and our periodic strategies guarantee a perfect heating of the pots. In fact, the main advantage of this solution is that it guarantees that the set of pots placed by the user receive the desired power. In particular, if each pot can have its own inductor assigned, the power is guaranteed all the time and, if the inductors have to be shared between the pots, it is guaranteed periodically or, equivalently, on average. Not only there are few previous attempts to find these kind of periodic coverage strategies with cooperation of the robots but none of them has been applied to induction heating before.

The main restrictions that have to be faced for mobile induction heating are twofold:

- **Requested power restrictions:** all the devices connected to the electric grid must comply with the European norm UNE-EN 61000-3-3, 2013, that limits the temporal changes in power requested from the grid. In this application, this changes occurs when an inductor arrives at a pot and starts heating it or stops heating it to move to the next one in its path. To overcome this limitation we minimize the simultaneous motion of the inductors with the cost function of the scheduling in Section 7.6. The changes of requested power are therefore minimized when there is no need to overlap movements since the minimum change occurs when only one inductor stops requesting power before requesting it again. Otherwise, finding the minimum overlap maximizes the possibility of compensating the requested power with other inductors to comply with the norm.

- **Spatial restrictions:** they come from the limited space that the inductors have to share inside the hob and the motion restrictions of the robotic arms. In this cluttered environment, an effective collision avoidance policy is critical for a good performance of the system and, for this reason, we modify the scheduling of the previous chapter to provide a more precise and efficient collision avoidance.

In summary, in this chapter we apply our periodic solution to the problem of persistently heating a finite set of pots with a team of mobile inductors. In the first place, we calculate a static assignment in which each pot is heated by the same inductor all the time, if it is possible. If not, we apply our periodic strategy. To do so, we introduce some particularities in the planning of the robot paths and a new collision avoidance method, more appropriate for the environment. Finally, we carry out a thorough experimental validation of the proposal with a real prototype.

The remainder of this chapter is structured as follows. In Section 8.2 we present the static assignment and in Section 8.3 the particularities of the periodic coverage strategy. Finally, we present the experimental results in Section 8.4 and conclusions in Section 8.5.

8.2 Static Assignment

In general, persistent coverage is applied in environments where the number of points to be covered is higher than the number of available robots. This always leads to strategies in which the robots have to move from one point to another to cover them. However, in domestic heating, most of the time the user cooks with only one or two pots at the same time, even though he usually has three or four heating zones. In a flexible hob like the one that we propose, this is not expected to change and at least three inductor can be placed in the hob. This means that, in many situations, there are more available inductors, i.e., robots, than pots, i.e., points. For these reason we need to find an static assignment,

$$\mathcal{S} = \big\{(\mathbf{q}, i_{\mathbf{q}}), \mathbf{q} \in \mathcal{R}_{i_{\mathbf{q}}} \mid \mathbf{q}_1 \neq \mathbf{q}_2 \iff i_{\mathbf{q}_1} \neq i_{\mathbf{q}_2},\ \forall\, \mathbf{q}, \mathbf{q}_1, \mathbf{q}_2 \in \mathcal{Q}\big\},$$

in which each pot is covered by only one inductor that remains static covering it, that is, $\forall\, k \geq 0$

$$\begin{aligned} \mathbf{p}_i(k) &= \mathbf{q}, \\ \rho_i(\mathbf{q}) &= P^*(\mathbf{q}). \end{aligned}$$

Such a static assignment guarantees that the original coverage objective, $Z(\mathbf{q}, t) = Z^*(\mathbf{q})$, can be maintained all the time.

In the first place, we formalize in which conditions the reachability of the robots allows such static assignment to exist.

Proposition 8.2.1. *Consider that $I \geq Q$. If for any pair $\mathbf{q}_1, \mathbf{q}_2 \in \mathcal{Q}$ with $\mathbf{q}_1 \neq \mathbf{q}_2$, the conditions*

$$|\mathcal{I}_{\mathbf{q}_1}| = 1, \tag{8.1a}$$
$$|\mathcal{I}_{\mathbf{q}_2}| = 1, \tag{8.1b}$$
$$\mathcal{I}_{\mathbf{q}_1} = \mathcal{I}_{\mathbf{q}_2} \tag{8.1c}$$

hold, then, a feasible static assignment does not exist.

It is clear that the condition $I \geq Q$ is necessary for the assignment to exist, since at least as many inductors as pots must be available, but it is not sufficient. The conditions from (8.1) are needed to guarantee that there is no pair of pots that can only be covered by the same inductor. Recall that $\cup_{i \in \mathcal{I}} \mathcal{Q}_i = \mathcal{Q}$.

Additionally, every inductor must be able to provide exactly the power required by its assigned pot.

Proposition 8.2.2. *A static assignment $\mathcal{S}$ must satisfy*

$$\rho_{i_\mathbf{q}}^{\max}(\mathbf{q}) \geq P^*(\mathbf{q}), \qquad \forall\, (\mathbf{q}, i_\mathbf{q}) \in \mathcal{S},$$

to be valid.

In the case that at least a feasible assignment exists, we seek the one that minimizes the time needed by the inductors to reach their assigned pots. We call $f_i(\mathbf{q})$ the function that gives the time needed by inductor i to arrive at pot $\mathbf{q}$. Thus, the optimal assignment is the solution to

$$\begin{aligned} \underset{\mathcal{S}}{\text{minimize}} \quad & \sum_{\mathbf{q} \in \mathcal{Q}} \sum_{i \in \mathcal{I}} f_i(\mathbf{q})\, x(i, \mathbf{q}), \\ \text{subject to} \quad & \sum_{\mathbf{q} \in \mathcal{Q}} x(i, \mathbf{q}) \leq 1, \qquad \forall\, i \in \mathcal{I}, \\ & \sum_{i \in \mathcal{I}} x(i, \mathbf{q}) = 1, \qquad \forall\, \mathbf{q} \in \mathcal{Q}, \end{aligned} \tag{8.2}$$

where $x(i, \mathbf{q}) = 1$ if $(\mathbf{q}, i) \in \mathcal{S}$ and $x(i_\mathbf{q}, \mathbf{q}) = 0$, otherwise. To find the optimal assignment, which solves (8.2) in polynomial time we make use of the Hungarian algorithm [146]. Following such assignment, the inductors move to their respective destinations and start covering.

The coverage action that they must in the transient is their maximum coverage action to reach the coverage objective in the minimum time and in the steady-state is equal to the instant power requested by the potsto maintain the desired level. Therefore,

$$\rho_{i_{\mathbf{q}}}(\mathbf{q}) = \begin{cases} P^*(\mathbf{q}), & \text{if } Z(\mathbf{q},t) = Z^*(\mathbf{q}), \\ \min(\rho_{i_{\mathbf{q}}}^{\max}(\mathbf{q}), Z^*(\mathbf{q}) - Z(\mathbf{q},t)), & \text{if } Z(\mathbf{q},t) < Z^*(\mathbf{q}), \end{cases}$$

8.3 Periodic Coverage

The main difficulty of heating a set of pots with mobile inductors appears when there are more pots than available inductors or, more generally, when a static assignment is not possible. Although it does not happen very frequently, it is required to make the hob flexible and provide the user total freedom in his daily cooking. In the cases that it happens, we transform the coverage objective in the periodic one from (7.7) and apply the strategy from Chapter 7. In this section we introduce the different modifications that it requires.

8.3.1 Water Boiling

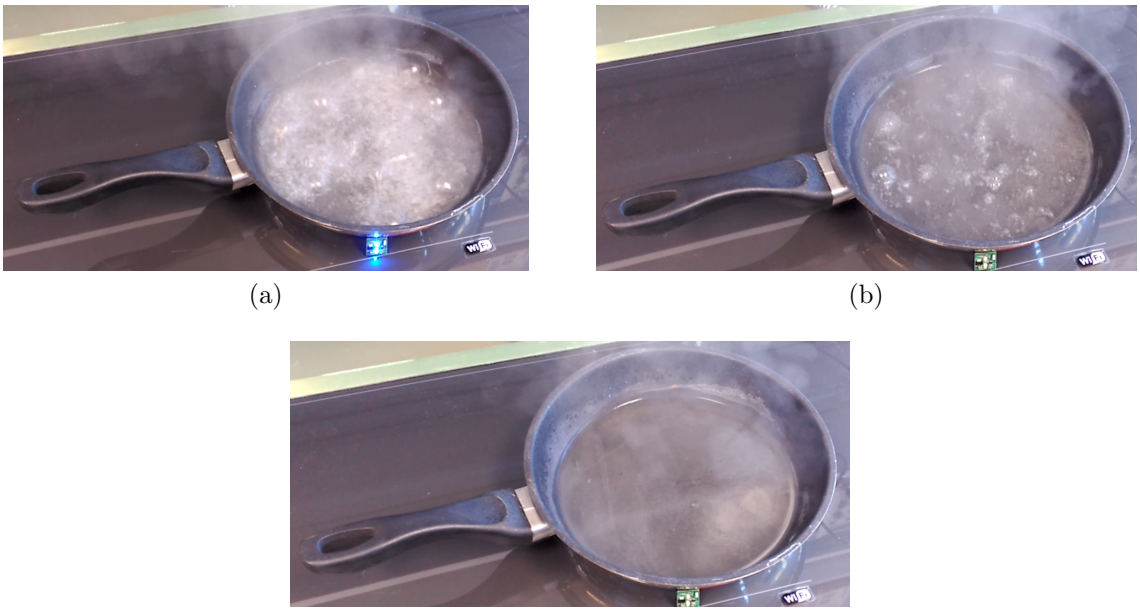
There is a particular situation in which a periodic strategy or any other approach that relies on heating the pots from time to time is not suitable. This situation is the process of boiling water, that is very frequent in domestic cooking.

The temperature of the pots suffers variations when the inductors start and stop heating them. In most cooking processes, such as deep frying or just warming, the variation of the temperature can hardly be appreciated by the user. However, this effect can be negative in boiling processes, where the bubbles can appear and disappear when the power changes. An example can be seen in Fig. 8.1. When the pot is receiving power from the inductor, the water is boiling gently (Fig. 8.1a) but as soon as the power is swithed off, the boiling stops (Fig. 8.1b) and the bubbles disappear (Fig. 8.1c).

In order to avoid this undesirable situations, when a boiling process is taking place, the assignment of the inductors can be adjusted to keep an inductor under the boiling pot all the time. From the team perspective, this can be solved in two different manners. The fist one is to directly assign an inductor to the boiling pot and solve the periodic coverage for the rest of the pots with the rest of the inductors. The second alternative is to solve the problem for all the pots and inductors but fixing $\mathcal{Q}_i = \mathbf{q}$ and $\theta_i^m = 0$, with $\mathbf{q}$, the pot in which we want to boil water and i, the inductor that we want to assign to the pot. Both solutions are simple and computationally negligible.

8.3.2 Configuration Change and Path Planning

The first step of the periodic coverage strategy presented in Algorithm 4 is to calculate the paths that the robots must follow. However, this is not straightforward for mobile inductors due to the robotic arms that are in charge of moving them.



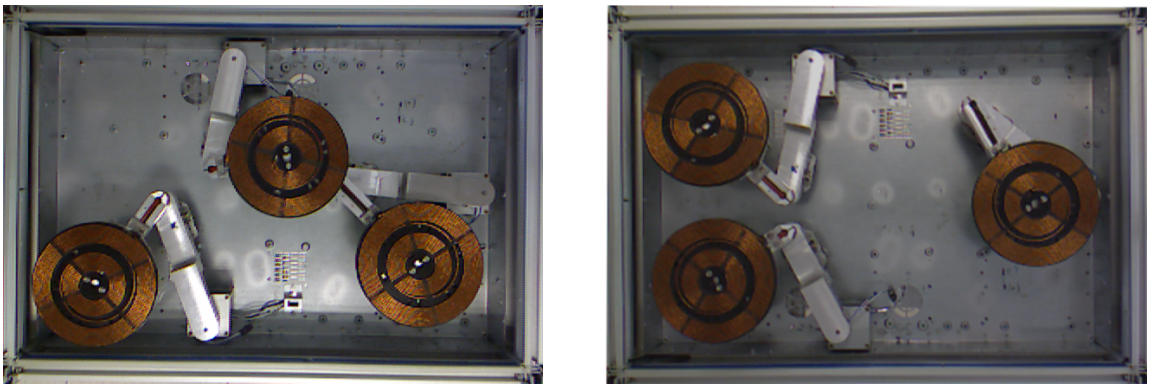
(a) (b)

(c)

Figure 8.1: Example of the effect of switching off the power of a pot with water boiling on it. (a) Water boiling at maximum power. (b) Water 5 seconds after switching off the power. (c) Water 10 seconds after switching off the power.

The arms are composed of two rotational joints and two link. The first link connects the joint that is attached to the hob chassis with the second joint, that we call *elbow*. The second link connects the elbow with the inductor. We define the origin of the inductor with the first link perpendicular to the hob side and the second link in a 180°-angle with the first one, that is, with the arm totally folded. From this origin there are two possible directions of turn for the motor of the second joint. Depending on the direction of the turn, the arm can adopt two different configurations as we show in Fig. 8.2 for the top arm. If we trace a line from the first joint to the inductor center, the elbow can be placed at the right side, leading to the elbow-right configuration (Fig. 8.2a), or at the left side, leading to the elbow-left configuration (Fig. 8.2b).

Most of the points of the hob can be reached with the arm in both configurations. Nevertheless, some areas are only reachable with one configuration because the arm would collide with the hob borders in the other configuration. An example can be seen in Fig. 8.2a for the bottom arm. The elbow-left configuration is not feasible for this position of the inductor because it is too close to the border. This forces to change the configuration of the arm when it tries to reach a position in which the current configuration is not feasible and, to this end, it is necessary to go through the origin to change the configuration without collisions. This has to be taken into account when planning the paths of the inductors between pots. In particular, when solving the TSP that gives the closed paths of the inductors, we include this configuration change in the cost of the path between any two pots. This should also be taken into account for the collision avoidance, since the path between two points may



(a) (b)

Figure 8.2: Example of the two different configuration of the top robotic arm. (a) Elbow-right configuration. (b) Elbow-left configuration.

no longer be the straight line between them. However, the new avoidance method that we introduce next is inherently capable of handling this.

8.3.3 Collision Avoidance

The collision avoidance policy presented in 7.6 is overprotective in the sense that, if the movements of two robots may result into a collision, one of them is executed first and the other one has to wait until it finished to start. For instance, in Fig. 7.1c the movement between $\mathbf{q}_1$ and $\mathbf{q}_3$ of i_1 (blue path) may collide with the movement between $\mathbf{q}_5$ and $\mathbf{q}_1$ of i_3 (green path), specifically at $\mathbf{q}_1$. The restrictions express that i_1 cannot start moving from $\mathbf{q}_5$ until i_1 has reached $\mathbf{q}_3$. This is more restrictive than needed. In fact, the is avoided collision if only i_1 starts moving before i_3 reaches $\mathbf{q}_1$.

The hob environment is significantly small in comparison with general environments and the number of robots and points is very limited. In these conditions, a discretization of the environment and an analysis of what happens at each point becomes affordable and permits to include spatio-temporal restrictions, instead of planning constraints, to avoid collisions between the inductors. These constraints provide a solution much less overprotective than the previous one, which is necessary in such an environment where the space occupied by the inductors is significant with respect to the free space. It is important to note that the arms are designed and attached to the hob in such a way that the first links of any pair of them cannot collide. Therefore, only the collisions between the inductors have to be avoided.

The idea of the new policy that we propose is to discretize the hob and determine the time that each inductor is occupying each cell of the hob where a collision may occur as a function of the scheduling time, φ_i. Then, the restrictions only avoids that any inductor starts occupying a cell that is already occupied by another inductor.

In the first place, the hob is discretized using a square grid, as shown in Fig. 8.3. Each cell is referred to as c.

Since the individual paths of the inductors are known, the cells in which a collision

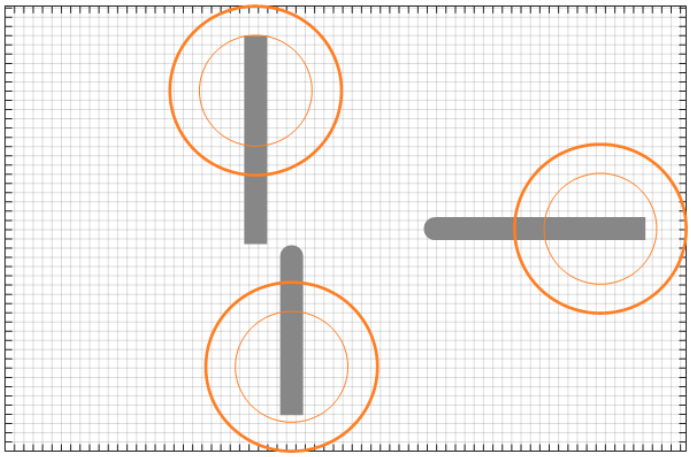


Figure 8.3: Example of the discretization of the hob.

between two inductors is possible are determined and the times in which the inductors arrive to those cells, $a'_{i,c}$, and depart from them, $d'_{i,c}$ are found. These arrival times can be transformed to the team plan as

$$\begin{aligned} A'_{i,c} &= \varphi_i + a'_{i,c}, && \text{if } \varphi_i + a'_{i,c} \leq 1, \\ A'_{i,c} &= \varphi_i + a'_{i,c} - 1, && \text{otherwise,} \end{aligned}$$

and the departure times as

$$\begin{aligned} D'_{i,c} &= \varphi_i + d'_{i,c}, && \text{if } \varphi_i + d'_{i,c} \leq 1, \\ D'_{i,c} &= \varphi_i + d'_{i,c} - 1, && \text{otherwise.} \end{aligned}$$

These times are calculated keeping in mind that an inductor can go through the same cell several times in the same period although we do not formulate it explicitly for the sake of simplicity of notation.

In order to make the arrival times amenable for the MILP formulation from Problem 7.6.1, we introduce the binary variables $c^{a'}_{i,c}$ so that $c^{a'}_{i,c} = 0$ if $\varphi_i + a'_{i,c} \leq 1$ and $c^{a'}_{i,c} = 1$, otherwise. Therefore, the times can be expressed as

$$A'_{i,c} = \varphi_i + a'_{i,c} - c^{a'}_{i,c},$$

subject to

$$\varphi_i + a'_{i,c} \leq 1 + R\, c^{a'}_{i,c}, \tag{8.3a}$$

$$-(\varphi_i + a'_{i,c}) \leq -1 + R\,(1 - c^{a'}_{i,c}), \tag{8.3b}$$

with R a sufficiently big number. Note that $c^{a'}_{i,c}$ has the effect of activating the appropriate constraint. The same can be done to the departure times, introducing $c^{d'}_{i,c}$:

$$D'_{i,c} = \varphi_i + d'_{i,c} - c^{d'}_{i,c},$$

subject to

$$\varphi_i + d'_{i,c} \leq 1 + R\, c^{d'}_{i,c}, \tag{8.4a}$$
$$-(\varphi_i + d'_{i,c}) \leq -1 + R\,(1 - c^{d'}_{i,c}). \tag{8.4b}$$

Finally, we are in the positions to impose the two sets of restrictions to these times that avoid collisions between inductors. The first set avoids that an inductor i_2 occupies a cell when inductor i_1 is already on it:

$$A'_{i_1,c} - A'_{i_2,c} \leq R\,(1 - c'_{i_1,i_2,c}), \tag{8.5a}$$
$$A'_{i_1,c} - D'_{i_1,c} \leq R\,(1 - c'_{i_1,i_2,c}), \tag{8.5b}$$
$$D'_{i_1,c} - A'_{i_2,c} \leq -\varepsilon + R\,(1 - c'_{i_1,i_2,c}), \tag{8.5c}$$
$$D'_{i_2,c} - A'_{i_1,c} \leq -\varepsilon + R\,(1 - c'_{i_1,i_2,c}) + R\,(1 - c^{d'}_{i_2,c} - c^{a'}_{i_2,c}), \tag{8.5d}$$

where ε is the minimum separation between a departure and an arrival to the same cell. The first restriction, (8.5a), activates the whole group if $A'_{i_1,c} \leq A'_{i_2,c}$, that is, if i_1 arrives to the cell before i_2. When this happens, the binary variable $c'_{i_1,i_2,c} = 1$. The second restriction, (8.5b), ensures that the time in which i_1 stays in the cell is not split by the end of the period. Following this, (8.5c) guarantees that i_2 does not arrive to the cell until i_1 has left it, i.e., $D'_{i_1,c} + \varepsilon \leq A'_{i_2,c}$. In the last one, (8.5b), the departure of i_2 is restricted to happen before the arrival of i_1 when the time that i_2 stays in the cell is split by the end of the period. In the same way, the second set avoids that i_1 occupies a cell when i_2 is already on it:

$$A'_{i_2,c} - A'_{i_1,c} \leq R\, c'_{i_1,i_2,c}, \tag{8.6a}$$
$$A'_{i_2,c} - D'_{i_2,c} \leq R\, c'_{i_1,i_2,c}, \tag{8.6b}$$
$$D'_{i_2,c} - A'_{i_1,c} \leq -\varepsilon + R\, c'_{i_1,i_2,c}, \tag{8.6c}$$
$$D'_{i_1,c} - A'_{i_2,c} \leq -\varepsilon + R\, c'_{i_1,i_2,c} + R\,(1 - c^{d'}_{i_1,c} - c^{a'}_{i_1,c}), \tag{8.6d}$$

With these restriction, the scheduling of the team plan is summarized in the following problem.

Problem 8.3.1. *The optimal periodic schedule, in which each robot executes its own periodic plan and which guarantees that no collisions occur and that the time in which two or more movements overlap is minimum, is the solution of the following MILP:*

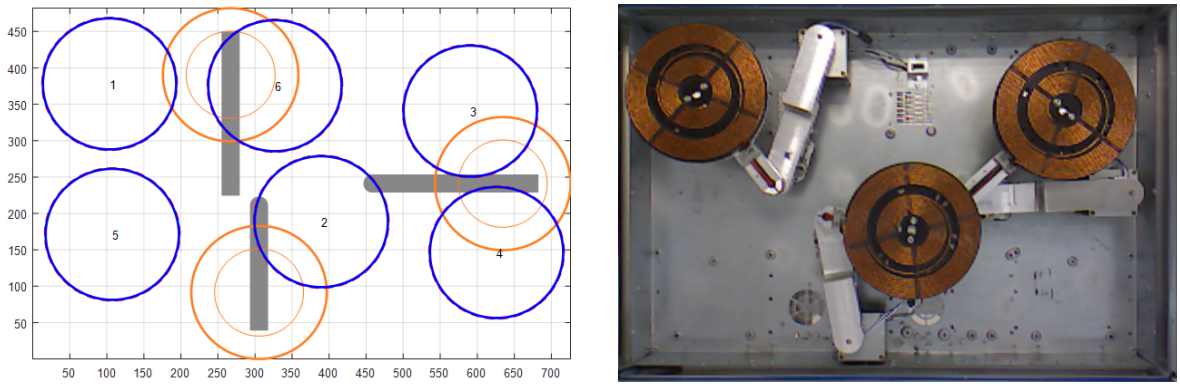
$$\min f_{schedule}$$

subject to the restrictions (7.23), (7.26), (7.27), (8.3) - (8.6) *and those coming from the cost function in Section 7.6.4.*

The number of restrictions in this case does not depend so strongly on the number of coverages and movements of the inductor, it does no the number of cells in which the environment is discretized.

8.4 Experiments

In this section we provide experimental results of the periodic heating strategy with the prototype of mobile inductors introduced before. Since the interest of this validation is in the movements of the inductors and the periodic strategy, in these experiments the pots have been simulated and the power has not been actually provided in order to accelerate the experimentation and guarantee the safety of the motion system. We have carried out rounds of 20 experiments for $Q = 4, 5$ and 6 pots. The results obtained for different configurations of the pots are very similar and, therefore, we only show results for the case of 6 pots.



(a) (b)

Figure 8.4: (a) Layout of one of the experiments. Blue circumferences represent the pots; orange circumferences, the two coils of the inductors; and grey segments, the two links of the robotic arms. (b) Snapshot of the prototype.

In Fig. 8.4a the layout of one of the experiments is shown. There are 6 virtual pots which are circles of diameter $18cm$. The pots require a constant $P^*(\mathbf{q}) = 1300, 1300, 1600, 800, 1500$ and $900W$, respectively. In Fig. 8.4b a snapshot of the prototype for the instant 0.3 of the period can be seen. Note that that, for instance, the right inductor has to go through its home position to change its configuration. Otherwise it could not reach pot 4 without colliding with the borders of the hob. This is one of the particularities of the system that have been addressed in this chapter.

The optimal schedule obtained for this configuration of inductors and pots, using the cost function (7.52) and solving Problem 8.3.1, is shown in Fig. 8.5. In this particular case each inductor was assigned to two different pots but this is not necessarily the result, as we show at the end of this section.

The power received by the pots can be seen in Fig. 8.6. One can see that all the pots receive power periodically for a period of time and then have to wait some time until the following coverage. This is a requirement of the system since there are more pots than inductors. However, the average received power quickly tends to the required power, meaning that, on average, the pots receive the desired power.

In Fig. 8.7 the power provided by the inductors is depicted along with the total power of the hob. The cost function of the scheduling that minimizes the time in which more than

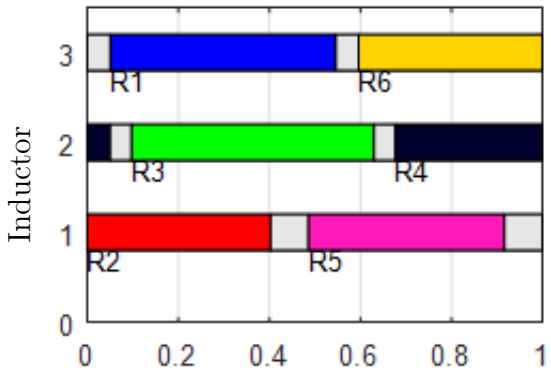


Figure 8.5: Optimal schedule obtained for the experiment. R1 to R6 refer to pots $\mathbf{q}_1$ to $\mathbf{q}_6$.

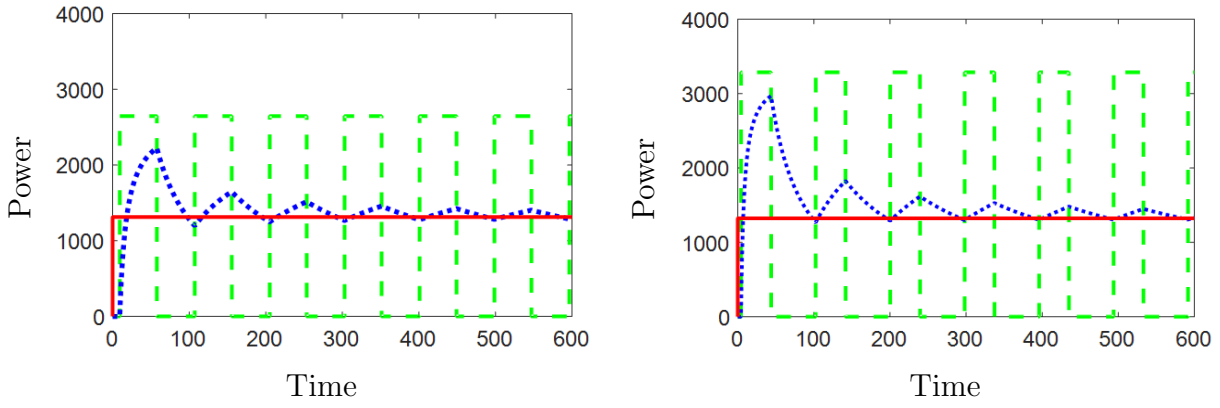
one inductor is moving at the same time favor that the variations of the total power are minimal. In addition, it allows us to adjust the total power of the hob when one inductor starts or stops requesting power with the other two inductors to satisfy the norm on the power requested to the grid.

Finally we show the results for four other experiments with the same cost function for the calculation of the optimal times and actions and different configurations of pots. One can see that each inductor is not necessarily assigned to two different pots.

8.5 Conclusions

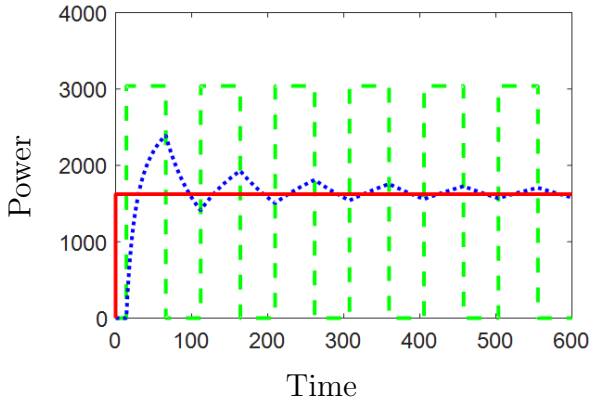
In this chapter we have particularized the periodic coverage solution from the previous one to a domestic heating environment with mobile inductors. We have introduced an optimal static assignment for the cases in which the inductors do not need to move to provide the power required by the pots. For the cases in which a periodic heating is needed, we have discussed the problem of boiling water in one of the pots and how the configuration of the arms affects the path planning. We have also presented a new collision avoidance policy for this application, that is much less overprotective and computationally affordable. This entire solution for mobile induction guarantees that the pots receive their required power periodically and a thorough experimental validation supports our proposal and demonstrates that a set of pots can be homogeneously heated with mobile inductors.

This proposal opens the possibility to create really innovative, high-value domestic hobs with immense opportunities for the designers and for the customers. In fact, it fulfills the technical solution of an innovation with a great potential.

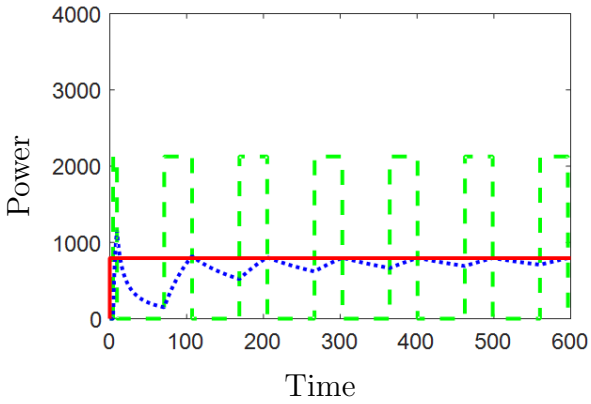


(a) Pot 1.

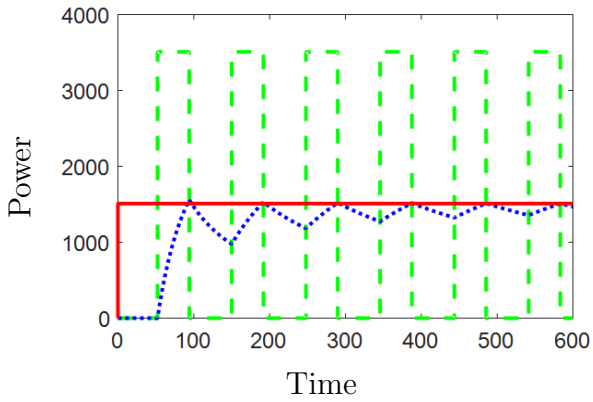
(b) Pot 2.



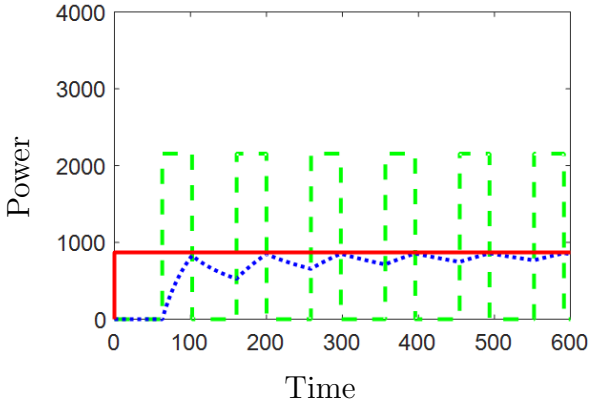
(c) Pot 3.



(d) Pot 4.



(e) Pot 5.



(f) Pot 6.

Figure 8.6: Power received by the pots during the experiment. The green line represents the instant received power; the red line, the requested power; and the blue line, the average received power.

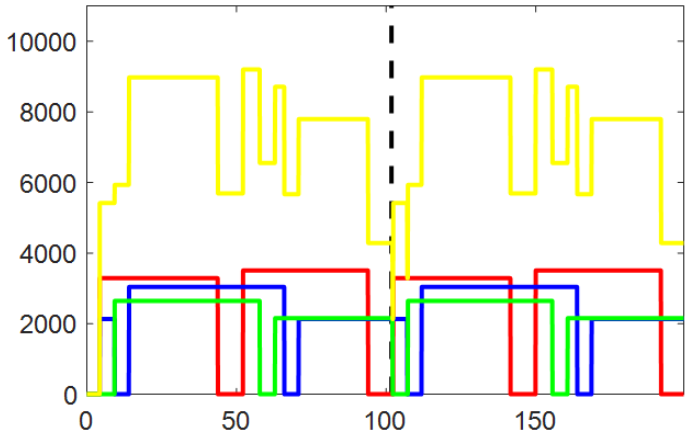


Figure 8.7: Power provided by the inductors during the experiment. The red, blue and green lines represent the power of each inductor and the yellow line, the total power.

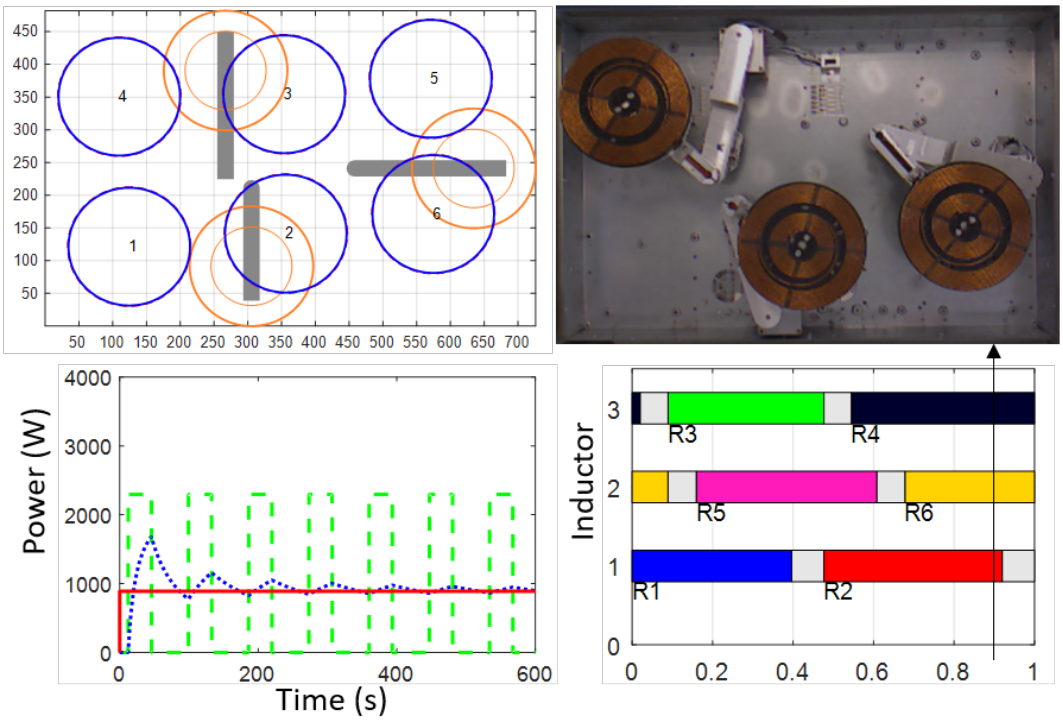


Figure 8.8: Experiment 2.

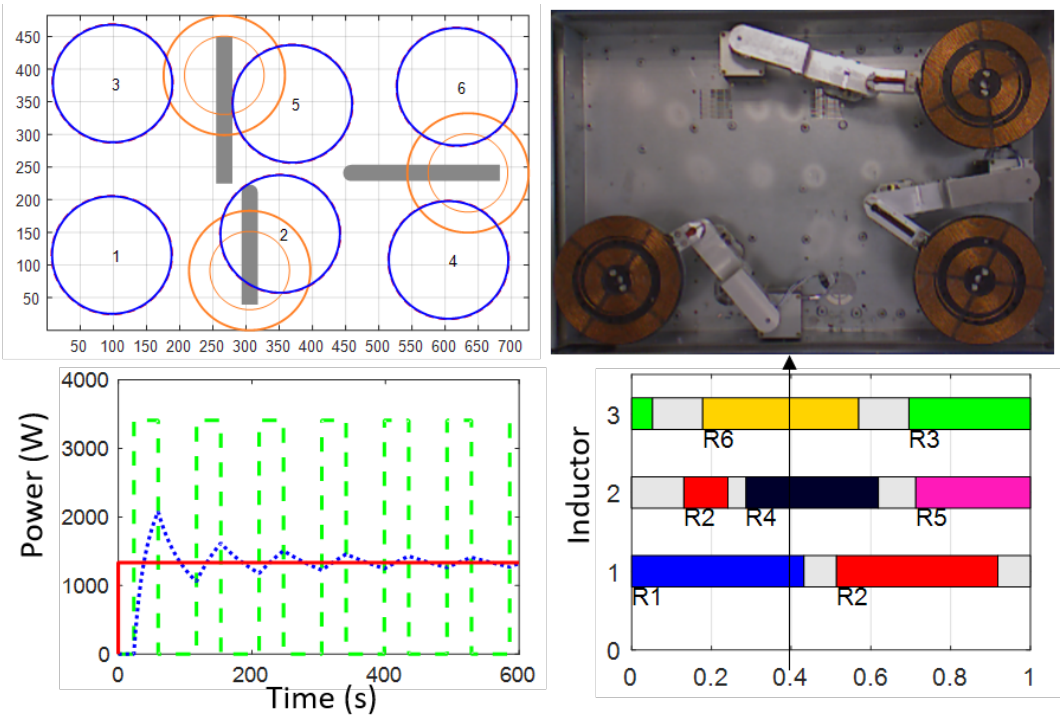


Figure 8.9: Experiment 3.

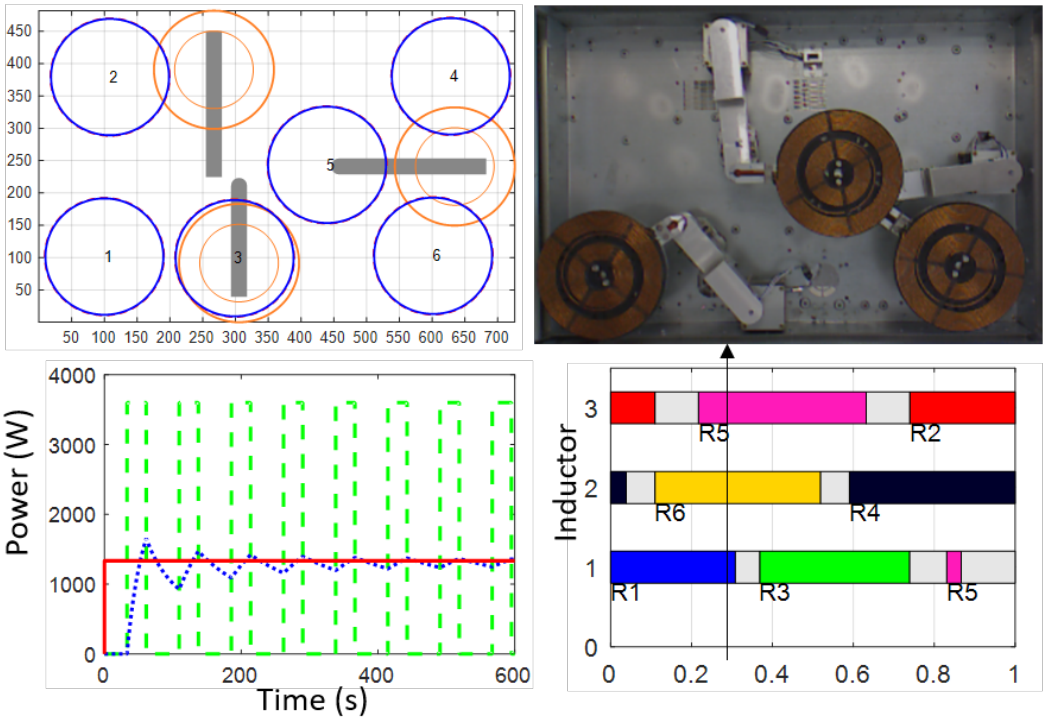


Figure 8.10: Experiment 4.

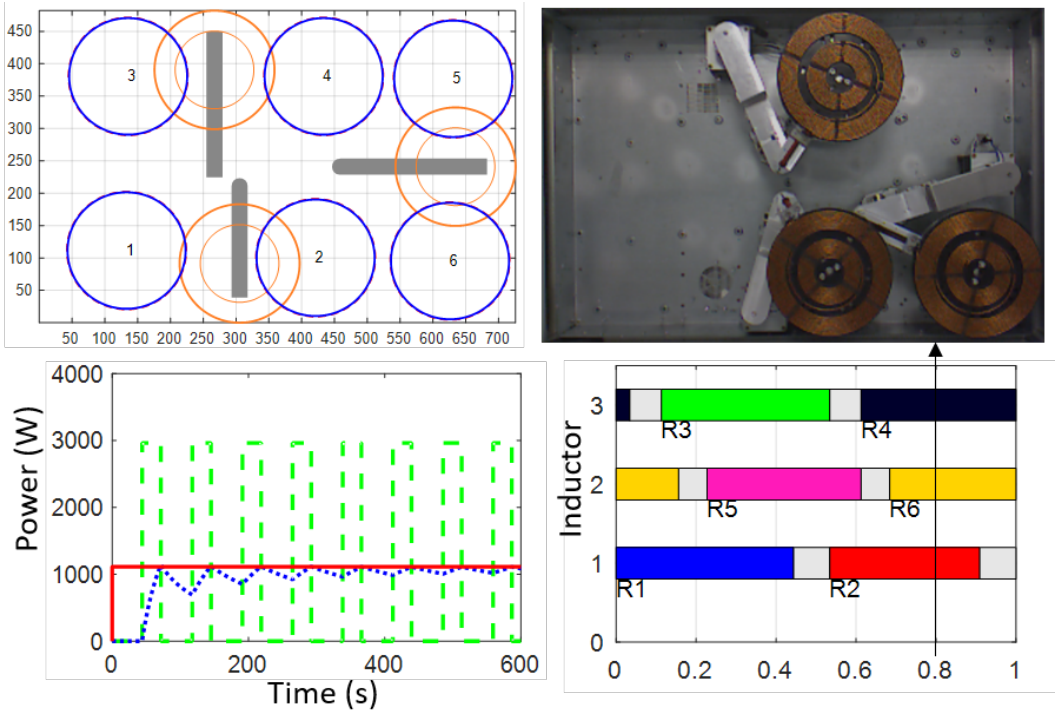


Figure 8.11: Experiment 5.

Chapter 9

Conclusions

Persistent coverage with a team of mobile robots has been proven to be NP-hard and many years of research have demonstrated that it is a difficult problem. In this thesis we have addressed those challenges with the intention of developing solutions which can be used for real life applications outside the laboratory while at the same time providing as many optimality guarantees as possible. We have tackled the problem for continuous environments and we have proposed solutions for increasingly more complicated settings, from a centralized solution in a convex environment to a distributed solution in a complex, non-convex environment, all of them with variable decay and importance of the coverage. Prior to this thesis, these issues were untackled. We have also worked in general discrete environments and focused in a particularly appealing application of persistent coverage: domestic induction heating, that is of interest for the home appliances industry.

In the beginning of this thesis, we have concentrated on the simplest version of the problem: a convex environment covered with a centralized system. We have demonstrated that a finite-horizon globally-optimal solution can be found for the whole team. Our proposed solution is based on a cost function that quantifies the quality of the coverage provided by the team in a finite horizon. Minimizing this function we obtain the paths and actions that the robots have to execute to optimize the coverage. We find them by transforming the problem in a discrete optimization problem with restriction solved with a branch-and-bound algorithm. This solution proves that the optimal can be found although it has the disadvantages of being centralized and computationally expensive.

Centralized solutions have robustness problems to failures of the central node and require really intense communications. For these reasons we have devoted our efforts to distributed solutions, which are robust, efficient and scalable. However, the first step to cover an environment distributedly is to have an estimate of the global coverage. To this end, we have introduced a distributed estimation algorithm based on max-consensus that allows each robot to have perfect information of the coverage in its surroundings and with a bounded error in the whole environment. This algorithm provides the robots enough accurate information to reach distributedly the same solution that a centralized gradient-based method. In addition, it has also proven to be communication-effective since each robot is capable of deciding which information is needed by its neighbors.

Using this estimation strategy, we have developed two different methods to define the

motion of the robots. The idea in both of them is to lead the robots to the points in which they can improve the coverage the most, that we call coverage goals. To quantify this, we have presented the improvement function, that is key to optimize the coverage. The first method includes a motion controller that combines an action based on the gradient of the improvement and an action to direct the robots to their goals. This method performs very well in convex environments, even with a fixed coverage action, although it does not provide any kind of optimality guarantees. The second method is based on planning the optimal coverage paths to a set of possible goals using an FMM and selecting the one that maximizes the overall improvement. Apart from giving an optimal path, this method is suitable for environment with obstacles and is capable of handling dynamic obstacles.

To deal with the most complex environments, we use a very common technique in coverage that consist in dividing the environment in as many regions as robots and assigning each area to one of them. Nevertheless, we go a step further and divide a complex, non-convex environment in equitable partitions based on the coverage that each robot is capable of providing, not based purely on geometry. Each robot is responsible of its own region and we have proposed a method to find the optimal coverage path through a graph of sweep-like paths that cover the entire partition. The whole approach takes into account the energy and the status of the robots, the shape of the environment and allows a very efficient path planning with existing methods.

In the final part of the thesis we have devoted our efforts to discrete environments where only a finite set of points has to be covered by the team of robots. In these environments, when there are more points than available robots, it is not possible to maintain the coverage level of all the point at their desired level at all times. For this reason, we have developed a periodic formulation with which we can guarantee that all the points receive their required coverage periodically or, equivalently, on average. Finding the periodic solution is still NP-hard and to counteract this complexity we use a divide-and-conquer approach and separate the problem into three smaller subproblem which can be solved optimally. For the first subproblem, that is planning closed paths for the robots, we use a state-of-the-art TSP algorithm. For the second subproblem, that is calculating the optimal times and coverage action that each robot has to spend and provide at each point, we solve a QCLP that guarantees the required coverage periodically. For the third subproblem, that avoids collisions scheduling a team plan with the individual plans of the robots, we solve a MILP that additionally minimizes the simultaneous motion of the robots. The entire solution is not only conceptually simple but also piece-wise optimal and computationally affordable for relatively big teams and environments. In fact, this is the first attempt to calculate simultaneously the paths and the coverage times and actions, that allows the collaboration of several robots at the same point and guarantees periodic coverage and collision avoidance.

This periodic solution has been applied to the novel problem of domestic cooking with mobile inductors. We have particularized the solution to heat a finite number of pots with some mobile inductor and to do so we have proposed a static assignment, when the user places on the hob less pots than inductors, a method to deal with pots boiling water, and a new collision avoidance method. This method is no longer overprotective and allows the inductor to make the most of the already cluttered space inside the hob. We have proven through real

experiments the great performance of our solution for a really innovative industrial problem with an enormous potential impact in the experience of domestic cooking.

Open Problems and Future Work

We believe that our solutions to the different setting in which persistent coverage can be applied provide a solid base for future developments in the field. Some interesting contributions that remain open are the following.

More elaborated coverage evolution models can be defined for individual applications. In some cases, such as lawn growing, or exponential model or even a simpler linear one may be enough. In this sense, it would be interesting to validate these ideas. Nevertheless, other phenomena that cause the need of persistence may not be well modeled with them and, therefore, new models would certainly provide more accuracy to the coverage solutions. For instance, one that takes into account the influence of each point on its neighbors as in heat transfer models. The main challenge with this phenomenon is that it introduces spacial dependencies that in the end become temporal dependencies of the coverage of each point and, as analyzed during this thesis, this is the most difficult issue in persistent coverage.

An alternative to modeling the evolution of the coverage is to execute persistent coverage based on sensor measurements. This variant can be probably addressed adopting some ideas from the exploration problem, where the uncertainty in the knowledge of the environment or, equivalently, on its coverage, is a key aspect.

Communication constraints and the possible loss of information are a current research topic in multi-robot systems. Since they can affect the performance of persistent coverage solutions, it would be interesting to quantify how and how much. The same happens with errors in the localization of the robots, which affect the actual coverage of the environment.

Another open problem that has the potential to outperform current solutions is planning infinite horizon paths that allow redundancy. This is the way to find an optimal solution for infinite time for environments where the decay and the importance are not constant.

A formal taxonomy of solutions and a framework to compare the multiple persistent coverage algorithms proposed so far in the literature and the ones to come would definitely gather the efforts of all the researchers that focus on the problem and provide an overview and the insights of the state of the art.

The last topic that is worth mentioning is the combination of persistent tasks with other tasks of higher instantaneous priority and high occurrence periods. An example is persistent coverage with target detection and tracking for surveillance and patrolling applications, such as UAVs and policemen protecting a city.

Bibliography

[1] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot task allocation: A review of the state-of-the-art," *Studies in Computational Intelligence*, vol. 604, pp. 31–51, 2015.

[2] F. Bullo, J. Cortés, and S. Martinez, *Distributed control of robotic networks: a mathematical approach to motion coordination algorithms*. Princeton University Press, 2009.

[3] R. Aragues, C. Sagues, and Y. Mezouar, *Parallel and distributed map merging and localization: algorithms, tools and strategies for robotic networks*. Springer, 2015.

[4] T. Balch and R. Arkin, "Behavior-based formation control for multirobot teams," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 926–939, 1998.

[5] J. Alonso-Mora, S. Baker, and D. Rus, "Multi-robot formation control and object transport in dynamic environments via constrained optimization," *The International Journal of Robotics Research*, vol. 36, no. 9, pp. 1000–1021, 2017.

[6] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Trans. Robot. Autom.*, vol. 20, no. 2, pp. 243–255, 2004.

[7] A. Okabe and A. Suzuki, "Locational optimization problems solved through voronoi diagrams," *European Journal of Operational Research*, vol. 98, no. 3, pp. 445–456, 1997.

[8] R. Mulligan and H. M. Ammari, "Coverage in Wireless Sensor Networks: A Survey," *Network Protocols and Algorithms*, vol. 2, no. 2, pp. 27–53, jun 2010.

[9] J. Urrutia, "Art gallery and illumination problems," *Handbook of computational geometry*, vol. 1, no. 1, pp. 973–1027, 2000.

[10] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, "Coordinated multi-robot exploration," *Robotics, IEEE Transactions on*, vol. 21, no. 3, pp. 376–386, 2005.

[11] E. M. Arkin, S. P. Fekete, and J. S. Mitchell, "Approximation algorithms for lawn mowing and milling," *Computational Geometry*, vol. 17, no. 1, pp. 25–50, 2000.

[12] X. Sun, C. G. Cassandras, and X. Meng, "A submodularity-based approach for multi-agent optimal coverage problems," *arXiv preprint arXiv:1708.04201*, 2017.

[13] H. F. Parapari, F. Abdollahi, and M. B. Menhaj, "Distributed coverage control for mobile robots with limited-range sector sensors," in *IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, 2016, pp. 1079–1084. [Online]. Available: http://ieeexplore.ieee.org/document/7576913/

[14] A. Gusrialdi, S. Hirche, D. Asikin, T. Hatanaka, and M. Fujita, "Voronoi-based coverage control with anisotropic sensors and experimental case study," *Intelligent Service Robotics*, vol. 2, no. 4, pp. 195–204, aug 2009.

[15] C. H. Caicedo-Nunez and M. Zefran, "A coverage algorithm for a class of non-convex regions," in *IEEE Conference on Decision and Control*, 2008, pp. 4244–4249.

[16] L. Pimenta, V. Kumar, R. Mesquita, and G. Pereira, "Sensing and coverage for a network of heterogeneous robots," in *IEEE Conference on Decision and Control*, 2008, pp. 3947–3952.

[17] M. Schwager, D. Rus, and J.-J. Slotine, "Decentralized, Adaptive Coverage Control for Networked Robots," *The International Journal of Robotics Research*, vol. 28, no. 3, pp. 357–375, Mar. 2009.

[18] A. Kwok and S. Martínez, "Deployment algorithms for a power-constrained mobile sensor network," *International Journal of Robust and Nonlinear Control*, vol. 20, no. 7, pp. 745–763, may 2010.

[19] Y. Kantaros, M. Thanou, and A. Tzes, "Distributed coverage control for concave areas by a heterogeneous RobotSwarm with visibility sensing constraints," *Automatica*, vol. 53, pp. 195–207, mar 2015.

[20] J. W. Durham, R. Carli, P. Frasca, and F. Bullo, "Discrete partitioning and coverage control for gossiping robots," *IEEE Trans. on Robotics*, vol. 28, no. 2, pp. 364–378, April 2012.

[21] S.-k. Yun and D. Rus, "Distributed coverage with mobile robots on a graph: locational optimization and equal-mass partitioning," *Robotica*, vol. 32, no. 02, pp. 257–277, mar 2014.

[22] A. Y. Yazicioglu, M. Egerstedt, and J. S. Shamma, "Communication-Free Distributed Coverage for Networked Systems," p. 1, 2016.

[23] R. Z. Farahani, N. Asgari, N. Heidari, M. Hosseininia, and M. Goh, "Covering problems in facility location: A review," *Computers & Industrial Engineering*, vol. 62, no. 1, pp. 368–407, feb 2012.

[24] H. W. Hamacher and Z. Drezner, *Facility location: applications and theory*. Springer, 2002.

[25] Y. Diaz-Mercado, S. G. Lee, and M. Egerstedt, "Distributed dynamic density coverage for human-swarm interactions," in *Amer. Control Conf.*, 2015, pp. 353–358.

[26] J. Derenick, N. Michael, and V. Kumar, "Energy-aware coverage control with docking for robot teams," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, sep 2011, pp. 3667–3672.

[27] H. Choset, "Coverage for robotics A survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1/4, pp. 113–126, 2001.

[28] P. F. Hokayem, D. Stipanovic, and M. W. Spong, "Dynamic Coverage Control with Limited Communication," in *2007 American Control Conference*. IEEE, 2007, pp. 4878–4883.

[29] N. Agmon, N. Hazon, and G. A. Kaminka, "Constructing spanning trees for efficient multi-robot coverage," in *IEEE International Conference on Robotics and Automation*, 2006, pp. 1698–1703.

[30] Y. Guo and M. Balakrishnan, "Complete coverage control for nonholonomic mobile robots in dynamic environments," in *IEEE International Conference on Robotics and Automation*, 2006, pp. 1704–1709.

[31] I. I. Hussein and D. M. Stipanovic, "Effective Coverage Control for Mobile Sensor Networks With Guaranteed Collision Avoidance," *IEEE Trans. on Control Systems Technology*, vol. 15, no. 4, pp. 642–657, 2007.

[32] D. Panagou, D. M. Stipanovic, and P. G. Voulgaris, "Dynamic Coverage Control in Unicycle Multi-Robot Networks Under Anisotropic Sensing," *Frontiers in Robotics and AI*, vol. 2, p. 3, 2015.

[33] G. M. Atnç, D. M. Stipanović, and P. G. Voulgaris, "Supervised coverage control of multi-agent systems," *Automatica*, vol. 50, no. 11, pp. 2936–2942, 2014.

[34] W. Burgard, M. Moors, C. Stachniss, and F. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 376–386, 2005.

[35] A. Ahmadzadeh, A. Jadbabaie, V. Kumar, and G. Pappas, "Cooperative Coverage using Receding Horizon Control," in *European Control Conference*, 2007, pp. 2466–2470.

[36] D. Mackenzie and T. Balch, "Making a clean sweep: Behavior based vacuuming," in *Proceedings of the AAAI Fall Symposium: Instantiating Real-world Agents*, 1993, pp. 93–98.

[37] H. Sahin and L. Guvenc, "Household robotics: autonomous devices for vacuuming and lawn mowing," *IEEE Control Syst. Mag.*, vol. 27, no. 2, pp. 20–96, 2007.

[38] N. M. Kakalis and Y. Ventikos, "Robotic swarm concept for efficient oil spill confrontation," *J. Hazardous Materials*, vol. 154, no. 1, pp. 880–887, 2008.

[39] M. Jadaliha and J. Choi, "Environmental monitoring using autonomous aquatic robots: Sampling algorithms and experiments," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 3, pp. 899–905, 2013.

[40] R. N. Smith, M. Schwager, S. L. Smith, B. H. Jones, D. Rus, and G. S. Sukhatme, "Persistent ocean monitoring with underwater gliders: Adapting sampling resolution," *J. Field Robotics*, vol. 28, no. 5, pp. 714–741, 2011.

[41] D. A. Paley, F. Zhang, and N. E. Leonard, "Cooperative control for ocean sampling: The glider coordinated control system," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 4, pp. 735–744, 2008.

[42] N. Nigam, "The Multiple Unmanned Air Vehicle Persistent Surveillance Problem: A Review," *Machines*, vol. 2, no. 1, pp. 13–72, 2014.

[43] A. Al-Wazzan, R. Al-Farhan, F. Al-Ali, and M. El-Abd, "Tour-guide robot," in *Int. Conf. Ind. Informatics Comput. Syst.*, 2016, pp. 1–5.

[44] J. Messias, R. Ventura, P. Lima, J. Sequeira, P. Alvito, C. Marques, and P. Carri, "A robotic platform for edutainment activities in a pediatric hospital," in *Int. Conf. Auton. Robot Syst. Compet.*, 2014, pp. 193–198.

[45] C. Song, L. Liu, G. Feng, Y. Wang, and Q. Gao, "Persistent awareness coverage control for mobile sensor networks," *Automatica*, vol. 49, no. 6, pp. 1867–1873, 2013.

[46] D. E. Soltero, M. Schwager, and D. Rus, "Decentralized path planning for coverage tasks using gradient descent adaptive control," *International Journal of Robotics Research*, vol. 33, no. 3, pp. 401–425, 2014.

[47] N. Mathew, S. L. Smith, and S. L. Waslander, "Planning Paths for Package Delivery in Heterogeneous Multirobot Teams," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 4, pp. 1298–1308, oct 2015.

[48] X. Zheng, S. Koenig, D. Kempe, and S. Jain, "Multirobot forest coverage for weighted and unweighted terrain," *IEEE Trans. Robot.*, vol. 26, no. 6, pp. 1018–1031, 2010.

[49] J. R. Peters, S. J. Wang, and F. Bullo, "Coverage control with anytime updates for persistent surveillance missions," in *American Control Conference (ACC)*, may 2017, pp. 265–270.

[50] F. Pasqualetti, J. W. Durham, and F. Bullo, "Cooperative Patrolling via Weighted Tours: Performance Analysis and Distributed Algorithms," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1181–1188, 2012.

[51] Y. Elmaliach, N. Agmon, and G. A. Kaminka, "Multi-robot area patrol under frequency constraints," *Annals of Mathematics and Artificial Intelligence*, vol. 57, no. 3-4, pp. 293–320, 2009.

[52] J. L. Fargeas, B. Hyun, P. Kabamba, and A. Girard, "Persistent visitation under revisit constraints," in *International Conference on Unmanned Aircraft Systems*, 2013, pp. 952–957.

[53] I. Maza and A. Ollero, "Multiple UAV cooperative searching operation using polygon area decomposition and efficient coverage algorithms," in *Distributed Autonomous Robotic Systems 6*. Springer Japan, 2007, pp. 221–230.

[54] A. Barrientos, J. Colorado, J. del Cerro, A. Martinez, C. Rossi, D. Sanz, and J. Valente, "Aerial remote sensing in agriculture: A practical approach to area coverage and path planning for fleets of mini aerial robots," *Journal of Field Robotics*, vol. 28, no. 5, pp. 667–689, 2011.

[55] X. Lan and M. Schwager, "Planning periodic persistent monitoring trajectories for sensing robots in gaussian random fields," in *IEEE Int. Conf. Robot. Autom.*, 2013, pp. 2407–2412.

[56] A. Breitenmoser, M. Schwager, J.-C. Metzger, R. Siegwart, and D. Rus, "Voronoi coverage of non-convex environments with a group of networked robots," in *IEEE Int. Conf. on Robotics and Automation*, may 2010, pp. 4982–4989.

[57] C. Franco, G. Lopez-Nicolas, C. Sagues, and S. Llorente, "Persistent coverage control with variable coverage action in multi-robot environment," in *IEEE Conf. Decision and Control*, 2013, pp. 6055–6060.

[58] C. G. Cassandras, X. Lin, and X. Ding, "An optimal control approach to the multi-agent persistent monitoring problem," *IEEE Transactions on Automatic Control*, vol. 58, no. 4, pp. 947–961, 2013.

[59] C. Franco, G. López-Nicolás, C. Sagüés, and S. Llorente, "Adaptive action for multi-agent persistent coverage," *Asian J. Control*, vol. 18, no. 2, pp. 419–432, 2016.

[60] N. A. Pantazis, S. A. Nikolidakis, and D. D. Vergados, "Energy-Efficient Routing Protocols in Wireless Sensor Networks: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 2, pp. 551–591, 2013.

[61] K. M. Lynch, I. B. Schwartz, P. Yang, and R. A. Freeman, "Decentralized environmental modeling by mobile sensor networks," *IEEE Transactions on Robotics*, vol. 24, no. 3, pp. 710–724, 2008.

[62] R. A. Freeman, P. Yang, and K. M. Lynch, "Stability and convergence properties of dynamic average consensus estimators," in *IEEE International Conference on Decision and Control*, 2006, pp. 398–403.

[63] S. Martínez, "Distributed interpolation schemes for field estimation by mobile sensor networks," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 491–500, 2010.

[64] S. L. Smith, M. Schwager, and D. Rus, "Persistent robotic tasks: Monitoring and sweeping in changing environments," *IEEE Trans. Robot.*, vol. 28, no. 2, pp. 410–426, 2012.

[65] J. Yu, J. Aslam, S. Karaman, and D. Rus, "Anytime planning of optimal schedules for a mobile sensing robot," in *IEEE Int. Conf. Intell. Robots Syst.*, 2015, pp. 5279–5286.

[66] X. Yu, S. B. Andersson, N. Zhou, and C. G. Cassandras, "Optimal dwell times for persistent monitoring of a finite set of targets," in *American Control Conference (ACC)*, 2017, pp. 5544–5549.

[67] P. Amstutz, N. Correll, and A. Martinoli, "Distributed boundary coverage with a team of networked miniature robots using a robust market-based algorithm," *Ann. Math. Artificial Intell.*, vol. 52, no. 2-4, pp. 307–333, 2008.

[68] M. Jager and B. Nebel, "Dynamic decentralized area partitioning for cooperating cleaning robots," in *IEEE International Conference on Robotics and Automation*, vol. 4, 2002, pp. 3577–3582.

[69] A. Pierson, L. C. Figueiredo, L. C. Pimenta, and M. Schwager, "Adapting to sensing and actuation variations in multi-robot coverage," *The International Journal of Robotics Research*, vol. 36, no. 3, pp. 337–354, mar 2017.

[70] S. Moon and E. W. Frew, "Distributed cooperative control for joint optimization of sensor coverage and target tracking," in *International Conference on Unmanned Aircraft Systems (ICUAS)*, 2017, pp. 759–766.

[71] K. Hungerford, P. Dasgupta, and K. R. Guruprasad, "A Repartitioning Algorithm to Guarantee Complete, Non-overlapping Planar Coverage with Multiple Robots," in *Distributed Autonomous Robotic Systems*, 2016, pp. 33–48.

[72] J. F. Araujo, P. B. Sujit, and J. B. Sousa, "Multiple UAV area decomposition and coverage," in *IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, apr 2013, pp. 30–37.

[73] M. Pavone, A. Arsie, E. Frazzoli, and F. Bullo, "Distributed Algorithms for Environment Partitioning in Mobile Robotic Networks," *IEEE Transactions on Automatic Control*, vol. 56, no. 8, pp. 1834–1848, aug 2011.

[74] B. Boardman, T. Harden, and S. Martinez, "Spatial load balancing in non-convex environments using sampling-based motion planners," in *American Control Conference*, 2016, pp. 5703–5708.

[75] A. C. Kapoutsis, S. A. Chatzichristofis, and E. B. Kosmatopoulos, "DARP: Divide Areas Algorithm for Optimal Multi-Robot Coverage Path Planning," *Journal of Intelligent & Robotic Systems*, pp. 1–18, jan 2017.

[76] V. Sea, C. Kato, and T. Sugawara, "Coordinated Area Partitioning Method by Autonomous Agents for Continuous Cooperative Tasks," *Journal of Information Processing*, vol. 25, no. 0, pp. 75–87, 2017.

[77] P. F. Hokayem, D. Stipanovic, and M. W. Spong, "On persistent coverage control," in *IEEE Conf. Decision and Control*, 2007, pp. 6130–6135.

[78] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[79] S. Bochkarev and S. L. Smith, "On minimizing turns in robot coverage path planning," in *IEEE Conf. Autom. Sci. Eng.*, 2016, pp. 1237–1242.

[80] Y. Gabriely and E. Rimon, "Spanning-tree based coverage of continuous areas by a mobile robot," *Ann. Math. Artificial Intell.*, vol. 31, no. 1/4, pp. 77–98, 2001.

[81] N. Hazon and G. Kaminka, "Redundancy, Efficiency and Robustness in Multi-Robot Coverage," in *IEEE International Conference on Robotics and Automation*, 2005, pp. 735–741.

[82] D. Portugal, C. Pippin, R. P. Rocha, and H. Christensen, "Finding optimal routes for multi-robot patrolling in generic graphs," in *IEEE/RSJ Int. Conf. Intell. Robots and Systems*, 2014, pp. 363–369.

[83] S. Alamdari, E. Fata, and S. L. Smith, "Persistent monitoring in discrete environments: Minimizing the maximum weighted latency between observations," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 138–154, 2013.

[84] X. Lin and C. G. Cassandras, "An optimal control approach to the multi-agent persistent monitoring problem in two-dimensional spaces," in *IEEE Conf. Decision and Control*, 2013, pp. 6886–6891.

[85] X. Lan and M. Schwager, "Rapidly Exploring Random Cycles: Persistent Estimation of Spatiotemporal Fields With Multiple Sensing Robots," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1230–1244, 2016.

[86] J. Yu, M. Schwager, and D. Rus, "Correlated Orienteering Problem and its Application to Persistent Monitoring Tasks," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1106–1118, 2016.

[87] D. Mitchell, M. Corah, N. Chakraborty, K. Sycara, and N. Michael, "Multi-robot long-term persistent coverage with fuel constrained robots," in *IEEE Int. Conf. Robot. Autom.*, 2015, pp. 1093–1099.

[88] R. Graham and J. Cortés, "Adaptive information collection by robotic sensor networks for spatial estimation," *IEEE Transactions on Automatic Control*, vol. 57, no. 6, pp. 1404–1419, 2012.

[89] N. Nigam, S. Bieniawski, I. Kroo, and J. Vian, "Control of multiple UAVs for persistent surveillance: algorithm and flight test results," *IEEE Trans. Control Syst. Technol.*, vol. 20, no. 5, pp. 1236–1251, 2012.

[90] N. Hubel, S. Hirche, A. Gusrialdi, T. Hatanaka, M. Fujita, and O. Sawodny, "Coverage control with information decay in dynamic environments," in *17th IFAC World Congress*, 2008, pp. 4180–4185.

[91] W. B. D. Fox and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Trans. Robot. Autom.*, vol. 4, p. 1, 1997.

[92] R. Ventura and A. Ahmad, "Towards Optimal Robot Navigation in Domestic Spaces," in *RoboCup 2014: Robot World Cup XVIII*. Springer, 2015, pp. 318–331.

[93] H. Fukushima, K. Kon, and F. Matsuno, "Model predictive formation control using branch-and-bound compatible with collision avoidance problems," *IEEE Trans. on Robotics*, vol. 29, no. 5, pp. 1308–1317, 2013.

[94] X. Wang, M. Kloetzer, C. Mahulea, and M. Silva, "Collision avoidance of mobile robots by using initial time delays," in *IEEE Conf. Dec. Control*, 2015, pp. 324–329.

[95] A. Richards and J. P. How, "Aircraft trajectory planning with collision avoidance using mixed integer linear programming," in *American Control Conf.*, vol. 3, 2002, pp. 1936–1941.

[96] M. Turpin, K. Mohta, N. Michael, and V. Kumar, "Goal assignment and trajectory planning for large teams of interchangeable robots," *Autonomous Robots*, vol. 37, no. 4, pp. 401–415, 2014.

[97] D. Panagou, M. Turpin, and V. Kumar, "Decentralized goal assignment and trajectory generation in multi-robot networks: A multiple Lyapunov functions approach," in *IEEE International Conference on Robotics and Automation*, 2014, pp. 6757–6762.

[98] C. Franco, D. M. Stipanović, G. López-Nicolás, C. Sagüés, and S. Llorente, "Persistent coverage control for a team of agents with collision avoidance," *European J. Control*, vol. 22, pp. 30–45, 2015.

[99] Y. Mei, Y. Lu, Y. Hu, and C. Lee, "Deployment of mobile robots with energy and timing constraints," *IEEE Transactions on Robotics*, vol. 22, no. 3, pp. 507–522, 2006.

[100] K. Sundar and S. Rathinam, "Algorithms for Routing an Unmanned Aerial Vehicle in the Presence of Refueling Depots," *IEEE Trans. on Automation Science and Engineering*, vol. 11, no. 1, pp. 287–294, 2014.

[101] J. Kim, B. D. Song, and J. R. Morrison, "On the scheduling of systems of UAVs and fuel service stations for long-term mission fulfillment," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 70, no. 1-4, pp. 347–359, 2013.

[102] B. D. Song, J. Kim, J. Kim, H. Park, J. R. Morrison, and D. H. Shim, "Persistent UAV service: An improved scheduling formulation and prototypes of system components," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 74, no. 1-2, pp. 221–232, 2014.

[103] D. Mitchell, N. Chakraborty, K. Sycara, and N. Michael, "Multi-Robot Persistent Coverage with stochastic task costs," in *IEEE/RSJ Int. Conf. Intell. Robots Systems*. IEEE, 2015, pp. 3401–3406.

[104] N. Mathew, S. L. Smith, and S. L. Waslander, "Multirobot Rendezvous Planning for Recharging in Persistent Tasks," *IEEE Transactions on Robotics*, vol. 31, no. 1, pp. 128–142, feb 2015.

[105] K. Leahy, D. Zhou, C.-I. Vasile, K. Oikonomopoulos, M. Schwager, and C. Belta, "Persistent surveillance for unmanned aerial vehicles subject to charging and temporal logic constraints," *Autonomous Robots*, vol. 40, no. 8, pp. 1363–1378, dec 2016.

[106] G. Hollinger and S. Singh, "Multi-robot coordination with periodic connectivity," in *International Conference on Robotics and Automation*, 2010, pp. 4457–4462.

[107] M. M. Zavlanos, "Synchronous rendezvous of very-low-range wireless agents," in *International Conference on Decision and Control*, 2010, pp. 4740–4745.

[108] M. Zavlanos and G. Pappas, "Distributed Connectivity Control of Mobile Networks," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1416–1428, 2008.

[109] G. De La Torre, T. Yucelen, and E. N. Johnson, "Hybrid protocols for maintaining connectivity of multiagent systems," in *IEEE Conference on Decision and Control*, 2013, pp. 805–810.

[110] J. Wang and S. Medidi, "Energy Efficient Coverage with Variable Sensing Radii in Wireless Sensor Networks," in *IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, 2007, pp. 61–61.

[111] S. Davari, M. H. F. Zarandi, A. Hemmati, and I. B. Turksen, "The variable radius covering problem with fuzzy travel times," in *International Conference on Fuzzy Systems*, 2010, pp. 1–6.

[112] A. D. Vicente, "Power control of multiple inverters applied to comestic induction heating," Ph.D. dissertation, Escuela de Ingeniería y Arquitectura, University of Zaragoza, 2017.

[113] M. Carmona, "Implementación y control de múltiples manipuladores en cocina de inducción," Ph.D. dissertation, Escuela de Ingeniería y Arquitectura, University of Zaragoza, 2016.

[114] F. S. Serrano, "Electromagnetic heating with uniform thermal distribution in flexible induction appliances," Ph.D. dissertation, Escuela de Ingeniería y Arquitectura, University of Zaragoza, 2017.

[115] A. Pérez, "Sistema basado en visión artificial y sensor de rango para control y reconocimiento en encimeras de inducción," Master's thesis, Escuela de Ingeniería y Arquitectura, University of Zaragoza, 2012.

[116] G. Arto, "Prototipo para detectar la posición de recipientes mediante sensores capacitivos en cocinas de inducción," Master's thesis, Escuela de Ingeniería y Arquitectura, University of Zaragoza, 2015.

[117] J. M. Palacios, C. Sagüés, E. Montijano, and S. Llorente, "Human-Computer Interaction Based on Hand Gestures Using RGB-D Sensors," *Sensors*, vol. 13, no. 9, pp. 11 842–11 860, 2013.

[118] J. M. Palacios-Gasos, E. Montijano, C. Sagüés, and S. Llorente, "Multi-robot persistent coverage using branch and bound," in *American Control Conf.*, 2016, pp. 5697–5702.

[119] J. M. Palacios-Gasós, E. Montijano, C. Sagüés, and S. Llorente, "Distributed coverage estimation for multi-robot persistent tasks," in *European Control Conference*, 2015, pp. 3681–3686.

[120] ——, "Distributed coverage estimation and control for multi-robot persistent tasks," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1444–1460, 2016.

[121] J. M. Palacios-Gasós, Z. Talebpour, E. Montijano, C. Sagüés, and A. Martinoli, "Optimal path planning and coverage control for multi-robot persistent coverage in environments with obstacles," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 1321–1327.

[122] J. M. Palacios-Gasós, E. Montijano, and C. Sagüés, "Equitable persistent coverage of non-convex environments with graph-based optimal planning," *International Journal of Robotics Research*, 2017, submission Pending.

[123] J. M. Palacios-Gasós, E. Montijano, C. Sagüés, and S. Llorente, "Multi-robot persistent coverage with optimal times," in *Conf. Decision and Control*, 2016, pp. 3511–3517.

[124] ——, "Multi-agent periodic persistent coverage with collision avoidance," *IEEE Transactions on Control Systems Technology*, 2017, under review.

[125] A. Inogés, S. Llorente, G. López, E. Montijano, J. Palacios, and C. Sagüés, "Domestic appliance comprising a gesture detection," European Patent EP3 187 785 (A1), 2015.

[126] S. Llorente, J. Palacios, and C. Sagüés, "Domestic appliance with gesture detection," Spanish Patent ES2 564 879 (B1), 2017.

[127] A. Inogés, S. Llorente, G. López, E. Montijano, J. Palacios, and C. Sagüés, "Domestic appliance," European Patent EP3 184 907 (A1), 2015.

[128] J. Galindo, S. Llorente, C. Obón, J. Palacios, E. Pérez, E. Ramírez, F. Sanz, and B. Villanueva, "Sistema de aparato de cocción," Spanish Patent ES2 603 780 (A1), 2015.

[129] D. P. Bertsekas, *Dynamic programming and optimal control*. Athena Scientific Belmont, MA, 1995, vol. 1, no. 2.

[130] A. H. Land and A. G. Doig, "An automatic method of solving discrete programming problems," *Econometrica: Journal of the Econometric Society*, pp. 497–520, 1960.

[131] K. J. Aström and R. M. Murray, *Feedback systems: an introduction for scientists and engineers*. Princeton university press, 2010.

[132] L. Sabattini, A. Gasparri, C. Secchi, and N. Chopra, "Enhanced connectivity maintenance for multi-robot systems," in *10th International IFAC Symposium on Robot Control*, 2012, pp. 319–324.

[133] N. A. Lynch, *Distributed algorithms*. Morgan Kaufmann, 1996.

[134] A. Jadbabaie, J. Lin, and A. Morse, "Coordination of Groups of Mobile Autonomous Agents using Nearest Neighbor Rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, 2003.

[135] L. Jaillet, A. Yershova, S. La Valle, and T. Simeon, "Adaptive tuning of the sampling domain for dynamic-domain RRTs," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 2851–2856.

[136] A. Solanas and M. Garcia, "Coordinated multi-robot exploration through unsupervised clustering of unknown space," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, vol. 1, 2004, pp. 717–721.

[137] N. Michael, M. Zavlanos, V. Kumar, and G. J. Pappas, "Distributed multi-robot task assignment and formation control," in *IEEE International Conference on Robotics and Automation*, 2008, pp. 128–133.

[138] J. A. Sethian, "Fast Marching Methods," *SIAM Review*, vol. 41, no. 2, pp. 199–235, 1999.

[139] F. Aurenhammer, "Power diagrams: properties, algorithms and applications," *SIAM Journal on Computing*, vol. 16, no. 1, pp. 78–96, 1987.

[140] T. H. Cormen, *Introduction to algorithms*. MIT press, 2009.

[141] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495–1512, 2013.

[142] K. Schüpbach and R. Zenklusen, "An adaptive routing approach for personal rapid transit," *Mathematical Methods of Operations Research*, vol. 77, no. 3, pp. 371–380, 2013.

[143] C. Prins, P. Lacomme, and C. Prodhon, "Order-first split-second methods for vehicle routing problems: A review," *Transportation Research Part C: Emerging Technologies*, vol. 40, pp. 179–200, 2014.

[144] N. Christofides, "Worst-case analysis of a new heuristic for the travelling salesman problem," Graduate School of Industrial Administration, Carnegie Mellon University, Tech. Rep., 1976.

[145] J. Tumova and D. V. Dimarogonas, "A receding horizon approach to multi-agent planning from local LTL specifications," in *2014 American Control Conf.*, 2014, pp. 1775–1780.

[146] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[147] I. Griva, S. G. Nash, and A. Sofer, *Linear and nonlinear optimization*. SIAM, 2009.

[148] "IBM-ILOG: CPLEX optimization studio." [Online]. Available: http://www.ilog.com/products/cplex