TESIS DE LA UNIVERSIDAD DE ZARAGOZA
2020
17

Yaroslav Marchukov Lepeha

Multi-robot deployment planning in communication-constrained environments

Departamento
Informática e Ingeniería de Sistemas

Director/es
Montano Gella, Luis

ISSN 2254-7606

Prensas de la Universidad
Universidad Zaragoza

ISSN 2254-7606

Universidad Zaragoza
1542

Tesis Doctoral

MULTI-ROBOT DEPLOYMENT PLANNING IN COMMUNICATION-CONSTRAINED ENVIRONMENTS

Autor

Yaroslav Marchukov Lepeha

Director/es

Montano Gella, Luis

UNIVERSIDAD DE ZARAGOZA

Informática e Ingeniería de Sistemas

2019

PhD Thesis

Multi-robot deployment planning in communication-constrained environments

Author

Yaroslav Marchukov

Supervisor

Luis Montano Gella

Grupo de Robótica, Percepción y Tiempo Real (RoPeRT)
Instituto de Investigación en Ingeniería de Aragón (I3A)
Departamento de Informática e Ingeniería de Sistemas
Escuela de Ingeniería y Arquitectura (EINA)
Universidad de Zaragoza

2019

Agradecimientos

En primer lugar me gustaría agradecer a mi director Luis Montano por darme la oportunidad de iniciarme en el mundo de la investigación. Por su guía a lo largo de todos estos años y por todas las veces que se ha quedado al pie de cañón los días de deadline.

También agradecer a todos los compañeros con los que he compartido laboratorio a lo largo de estos años, a los que siempre he podido acudir cuando he necesitado ayuda: Pablo, Carlos, Jesús, Marta, Mayte, Wajahat, Maite, Luis, Danilo, Alex.

Gracias a todos mis amigos por los viajes y por todas las quedadas los fines de semana. Sin duda, sin esos momentos, sin esas intensas sesiones de desestrés, me habría resultado mucho más difícil continuar con la tesis.

Y sobre todo me gustaría agradecer el apoyo de mi madre. Persona sin la que nunca hubiese conseguido realizar esta tesis, por todos sus consejos y por todas las charlas que me han motivado seguir adelante en los momentos más difíciles.

Project framework

The work of this thesis has been developed within the Robotics, Perception and Real Time Group of the Instituto de Investigación en Ingeniería de Aragón of Universidad de Zaragoza. This research was supported by Ministerio de Economía y Competitividad (MINECO) with "Ayudas para contratos predoctorales para la formación de doctores" under grant BES-2013-067405, as well as within the framework of two research projects:

- **TEams of robots for LOgistics, MAintenance and eNvironment monitoring (TELOMAN) DPI2012-32100**

  The project involves deployment and actuation techniques of a multi-robot team. It is necessary to address problems of task planning and allocation, coordinated execution of the navigation, perception of the environment from each of the team members and maintaining communication between all system components – robots, infrastructure, bridges, monitoring equipment, etc. This project addresses new goals and challenges for research and their application in real, large and complex scenarios.

- **Despliegue de Robots en Entornos Desafiantes (ROBOCHALLENGE) DPI2016-76676-R-AEI/FEDER-UE**

  This project addresses theoretical and experimental research in the field of intervention and exploration of challenging environments by means of aerial and ground robots. A challenging environment is one where, due to different reasons, current robotic techniques fail or do not work properly and continuously. This failure is due to the characteristics of the environment, such as lack of coverage GNSS (any environment confined), absence of visual or geometric discriminant characteristics (tunnels, pipes), dynamic environments (presence of workers or other vehicles), or large (any real environment). Experimental environments considered are confined and structured (tunnels, galleries), or unstructured (tunnels under construction, underground areas in buildings). Realistic demonstrators are developed to evaluate the applicability of the techniques developed in these environments. The adaptive integration of the information from the sensors allows the environment understanding for the robot localization and navigation during the whole mission.

Resumen

A lo largo de las últimas dos décadas hemos podido observar el aumento del uso de sistemas robóticos tanto en el entorno industrial como en nuestra vida cotidiana. Ello se debe a una mayor eficiencia de los robots frente a los humanos, para la realización de tareas que resultan tediosas, pesadas, repetitivas, etc. Claros ejemplos de ello son: los numerosos brazos robóticos utilizados en las plantas industriales, o los robots aspiradores que limpian nuestras casas. Con el paso de los años hemos visto como cada vez más y más, se le ha ido dotando de una mayor inteligencia a los robots para poder realizar las tareas de manera más autónoma. Con ello se pretende que en un futuro no se requiera una intervención humana para la realización de dichas tareas.

De este modo también podrían intervenir en tareas en las cuales es imposible o poco eficiente la intervención de los humanos, e incluso que implicase un cierto grado de riesgo para una persona. Por ejemplo, monitorización de entornos de difícil acceso, como podrían ser túneles, minas, etc. Éste es precisamente el tema en el que se ha enfocado el trabajo realizado durante esta tesis: la planificación del despliegue de un equipo de agentes para la monitorización de un entorno.

La misión de los agentes consiste en alcanzar unos puntos o localizaciones de interés y transmitirle la información observada a una estación base estática. Debido a la ausencia de una infraestructura de comunicaciones, una transmisión directa es imposible. Por lo tanto, los agentes se deben coordinar de manera autónoma, de modo que algunos de ellos alcancen los objetivos y otros realicen la función de repetidor para que sea posible el envío de la información.

Nos hemos centrado en dos líneas de investigación principales, relacionadas con dos maneras del envío de la información a la estación base. En el primer enfoque, los agentes deben mantener un enlace de comunicación con la base en el momento de alcanzar los objetivos. La finalidad de este enfoque es que, ya sea un humano u otro agente autónomo con mayores capacidades computacionales, pueda interactuar desde la estación base con un robot que ha alcanzado el objetivo. Para ello hemos desarrollado un método para el cálculo de las posiciones óptimas para los agentes que se dediquen a tareas de repetidor, en base a dos criterios: la distancia recorrida por los robots y la cantidad de agentes que se dediquen a tareas de repetidores. A continuación, hemos implementado un método de planificación de caminos de modo que los agentes pudiesen navegar el máximo tiempo posible dentro de zonas con señal. Al utilizar los dos métodos conjuntamente, el equipo de agentes extiende el área de cobertura de la estación base, de modo que es posible establecer un enlace de comunicación desde la base hasta los objetivos marcados.

Utilizando este último método, el equipo de agentes es capaz de lidiar con variaciones del entorno si la comunicación entre los agentes no se pierde. Sin embargo, los eventos tan comunes e irrelevantes para los seres humanos, como podría ser un simple cierre de una puerta, pueden llegar a ser críticos para el equipo de robots. Ya que ello podría provocar una interrupción de la comunicación entre el equipo. Debido a ello, hemos propuesto un método

distribuido para que los agentes sean capaces de reconectarse formando una cadena, hacia un objetivo, en escenarios donde haya variaciones con respecto al mapa inicial que los robots poseían.

En el segundo enfoque de la presente tesis, hemos planteado misiones de recopilación de datos de un entorno. Aquí la comunicación con la estación base, en el instante de alcanzar un objetivo, no es necesaria y a menudo imposible. No obstante, y con el fin de tener una información más "fresca", la entrega de los datos recopilados a la base debe realizarse en el menor tiempo posible. Este tipo de enfoques resulta útil en casos en los cuales los escenarios sean grandes y la cantidad de agentes disponibles no permita desplegar cadenas hacia todos los puntos deseados con comunicación.

En este tipo de escenarios, resulta más eficiente que algunos agentes se encarguen de trabajar, recopilando datos del entorno, y otros de reunir la información de los que trabajan para posteriormente retransmitirla a la estación base. A los primeros los llamamos trabajadores y a los segundos colectores. De este modo tan solo los colectores realizan largos viajes a la estación base, mientras que los trabajadores emplean la mayor parte de su tiempo exclusivamente a la recopilación de datos.

Puesto que para el intercambio de datos entre colectores y trabajadores se deberán encontrar en un punto del escenario, hemos desarrollado dos métodos para la planificación de caminos para la sincronización entre los agentes, tanto en un punto de encuentro estático como con los agentes en movimiento. El primer método se ha basado en una técnica de muestreo aleatorio del espacio, los denominados RRTs, con el fin de obtener una solución en el menor tiempo posible. Para el segundo, se ha utilizado de base el Fast Marching Method (FMM), el cuál a pesar de ser más lento, proporciona una solución óptima.

Finalmente, hemos propuesto una técnica global para la recopilación de datos, transmitiéndolos a la estación base de manera periódica. Este método consiste en: encontrar el mejor balance entre la cantidad de agentes trabajadores y colectores, la mejor división del escenario en áreas de trabajo de los agentes trabajadores, la asociación de los trabajadores a transmitir los datos recopilados a los colectores o bien directamente a la estación base, así como los caminos de los colectores de modo que el intervalo de espera de los trabajadores sea el menor posible. El método propuesto trata de encontrar la mejor solución con el fin de entregar la mayor cantidad de datos y que el tiempo de "refresco" de los mismos sea el mínimo posible.

Abstract

In the last two decades, we humans have seen an increase of the usage of robotic systems in both, the industrial environment as well as our daily life. The main reason of this growth, is the higher efficiency of the robots to perform tedious, heavy or repetitive tasks. Clear examples are: the robotic arms present in many industrial factories or autonomous vacuum cleaners at our homes. Over the years, we have observed the evolution of the robots which have been gradually endowed with more intelligence in order to perform different tasks autonomously. The aim is to avoid the human intervention in these tasks in the near future.

In this way, the robots would contribute in tasks where a human intervention might be impossible or inefficient, and even involve a degree of risk for a person. Examples of these kind of missions are monitoring in environments difficult to access, such as tunnels, mines, etc. This is the main topic of the work developed in the present thesis: the planning of the deployment of a team of agents for environment monitoring missions.

The mission of the agents consists in reaching some points or locations of interest and to transmit the information from there to a static base station. Due to the absence of a communication infrastructure, a direct communication with the base is impossible. Therefore, the agents must coordinate, in such a way that some of them will reach the goals and the others will be used in role of relay, in order to ensure a communication link from the goals to the base station via a multi-hop network.

In this work we have focused in two main research lines, directly related to the information transmission to the base station. The first approach considers that the agents must maintain a communication link with the base station at the moment of reaching the goals. The purpose of this approach is that a human or some autonomous agent, with greater computational capacities, could interact from the base station with the robot that has reached the goal. To this end, we have developed a method to compute the optimal positions where some agents of the team will be placed in order to act as relay, based on two different criteria: the total number of agents that are being used in role of relays and the travelled distance by the agents. Then, we implement a path planning method that the agents use to navigate as long as possible within areas with signal. The team of agents, jointly using both methods, extends the coverage area of the base station, so that it is possible to establish a communication link from the base to the goal locations.

With this approach, the team is able to cope with the possible variations in the environment, if the communication between the agents is not broken. However, common and irrelevant events for humans, such as a simple closure of a door, may become critical for the team of robots. It possibly will breakdown the communication between the team, dividing them into different groups. Thus, we have proposed a distributed method to reconnect the team of robots in a chain formation to reach some goal, in scenarios that are subject to change without notice, with respect to the initial map that has the team.

For the second approach of this thesis, we have proposed data gathering missions with a team of robots. Here, the communication with the base station is no necessary and generally

impossible, at the moment of reaching the goals. However, to preserve a "fresher" information, the robots must deliver the gathered data at the goals as fast as possible to the base. These kind of approaches are useful in cases of large scenarios and if the number of available robots is not large enough to extend a chain to all the desired locations of the scenario. So the base station periodically request data from some goal locations and the robots must upload the gathered data from these positions to the base.

In this kind of scenarios it is more efficient to devote some agents to work, gathering data from goal positions, and the others to collect the information of the working agents and deliver it to the base station. We call the first of them, as workers, and to the second ones, collectors. In this way, only the collectors are travelling large journeys to the base station, while workers devote much of their time and energy exclusively to gather data from the requested goal locations.

Since the collector and worker agents must meet each other somewhere in order to exchange data, we have developed two path planning methods in order to assure synchronization between the agents. Both approaches consider synchronization at some static point as well as information exchange with the agents in movement. The first one is based in one of the most famous random sampling techniques, the so-called Rapidly-exploring Random Trees (RRT), in order to provide a possible solution in the minimum time. The second one uses as a base method the Fast Marching Method (FMM), that despite of being slower, obtains optimal solutions.

Finally, we have proposed a global technique to gather data from a scenario, periodically uploading the information to the base station. The method: finds out the best balance between the number of agents used in collector and worker roles, splits the scenario into working areas for the worker agents, associates the workers to deliver the data to the collectors or directly to the base station, as well as the trajectories of the collectors in the way that the waiting to transmit of the workers will be as short as possible. The proposed combination of these techniques attempts to find out the best possible solution in terms of delivered information packages to the base station as well as to keep the "refreshing time" of these packages as small as possible.

Index

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

The usage of robots is becoming more frequent for multiple tasks in the everyday live. And it is even more notably in industrial environments, where the robots replace the humans, in order to perform repetitive, tedious or heavy jobs. The robots are programmed specifically to carry out these specific tasks. However, the last decade we observe how the researchers are giving higher intelligence to the robots, in order to allow them to operate more autonomously. A clear example are semi-autonomous warehouses where the robots perform the heaviest tasks [1]. The purpose is that the robots could perform all type of jobs without human supervision.

In this way, the robots would contribute in tasks where a human intervention might be impossible or inefficient, and even involve a degree of risk for a person. Examples of these kind of missions are monitoring in environments difficult to access, such as tunnels, mines, etc. Or, for instance the monitoring of a firework, where it would be a risk to send humans. This is the main topic of the work developed in the present thesis: the use of a team of robots for environment monitoring.

Our aim is to endow more intelligence to a team of robots, also called as agents, so that they will be able to autonomously coordinate a deployment in scenarios that lack of a communication infrastructure, in order to monitor them and delivering the information to a static Base Station. The mission of the robots is, having a map of the environment, planning a deployment to reach some locations of interest, where they make observations and then transmit this data back to the base.

In order to do this, several details should be considered: the size of the scenario altogether with the distribution of the obstacles, the position of the Base Station, the amount of available robots and the features of the communication devices, mainly the communication range. Obviously, in the majority of cases, some or almost all the aforementioned details of the environment, prevent to establish a direct communication link between the base and the agents, when they reach the goal locations.

Due to that, two situations may occur on the basis on the aforementioned features. In the first situation it will be possible to reach all the desired positions of the scenario maintaining connectivity with the base from the goal locations. And in the second case, it is absolutely impossible to maintain the connectivity. These two situations define the two main research lines of the present thesis.

In the first situation, the team of robots must reach some locations of interest in order to take measurements or observations of the environment. At the moment of reaching these

[1] https://www.theverge.com/2018/5/8/17331250/automated-warehouses

locations, the agents must ensure a communication link with the base. To do so, the agents are coordinated in order to form a multi-hop network, where some agents reach the locations of interest and the rest act as relays, re-transmitting the data down to the Base Station. Thus, some human operator or another autonomous artificial agent might interact with the robots at these positions. A motivating example of application of this approach, apart from simple monitoring, is a telemanipulation of a robot, using a video steam, from a base station in some hazardous environment. In this work, we refer to the multi-hop network of robots as a chain.

Here, the objective of the robot team is to fulfill the mission of reaching all the possible goals as fast as possible. So, in the present thesis we develop several methods that attempt to reduce the global time of the mission. The proposed methods to form chains of robots, reduce the number of agents that will be used for relay tasks. The idea is to dispose of larger number of agents to accomplish the primary task of visiting the locations of interest. The downstream of information to the Base Station is mandatory from the goal locations, but the robots are free to autonomously navigate in areas without connectivity going to the goals. However, changes in the scenario may occur, so it is preferable to keep the connectivity between the robots if this does not sacrifices many time. We use a communication-aware algorithm to deviate the robots to the areas with connectivity in the case if it does not deviate the robots from their shortest paths. This way, the motions of the robots are not strictly limited by the communication. All these issues are discussed in Part I.

In the second situation, where it is impossible to reach all the desired locations with connectivity with the base, even extending a chain, the agents must go to the locations to gather data and return to the base to deliver the information. Obviously this idea may not be so effective if the scenario is large, where the robots have to travel long paths in order to deliver the information. Therefore, we propose a simple but effective approach, where the robots of the team may act in two roles, workers or collectors. The workers are devoted to gather the data requested from the Base Station of the scenario. And the collectors are responsible of travelling constant tours, collecting the data gathered by the workers to later retransmit it to the base. New locations from where make observations are requested from the Base Station and the collector agents communicate them to the workers when they exchange information.

Therefore, the data is periodically requested from some goal locations and delivered by the agents to the base. The requested goals to reach change during the mission, with each gathering cycle. So, two factors must be considered: the amount of data to deliver and the "refreshing time" of this data. This is the elapsed time from the instant of request until the reception of the data. Thus, the aim of the team is to find the best balance between the number of delivered data packages and their refreshing time. An example of this kind of situations can be some critical missions such as fire monitoring. In that scenarios, some human operator is periodically requesting to a team of UAVs to take pictures from a fire in a forest. In some cases it will be more profitable to deliver more recent information. So the deployment will be coordinated for faster deliveries, despite of delivering all the tasks. All these aspects are dealt with in Part II.

1.1 Contributions

In the present thesis we tackle with the two most frequent situations in the monitoring missions using teams of robots. Along the document, we develop several methods to solve both aforementioned situations. In Part I of the document, we present the proposed methods to adopt chain formations with a group of robots, in order to be able to establish a communication link from a Base Station with the robots when they reach goal locations. The methods solving the second situation, for data gathering missions without continuous communication, is presented in Part II of this work. Thus, the contributions of the work developed during the present thesis, referred to both parts, are:

1. Deployment planning for communication with a static Base Station.
   - In Chapter 2 we develop a method to compute the optimal positions to place the relays in order to reach the goal locations providing a communication link to the Base Station. The proposed method considers two parameters to minimize: the number of robots that will be used in relay role and the estimated distances travelled by the robots from the base to the goals. This method is related to publication [1].
   - Chapter 3 presents a method to plan the deployment of the team, considering the communication with the base. We propose a path planning method for the agents in order to navigate within communication areas. Furthermore, we develop a method to allocate the relay tasks for the robots and the order of the goals visit in order to extend the communication area of the Base Station. This contribution is based on the publication [2].
   - In Chapter 4 we develop a method to coordinate big fleets of robots to visit large amounts of goals. The proposed approach obtains sub-optimal solutions to visit hundreds of primary goals where making observations of the environment with teams of dozens of robots, in a short time. The work of this chapter is related to the publication [3].
   - Possible connectivity failures due to appearance of new obstacles are considered in Chapter 5. We develop a distributed method for a team of robots, which starting from different locations of the scenario, make observations of the new observable obstacles in order to reconnect in a chain formation from the base to some goal location. The proposed technique is related to publication [4].
2. Data gathering with periodic information delivery to a static Base Station.
   - In Chapter 6, we present two techniques to synchronize two agents in movement. We introduce the concept of dynamic communication area. Then we develop a sampling-based trajectory planner for communication with time constraints. This technique is based on the Rapidly-exploring Random Trees (RRT) in order to provide fast and sub-optimal solutions. The second technique is an optimal version of a trajectory planner, which uses the Fast Marching Method as the base, in order to find out the optimal trajectories for synchronization. The sampling-based technique is based on the work published in [5].

– In Chapter 7 we develop a complete method for a data gathering mission using the robots in roles of workers and collectors. The proposed approach consists in a sequence of methods to plan the coordination of the robots. Firstly, we propose and evaluate three different area partition methods, to split the scenario into working areas of for the worker agents. Secondly, we present an iterative procedure to find out the best balance between collectors and workers required for the scenario. Then, we provide an algorithm to associate the workers to deliver their data either to a collector or to the base, based on the estimated workload of the workers. This procedure, establishes the associations altogether with the paths of the collectors, that are fixed balancing their time to return to the base and the time of the workers. This part is associated to the publications [6][7].

1.2 Related publications

1. Y. Marchukov and L.Montano, Multi-robot Optimal Deployment Planning Under Communication Constraints, In Proceedings of ROBOT2015: Second Iberian Robotics Conference, November 2015.

2. Y. Marchukov and L.Montano, Communication-aware planning for robot teams deployment, IFAC2017 World Congress: The 20th World Congress of the International Federation of Automatic Control, Toulouse, France, July 9-14, 2017.

3. Y. Marchukov and L.Montano, Fast and scalable multi-robot deployment planning under connectivity constraints, ICARSC2019: 19th IEEE International Conference on Autonomous Robot Systems and Competitions, Porto, April 2019.

4. Y. Marchukov and L.Montano, Multi-robot coordination for connectivity recovery after unpredictable environment changes, IAV2019: 10th IFAC Symposium on Intelligent Autonomous Vehicles, Gdansk, July 2019.

5. Y. Marchukov and L.Montano, Trajectory planning under time-constrained communication, In Proceedings of ROBOT2017: Third Iberian Robotics Conference, November 2017.

6. Y. Marchukov and L.Montano, Multi-agent coordination for on-demand data gathering with periodic information upload, PAAMS2019: 17th International Conference on Practical Applications of Agents and Multi-Agent Systems, Avila, June 2019.

7. Y. Marchukov and L.Montano, Multi-agent coordination for data gathering with periodic requests and deliveries, PAAMS2019: 17th International Conference on Practical Applications of Agents and Multi-Agent Systems, Avila, June 2019. Invited paper.

8. Y. Marchukov and L.Montano, Distributed algorithm for multi-robot regrouping in changing environments, Autonomous Robots, in preparation.

1.3 Videos

The videos of the simulations correspondent to each publication:

1. Relays placement: http://robots.unizar.es/data/videos/robot15yamar.avi
2. Communication-aware deployment planning:
   http://robots.unizar.es/data/videos/ifac17yamar.mp4
3. Fast and scalable deployment planning:
   http://robots.unizar.es/data/videos/icarsc19yamar.mp4
4. Connectivity recovery: http://robots.unizar.es/data/videos/iav19yamar.mp4
5. Time-constrained trajectory planning:
   http://robots.unizar.es/data/videos/robot17yamar.mp4
6. Data gathering: http://robots.unizar.es/data/videos/paams19yamar/simulations/

Part I

Deployment planning for communication with a static Base Station

Chapter 2

Relay positions computation

2.1 Introduction

In this work we have focused in environment monitoring missions. We consider the usage of a team of robots for this purpose. The mission of the agents is to reach some locations of interest and make observations of the environment. The agents must send the observed information to a static Base Station from the goals locations. Along the document the robots are also called as agents, the base station is denoted as BS and the primary goals are the locations that the agents must reach.

The way to send the information to the BS from the primary goals is establishing a communication link between the robots at the goals and the BS. Due to the connectivity constraints, such as limited communication ranges of the wireless sensors and the obstacles present in the environment, it is impossible to always establish the connectivity link. Therefore, some agents have to be used in role of relay, forming a multi-hop network, retransmitting the information of the agents that are taking measures at the primary goals. Thus, in this chapter we develop a method to compute the positions where will be placed the agents used as relays. These positions are called relay goals. In other words, we compute the relay goals that connect the primary goals.

In the proposed approach, the multi-hop connectivity network formed by the agents of the team has a tree topology. This means that one relay agent can provide connectivity to several agents. In other words, one relay may have several descendants, but each relay has only one parent. An illustrative example of a tree topology is depicted in Fig.2.1.

The robots can have both roles, visiting the relay and primary goals. So, when some agent has reached its primary goal for data acquisition, it is free to be used as a relay, serving to the teammates. Using lowest possible number of agents for relay tasks, it is advantageous from the point of view of the mission time. A smaller number of agents in relay role involves that more agents are available to reach the primary tasks. This, a priori, favours to fulfill the mission in lower time. Therefore, we need to compute the minimum number of relay goals to connect the primary ones.

Since the agents are used for both tasks, relay and primary, we study how to minimize the distances between the goals. This way, the displacements of the agents between the goals will be minimized. This can be beneficial in the case of being necessary to save energy of the agents.

In summary, we present a method to compute the relay goals positions to connect the

primary goals, with the purpose of enabling the robots to establish a connectivity link with the BS through some agents used as relays. We propose several heuristics to minimize either the number of agents used as relays or the estimated distances to the goals. Note, that robots are not used in this chapter and the method only obtains the positions of the relay goals to fulfill the deployment mission with connectivity.

The rest of the chapter is organized as follows. Some related works of the literature are presented in Section 2.2. In Section 2.3 we describe our problem. Section 2.4 describes the connectivity graph computation and the tree solution to deployment problem and shows some simulated tree configurations. In Section 2.5 we consider the optimization of the connectivity trees, in order to maintain the connectivity between the team when visiting the goals. Section 2.6 presents the simulated results, followed by the conclusions in Section 2.7. The work developed in this chapter is related to the publication [1].

2.2 Related works

In last decades many techniques were developed in order to control multi-robot teams executing deployment missions. These techniques pursue different purposes, as control of formations [8][9][10], when a team of robots must perform a task without losing communication and the goals of the robots are close one to each other. These methods are common in *master-slave* configurations, where a leader plans its trajectory and the rest of the team uses some reactive method to follow it [11].

Other type of task is exploration, which consists on covering some terrain and discovering the obstacles. In order to cover maximum area, the multi-robot team is separated [12]. These tasks may be executed in different ways, planning the paths of the robots to make them concur and share information as in [13]. Or assuring to maintain connectivity all the time as in [14][15][16][17]. However, in this works, it is necessary to avoid motions that lead to the disconnection of some member of the team.

A quite recent line of research in constant communication is an efficient channel modelling to have a better knowledge of the signal strength at every point of the map [18][19][20][21], in order to assure the connectivity during the deployment. Others make an effort developing robust algorithms to keep communication, based on the minimal binary rate [15], bit error ratio (BER) [19], Received Signal Strength Indicator (RSSI) to assure packet transmission [14] or the Signal-to-Noise Ratio (SNR) [22]. However, most of these works are based on a leader-followers configuration of the team. Only one predefined leader agent performs some kind of task and the rest of the agents are used in role of relay, providing connectivity. The relay agents are only executing a control law to follow the leader, in order to maintain a connectivity link between BS and leader. The problem here is that these agents are never used for other purposes, as for example visiting some primary goals. At the same time, due to the use of a control law, the relay agents perform reactive movements to provide connectivity to the leader, recurrently visiting the same positions several times.

Some works in the literature avoid these situations, computing the positions where to place the relays in order to provide connectivity to the primary goals and avoid unnecessary reactive motions. From these positions, some robots will make measurements and send information to the BS. In [23], the authors propose a method to find positions to interconnect different groups of disconnected agents. In this work all the agents can move and there no exists a static

anchor point to connect the agents, as is the case of the Base Station in our work. In [24], the BS is interconnected with one destination point through a relay chain of robots. However there is a unique destination point, so the relay positions are obtained for only a unique chain of relays. In [25], the authors propose a method to find relay goal positions to interconnect a small group of independent robots that are moving along the scenario. In this case, a tree topology for relays is used, as in our work. But, their solution become too reactive, since they do not control the robots to interconnect with the BS. That is, from the moment that the independent agents are connected, the subsequent obtained relay positions are adjacent to the previous ones. Even with a smaller number of goals, more similar approaches are related to exploration missions developed in works [26][27][28][29][30]. In [26] and [27], the authors propose a technique to solve the relay placement in exploration missions using Integer Linear Programming (ILP). In these works the number of goals is quite small, because all of them lay over the limits between the explored and unexplored areas. And at the same time, ILP is a time consuming process. Only for instances of 20 goals it requires up to minutes to obtain a solution. In [28] and [29], a faster approach is proposed. It is in large part thanks to the simplified communication model employed in both works. Only the distance range is taken into account, without considering obstacles that might prevent communication. The authors of [30], also solve the relay placement in exploration by using a recursive tree configuration, where the tree is built iteratively. At each iteration, the maximum number of agents to visit the primary goals is employed, leaving the minimal number of relays. When the primary goals are reached by the agents, the branch is contracted to the point when it is possible to reach another primary location. This methodology produces unsatisfactory behaviours of the relays deployments. Primary locations, that require just a couple relays, are reached employing a dozen of agents used as relay. In [31] also a tree topology is used. However, the root in that work is a moving agent, instead of a static BS used in our work.

Besides of all the aforementioned weaknesses, all the cited methods use a simplified communication model. The communication distance range altogether with the *line-of-sight* (LoS) between transmitter-receiver are employed. Nevertheless, the obstacles does not completely attenuate the signal and it is possible to establish a connectivity link through a wall.

Therefore, our proposed approach inspired by the optimal router positioning problem [32]. The router locations are optimally computed in order to cover all the map using the minimal number of routers and considering a more realistic communication model. But in our case the method will be applied to a team of robots. So, the estimated travelled distance by the agents it is considered in the method and the relays are extended from a static BS.

2.3 Problem statement

Consider N primary goal positions in a static environment. A position in the environment is denoted as x. A team of M robots has the objective of reaching the the primary goals locations, being $M \leq N$. The primary goals to reach are denoted by $\mathbf{X}$. When a robot reaches a goal, it must send the information to the BS from this position. Placing a robot in some position x_i it can provide connectivity to other robot that will reach another goal at the position x_j. Therefore, we can obtain a connectivity graph of the goal positions defined as $\mathcal{G}(\mathbf{x}) = (\mathcal{V}, \mathcal{E})$, where:

– $\mathcal{V}$ is the set of vertices defined as all connected goals $\mathbf{x}$ to the BS, being $\mathbf{x} \in \mathbf{X}$. n denotes the number of goals $\mathbf{x}$ and $n \leq N$. $\mathbf{x}(0)$ represents the position of the root or the BS.

– $\mathcal{E}$ is the set of edges between the vertices that is defined as $\{(x_i, x_j) \mid \gamma(x_i, x_j) \geq \gamma_{th}, x_i \in \mathbf{x}, x_j \in \mathbf{x}\}$, where $\gamma(x_i, x_j)$ is the signal strength between the positions x_i and x_j and γ_{th} is a constant signal strength threshold to establish a connectivity link.

An example of the connectivity graph is depicted in Fig.2.1(a). Since the goals positions are vertices in the graph, we refer to them also as vertices or nodes in this chapter. When a node of the graph is connecting other nodes, i.e. being used as relay, we call it link node, and the connected nodes are called as linked nodes.

The transmission channel is considered symmetric, thus $\gamma(x_i, x_j) = \gamma(x_j, x_i)$. We consider all the goals $\mathbf{x}$ connected because there exists a path in the graph from BS to all these goals. The path in the graph to some node or goal i is defined as a sequence of adjacent vertices in the graph $\mathcal{G}(\mathbf{x})$ and it is expressed as $P_i = \{x_1, \ldots, x_k\}$, where k is the the number of vertices in the path. The distance of this path can be defined as the sum of Euclidean distances between every pair of vertices that form a path to i:

$$d_i = \sum_k ||x_{k+1} - x_k||, \quad x_{k+1} \in P_i, x_k \in P_i \tag{2.1}$$

On the other hand, we define $\mathbf{x}_{disc}$ as the set of goals of $\mathbf{X}$ that do not have a path to the BS. So $\mathbf{x}_{disc}$ do not belong to the graph $\mathcal{G}(\mathbf{x})$, accomplishing $\mathbf{x} \cup \mathbf{x}_{disc} = \mathbf{X}$. Therefore, the first problem is to connect all the primary goal locations to the BS, formally defined as:

Problem 1 (Goal connection): Given the initial connected graph $\mathcal{G}(\mathbf{x})$, compute the new distribution of $\mathbf{x}$ so that $\mathbf{x}_{disc} \in \mathbf{x}$. In other words, we want to find a new $\mathcal{G}'(\mathbf{x})$ configuration that connects all N nodes, that correspond to the primary goals to reach.

In order to solve the *Problem 1*, connecting all the primary goals by moving some of already connected goals $\mathbf{x}$, it is necessary to know which of these goals can move. At the same time, the initial graph configuration may be suboptimal. That is, some goal positions are connected by more link nodes than actually needed. Therefore, for both cases, it is necessary to know which nodes perform the relay task and which ones are the linked nodes at each moment of the mission. For this purpose we define the second problem:

Problem 2 (Optimal goal linking): Given an initial graph, compute the optimal tree $\mathcal{T}^*$ from all the possibles $\mathcal{T}(\mathbf{x}) \in \mathcal{G}(\mathbf{x})$, so that the path to every node contains minimal number of nodes. At the same time, the costs of the edges, denoted as w, are taken into consideration for minimizing the distance or the number of nodes.

$$\mathcal{T}^* = \underset{\mathcal{T}}{\operatorname{argmin}} \sum_{u \in \mathbf{U}} \left(|\mathbf{P}_u| + \sum_{(x_i, x_j) \in P_u} w(x_i, x_j) \right) \tag{2.2}$$

where $\mathbf{U} : \{\mathbf{x} \in \mathcal{T} | dp(\mathbf{x}) > 2\}$ represents all the unreachable goals directly from BS, dp represents the depth of each node $\mathbf{x}$ in the tree, i.e. the number of hops from BS to each $\mathbf{x}$, $\mathbf{P}_m$ is a set of paths from the root of the tree to the node m, $|\mathbf{P}_m|$ is the number of nodes in each path, the pair (x_i, x_j) represents each consecutive pair of nodes of the path P_m and costs w are chosen according to different criteria, described in the next section.

Now, the order of the movements of the agents in order to sequentially extend the coverage area during the mission is obtained. Let us define the variables for this process. The positions $\mathbf{x}_{visit}$ represent the set of visited primary goals, i.e. some robots have visited these goals, and from these positions can provide connectivity to other primary goals. Only the base station is considered as visited goal at $t = 0$. $\mathbf{x}_{visible}$ is a set of nodes that are inside the coverage area and may be visited, formally defined as $\mathbf{x}_{visible} : \{\mathbf{x}|dp(\mathbf{x}_{visit})+1\}$. The visible nodes are descendants of the visited goals, that is, $dp(\mathbf{x}_{visible}) = dp(\mathbf{x}_{visit}) + 1$. Despite the fact that it is possible to provide connectivity to other nodes, these positions may be suboptimal. Here, suboptimal means that these nodes can be connected with lower number of link nodes, i.e. using less relay robots. Thus, we define $\mathbf{x}_v$ as virtual goals or nodes, that are obtained after optimizing the positions $\mathbf{x}_{visit}$ in order to provide connectivity to the next set of primary goals in a more efficient way.

Thus, the visited primary goals are extracted from the goal list and new virtual goals $\mathbf{x}_v$ are inserted as relay tasks. This can be formulated as:

$$\mathbf{x}_k^+ \leftarrow (\mathbf{x}_k \backslash \mathbf{x}_{visit}) \cup \mathbf{x}_v \tag{2.3}$$

where $\mathbf{x}_k$ and $\mathbf{x}_k^+$ are the nodes before and after optimization, respectively. And the procedure of computation of optimal positions where the robots will link other agents is formally defined as follows.

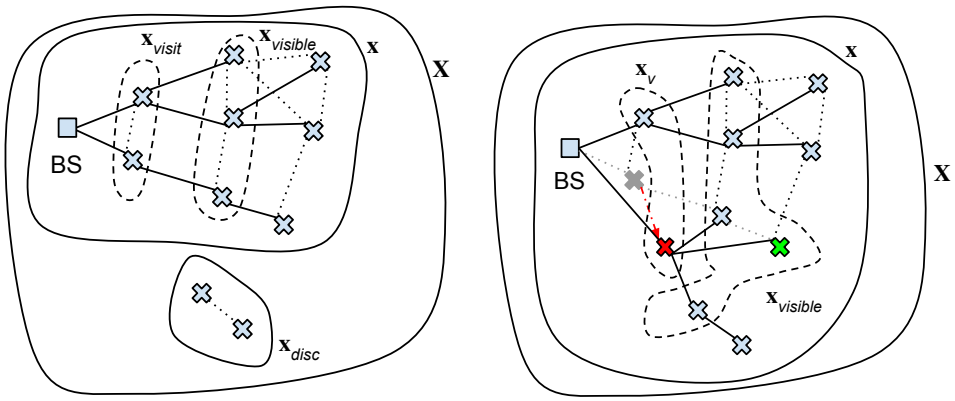
Problem 3 (Optimal positions): Let $\mathbf{x}_{visible}^+$ be all desirable visible goals after visiting the computed virtual goals. Formulated as $\mathbf{x}_{visible}^+ : \{\mathbf{x}|dp(\mathbf{x}_v) + 1\} \cup \{\mathbf{x}_{disc}|\gamma(\mathbf{x}_v, \mathbf{x}_{disc}) \geq \gamma_{th}\}$. We determine all the virtual goals that allow to maximize the number of visible nodes and guarantee their connectivity:

$$\mathbf{x}_v^* = \underset{\mathbf{x}_v}{\operatorname{argmax}}(|\mathbf{x}_{visible}^+| \ni \gamma(\mathbf{x}_v, \mathbf{x}(dp(\mathbf{x}_v) - 1) \geq \gamma_{th}) \tag{2.4}$$

All the process of optimization is illustrated in a simple example in Fig.2.1. The crosses represent all the primary goals, that is, all the locations that should be visited. The discontinuous line depicts the possible links between nodes, that is the graph $\mathcal{G}(\mathbf{x})$. And the continuous line depicts communication links, or the optimal tree $\mathcal{T}^*(\mathbf{x})$ for the distribution of the nodes $\mathbf{x}$. In 2.1(a) the initial nodes distribution is represented, the minimal depth tree is obtained with eq.2.2. Some nodes are initially disconnected, so can not be visited with communication. Therefore, using eq.2.3-2.4, the optimal positions of the virtual goals $\mathbf{x}_v$ are computed, so that they maximize the number of visible nodes and connect disconnected ones, depicted in Fig. 2.1(b). As we can observe, only one agent will need to change the position in order to improve the connectivity. It connects the disconnected nodes, so now these goals are reachable with communication. And reducing the number of links to reach some goals, thus requiring less robots in relay role.

2.4 Connectivity tree computation

In this section, we describe our method to compute the graph and the tree, linking the goals. First, in Section 2.4.1 we define the path-loss model, used to compute the possible links between the nodes. This contributes with binary information to each edge, 1 when exists a link and 0 when there is no connection between the nodes. In order to add more flexibility to



(a) Initial tree configuration (b) Tree configuration after optimization

Figure 2.1: Optimization process. In 2.1(a) the visited nodes $\mathbf{x}_{visit}$ provide signal to the nodes of next level of the tree, $\mathbf{x}_{visible}$, and some nodes are disconnected $\mathbf{x}_{disc}$ from the rest of the team. Fig.2.1(b) shows the location for the optimized virtual nodes $\mathbf{x}_v$ providing signal to the next level of the tree. As the upper node didn't find better location thus it keeps at the same position, and the lower node, highlighted as a red cross, changed its location in order to connect the disconnected nodes ($\mathbf{x}_{disc}$) and reduce the depth of one node (green cross), so reducing the number of agents to be used to link this goal.

the tree configuration, the different ways to weight the edges and the tree computation are defined in Section 2.4.2 and, finally, we show some tree examples in Section 2.4.3.

2.4.1 Path-loss model

As defined in Section 2.3, the existence of the edges depends on the signal strength. Many of works use simplified signal model that only considers the distance. But the obstacles information is not considered. In order to obtain a more realistic signal approximation to obtain the links, we take into consideration a geometric information of the scenario including the shadowing effect. It considers the traversed obstacles by the signal. So, we use the model proposed in [33]. This work defines a Multi-Wall-and-Floor model (MWF), considering the attenuation due to the number of traversed walls and floors, where the loss of the signal is:

$$L_{MWF} = L_o + 10\mu \log_{10}(d) + \sum_{i=1}^{I}\sum_{k=1}^{K_{wi}} L_{wik} + \sum_{j=1}^{J}\sum_{k=1}^{K_{fj}} L_{fjk} \tag{2.5}$$

where L_o is the path loss at a distance of 1m, μ is the path-loss exponent, d represents the distance between transmitter and receiver, L_{wik} is the attenuation due to the material type i and $k-th$ traversed wall, I is the number of traversed walls, K_{wi} the number of traversed walls of category i, L_{fjk} is the attenuation due to the material type j and $k-th$ traversed floor, J is the number of traversed floors, K_{fj} the number of traversed floors of category j.

Therefore, eq.2.5 is used to compute the received signal power and then, using a threshold, the edges of the graph are obtained. The signal is obtained as:

$$\gamma(x_i, x_j) = \gamma(x_i) - L_{LMF}(x_i, x_j) \tag{2.6}$$

where x_i and x_j are the transmitter and receiver positions, respectively. $\gamma(x_i)$ is the transmitter power and $L_{LMF}(x_i, x_j)$ is obtained using eq.2.5.

2.4.2 Connectivity tree and weight function

Algorithm 1 summarizes all the procedure of the tree computation. For notation, $\mathbf{p}$ represents all the primary goal positions and their links $\mathbf{l}$, so $\mathbf{p} = [\mathbf{X}^T \ \mathbf{l}]^T$. Initially, the value of each link l is fixed to zero. But, once the tree is constructed, the values are assigned to every node. As $\mathbf{X}$ contains both, the connected $\mathbf{x}$ and disconnected goal locations $\mathbf{x}_{disc}$, the link values of the disconnected nodes are $l = \infty$. The signal strength is computed in order to obtain the connectivity graph using eq.2.6, in lines 1-3 of the algorithm. Nevertheless, this results in a binary information, 1 value when exists a link and 0 when there is no connection between the nodes. Denoting existence or not of edges. As described in Section 2.3, our purpose is to obtain the shortest paths in the graph, i.e. minimize the number of nodes in each branch of the tree which is tantamount to reduce the number of agents used as relay. Furthermore, we are interested in using some criterion for local minimization of distance or links. Therefore, a weight function with these restrictions is computed to each edge of the same depth of the tree using (2.2). Thus, the costs of the edges are more flexible and vary within the interval [0,1]. The higher values are interpreted as the worst option and the lower ones are the potential candidates to pertain to a trees branch. This procedure is stated in lines 4-8 of the algorithm. First, the method computes the depths of all the connected goals in l.4. Then, it obtains the costs of the nodes in a sequential way by depths of the nodes in the tree. At each iteration, or each depth l, it selects the edges between the nodes depths l and $l+1$ denoted as $\mathcal{E}_l \in \mathcal{E}$, and adjusts their values, l.5-7.

The *cost_edges* function in line 8 calculates these values following three different criteria:

- Connectivity: number of descendants of a node. Using this criterion the linked nodes are connected to nodes which have the highest number of possible connections.
- Exclusiveness: degree of a node, i.e. number of edges incident to the vertex or number of possible links providing communication to the node. With this criterion we force the nodes to connect to links which are the only ones able to provide connectivity to some nodes.
- Distance: the distance between relay and linked nodes. This criterion is used for

Algorithm 1 Tree computation

Require: Goal positions $\mathbf{p}$, threshold γ_{th}, path-loss exponent μ

1: **for each** $p \in \mathbf{p}$ **do**
2: $\quad \mathcal{G} \leftarrow signal\_strength(p, \mathbf{p}, \gamma_{th}, \mu)$ ▷ Using eq.2.6
3: **end for**
4: $dp(\mathbf{x}) \leftarrow compute\_depth(\mathcal{G})$
5: **for** $l = 1, .., max(dp(\mathbf{x}))$ **do**
6: $\quad \mathbf{w} \leftarrow cost\_edges(\mathcal{E}_l)$
7: **end for**
8: $\mathcal{T} \leftarrow graph\_lower\_cost(\mathbf{w}, \mathbf{p}(0))$
9: **return** Minimum Depth Tree $\mathcal{T}$

distance reduction from the BS to the goals.

Hence, the defined criteria that is used in the *cost_edges* function, is formulated as:

$$\mathbf{w}_t = \alpha_c \mathbf{w}_c + \alpha_e \mathbf{w}_e + \alpha_d \mathbf{w}_d \tag{2.7}$$

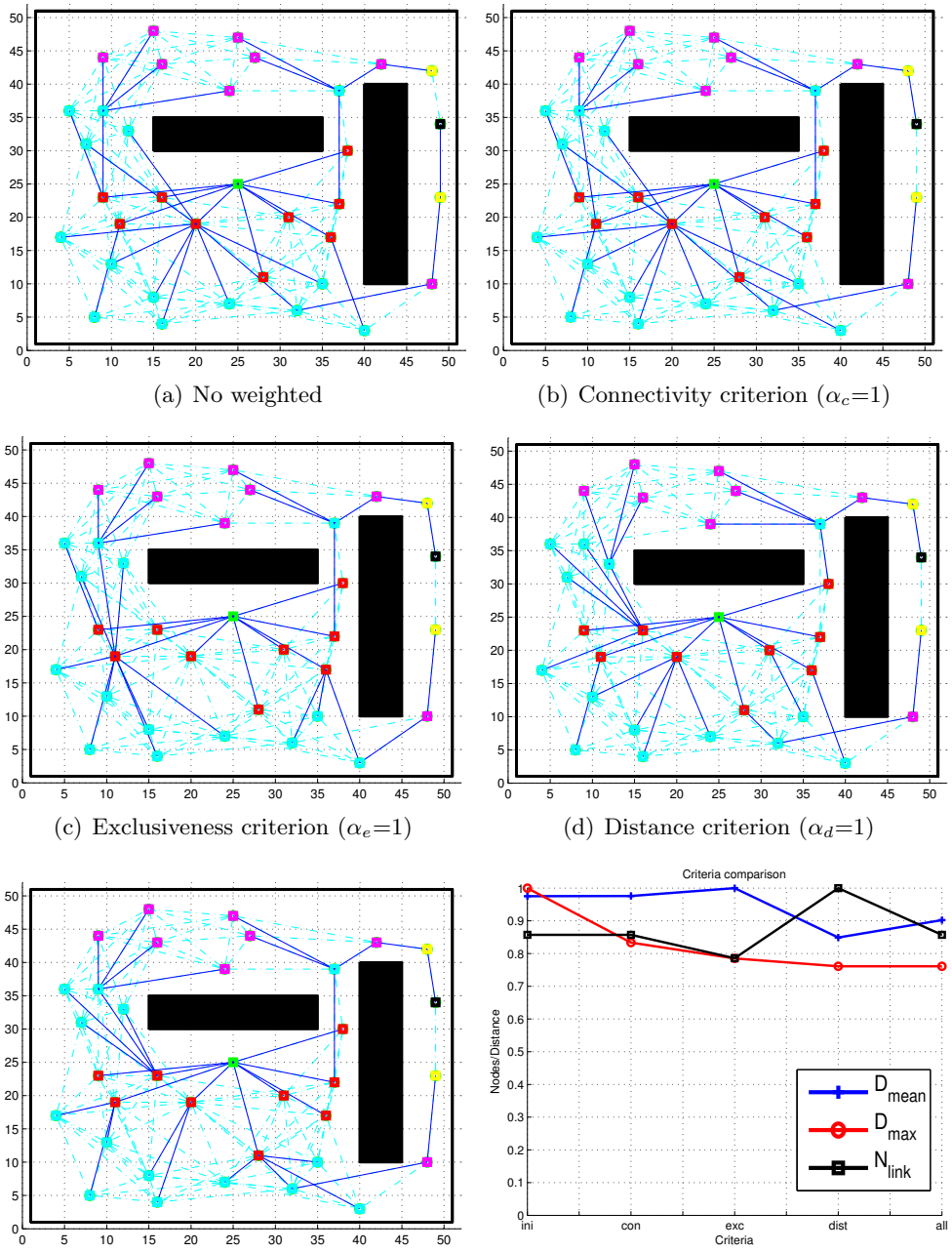
where $\alpha_c, \alpha_e, \alpha_d$ represent the weighting coefficient of the different criteria.

Finally, in the line 8 is executed a shortest path algorithm to build the tree. The root of the tree is BS, which is the unique member of the team able to communicate with all the members. Since all the obtained costs are positive in presence of a link and null in its absence, Dijkstra algorithm is used for the shortest path computation.

2.4.3 Connectivity tree examples

The tree is obtained finding the shortest path to each node based on the nodes depth in the graph, as formulated in eq.(2.2). That is, minimizing the number of components participating in every path or branch of the tree. This assures to save up the maximum number of robots for primary tasks. For accurate signal simulation, the real signal parameters are extracted from [18] and [33]. As our problem is a deployment of multi-robot teams in urban or building scenarios, only the walls are considered for the *MWF* propagation model in eq.2.5, so floor component L_{fjk} is not considered. We consider 10dB of attenuation due to each traversed wall, so $L_{wik} = 10$ in eq.2.5. The path-loss exponent is considered as in free space, being $\mu = 2$ [33]. As proved in [18], the RSSI threshold (γ_{th} in our work) should remain at least at -70dBm in order to assure 100% of Packet Delivery Ratio (PDR).

An example for a tree configuration, using different criteria in eq.2.7 described in Sect.2.4.2, is represented in Fig.2.2. The obtained configurations depict different trees due to the weighting criterion. The first configuration of Fig.2.2(a), was obtained using eq.2.6 and computing the tree with Dijkstra algorithm computed, without applying the weighting function. So the edges take values of 0 or 1. The next two criteria, in Fig.2.2(b)-2.2(c), have a node saver performance, that is, attempting to use the minimum number of link nodes (relays). The tree applying the distance criterion is illustrated in Fig.2.2(d). And finally, the tree of equitable weighting for all the criteria is depicted in Fig.2.2(e). The normalized results, for each criterion are depicted in Fig.2.2(f). The evaluated metrics are: the number of relays or link nodes (N_l) required for the deployment mission, and the mean (D_{mean}) and maximum (D_{mean}) distances from BS to the nodes. The distances are computed using eq.2.1, that represents an approximated travelled distance to deploy each branch of the tree. We can observe, that any of the criteria has better performance than no weighted scenario. It obtains a configuration with larger distances and using more link nodes. Connectivity and exclusiveness criteria save up relay nodes in exchange of obtaining larger distances. Otherwise, the distance criterion computes paths with shorter distances, but using more nodes as relay. The results of the equally distributed weights reach a compromise between the distance and links saving. Thus, this combination can be adopted in scenarios that present difficulties due to the obstacles.



(a) No weighted

(b) Connectivity criterion (α_c=1)

(c) Exclusiveness criterion (α_e=1)

(d) Distance criterion (α_d=1)

(e) All criteria ($\alpha_c = \alpha_e = \alpha_d = 0.3$)

(f) Nodes and distance comparison

Figure 2.2: Tree configuration using different criteria. Fig.2.2(a)-2.2(e) highlight configurations adopting different criteria, each color of the nodes represents different tree depth. Cyan dotted lines represent possible links or the graph $\mathcal{G}$ and blue lines represent the chosen links or the tree $\mathcal{T}$. 2.2(f) depicts the normalized values of link nodes (N_l) and normalized mean and maximum distance (D_{mean} and D_{max}) of each variable for different criteria.

2.5 Optimal deployment

The previous algorithm obtains the best connectivity tree to interconnect the nodes, that can be reached using different criteria. But this first configuration still may leave some disconnected nodes and use more relays than actually required for the mission, as described in Sect.2.3. Therefore, here we develop the next step of the method, the improvement or optimization of the connectivity tree.

All the iterative process of optimization is described in Algorithm 2 and it consists in finding the solution to eq.2.4. As described in Section 2.4, the first step is to compute the tree, in l.1, i.e. obtain the links or edges between every node. In this section, we are using points $\mathbf{p}$ instead of $\mathbf{x}$, because $\mathbf{p}$ includes the links of the nodes. The depth of each node is extracted from the tree in l.2, and the visible nodes are selected in order to plan the trajectory to these points. The algorithm is executed for all the nodes belonging to the same depth of the tree, l.4. In Section 2.3 we established that only the BS is considered as visited at the beginning. Thus, when some primary nodes are considered visited, the new optimal positions $\mathbf{p}_v$ for the relay goals are computed. The visited nodes are sorted according to their occupation, in line 5. That is, a node is idle if it doesn't have descendants in the tree and is considered occupied or busy if it has some descendant node. Then, the first robots, that look for their optimal positions, are the idle robots. When the virtual goals are visited, placing some robots at these locations to provide signal to other members, considered now as visible $\mathbf{p}_{visible}$ and the some robots can reach these positions and send information to the BS. This procedure is repeated until obtaining the optimal positions in all the depths of the tree.

The computation of virtual goals is described in the Algorithm 3. The first step is to delimit the area that contains all the possible candidate positions where to place these goals. This area must be reachable from the lower and higher levels of the tree and it is defined as A_l in line 2 of the algorithm. This area corresponds to the intersection between the signal coverage areas of the nodes of lower ($A_{level<l}$) and higher ($A_{level>l}$) depths than the optimized depth A_l. Since the optimization process also includes the disconnected nodes, as described in (2.4), the intersection area includes the signal of all the reachable disconnected goals A_{disc}. The function discards the obstacles positions A_{obst} and also the areas where the fading due to the shadowing degrades substantially the signal A_{shadow}.

Once the area is computed, each of these points is a candidate to be a virtual goal,

Algorithm 2 Main algorithm

Require: Initial positions $\mathbf{x}$

1: $\mathcal{T} \leftarrow tree\_comp(\mathbf{x})$ ▷ Using Alg.1
2: $dp \leftarrow compute\_depth(\mathcal{T})$
3: $\mathbf{p} \leftarrow \mathcal{T}$
4: **for** $l = 2, .., max(dp)$ **do**
5: $\quad \mathbf{p} \leftarrow sort(\mathbf{p}_{visit})$
6: $\quad \mathbf{p}_v \leftarrow compute\_vp(\mathbf{p}, l)$ ▷ Using Alg.3
7: $\quad \mathbf{p} = \mathbf{p} \backslash \mathbf{p}_{visit}$
8: $\quad \mathbf{p} = \mathbf{p} \cup \mathbf{p}_v$
9: **end for**
10: **return** Optimized points $\mathbf{p}$

Algorithm 3 Virtual points search

1: **for each** $compute\_vp(\mathbf{p}, l)$ **do**
2: $A_l = (A_{level<l} \cap A_{level>l} \cap A_{disc}) \notin (A_{obst} \cup A_{shadow})$
3: **for each** $p \in A_l$ **do**
4: $\mathbf{p} \leftarrow insert(p)$
5: $\mathcal{T} \leftarrow tree\_comp(\mathbf{p})$
6: $dp \leftarrow compute\_depth(\mathcal{T})$
7: $\mathbf{ld} \leftarrow |dp(l+1)|$
8: **end for**
9: $\mathbf{pl} \leftarrow \{A_l | max(\mathbf{ld})\}$
10: $\mathbf{d} \leftarrow dist(\mathbf{p}_l, \mathbf{pl})$
11: $\mathbf{vp} \leftarrow \{\mathbf{pl} | min(\mathbf{d})\}$
12: **end for**
13: **return** Virtual points **vp**

including the initial visited goals, expressed as $\mathbf{p}_l$. There are cases where it doesn't exist any virtual goal point $\mathbf{p}_v$ which is able to improve the present configuration, reducing the number of relays. Therefore every point of A_l is analyzed, l.3, and inserted into the points vector $\mathbf{p}$ in l.4. Then the Minimum Depth Tree $\mathcal{T}$ in l.5 is computed using Algorithm 1 and the depth of every node is obtained, l.6. Greedy criterion is adopted in order to maximize the number of linked nodes, so that, the amount of the linked nodes is stored in the variable **ld** in l.7. All the candidates that are providing the highest number of connections, are selected to be candidates for virtual goals in l.9 denoted as **pl**. This procedure leads us to compute the solution of (2.4), thus the number of visible nodes is maximized and, at the same time, communication is preserved. These maximized visible nodes include disconnected nodes and nodes which pertain to higher tree depths, now achievable employing minimal number of link nodes.

From the list all the possible link candidates **pl**, in l.9, the algorithm selects the nodes which are closer to the original visited nodes in l.10-11, in order to reduce the distance of displacement of each robot.

2.6 Simulations and discussion

In this section some simulated results of virtual goals search are presented in order to validate the developed algorithm. As in section 2.4.3, the parameters are fixed in $\gamma_{th} = -70$dBm, $\mu = 2$ and the strength of the transmitted signal is $\gamma_{tx} = -42$dBm. The algorithm is executed in the scenario depicted in Fig.2.3(a). The simulations were performed for several number of randomly distributed goals over the map. The disconnected nodes should be

Table 2.1: Weight coefficients

	ini	con	exc	dist	con+dist	exc+dist	con+exc	all
α_c	0	1	0	0	0.5	0	0.5	0.3
α_e	0	0	1	0	0	0.5	0.5	0.3
α_d	0	0	0	1	0.5	0.5	0	0.3

reached with communication and the depth of the tree, should be reduced, so that reducing the number of links of every branch. The search of virtual goals is executed using different weighting coefficients, in order to compare the performance of the algorithm for different goals distributions and in different scenarios. Thus, the values of the coefficients used in the experiments are depicted in the Table 2.1. A video with several examples is attached for better understanding of the algorithm.[1]

The evaluation of every point of the map in the process of optimization of connectivity would make the problem intractable in terms of computation time. Therefore, the intersection of the coverage area, in line 2 of Algorithm 3, substantially reduces the number of evaluated points.

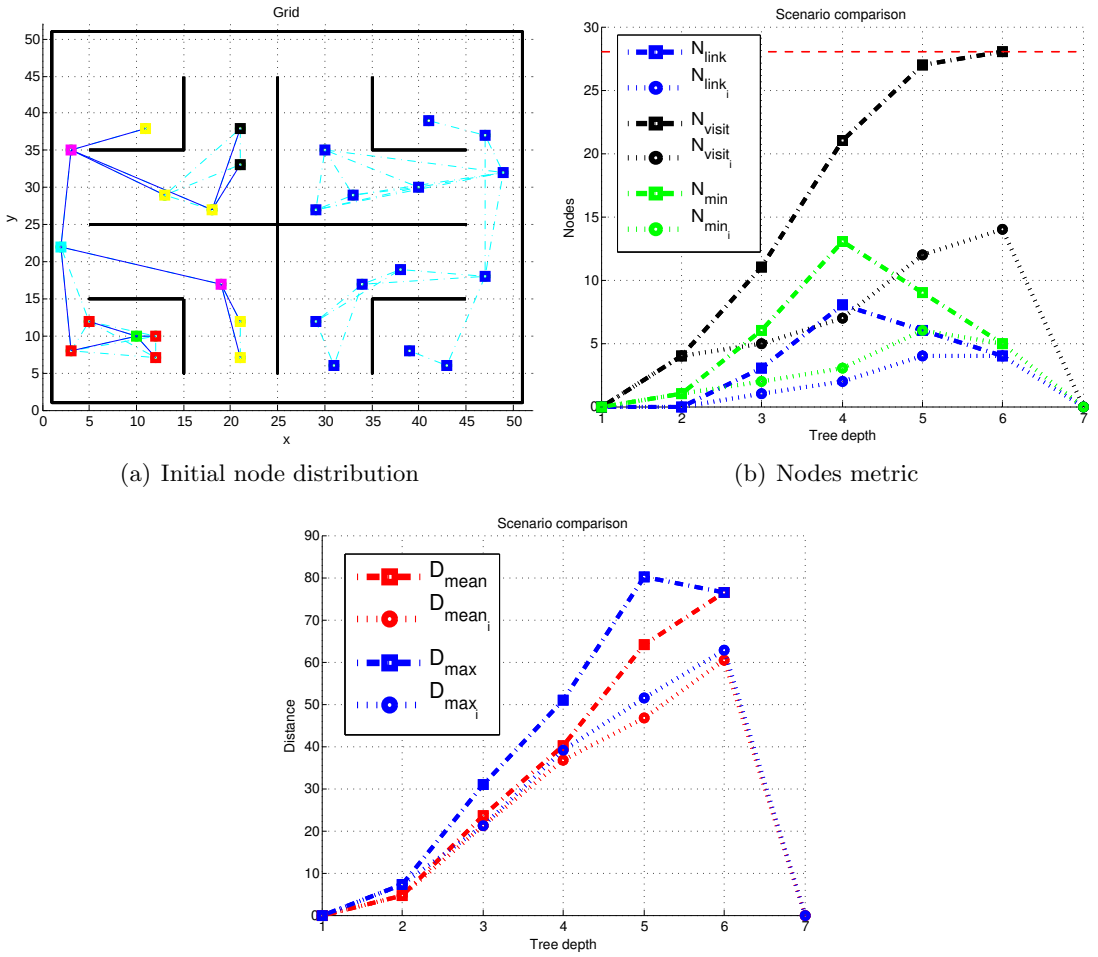
Firstly, we show the advantages of the proposed optimal tree planner (Alg.2) for the deployment in the scenario of Fig.2.3. The algorithm computes the virtual goals to reduce the number of link nodes (relays) or the distance to the nodes, using eq.2.1. Independently of the optimization criterion, the usage of the optimized goals reduces both values, the links and the distance. In Fig.2.3(b) the node metric is depicted and Fig.2.3(c) represents the distance of each level of the tree. For nodes metric, we evaluate three parameters. The number of link nodes required for the deployment defined as N_{link}, representing the positions where some agents will act as relays. The number of nodes reachable from each depth, denoted as N_{min}. And the total number of visited nodes, or primary goals, at each depth N_{visit}. Regarding the distances, we evaluate the mean(D_{mean}) and maximum (D_{max}) distances for each depth. The tree depth minimization for optimized results is observed. The optimal tree improves the connectivity of the nodes, reducing their depth. This reduces the number of link nodes to reach the goals, consequently reducing the number of agents used as relay. The depth reduction and the connection of previously disconnected nodes, are the reasons of why at each level the number of visited node N_{visit} is so high. As we can see, all the goals are reachable with communication with the BS. The distances are higher at each depth because there are mode achievable goals at each depth of the optimal tree.

The connectivity and exclusiveness criteria often perform similar results, although compute different weights. This is because of the dense distribution of the nodes. That is, when a link node is close to a group of nodes to be linked, it has many possible links with the next depth of the tree, so it has high connectivity. At the same time, it is probably that this node is the unique able to provide signal to some distant node of the group, so it has higher exclusiveness than the other nodes of the same depth.

Now we evaluate the proposed method for different instances of goals. Fig.2.4 shows the average results obtained for 50 trails of randomly distributed goals. We test the method for 10, 25 and 50 goal nodes. Here we evaluate the distances and only the nodes used for the mission and the maximum number of nodes required. The results highlight the performance of the three criteria and different combinations of them, proposed in Table 2.1. Both, the connectivity and exclusiveness, reach the purpose of link nodes saving. Predictably, the distance criterion reduces the distances to reach the goals in exchange of employing more nodes as relay. We observe a trend that in scenarios with higher node density, the differences between different criteria are more remarkable. This is due to amount of the nodes, that is, in these scenarios there are many choices for the possible locations of virtual goals.

In maps with many obstacles a better performance is found for combinations of distance

[1] http://robots.unizar.es/data/videos/robot15yamar.avi



(a) Initial node distribution

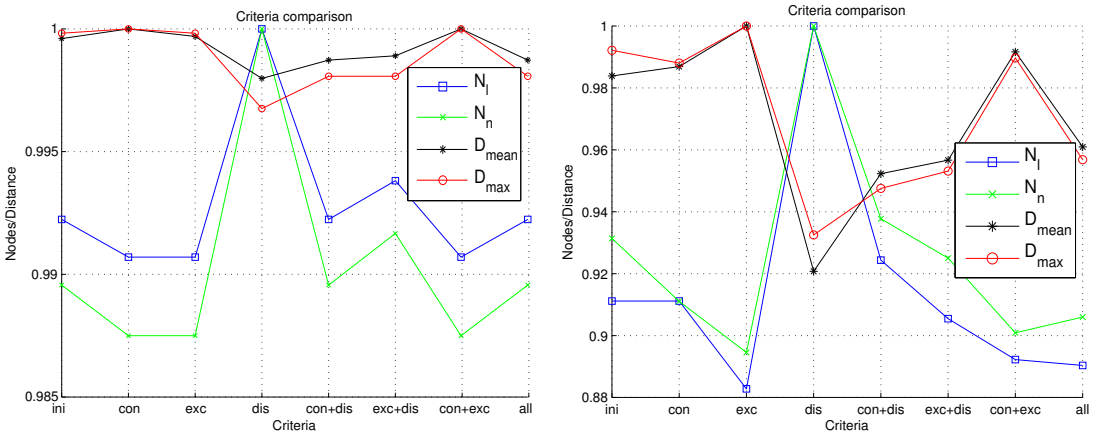
(b) Nodes metric

(c) Distance metric

Figure 2.3: Comparison between initial tree in 2.3(a) performance and the tree with optimization at each depth using virtual goals. The example contains 28 random goals to reach. The BS is depicted with green square, located at (10,10), and the disconnected nodes are represented with blue squares. Circles in 2.3(b) and 2.3(c) and sub-index i represent the results for the tree without optimization and squares depict the results using the optimization where all the nodes are connected. 2.3(b) highlights the nodes usage, in blue the number of link nodes (N_{link}), in black the number of visited nodes with communication (N_{visit}) and in green, the minimal number of required nodes for every tree depth (N_{min}). In Fig. 2.3(c) the results of distance metric are represented, mean (D_{mean}) and maximum (D_{max}) distance, in meters, for each tree depth are represented in blue and red respectively.

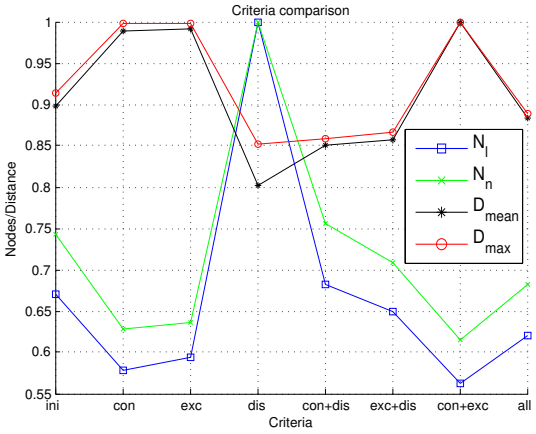
and any of node saving criterion. The connectivity approach saves more relay nodes in high goal density scenarios. On the contrary, in these scenarios, the combination of exclusiveness and distance criteria shows to accomplish the task with better balance of link nodes and distance.

Due to the geometric distribution of the obstacles, the different criteria provide suboptimal solutions, specially the distance criterion. The paths formed by the links between the nodes



(a) 10 nodes

(b) 25 nodes



(c) 50 nodes

Figure 2.4: Average results for different number of randomly distributed goals using the weights of Table 2.1. Every figure depicts normalized values with respect to the value of that variable for all the criteria: the total number of nodes used as links (N_l), the total number of nodes required for the mission (N_n), the mean and maximum distance travelled by the team (D_{mean} and D_{max}, respectively).

may traverse some wall and do not take into account the cost of surrounding this wall. Thus, one limitation of our method, is that the algorithm may compute points which will not be reachable with the shortest path or might lead to scenarios where all the disconnected nodes can not be connected, due to the obstacles distribution. As future work it may be profitable to compute a penalization factor dependant of the obstacle density. This factor will be included in every traversing wall edges, so that, avoiding to build a path through walls which are hard to surround.

Our algorithm carries has a greedy nature. That is, it attempts to connect a higher number of nodes and, at the same time, to minimize the distance to the optimal position. So that, if a better position for a node is not found, these node remain in the same locations. For future works, some informative positions may be computed, instead of minimal distance,

in order to direct the node to positions that will be beneficial for future steps of optimization process.

Another parameter to study is the task allocation when starts the algorithm. We tested our method using the criterion of the *idles first*, described in section 2.5. With this approach, the nodes that do not perform the task of link are the first to be optimized. It is practical in scenarios where there are few nodes per tree level. But denser distributions may lead to configurations where a virtual goal is found for a robot, that is the idle one, but is too far from the goal. However, there exist other agents considered busier, but are closer to these position.

2.7 Conclusions

In the present chapter we have developed a method to compute the relay goals for a team of robots, where some of the members of the team will act as relays during the deployment in a monitoring mission. The robots that reach these positions will act as relays, providing connectivity to their teammates, that are visiting the primary goals. This way, it is possible to establish a connectivity link, through a multi-hop communication network, with the Base Station.

We have presented an algorithm to compute a connectivity tree, that obtains the best links between nodes (goals), according to nodes or distance saving criteria. The virtual goals computation, allows to reach all the goal positions with connectivity.

With the proposed approach, we attempt to reduce the number of the required relays used for the mission. This way, a larger number of robots are available for the primary tasks. We have used the Multi-Wall-and-Floor model, that includes the obstacles information, to obtain more accurate simulated signals to plan the deployment. The simulated results show that the algorithm allows flexibility in terms of relay nodes or distance that will travel the agents depending on the scenario.

Our algorithm obtains the sequence of relay or link goals, in order to assure the connectivity when the agents will reach the primary goals. However, we have not considered moving agents in a deployment mission. The following chapter includes a communication-aware path planner and a goal allocation method, to be used altogether with the algorithm developed in here, for a complete deployment mission.

Chapter 3

Communication-aware deployment planning under connectivity constraints

3.1 Introduction

After obtaining the positions for the relay goals, in order to establish the connectivity links from the primary goals and the Base Station, there are multiple ways of visiting the goals. The agents must take into account the kind of tasks when visiting them, the relay and primary goals. The agents are able to transmit the information from the primary goals, only if all the required relay agents are at their place at that moment. So, the relay agents reach their correspondent goals and enable the communication at the primary goals. And then, the agents aimed to visit the primary goals, reach them with connectivity with the BS.

We assume that initially there are as many robots as places to be reached. But it is also possible to find a solution even when not all the robots are used. In this chapter, we propose two approaches: (i) employing all the agents to visit the goals, so the time of the mission will be reduced; (ii) using the minimum number of agents for the mission, to minimize the resources. This latter approach is useful in cases when not all the robots are available for the mission.

This planning procedure is done previously to deploy the team in the scenario. So, the plan is obtained using the map of the scenario. However, some changes in the environment may occur. Particularly in disaster scenarios where the corridors may be blocked by some obstacles or in scenarios with human presence that, for example, may close the doors. These events may alter the communication signal that interrupts the connectivity links between the agents. In order to prevent the communication breakdown, it is more profitable to try keep the team connected as long as possible. We propose a communication-aware path planning method to maintain the agents within the communication areas of their relays in order to be able to replan in case of detecting changes of the map.

The rest of this chapter is organized as follows. Section 3.2 summarizes the works of the literature related to the deployment missions. Section 3.3 presents the overview of the system and its parts. In Section 3.4, we describe the employed communication model. Section 3.5 develops the communication-aware path planning method. In Section 3.6 the complete deployment planner and its parts are presented. Finally, in Sections 3.7 and 3.7 we present

the results and discuss them. The results of this chapter were presented in [2].

3.2 Related works

As described in the previous chapter, there exist two main ways to coordinate a team of robots considering the type of communication between the agents: permanent or intermittent. In both cases, some relay agents are used in relay role, in order to build a connectivity link from the agents that are reaching the goals, for monitoring, to the BS. In the case of permanent connectivity, the most common way to keep the primary agents connected to the BS, is to apply a control law for the relay agents, as in works [15][19][34]. The problem of this, as mentioned in the previous chapter, is the reactivity of the movements of the relay agents. This is avoided if using intermittent type of communication between the agents. In other words, obtaining the positions where to place the agents that act as relay and only establishing connectivity at these points, without considering the communication when the team travels between the goals. This is the case of most of exploration missions developed in works [26][27][28][29][30]. The problem of completely ignoring the connectivity during the navigation between goals is that at any given time the scenario may change, when the robots are not connected. For instance, a simple door closure may prevent the connection between agents. So, the communication does not exist as expected by the plan and the agents should reestablish connectivity from the changed scenario, resulting in a loss of time. Thus, the proposed methodology in this chapter, provides a more flexible solution. The relay goal positions are computed by the algorithm proposed in Chapter 2 and the agents obtain paths that attempt to keep the robots within a communicated area if the deviation from the direct path is not substantial. This way, the connectivity between the agents is maintained as long as possible but not being a strict constraint, in order to prevent being disconnected when some environment change it is produced. This is similar in spirit to the idea proposed in [35] in the aspect of disconnection anticipation. The authors propose a decentralized control strategy to anticipate possible disconnections of some agents of the team, in the case of failures of some agents.

A path planning technique with intermittent connectivity was developed in [13]. Here the authors propose a method to plan paths for periodic meetings for a team of agents, in a searching mission. In our work, the periodic connectivity is not required, the robots must create a connectivity link with the BS at the moment of reaching the primary goals. Being this process asynchronous in the different branches of the relays tree. In [36], a method to compute a path to a destination while communicating with other independent agents navigating in the environment is developed. The proposed solution is based on Mixed Integer Linear Programming (MILP) to predict the possible wireless links with the independent agents and to define the reward of the communicated areas. In [19], a control law is developed in order to move a formation of a team of agents using BER in order to interconnect a BS with a static destination point. In [15], the authors propose to move a formation of agents by predicting the higher bit-rate motions. However, all these communication parameters rely on Received Signal Strength (RSS). As shown in [18], there exist a RSS level that guarantees an acceptable Packed Delivery Ratio (PDR) and, consequently, the data transmission. Therefore, in this work we employ only the signal estimation to obtain the communication-aware paths for the agents.

3.3 Proposed approach

3.3.1 System overview

The easiest way to deploy the team to visit a sequence of goals, it is planning the shortest path to every goal location. We employ the Fast Marching Method (FMM) [37] as the base method for path planning, as explained in Sect.3.5.1. This method can be easily adapted for communication constraints planning. But the connectivity is not considered in that basic technique. Thus, the first contribution in this chapter is a distributed communication-aware path planner, named as Communication-aware FMM (CA-FMM), which is employed by each robot to obtain the shortest path within coverage area, described in Sect.3.5.2. Using CA-FMM, the robots are deployed as independent agents which use the signal of other robots to communicate with the BS. So, as a second contribution, we present a centralized deployment planner to coordinate the team deployment. It is based on the relay positions computation method, developed in the previous chapter. It obtains the possible connections for the team, the sequential coverage enhancement, and the optimal positions for the relays. Here, we extend this work, improving the relay task assignment and taking into account the difficulty to reach the goals in presence of obstacles. Furthermore, we employ a more accurate signal propagation model to obtain the coverage area, described in Sect.3.4. The joint use of CA-FMM with the deployment planner, is called Deployment Planning FMM (DP-FMM) and is described in Sect.3.6.1. If the number of available robots is equal to the number of goals (tasks) to be reached, the method ensures that the mission is executed in minimum time under conditions specified in Sect.3.6.2. But the objective might be to use the minimum number of robots as relays. For this case, the last contribution of this work is a centralized clustering algorithm, which allocates several goals to each robot and computes their optimal visit order. The use of this procedure with the previous DP-FMM, is named Deployment Planning and Allocation FMM (DPA-FMM), presented in Sect.3.6.2.

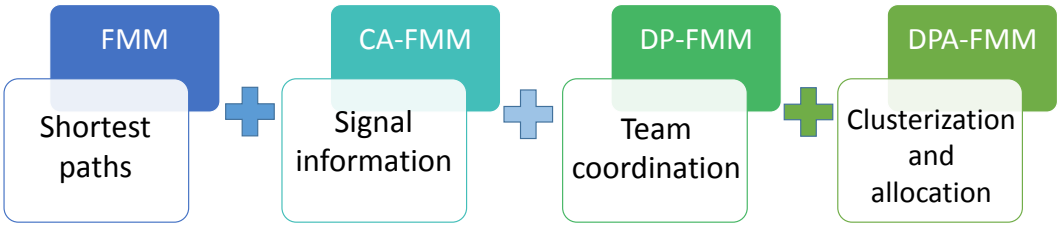


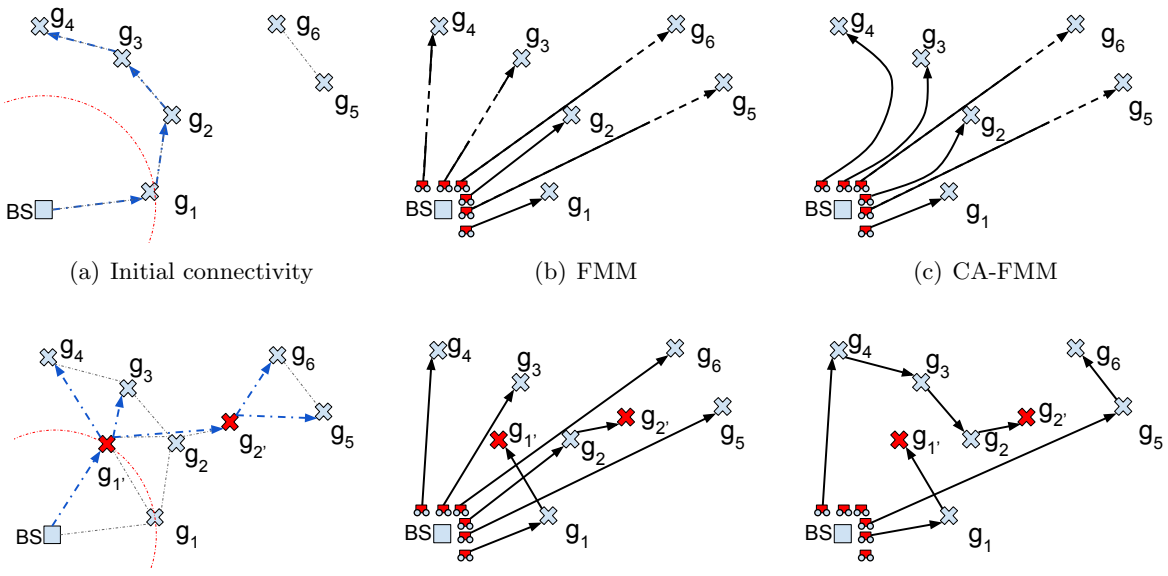
Figure 3.1: Sequence of algorithms

Each of the presented algorithms is an extension of the previous one, see Fig.3.1. A simple example of the complete procedure is depicted in Fig.3.2. After obtaining the initial connectivity in Fig.3.2(a), only g_{1-4} are connected with the BS, and can be reached by placing robots as relays at these positions. In Fig.3.2(b), FMM computes the shortest paths without taking into account the connectivity, so four goals are not reachable with connectivity. In Fig.3.2(c), CA-FMM obtains larger paths to g_{1-4} deviating the robots towards coverage areas, but the goals $g_{4,5}$ are not reachable with communication. In Fig.3.2(d), the deployment planner computes new goal positions for relay tasks $g_{1',2'}$, to improve connectivity. These

positions are the minimal to cover all the goals, as well as those that involve minimal displacement for the robots. Thus, after visiting g_1, the robot is considered free and it moves to serve as relay in $g_{1'}$. From this position it provides connectivity to $g_{2,3,4}$, thus the number of relays is reduced, and the shortest paths are within the coverage area. Likewise, after reaching g_2, the robot moves to $g_{2'}$ to provide connectivity to the teammates that will visit $g_{5,6}$. DP-FMM employs all the robots in order to minimize the mission time, Fig.3.2(e). In Fig.3.2(f), DPA-FMM classifies the goals into 3 clusters $\{g_{4,3,2,2'}\}$,$\{g_{1,1'}\}$,$\{g_{5,6}\}$, so uses only 3 robots to accomplish the mission, at the expense of increasing the mission time. The presented algorithms are also able to reactively respond to changes in the signal strength or obstacles in the environment. Metrics for the possible deployments, obtained with the presented algorithms, are presented in Sect.3.7.

3.3.2 Communication-aware paths

Consider an autonomous robot, with x_r denoting its position. The mission of the robot is to reach a goal, located at x_g. There are some signal sources or transmitters present in the environment, whose coverage area is the set of positions where it is possible to establish a connectivity link, defined as $\mathbf{x}_c : \{\mathbf{x} \mid \gamma(\mathbf{x}_{tx}, \mathbf{x}) \geq \gamma_{th}\}$, where $\gamma(\mathbf{x}_{tx}, \mathbf{x})$ is the RSS between the transmitters $\mathbf{x}_{tx}$ and the points of the environment $\mathbf{x}$, and γ_{th} is the RSS threshold to



(a) Initial connectivity (b) FMM (c) CA-FMM

(d) Connectivity improvement (e) DP-FMM (f) DPA-FMM

Figure 3.2: Proposed deployment methods to visit 6 goals. In (a), the coverage area of BS is depicted with red circle, black dashed lines are the possible connections or connectivity graph $\mathcal{G}$, and blue arrows the chosen ones or connectivity tree $\mathcal{T}$. The paths obtained by FMM and CA-FMM are represented in (b) and (c); the dashed lines represent the stretch of the path travelled without connectivity with the rest of the team. In (d), the deployment planner computes new positions (red crosses) to improve the connectivity. The paths obtained by DP-FMM and DPA-FMM are depicted in (e) and (f), respectively.

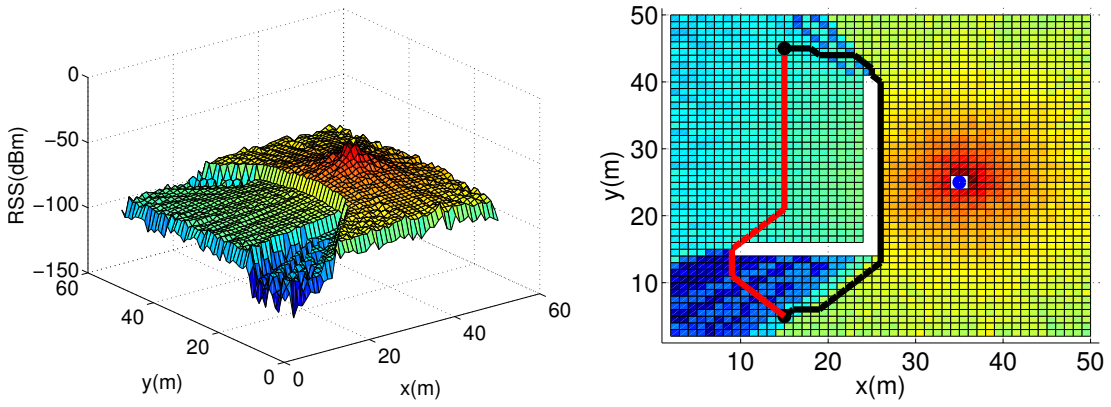
assure communication. Let us define π as a possible path from the initial position of the robot x_r to the goal x_g. The standard FMM formulation provides the shortest path based in a wavefront propagation in all the directions using a propagation velocity F. The proposed technique modifies the velocity function F in order to deviate the wavefront to areas within signal coverage areas. The result applying this new method, called CA-FMM, is depicted in Fig.3.3 and explained in detail in Sect.3.5. The path solution is not the shortest, but the one that avoids the obstacles and leads the robot through the communicated area.

3.3.3 Communication-aware deployment

The planning of the connectivity during the deployment, was developed in Chapter 2. It consists in computing the relay positions and building a connectivity tree of the goal positions. Let us briefly describe the procedure again in the context of the work of this chapter.

Consider a team of N robots with $\mathbf{x}_r$ denoting their positions, $\mathbf{x}_{r_i} := (x_{r_0}, ..., x_{r_N})$. The team includes a static base station (BS), indexed with $i = 0$, which is able to communicate with the robots. The mission of the team is to visit N goals with $\mathbf{x}_g$ representing their locations, $\mathbf{x}_{g_i} := (x_{g_0}, ..., x_{g_N})$. Therefore, the team is self-coordinated by allocating every robot i to every goal i. So the path between each pair of x_{g_i} and x_{r_i} is expressed as π_i.

Instead of a prior role assignment for the robots to primary and relay tasks, the team automatically distributes these roles. So all the robots can be used for both types of tasks. While the robots are moving they expand the coverage area within the environment. So it is necessary to compute the goals visit order in order to know which robots are going to provide communication to others. We can define a graph, expressed as $\mathcal{G}(\mathbf{x}_g)$, whose vertexes are the goal positions and the edges are the connectivity links between the goals positions. We consider that it exists a connection between positions x_i and x_j when $\gamma(x_i, x_j) \geq \gamma_{th}$ and $\gamma(x_j, x_i) \geq \gamma_{th}$. All those goals that are not connected to the BS, directly or through others,



(a) Signal propagation

(b) Obtained paths

Figure 3.3: Example of communication-aware paths. The signal propagated from a BS in $(35, 25)$ is depicted in (a). In (b), x_r and x_g are depicted with black circles, the position of BS is represented with blue circle. Considering only the distance, FMM obtains the shortest path, depicted as a red line. CA-FMM computes a larger path, depicted as a black line, that maximizes the stretch within the coverage area $\mathbf{x}_c$.

are not reachable with communication. The team should dedicate the minimum number of robots for relay tasks, so that greater number of robots are visiting primary goals. Therefore, given a graph $\mathcal{G}(\mathbf{x}_g)$, we compute a tree $\mathcal{T}(\mathbf{x}_g)$, which minimizes the number of hops, or depth in the tree, from the root (BS) to every node. The number of hops to some position x_{g_i} indicates the number of relay robots needed to reach this position with connectivity, and is expressed as $n_r(x_{g_i})$.

Only after visiting the primary goal, a robot is considered free and can change the position to enhance $\mathbf{x_c}$. Hence, each robot attempts to find a new position where the maximum number of goals will be in its coverage area. In other words, the robot will provide communication to goals which were connected to another relay, thus the number of hops to these goals is reduced. At the same time, this procedure is applied to disconnected goals, in order to complete the mission with connectivity to all the goals when the robots reach them.

An example of this method is shown in Fig.3.2(a) and 3.2(d). The connections of $\mathcal{G}(\mathbf{x}_g)$ are depicted with black dashed lines, and $\mathcal{T}(\mathbf{x}_g)$ is represented with blue arrows. In Fig.3.2(d), after visiting g_1, a robot will move to $g_{1'}$, increasing the coverage area. In consequence, the number of required relays to reach $g_{3,4}$ will be reduced with respect to the situation in Fig.3.2(a). The new position $g_{2'}$ will be used to connect the previously disconnected goals, $g_{5,6}$. As a result, the minimum number of relays locations are used for the mission and all the goals are achievable with connectivity.

3.3.4 Clustering

The number of the available robots for the mission initially corresponds to the number of goals, so the fastest way to complete the mission is using all the robots. However, it is not always necessary, or possible, to employ the entire team. Thus, in this chapter we study the minimization of the number of employed robots for achieving the proposed mission, that will be used to visit several goals. Firstly, the deployment planner obtains the relay locations, and allocates them to the robots. Then, the robots used for this purpose should remain in these positions, and the rest of the team can be freely used to reach other goals.

The first step is to find all those goals which can be visited by the same robot. This process of clustering is made depending on three factors: the initial point, the deviation distance from the straight path between goals, and the occupation at these positions. This can be observed in Fig.3.2(d) and 3.2(f). Possible destinations for the robots are $g_{1'}$ and $g_{2'}$, where they are occupied as relays in Fig.3.2(d). The rest of the goals only must be visited. In other words, these goals can be considered as waypoints to other destinations. Thus, the algorithm splits the goals into three different clusters, which will be visited by three robots. Once the clusters are obtained, each robot has multiple possibilities to visit the goals. Consequently, the goals visit sequence is computed in order to minimize the travelled distance. Fig.3.2(f) depicts the order that the robots follow for each cluster, visiting the goals. When the plan is already obtained, the path travelled by every robot is known by the rest of the teammates. So that, when a robot is aware that its relay will be delayed, it waits until the communication area is extended to its goal. For instance, in Fig.3.2(f), the robot which visits g_{5-6} waits until the robot, which visits $g_{4-3-2-2'}$, reaches the last position, where provides coverage to g_{5-6}. This way, the travelled distance is minimized and at the moment of visiting the goals, the robots are connected with BS. The detailed procedure of clustering is explained in Sect.3.6.2.

3.4 Signal propagation

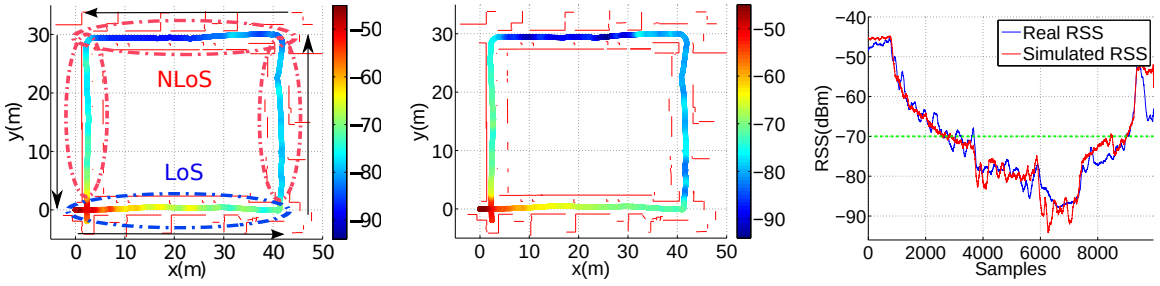
In this work we use a more accurate signal propagation model, instead of the multi-wall model employed in the previous chapter. Every signal propagation depends on three main factors: the attenuation due to the distance, the shadowing because of obstacle traversing, and multipath effect, due to the reflections of the signal. Thus, the RSS is obtained subtracting the path loss to the emitted power by the antenna. The path loss suffered by the signal and the received power can be defined as [33]:

$$L(x_i, x_j) = L_o + 10\mu log_{10}(d(x_i, x_j)) + S + M, \qquad \gamma(x_i, x_j) = \gamma(x_i) - L(x_i, x_j) \tag{3.1}$$

where L_o is the path loss at distance of 1m, μ is the path-loss exponent, $d(x_i, x_j)$ is the distance between transmitter x_i and receiver x_j, S represents the fading due to the shadowing and M the fading due to the multipath effect. The shadowing can be obtained as $S = n_w * a_w$, where n_w is the number of traversed walls and a_w is the attenuation per traversed wall.

The attenuation and shadowing are easily predictable, but the multipath effect is computationally intractable using multiple mobile transmitters. Some advanced techniques, as ray-tracing, can be employed to obtain accurate approximations of the signal, but it is out of the scope of present work. Thus, we propose an approximation of the signal, which considers the main features of the signal propagation. In [38], the authors average the real RSS to reduce the small variations of the signal, in order model the spatial distribution of the connectivity area of a single BS. In our approach, we also extract the main features of the averaged signal obtained from real world, in order to accurately simulate the coverage areas for path planning.

Our algorithm is intended to use in indoor scenarios, emulating some commercial technology, as WiFi. We have validated the model from the real signal data collected in a building at the University of Zaragoza. We consider LoS and NLoS components of the signal in order to increase the precision of our simulations. The building map and the collected signal data are depicted in Fig.3.4(a). In that scenario, the inner obstacles are glass windows and the exterior obstacles are concrete walls. From [18] we extract $\gamma_{th} = -70$dBm, in order



(a) Real RSS (dBm) (b) Simulated RSS (dBm) (c) Comparison

Figure 3.4: RSS in a real environment. In (a), it is represented the collected data, the transmitter is located at (0,0), black arrows depict the walk through of the robot, LoS and NLoS areas surrounded with blue and red dashed lines, respectively. The simulated RSS, with eq.3.1, is depicted in (b). The comparison between smoothed real and simulated RSS is represented in (c).

to assure a package transmission between the agents. We take the values for shadowing of 10dB/wall and 2.5dB/glass plate, from [33]. The used path-loss exponents are $\mu_{LoS} = 1.7$ and $\mu_{NLoS} = 1.4$; and multipath variances are $\sigma^2_{LoS} = 3.45$ and $\sigma^2_{NLoS} = 3.25$.

Firstly, we use the moving average to smooth the real signal, depicted with blue line in Fig.3.4(c). Secondly, we subtract the average signal to raw data, obtaining the fast fadings of the signal produced by multipath effect. Extracting the variance, it is possible to approximate the multipath component by a Gaussian distribution $\mathcal{N}(0, \sigma^2_{mp})$. As we have the RSS and the distance to the emitter, the path-loss exponent μ can be computed employing polynomial regression. Therefore, the polynomial coefficients are obtained with least-squares fit. The simulated signal, using eq.3.1, is depicted in Fig.3.4(c). As we can observe, the simulation captures the main shape of the real signal. This signal model and the adjusted parameters have been used to obtain the connectivity between the teammates, by the deployment planner, as well as by the path planner developed in the following section.

3.5 Communication-aware path planning

In this section we develop the method to find paths leading to the goals with maximum connectivity, outlined in Sect.3.3.2. We briefly describe the FMM and present our CA-FMM approach.

3.5.1 Fast Marching Method (FMM) for path planning

The Fast Marching Method (FMM) was proposed as an approximated solution to Eikonal equation by Sethian in [37]. It consists in the computation of a distance function D, from some source point for every point of the grid $\mathbf{x}$, obtaining as solution the minimum cost to reach these points. Thus, a wavefront is propagated over all the grid, computing the distance function D, with the metric F representing the speed of the wavefront propagation. The variable F contains the obstacle information, so that $F(\mathbf{x}) = 0$ for all the positions of $\mathbf{x}$ which contain obstacles and $F(\mathbf{x}) = 1$ for free space. Therefore, the wave is initialized at goal x_{g_i}, assigning $D(x_{g_i}) = 0$, and it propagates uniformly in all the directions, computing:

$$|\nabla D| F(\mathbf{x}) = 1 \tag{3.2}$$

The FMM can be interpreted as a continuous Dijkstra method. The main advantage is that FMM uses the values of two neighbours to interpolate the distance, instead of one, used by Dijkstra. Therefore, FMM obtains more accurate approximations of distance. As Dijkstra, FMM computes the shortest paths to the goals and not suboptimal solutions, as provided by randomized algorithms, such as different versions of RRTs (Rapidly exploring Random Trees). In the present work we have used the Fast Marching Method for path planning for the agents in two dimensional scenarios. However, the FMM it is also employed for multiple purposes in different research topics. The most popular is the application of FMM in image segmentation [39].

The algorithm of the wavefront propagation and the distance approximation of the FMM are described in detail in the Appendix A. In this part we employ FMM only for path planning, thus the wavefront is extended just from the agent position to the goal, using eq.A.3, in order to do not waste time for computation.

3.5.2 Communication-aware FMM (CA-FMM)

Since the velocity function F contains only obstacles information, the obtained gradient ∇D, represents the Euclidean distance to the goal. However, our approach must include the signal information in order to take it into account for the path planning. To this end, we modify the wavefront velocity in the areas with signal coverage. So that, the gradient will deviate the paths to areas with connectivity. The idea of changing the wavefront velocity to lead the propagation to some areas of interest was originally proposed also by Sethian in [40]. In that case the wavefront speed F was changed for tracking moving interfaces.

Therefore, we modify the wavefront velocity, including the signal information in the velocity F. This new speed with the communication factor is denoted as $F_c(\mathbf{x})$ and it is obtained as:

$$F_c(\mathbf{x}) = F(\mathbf{x}) + f(\mathbf{x}_c) \tag{3.3}$$

where the function $f(\mathbf{x}_c)$ normalizes the signal of the coverage area and fix to zero the positions of the other robots, considered as obstacles. The new velocity function F_c, is used in eq.3.2 to obtain gradients with a new metric, depending on the distance and communications. The wavefront is propagated faster in areas with higher values of F_c, which correspond to the coverage area. Therefore, the lower values of the gradient of distance function ∇D, computed using the new metric F_c, will find shorter paths within coverage area.

As the signal sources are mobile, the CA-FMM (Alg.4) takes into account the movement of the robots and the variations of the signal. Firstly, the algorithm obtains the coverage area $\mathbf{x}_c$, in l.2, and this information is inserted into F_c in l.3. The coverage area of is computed employing eq.3.1. Although the used signal model is defined by eq.3.1, any other model can be used or even the stored real signal data, in the case of the execution in real world. Then, in l.4, the wavefront is propagated to the relay goal positions with every movement of the relays. The algorithm iterates until reaching the goal position, using the coverage of already deployed robots and, finally, it builds the path in l.6, descending the gradient.

3.6 Deployment planning

In the present section we present our method to assign coverage tasks to each robot devoted to do it, and to compute the clusters and the sequence of visit of the primary goals in each of them.

Algorithm 4 CA-FMM (for one robot reaching one goal)

Require: Robot position x_{r_i}, Goal position x_{g_i}, Paths π_l (travelled by the relays)

1: **for each** $\mathbf{x}_{r_l} \in \pi_l$ **do**
2: $\quad \mathbf{x}_c = coverage\_area(\mathbf{x}, \mathbf{x}_{r_l}, \gamma_{th}, n)$ ▷ Using eq.3.1
3: $\quad F_c \leftarrow compute(\mathbf{x}_c)$ ▷ Using eq.3.3
4: $\quad \nabla D \leftarrow compute\_gradient(x_{g_i}, \mathbf{x}_{r_l}, F_c)$ ▷ Using Alg.21
5: **end for**
6: $\pi_i \leftarrow gradient\_descent(\nabla D)$
7: **return** Path π_i

3.6.1 Sequential deployment and coverage enhancement (DP-FMM)

When some robot is going towards its goal, it modifies the coverage area, providing signal to other robots. As explained in Sect.3.3.3, the connectivity tree $\mathcal{T}(\mathbf{x}_g)$ provides the connections between the goals and the depth of each one. Hence, the order of goal visitation is obtained to ensure that the coverage is sequentially enhanced. Besides the sequence, we are able to know which robot must remain at the same position providing connectivity. This process is illustrated in Fig.3.2(a). When a robot visits the position of g_1 it it extends the communication area of the BS to g_2 and so on. Until a robot reaches g_4, the relays must remain at the positions g_{1-3}. Once the primary goals are visited, the deployment planner computes new goal locations where the robots, serving as relays, improve the connectivity of the team in terms of distance and required relays. The number of these new goal positions is minimum, thus it involves the minimum number of robots devoted to relay tasks. Simultaneously, the tasks are allocated to those agents, which require minimal displacement to relay positions. From Fig.3.2(a), we extract that the approximated distance to g_4 is $d(\pi(x_{g_4})) = d_{01} + d_{12} + d_{23} + d_{34}$ and the depth is $n_r(x_{g_4}) = 4$. In Fig.3.2(d), after reaching g_1, the robot moves to $g_{1'}$. So the new distance to goal g_4 is $d(\pi(x_{g_4})) = d_{04}$, considerably smaller than initially, and $n_r(x_{g_4}) = 2$. At the same time, the second robot enhances the coverage area at $g_{2'}$, connecting g_5 and g_6.

The deployment planner (DP) assures the connectivity to all the goals if the condition $d(\pi_f)/d_{cov} \leq N$ is accomplished, where π_f is the minimum distance path from the BS to the farthest goal (in the sense of the shortest path computed, for instance, by an A^* algorithm), d_{cov} is the minimum coverage distance where the connectivity with a robot is always assured, and N is the number of robots. Note that in the worst case the robots form a chain, up to the farthest goal. So under that condition all the goals will be always reached. The chain can be obtained using the chain planner of the next Chapter 4.

In the previous chapter, the *idle* agents were allocated to visit the obtained relay goals. That is, the robots which were not used for relay tasks in that moment. As a result, the nearby relay robots were automatically discarded, even being optimal for these tasks. Now, we use the distance as the cost to reach some goal. In addition, we incorporate a penalty distance which corresponds to the difficulty to surround the obstacles between the robot and the goal. Therefore, the cost to go to some position j from position i is:

$$c(x_i, x_j) = d(x_i, x_j)(1 + n_o(x_i, x_j)) \tag{3.4}$$

where d is the Euclidean distance and n_o represents the number of obstacles in between x_i and x_j. More sophisticated cost functions could be used for taking into account obstacle characteristics, as size, but it does not change the essence of the basic proposed algorithm.

The Hungarian algorithm [41] is used to allocate the tasks according to the computed costs. It consists in the allocations of a set of works to a set of workers, based on the cost of the works. With this method, the sum of the costs of the allocation is the minimal. In this case, the costs are represented by the distances between the goals positions and the agents, obtained with eq.3.4.

The deployment planning algorithm working altogether with the previously described, CA-FMM, is named as DP-FMM. At first, the centralized deployment planner obtains the connectivity tree, the list of first goals to visit and the new positions for relay tasks. This information is shared with the robots. Each robot computes the path using the distributed

CA-FMM, sharing this information with the team. Note that in the first iteration, the coverage area is provided only by BS. The robots use the coverage area of their mates in each iteration. The algorithm iterates until obtaining the paths to all the goals, as illustrated in Fig. 3.2(d)-3.2(e).

3.6.2 Clustering and visit sequence (DPA-FMM)

The previous algorithm can minimize the mission time, because all the robots of the team are used, planning the shortest paths within the coverage areas. By contrast, we could want to minimize the number of robots required to complete the mission, instead of minimizing the mission time, as described in Sect.3.3.4. As a result, the number of goals visited by each robot is maximized. To this end, it is necessary to compute which goals should visit each robot. Thus, the goals are clustered according to different parameters and each cluster will be visited by only one robot, minimizing the total number of used robots.

We are going to group the goals that are connected to the same relay, where some robot is providing signal to the rest of the goal positions. The robots will come from this direction, following the trace of the signal, so we consider it as the starting point, g_0 in Fig.3.5. The number of clusters is determined by the number of relay goals to achieve. That is, if there are two goals where the robots will have to remain providing connectivity (relay goals), at least two robots are required, and these goals are the destinations of those robots, $g_{1,2}$ in the figure. All those goals that are not used for relay tasks are considered waypoints for one robot to a destination, g_{3-5}. The waypoints included in that cluster, will be those that will provide the smallest deviation distance with respect to the direct path to the destination point of this cluster. Mathematically, the clustering process may be expressed as:

$$s_i = \{x_p : (c_{lp} + c_{pi}) - c_{li} < (c_{lp} + c_{pj}) - c_{lj},\ \forall j,\ 1 \geq j \geq k\}, \tag{3.5}$$

where c represents the cost computed with eq.3.4, k is the number of destinations, i and j represent two possible destinations, p is a waypoint and l is the link of p and k.

After obtaining the clusters, it is necessary to assign the best order to visit the waypoints. Given the limited signal range of the relays, the dispersion of the goals and the existence of several destination points, the clustering process greatly reduces the computational cost in the process of obtaining the visit sequence. Hence, we assess all the possible combinations of visit order of the waypoints from starting point to the destination in each cluster. The optimal sequence is that, which provides the shortest distance, depicted with green arrows in Fig.3.5. The distance cost between every pair of points is also computed using eq.3.4.

The DP-FMM including the goal allocation receives the name of DPA-FMM and the procedure is shown in Alg.5. At each iteration the deployment planner obtains the goals to visit and the new positions for relay tasks (l.1). The goals are assigned to each robot (l.4) and this information is shared with the robots. Each robot obtains the best visit order, computes its path using CA-FMM (l.5-9) and shares this information with the team. This solution minimizes the number of robots employed for the mission, as well as the total distance travelled by the team.

While the team is executing the mission, the robots can detect variations of the environment, as well as handle with variation of the signal. Therefore, our planner is used to replan in case of detecting changes in the scenario, reactively solving this kind of situations. The only difference is that the robots which have reached their goals are extracted from the

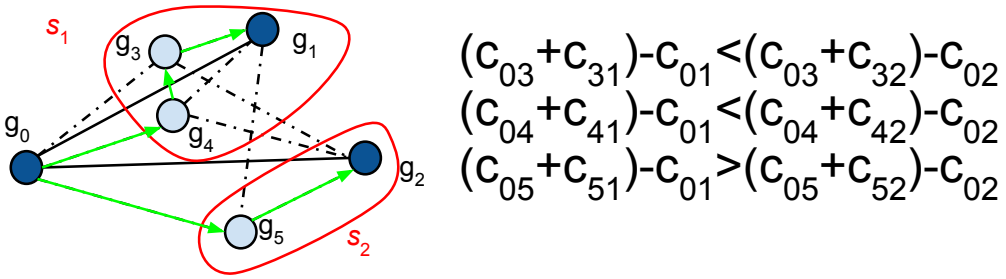


Figure 3.5: Clustering and goal assignment procedure for 2 robots visiting 5 goals, g_{1-5}. The visit sequence for each cluster s_1, s_2 is depicted with green arrows.

Algorithm 5 DPA-FMM

Require: Robot positions $\mathbf{x}_r$, Goal positions $\mathbf{x}_g$, Reached goals $\mathbf{f}$

1: $\mathcal{T}(\mathbf{x}_g) \leftarrow DP(\mathbf{x}_g)$ ▷ Optimal connectivity tree using Alg.2
2: **for** $dp = 1, .., max(n_r(\mathbf{x}_g))$ **do**
3: $\mathbf{x}_g(dp) \leftarrow \mathbf{x}_g(dp) \setminus \mathbf{x}_g(\mathbf{f})$ ▷ $\mathbf{x}_g(\mathbf{f})$ represents already planned goals
4: $\mathbf{S} \leftarrow cluster(\mathbf{x}_g(dp))$ ▷ $\mathbf{S}$ represents the clusters of the same tree depth
5: **for each** $s_i \in \mathbf{S}$ **do** ▷ $s_i : \{x_{r_i}, x_{g_{w_1}}, ..., x_{g_{dest_i}}\}$, robot position, waypoints, destination
6: **for each** $x_j \in s_i$ **do**
7: $\pi_i \leftarrow CA - FMM(x_j, x_{j+1}, \pi_{r_l})$ ▷ r_l is the relay of the goals of s_i
8: **end for**
9: **end for**
10: $\mathcal{T}(\mathbf{x}_g) \leftarrow DP(\mathbf{x}_g)$ ▷ tree update for new relay positions
11: **end forreturn** Paths π

list of goals to visit (l.3), being the planner DPA-FMM re-launched again.

Every robot computes the shortest path under communication constraints, as defined in Sect.3.3.2. So that, the distance of the obtained path is considered the minimum. The farthest goal location constraints the time of the mission. Therefore, the condition to execute the mission in minimum time, is to use all the robots, thus each robot reaches its own goal. In the Fig.3.2, g_6 is the restrictive goal. As DP-FMM considers all the robots for the deployment, the distance is the direct distance to this goal, $d(\pi(x_{g6})) = d_{06}$ in Fig.3.2(e). Employing DPA-FMM the robots visit several goals. From Fig.3.2(f), we extract $d(\pi(x_{g6})) = d_{05} + d_{56}$. Clearly, the time is increased with respect to DP-FMM solution, at the expense of reducing the number of robots.

3.7 Results

We have evaluated the method by means of simulations. We test the four described algorithms for possible deployment missions: FMM, CA-FMM, DP-FMM and DPA-FMM. The results are evaluated according to different metrics: travelled distance (d), mission time (T), connectivity (C), and robot occupation during the mission (O). The last two metrics correspond to communication of the team. The connectivity represents the time which all

the team remains connected to BS. The occupation represents the time employed by every robot to provide connectivity to the rest of the team. The results of time, connectivity and occupation are normalized to the maximum value obtained by all the algorithms. C_{mean} and O_{mean} represent the average time in which a robot is connected and employed as relay, respectively. The worst cases of distance and connectivity are d_{max} and C_{min}, respectively. d_{tot} is the sum of travelled distances by the entire team.

An example of the deployments obtained by each algorithm is depicted in Fig.3.6, and numerical results are presented in Fig.3.7 and Table 3.1. The reader can find a video in the link[1]with the simulation of the proposed methods in this chapter.

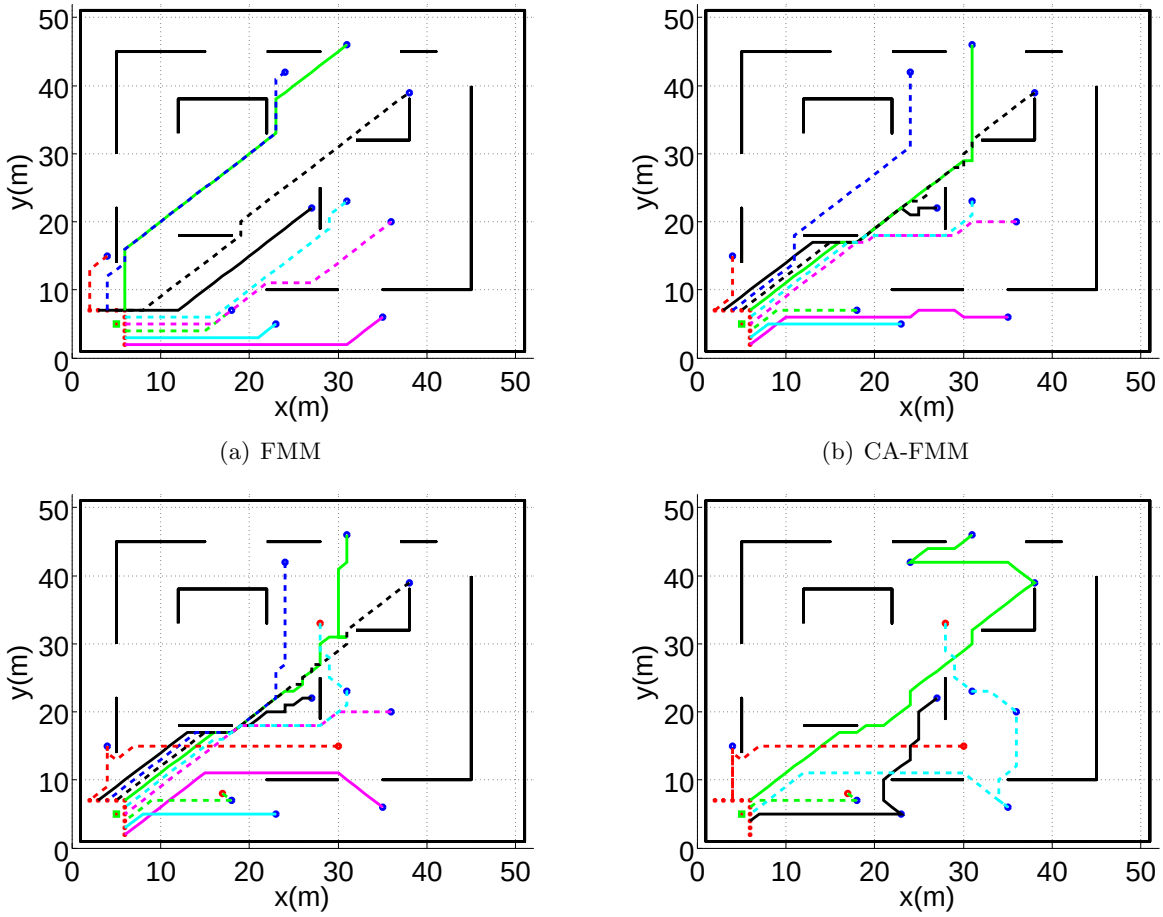
We use an Hungarian algorithm for task allocation in FMM and CA-FMM, where the costs are computed by eq.3.4. FMM obtains the shortest paths, because it considers only the distance for planning, Fig.3.6(a). CA-FMM deviates the robots to areas where connectivity is maximized, Fig.3.6(b), although it is not able to finish the mission reaching all the goals with connectivity. In Fig.3.6(c), DP-FMM computes and allocates the new positions for relay tasks, thus the mission is accomplished with complete connectivity for the whole team. This involves a delay of 6%, and 14.7% of extra travelled distance (Table.3.1) with respect to the basic solution without connectivity issues (FMM). Employing the complete planner DPA-FMM, the mission is accomplished reducing 23% the total distance travelled by the team. As a result, the longest distance travelled by a robot, which delimits the time of the mission, is increased in 31% with respect to the FMM algorithm, but the number of used robots is strongly reduced, 5 instead of 10. Moreover, some robots are temporarily disconnected from the rest of the team during the motion. This occurs when a robot visits several goals and considerably deviates from the direct path to its relay. Consequently, the robots that were connected to this robot are disconnected too. This situation is shown in Fig.3.6(d), losing the connectivity only during 3% of time, Fig.3.7(c).

The method allows disconnections during the motion between the goals, but the robots always establish a connectivity link with the BS (see the video). Due to the deployment planner, both, DP-FMM and DPA-FMM better allocate the relay tasks. Therefore, the occupation is concentrated in robots which are the best for relay tasks.

The planner is able to resolve situations when some variations in the scenario are produced during the deployment. If some robot of the team detects new obstacles, the signal changes and the planner re-plans new paths, resolving reactively these changing situations. The new paths are depicted in Fig.3.8 and the results are shown in Table.3.2. The distances and the time increase, and the connectivity, particularly C_{min}, is reduced. In case of FMM, C_{mean} increases because of the obstacles, the agents are unintentionally deviated to coverage areas. Now, DP-FMM loses 1% of connectivity in areas between goals, and DPA-FMM employs an extra robot with respect to the initial plan.

Table 3.3 depicts the average results for 50 random trials, in missions to visit 15, 20 and 30 randomly distributed goals. The number of employed robots for the mission is denoted with R. We can observe that the results follow the trend of the Table 3.1. Using the complete algorithm DPA-FMM, the mission is accomplished without employing the entire team, reducing the ratio robots-goals; and travelling shorter distance d_{tot}. But in exchange, the time of the mission is increased. The worst cases of connectivity for DP-FMM and DPA-FMM, are always above 80% of time, considered assumable. In absence of deployment

[1]http://robots.unizar.es/data/videos/ifac17yamar.mp4



(a) FMM (b) CA-FMM

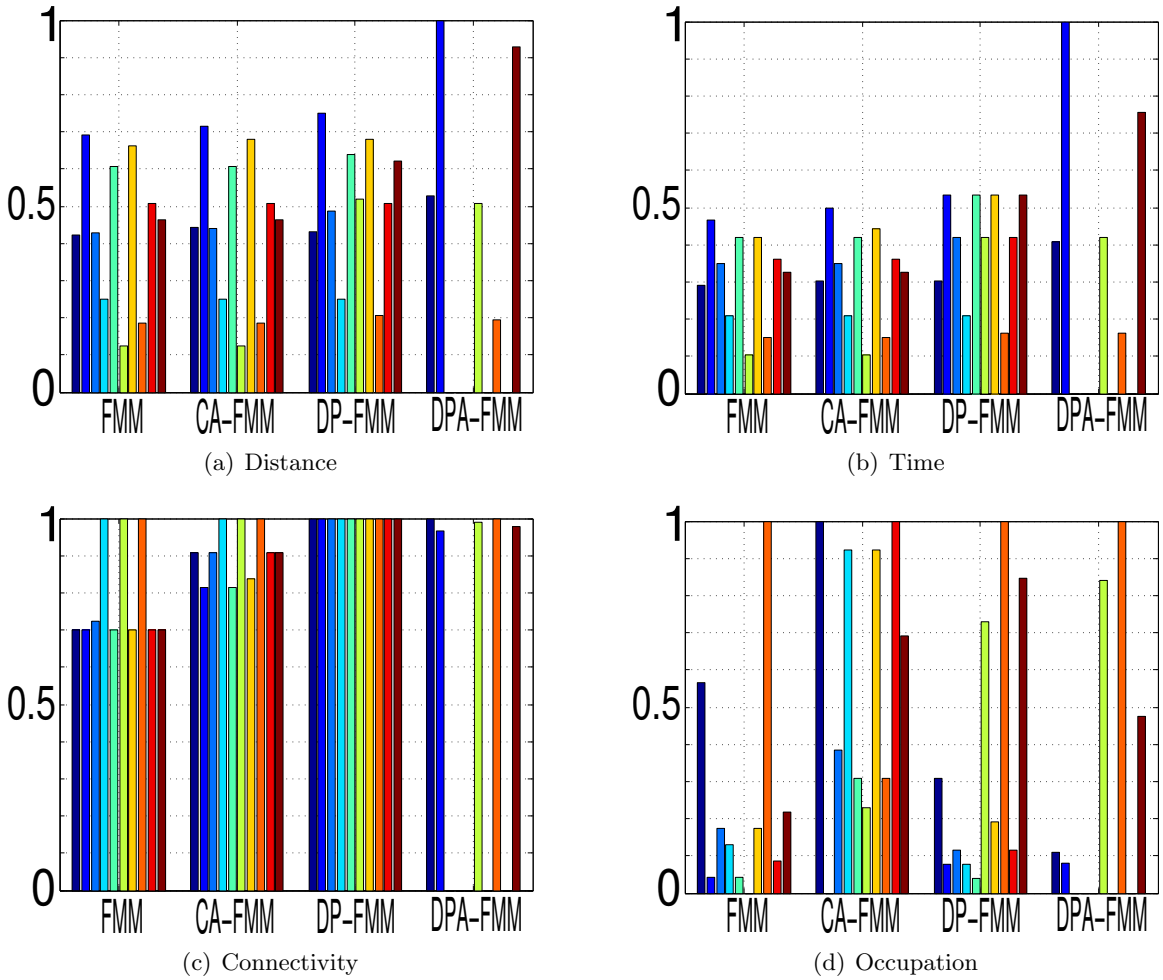
(c) DP-FMM (d) DPA-FMM

Figure 3.6: Possible deployments visiting 10 randomly distributed goals. The paths obtained by each algorithm are illustrated in (a)-(d). Green square depicts the BS, red crosses represent the initial robot positions, blue and red circles are the primary and relay goals, respectively. DPA-FMM requires only 5 robots for the mission.

Table 3.1: Obtained results for the scenario represented in Fig.3.6.

	d_{max}	d_{tot}	T	C_{mean}	C_{min}	O_{mean}
FMM	49.36	310.26	0.47	0.79	0.70	0.24
CA-FMM	51.11	315.43	0.50	0.91	0.84	0.53
DP-FMM	53.70	363.75	0.53	1	1	0.33
DPA-FMM	71.50	225.58	1	0.99	0.97	0.50

planner, CA-FMM may obtain worse connectivity than for FMM, as for 20 robots case. This occurs when a robot is using the coverage area of an independent relay, which is accomplishing its own task. Thus, it is deviated from the goal to finally travel the remaining path without connectivity.



(a) Distance (b) Time

(c) Connectivity (d) Occupation

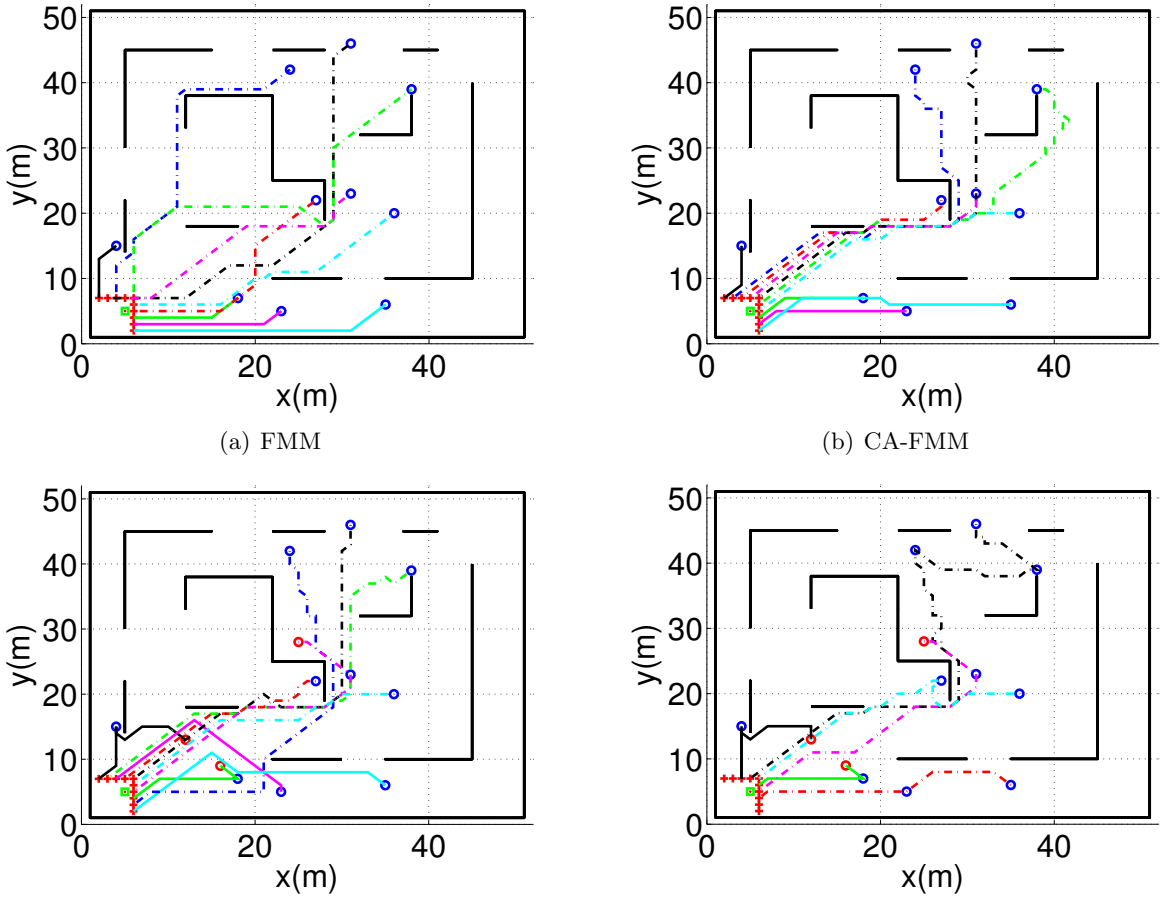
Figure 3.7: Normalized results of the possible deployments of Fig.3.6. Since DPA-FMM employs 5 robots for the mission, thus only 5 bars are depicted.

Table 3.2: Obtained mean results for the scenario represented in Fig.3.8. Now DPA-FMM requires to use 6 robots instead of 5 of the initial plan.

	d_{max}	d_{tot}	T	C_{mean}	C_{min}	O_{mean}
FMM	59.46	338.52	0.74	0.92	0.53	0.42
CA-FMM	59.18	345.63	0.72	0.84	0.42	0.49
DP-FMM	58.8	379.06	0.74	0.99	0.98	0.45
DPA-FMM	83.77	234.31	1	0.93	0.65	0.57

3.8 Conclusions

In this chapter we have presented a method to plan the deployment of a team of mobile robots, to visit some locations of interest in an environment with obstacles and under communication constraints, using some robots in role of relay. The team communicates with a static base station, at the moment of visiting the goals. The plan is obtained previously to deploy the



(a) FMM (b) CA-FMM

(c) DP-FMM (d) DPA-FMM

Figure 3.8: New deployments after changes in the environment. During the deployment, a new obstacle is detected. The way is now blocked at the center of the scenario, and the signal also changes. The planner is re-launched from this new situation to achieve the rest of the tasks.

team, using only a map of the environment and a signal propagation model. The presented method is composed by three algorithms: a communication-aware path planner (CA-FMM), a deployment planner (DP-FMM), and a robot and goals clustering algorithm (DPA-FMM). With DP-FMM, the team fulfills the mission employing all the robots, so that reducing the time. In contrast, DPA-FMM allocates several goals per robot, so the amount of required members for the mission is reduced.

The proposed approach is able to obtain a plan for the deployment of small teams of robots and goals. Being both of them up to 30 agents and goals. In the next Chapter 4, we develop a method in order to obtain a solution in short time for large instances of robots and goals.

We have shown that the proposed method is able to reactively respond to environment or signal changes during the deployment, by replanning in the case of detecting the changes. This is possible only in the case of detecting the changes before the connectivity loss. That is,

Table 3.3: Average results visiting different number of goals N.

N	**Algorithm**	d_{max}	d_{tot}	T	C_{mean}	C_{min}	O_{mean}	R
15	**FMM**	57.7411	515.2154	0.6503	0.7492	0.4511	0.2164	15
	CA-FMM	62.5775	533.6519	0.7088	0.7365	0.4634	0.2211	15
	DP-FMM	68.2503	599.3198	0.7682	0.9827	0.8501	0.2795	15
	DPA-FMM	85.6530	488.6692	1.0000	0.9745	0.8051	0.4037	11
20	**FMM**	61.1598	714.9101	0.6927	0.8392	0.5535	0.2254	20
	CA-FMM	66.8139	746.3166	0.7453	0.8417	0.5268	0.2376	20
	DP-FMM	69.8790	819.1511	0.8029	0.9879	0.8818	0.2597	20
	DPA-FMM	78.4160	583.9208	1.0000	0.9846	0.8700	0.4115	13
30	**FMM**	61.3190	1050.02	0.6715	0.9172	0.7536	0.1907	30
	CA-FMM	66.4488	1093.3	0.7475	0.9230	0.7435	0.2063	30
	DP-FMM	69.6423	1162.5	0.7845	0.9964	0.9372	0.2104	30
	DPA-FMM	85.9442	752.3	1.0000	0.9880	0.8690	0.3803	17

if the team detects the changes at the moment of being connected. In the following Chapter 5, we develop a simple method to regroup the team in the case of loosing the connectivity due to the changes in the environment.

Chapter 4

Fast and scalable multi-robot deployment planning under connectivity constraints

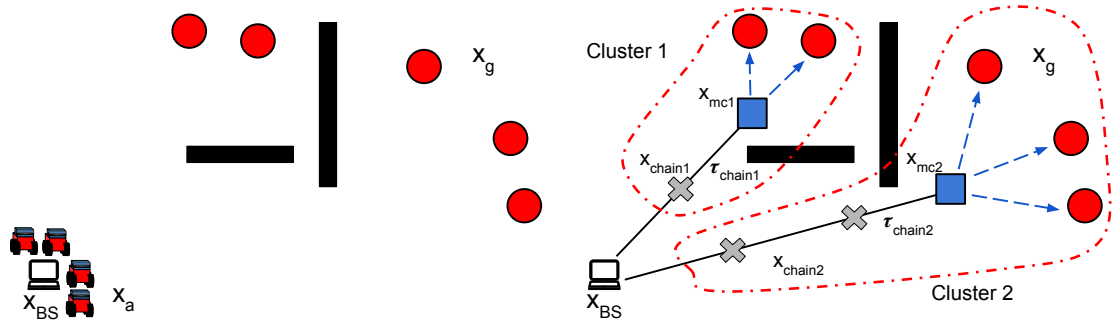
4.1 Introduction

In the previous chapters we have proposed the relay goals computation in order to establish a connectivity link with a static Base Station. The agents adopt a mobile multi-hop connectivity network, with a tree topology, in order to reach the primary goals with connectivity. Furthermore, we have proposed a communication-aware path planning method to navigate within connectivity areas. Altogether with the developed allocation approach, the agents maximize the connectivity during the mission. These sequence of algorithms was focused for missions where the number of agents is lower or equal to the number of primary goals. Furthermore, this approach is too slow due to the large number of candidates to analyze where to place the relays, when computing the tree.

In this chapter we generalize that approach, making it scalable for all the instances of agents and goals. The new method is faster in comparison to the previous approach. Moreover it may even obtain solutions in real-time for small instances of agents and goals, as will be shown in the results section. For example, for instances up to a couple of dozens of goals and for a ten agent team, the solution can be found in less than 500 milliseconds.

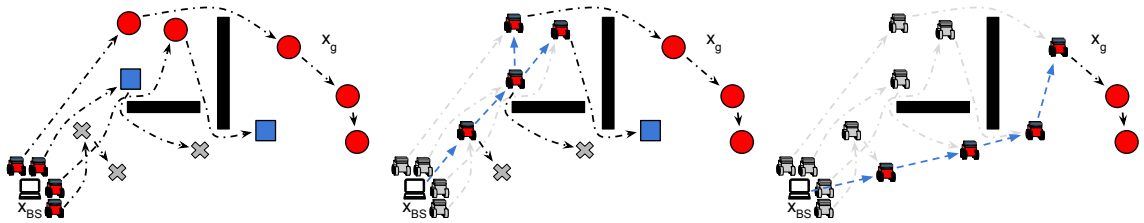
The method is also based in a sequence of algorithms. An illustrative example of the proposed approach is depicted in Fig.4.1-4.2. As in the previous chapter, the method uses the map of the environment, containing the obstacles, and the primary goals to reach, Fig.4.1(a). Firstly, it computes the relay positions to connect the primary goals with the BS, through what we call relay chains. As can be seen, the chains can be interpreted as a part of the tree, proposed in the previous chapters. However, the time to compute the chain is lower, because all of them have the same root, so there is no need to evaluate where to connect the chain of robots, since it is always the BS. The relay chains and the primary goals connected by these chains are grouped into different clusters of goals.

Here the problem of how to visit the different clusters of goals arises, Fig.4.2. In the previous chapter, the visit order was obtained based on the deviation from the destination point, being in that case the next relay position. Here, we are evaluating hundreds of points, so this kind of assessment cannot be considered due to the time limits for computation.



(a) Initial scenario (b) Relay deployment

Figure 4.1: Relay deployment illustration. In (a) the initial scenario is illustrated: the positions of the BS, the robots and the goals to reach $\mathbf{x}_g$, depicted with red circles. (b) depicts the process of relay positions computation, where two chains of relay robots must be deployed to reach all the goals. The relays to maximize the connectivity $\mathbf{x}_{mc}$ are blue squares, the links of communication are blue dashed arrows. The chain paths π_{chain} are depicted with black lines and the relays $\mathbf{x}_{chain}$ are gray crosses.



(a) (b) (c)

Figure 4.2: Order of cluster visit. (a) initial allocation, (b) visiting the first cluster, (c) visiting the second cluster. The black dashed arrows depict the allocated goals to the robots and the travelled paths. The blue dashed arrows represent the communication links between the agents.

Therefore, for the second step of our approach we propose and analyze different ways to visit the clusters, Fig.4.2(a): considering the distance to move the robots between clusters, the number of required robots, and the time to visit the goals of each cluster. It is scalable to the number of goals and agents, and fast due to the use of heuristics. The clusterization can be interpreted as a local optimization, since we firstly group the goals, and then efficiently distribute the agents between them, based on the number of goals within each cluster. And the proposed heuristics substantially reduce the time to obtain a visit order of the clusters, instead of analyzing different possible tours.

As in the previous chapters, some robots must reach the relay goals, just after that, their teammates can visit the primary goals of the cluster to send the information to the BS, depicted in Fig.4.2(b)-4.2(c).

The proposed approach requires two steps:

– *Relay chains computation and clustering* (Fig.4.1): to deploy relay chains and group them with the primary goals forming clusters, minimizing the number of robots devoted as relays. Explained in Sect.4.4.

– *Ordering the visits to the clusters* (Fig.4.2): to obtain the order to reach the positions in the clusters and the goals belonging to them, by minimizing the time spent in the cluster. Detailed in Sect.4.5.

The combination of this two-step strategy, provides a low time-consuming method, scalable to the number of robots and goals. The rest of the chapter is organized as follows. In the next Section 4.2 we review the works of the state of the art. Section 4.3 describes the proposed method. In Section 4.4 the technique to compute the positions where will be placed the relays. In section 4.5 different heuristics are proposed to obtain the sequence of visit of the clusters of goals and a method to visit the relay and primary goals within each cluster is developed. Sections 4.6 and 4.7 present the results and conclusions, respectively. The proposed technique was published in [3].

4.2 Related works

In order to generalize the proposed approach to all the amounts of goals and agents, and, at the same time, obtain a solution in minimum time, we need to lighten the calculations of the different parts of the proposed algorithms. So, the first simplification is to change the communication model. In the previous chapters we was using the multi-wall-and-floor model of [33]. The simulation of the multipath fading in order to obtain a more realistic simulated signal and the counting of the traversed walls are time consuming. Here we propose to use a more simple model where only the distance threshold altogether with the LoS are taken into account to predict the connectivity links. This kind of connectivity is a widely used model in robotics [42][26].

The relays positions computation also needs to reduce the number of possible candidates where to place the agents used in relay role. In [26], all the free positions of the scenario are considered as candidates to place the relays and using ILP, the relay placement problem is solved. In [42], the authors propose to reduce the candidates using different discretizations and by using the Mutual Visibility Graph (MVG). However, their heuristics evaluate different relay topologies, which is still too time consuming. We propose a faster method to compute the relay locations, significantly reducing the number of potential candidates for this purpose, by computing relay chains directly from the BS to the primary goals.

Furthermore, the problem of planning paths taking into account the connectivity of the agents also increases the computation time. Even fast approaches based on random sampling of the space, as the one developed in [43], are too time consuming because of evaluation of the possible connectivity-aware movements. Requiring up to minutes to obtain a path, that it is unfeasible in real deployments missions. Therefore, in this work we do not take into account the connectivity between the agents during the navigation between the goals, but it is considered when the robots are at the goals, as in most of the exploration works [26][29].

Regarding the goals visit, in the proposed strategy in the previous chapter the agents were distributed considering the depth of the goals in the tree and an Hungarian algorithm to distribute the agents. This cannot be used in the present strategy since there no exists a tree.

The number of goals used in exploration [26] used to be too few. Because the goals lay over the frontier between the explored and unexplored areas. Thus, the number of goals is not very high, although it increases with the new explored areas. The proposed approach is intended to be used to visit hundreds of goals. Some general Vehicle Orienteering techniques, as Tabu Search or Genetic Algorithms require up to minutes to obtain a solution [44]. Our proposed approach groups the goals into different clusters and tests different heuristic methods in order to find out the best visit order in short time.

4.3 System overview

4.3.1 Problem setup

The scenario is modeled as a grid, because the *Fast Marching Method* is used in different parts of the method. The robots move in a scenario with static obstacles, where x and $\mathbf{x}$ denote a position and a set of positions in the grid, respectively. There are M robots with $\mathbf{x}_a = [x_{a_1}, ..., x_{a_M}]$ denoting their positions. As in the previous chapter, the mission of the robots is to reach N goal locations $\mathbf{x}_g = [x_{g_1}, ..., x_{g_N}]$, and to transmit the information from the goals to the BS located at the x_{BS}. In this work we consider $M \leq N$. We consider that each agent is equipped with a wireless sensor to communicate with the rest of the team. In this work the employed communication model is a limited distance line-of-sight. In other words, in order to establish a connectivity link between two agents the maximum distance between them must be d_γ. And the *line-of-sight* between them has not to be occluded by some obstacle. The mission of the agents is to adopt a chain formation from the primary goals positions to the BS. Then, they have to obtain the visit order of the chains, in order to fulfill the mission as fast as possible.

4.3.2 Proposed approach

The general procedure of the proposed method is described in Alg.6. Firstly, the positions providing maximum connectivity where to connect the maximum number of primary positions are obtained, denoted as $\mathbf{x}_{mc}$ in l.1. As in the previous works, with this property the minimal number of robots for relay tasks are devoted. Therefore, most of the agents are used to visit the primary goals.

From the positions $\mathbf{x}_{mc}$, the agents that will act as relay, will provide connectivity to their teammates to visit the primary goals and to transmit the information to the BS. Due to d_γ, most of $\mathbf{x}_{mc}$ points are outside the communication area of the BS. Therefore, chains of relay goals $\mathbf{x}_{chain}$ from x_{BS} to each $\mathbf{x}_{mc}$ have to be formed, in l.2. We group the relay chain goals

Algorithm 6 Deployment general procedure

Require: $grid$, x_{BS}, $\mathbf{x}_a$, $\mathbf{x}_g$

1: $\mathbf{x}_{mc} \leftarrow max\_con\_relays(grid, x_{BS}, \mathbf{x}_g)$ ▷ Sect.4.4.1
2: $\mathbf{x}_{chain} \leftarrow relay\_chains(grid, x_{BS}, \mathbf{x}_{mc})$ ▷ Sect.4.4.2
3: $\mathbf{x}_{cl} \leftarrow \mathbf{x}_{mc} \cup \mathbf{x}_{chain} \cup \mathbf{x}_{pg}$ ▷ Form clusters (Sect.4.4)
4: $\mathbf{x}^*_{cl} \leftarrow visit\_order(grid, \mathbf{x}_a, \mathbf{x}_{cl})$ ▷ Sect.4.5.2
5: $< \mathbf{x}_a, \mathbf{x}^*_{cl} > \leftarrow allocation(grid, \mathbf{x}_a, \mathbf{x}^*_{cl})$ ▷ Sect.4.5.1
6: $\pi \leftarrow compute\_paths(grid, \mathbf{x}_a, \mathbf{x}^*_{cl})$

$\mathbf{x}_{chain}$, the maximum connectivity positions $\mathbf{x}_{mc}$ and the primary goals $\mathbf{x}_{pg}$ connected from $\mathbf{x}_{mc}$, forming different clusters of goals $\mathbf{x}_{cl}$ in l.3. After the clustering process, there exist different ways to visit the goals with communication. First, the algorithm allocates the best visit order of clusters to fulfill the mission in minimal time, in l.4. Then, the cluster goals are allocated to be visited by the agents, taking into account the type of the goals, primary or relays, in l.5. Finally, the algorithm computes the shortest paths to visit all the goals, in l.6.

4.3.3 Fast Marching Method (FMM) for chains computation

In this chapter the FMM [37] is used for different tasks apart from simple path planning. The relay positions for the chains and the evaluation of the costs for the allocation algorithms are procedures that are using the FMM. We review the properties of FMM employed for these purposes.

As described in Sect.3.5.1, FMM propagates a wavefront from a start position over every point of the grid, computing the distance gradient ∇D to every point. The wavefront propagates uniformly in all the directions with a velocity F, avoiding the obstacles. With $F = 0$ in cells that contain an obstacle and $F = 1$ in a free space cell. An example is depicted in Fig.4.3. The source is the blue point in Fig.4.3(a) and the obtained gradient ∇D is illustrated in in Fig.4.3(b). Descending the gradient ∇D, the direct path is obtained to the origin of the wavefront, depicted with red line in Fig.4.3(e). Furthermore, we consider advantageous FMM for goals allocation to the robots. With a unique gradient computation we know the distances to all the positions of the grid from the source. So, the allocation algorithms used in this work employ FMM to analyze the costs to assign the goals to the robots, as will be described in Sect.4.5.

The wavefront also can be initialized from multiple source positions. Therefore, if we propagate the wavefront from the obstacles, we obtain the distances to the closest obstacle, ∇D_{obst} in Fig.4.3(c). If the wavefront is propagated again from the same source, but fixing $F = \nabla D_{obst}$ the resulting gradient $\nabla D'$ of Fig.4.3(d) considers the distance to the obstacles. The path obtained by descending this gradient is maintained away from the obstacles, depicted with blue line in Fig.4.3(e). As can be seen in the image, the resulting path corresponds to the positions of the Voronoi Graph (VG) of these area. This path computed from BS, hereinafter referred to as Voronoi Path (VP), is used in our method to obtain the possible candidate positions where to deploy the relay chains, as will be described in Sect.4.4.2.

4.4 Relay chains and clustering

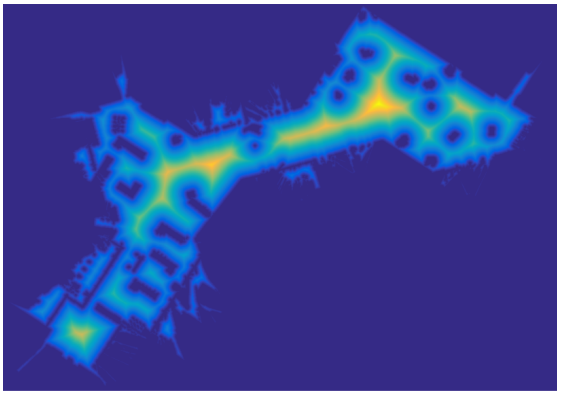
In the present section we describe the computation of the relay locations to visit the primary goals establishing a connectivity link with the BS.

4.4.1 Maximum connectivity relays

As mentioned in Sect.4.3.2, the idea is to devote the minimal number of robots for relay tasks during the mission. Thus, the maximum possible number of robots is used to visit the primary goals. For this purpose, we use the Minimum Steiner Tree (MST), since it is the best option for connectivity maximization, as it has been demonstrated in [42][26][29].

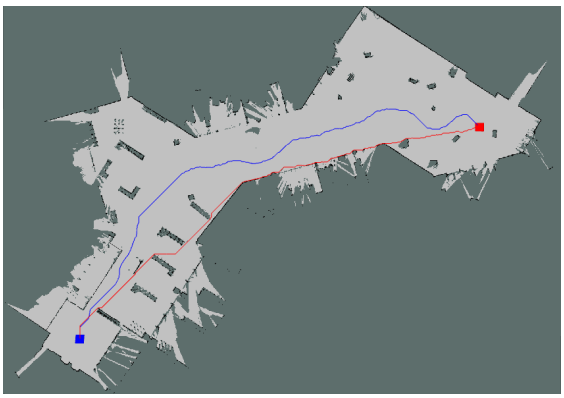
(a) Scenario

(b) ∇D



(c) ∇D_{obst}

(d) $\nabla D'$



(e) Paths

Figure 4.3: Path computation with FMM. In (a), blue and red squares are start and goal positions respectively. (b) gradient from start. The yellow cells represent the farthest distances from the start positions. (c) gradient from obstacles. (d) gradient with $F = \nabla D_{obst}$. (e) red path obtained from ∇D of (b), blue path (VP) descending $\nabla D'$ in (d).

First, we need to find out the possible candidate positions where placing the relays to reach the primary goals with connectivity. The algorithm obtains all the possible positions where

it will be possible to establish a communication link with the primary goals, by computing all the positions that satisfy the *LoS* communication model.

Then, the algorithm iteratively computes the positions of maximum connectivity, $\mathbf{x}_{mc}$. Firstly, the algorithm discards the primary goals within the communication range of the BS, because no relays are needed to reach these goals. The visit of the remaining primary goals will require the use of relays. At each iteration, the algorithm selects the position where there are more communication links from more not yet connected primary goals. If there are multiple positions that provide the same connectivity, the method chooses the one closest to the BS, which a priori reduces the number of required relay robots and their displacements. The algorithm iterates until connecting all the primary goals. As depicted in Fig.4.1(b), the algorithm firstly obtains x_{mc_2}, because it has communication links with 3 primary goals. Then it computes x_{mc_1}, to provide connectivity to the remaining two goals. The obtained $\mathbf{x}_{mc}$ correspond to the points of MST because they are the minimum necessary to interconnect the primary goals with the BS.

4.4.2 Relay goals computation

As stated in Sect.4.2, the main drawback of the relay placement is the high computational cost to analyze the suitable candidates for this purpose. In [42], the authors evaluate four different discretizations of the environment to obtain the possible candidates for relay positions: two polygonal discretizations, a grid of a desired resolution, and points of the Voronoi Graph (VG). The VG provides a good balance between the cost of the solutions and the execution time. This is because the potential candidates to place the relays are those positions that are distant from the obstacles, with wider field of view, and the number of points to evaluate is low, in comparison to evaluate the entire grid as in [26][29].

At the same time, it is logical that the relay chain will be placed somewhere along or close to the direct path from the BS to the goal for *LoS* communication. Therefore we use the paths from x_{BS} to each $\mathbf{x}_{mc}$ to place the chain relays. We use the Voronoi Paths (VP), defined in Sect.4.3.3. The advantage of using VP with respect to compute the entire VG, is that the path will guide the relay chain directly to the desired x_{mc}, instead of computing the complete VG. At the same time, VP only considers the useful points of VG to the desired x_{mc}, instead of considering the points of the entire graph. Since VP contains fewer points than VG, the search of the relay positions is faster.

After obtaining all the VPs connecting $\mathbf{x}_{mc}$ with x_{BS}, the algorithm computes the positions of the relays on these paths, placing the relay positions with an interval d_γ and with *LoS* along the path from x_{mc} to x_{BS}.

An illustrative example of the procedure of connection of primary goals and clusterization is depicted in Fig.4.1(b). For two $\mathbf{x}_{mc}$, two chains π_{chain} are extended. The resultant configuration has two clusters composed by: primary goals, maximum connectivity relay x_{mc} and the relay chain goals x_{chain}. This approach minimizes the movements of the chains; once the robots acting as relays are deployed, they remain in their position until their teammates finish the visit of primary goals in the cluster.

4.5 Goal and cluster allocation

Our approach employs two allocation procedures: the allocation of relays and primary goals to the agents within the clusters, and the order in which the clusters have to be visited.

4.5.1 Relay and primary goals allocation

The robots sequentially visit the relay and primary goals in different clusters, using the Hungarian method to optimally allocate the goals. Here the distances between robots and goals positions are the working costs. The distances from every goal to the agents are computed by the FMM, being the value of the gradient at the position of the agent the cost to reach the goal for this agent. A distance matrix D, which is used by the Hungarian method to allocate the goals, is computed.

Generally, the number of goals in the clusters is higher than the number of robots. The relay goals have priority over the primary ones, because without deploying the relays, the robots that visit the primary goals cannot transmit data. In order to ensure this order, we modify the costs in matrix D corresponding to relay goals with the expression:

$$D^*_{relay} = \frac{D_{relay} \cdot min(D)}{max(D_{relay})} \tag{4.1}$$

where D_{relay} are the values of the matrix that correspond only to the distances to the relay goals, $min(D)$ is the global minimum of the matrix D, and $max(D_{relay})$ denotes the maximum distance to the relay goals. This way, the costs that correspond to reach the relay goals, D^*_{relay}, are always lower than the costs of primary goals. And consequently the Hungarian algorithm always allocates the relay goals to the agents.

After this allocation with priorities, the team is able to extend a relay chain and it must have at least one agent to start visiting the primary goals of the cluster. We also use the Hungarian method to iteratively assign the primary goals to the robots that does not belong to the relay chains. The costs are also the distances.

4.5.2 Cluster visit order

After obtaining multiple clusters of goals, there exist different way to visit them. We propose two ways of doing this: in sequential and concurrent manner. With the sequential visits, very few agents are devoted for relay tasks, since only one chain of relays is extended. So, the workload is distributed within the clusters, distributing the remaining agents to visit the primary goals. Despite the fact that the advantages of using this type of visit are not very obvious, this visit type can be useful in some specific scenarios. For example, for stretched scenarios and where the BS is placed in some extreme location. On the other hand, the concurrent way of visits distributes the workload between the clusters, extending several relay chains. This kind of visit is advantageous for scenarios where the BS is placed in the middle, allowing to extend several relay chains in different directions.

In this section we develop different heuristics to visit the different goals clusters, in sequential and concurrent manner. The use of heuristic approaches is motivated by the need to be able to allocate the tasks to the agents in a short time, few seconds, against classic orienteering problem methods, that require up to minutes to find a solution [44].

Sequential cluster visit

For this type of cluster visit we evaluate different costs based on the distances between clusters, the amount of required robots and the workload in each cluster. The costs are the following:

- *Distance between clusters (CD).* This cost measures the displacements between the relay chains. The distance matrix is obtained using the distances between maximum connectivity relay positions of the clusters:

$$CD = \|x_{mc_i} - x_{mc_j}\|, i = 1, ..., K, j = 1, ..., K, i \neq j \tag{4.2}$$

where K denotes the number of clusters. Again, FMM is used to obtain the distances. FMM is executed K times. The solution is provided solving the Travelling Salesman Problem (TSP). As we want to obtain the solution as fast as possible, we employ different solvers based on the number of instances of the problem, empirically adjusted for our computer. If the number of instances is at most 12, we use the brute force, obtaining the optimal solution. For instances greater than 12 and lower than 20, we use the branch and bound method, and for greater instances we employ the Nearest Neighbor initialization altogether with the local optimization using 2-opt method [45]. It consists in a local optimization of the route, swapping every two edges of the route, goals in our case, checking if the cost of the new route improves the previous one. The algorithm selects the different routines in order to obtain a solution within an interval time of 50 milliseconds.

- *Required relays (RA)*, obtained as:

$$RA = (M_{rcl_i} + 1)/N, \quad i = 1, ..., K \tag{4.3}$$

where M_{rcl} denotes the number of relays of the cluster and one agent to visit the primary goals, N is the number of robots. With this procedure the robots visit the clusters based on the required relays in ascendant order. For example, for clusters that require $\{3, 5, 1, 0, 1, 5\}$ relays, the visit order will be $\{0, 1, 1, 3, 5, 5\}$. This approach can be interesting in corridor-like environments with rooms.

The following two costs measure the workload within the clusters.

- *Mean cost of the cluster*:

$$CMD = \overline{\|x_{mc_i} - \mathbf{x}_{p_i}\|}, \quad i = 1, ..., K \tag{4.4}$$

where $\mathbf{x}_{p_i}$ are the positions of the primary goals of cluster i. It measures the estimated cost of visiting the primary goals in each cluster.

- *Worst cost of the cluster*:

$$CWD = max(\|x_{mc_i} - \mathbf{x}_{p_i}\|), \quad i = 1, ..., K \tag{4.5}$$

The same as the previous one but it considers the worst cost for each cluster, represented by the largest distance between primary goals and the maximum connectivity relay of each cluster.

We evaluate different costs to obtain the sequential visit order of the clusters, listed in the Table 4.1. The overbar stands for the normalized values of the variables, to the maximum. The symbol $\times$ denotes the product of the proposed costs. PA is the number of primary goals in the cluster, that measures the amount of workload within each cluster. Heuristic $S3$ is the combination of $S1$ and $S2$, that sorts the cluster visit based on distance from BS, and $S4 - S8$ consider the displacements and the workload within each cluster.

Concurrent cluster visit

In this kind of cluster visit we propose different techniques to extend multiple chains of relays in parallel in order to reach the primary goals of the clusters. We define three different ways to obtain the adjacency graph between the clusters. The BS is the root, from where the team starts, and the vertices are $\mathbf{x}_{mc}$ of the clusters. The algorithm, starts from the root, and iteratively connects the disconnected vertices to the connected ones by levels. The levels represent the number of hops from BS. An example of the graph computations of the scenario of the Fig.4.4 are depicted in Fig.4.5.

- *Relay Number Level (RL)*: connecting the vertices based on the number of required relays to reach the clusters, in ascending order. For the previous example, where $\{0, 1, 1, 3, 5, 5\}$ relays are required to reach the clusters, they will be $\{0, 1, 1, 2, 3, 3\}$ from the BS.

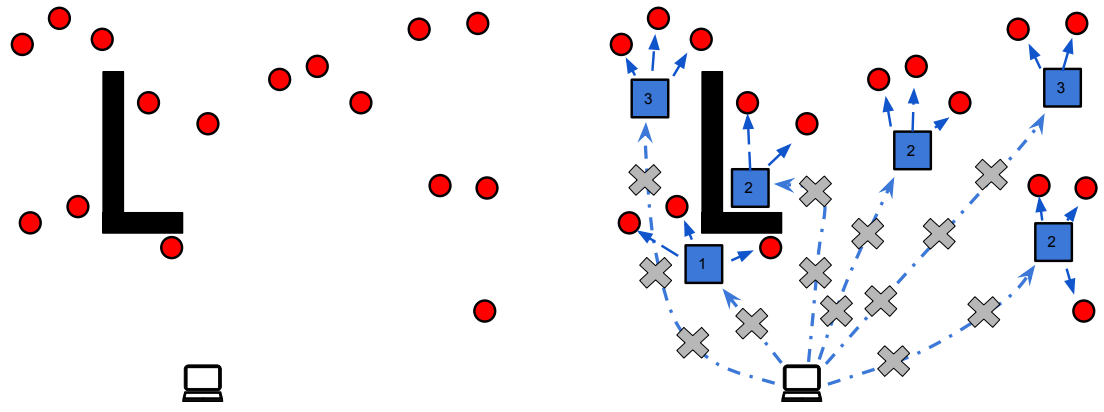
- *Cluster Distance Level (DL)*: connecting the vertices based on the shortest distance between clusters, eq.(4.2). The algorithm, at each iteration, connects one disconnected vertex to the closest connected vertex.

- *Relay and Distance Level (RDL)*: combines the previous *RL* and *DL* graph computation. At each iteration, the algorithm selects one non-connected vertex from the next relay level, obtained with *RL*, and connects it to the closest connected vertex.

After obtaining the graph of clusters, the team distributes the robots between them, extending several chains. We propose three different strategies of chain extensions:

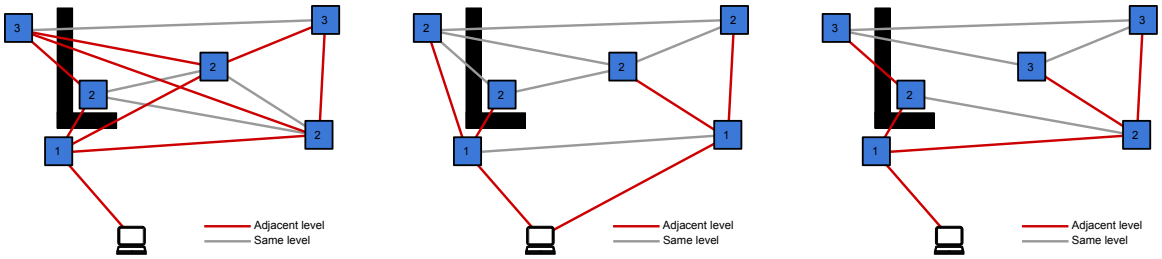
- *LC*: extending all the possible chains only to the next level.

- *MC*: extending the maximum number of possible chains.



(a) Scenario (b) Obtained relay chains

Figure 4.4: Example scenario for graph computation. The red circles are primary goals. Blue squares are maximum connectivity relay positions. Gray crosses are the remaining relays of each chains.



(a) Relay Number Level (RL) (b) Cluster Distance Level (DL) (c) Relay and Distance Level (RDL)

Figure 4.5: The possible adjacency graphs for the scenario of Fig.4.4. The numbers represent the adjacency levels or hops to the clusters from the Base Station.

– MP: extending the chains needed to visit the maximum number of primary goals.

With LC and MC criteria, the agents that will visit the primary goals are distributed proportionally to the number of primary goals of the reachable clusters. Reachable clusters are those that have already extended the relay chains.

In similar way to the sequential techniques, we consider different combinations of the graph computations and relay chain extensions, listed in Table 4.1.

4.6 Evaluation

We test our method in simulation for different situations and for different $\#robots/\#goals$ ratios. The results are evaluated with respect to the total mission time. The scenario is shown in Fig.4.3(a), with dimensions 45.8×65.65m and a resolution of 20 cm. The team of the agents starts from randomly generated positions around the BS. The goals are generated from an uniform distribution. The velocity for all the robots is fixed to $0.2m/s$. The communication range is $d_\gamma = 10m$. Some examples of the team deployment can be found in the link.[1]

4.6.1 Scalability and influence of the BS position

We test the strategies for $N = [5, 10, 20, 50]$ robots and $M = [100, 200, 500]$ primary goals and for two different positions of BS, Fig.4.6: one in an extreme area of the scenario and another one centered. In Fig.4.6(b) there are more possibilities to extend multiple relay chains in different directions, to execute in parallel the visits to clusters. As the purpose of this work is to reach the goals with connectivity for large teams of robots and number of goals, the avoidance between agents is not considered. We assume a precise continuous localization for the agents. The collision avoidance during the navigation towards the goals is not considered here. An attitude control for aerial robots or a reactive collision avoidance for ground robots might be applied during the execution of the mission. We run 10 simulations for randomly distributed goals for each scenario. The mean results of total mission time for the heuristics of Table 4.1 are depicted in Fig.4.7.

[1] http://robots.unizar.es/data/videos/icarsc19yamar.mp4

Table 4.1: List of tested methods

Sequential		Concurrent	
S1	CD	C1	*RL-LC*
S2	RA	C2	*RL-MC*
S3	$\overline{CD} \times RA$	C3	*RL-MP*
S4	$\overline{CD} \times \overline{PA}$	C4	*DL-LC*
S5	$\overline{CD} \times \overline{CMD}$	C5	*DL-MC*
S6	$\overline{CD} \times \overline{CWD}$	C6	*DL-MP*
S7	$\overline{CD} \times \overline{PA} \times \overline{CMD}$	C7	*RDL-LC*
S8	$\overline{CD} \times \overline{PA} \times \overline{CWD}$	C8	*RDL-MC*
		C9	*RDL-MP*

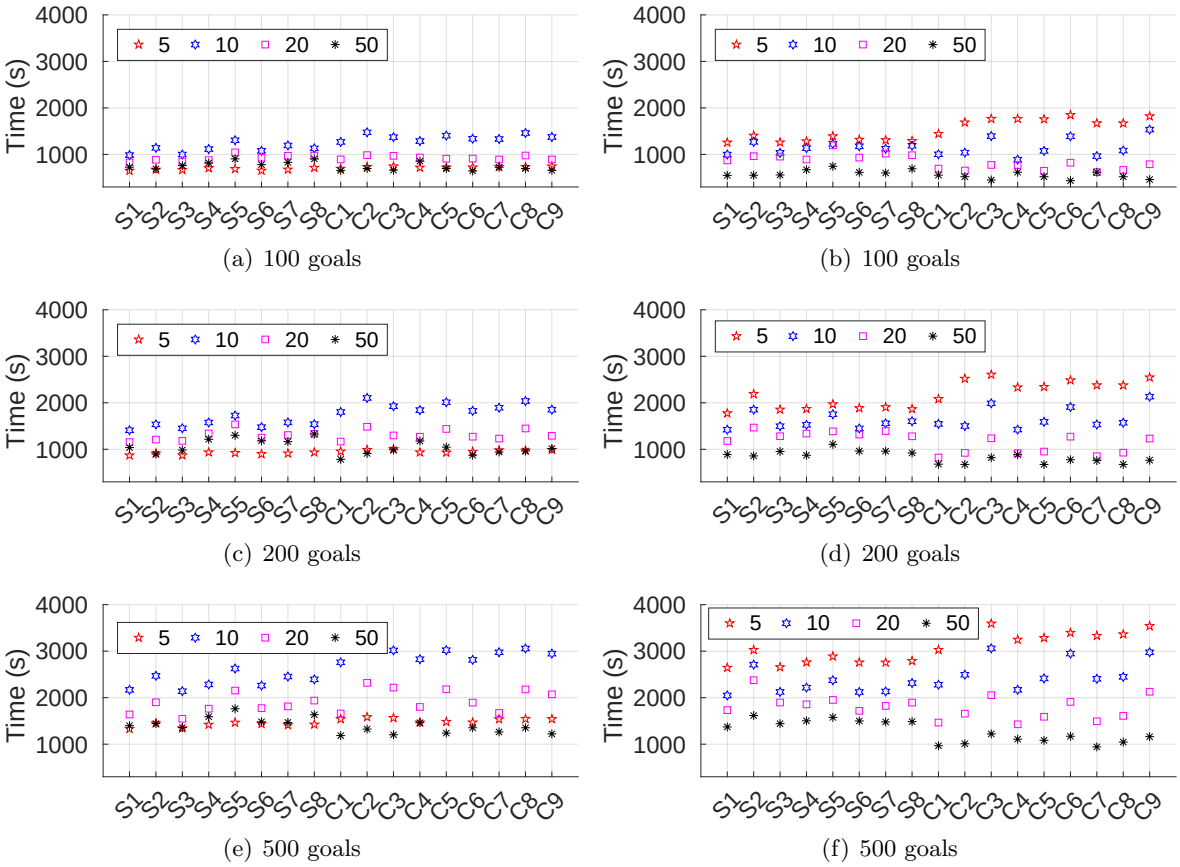
(a) BS in the extreme

(b) BS in the center

Figure 4.6: Snapshot of the tested scenarios. Big green rectangle is the BS, red squares are the agents and the rest of the markers are clusters of goals.

With 5 robots, only 50% and 90% of the goals are reached with communication in Fig.4.6(a) and Fig.4.6(b), respectively. Thus, there cannot be a fair comparison with the cases with larger robots instances, in which all the goals are reached.

The influence of the BS position is observed in Fig.4.7(a),4.7(c),4.7(e). For Fig.4.6(a), it can be seen a better performance of the sequential methods, because there are very few possibilities to extend several chains, parallelizing the cluster visit. Only for large teams (50) the concurrent methods outperform the sequential ones. When the BS is centered, Fig.4.6(b), the teams of few robots (10) fulfill the mission in similar time using sequential and some of the concurrent methods, specially with LC chains extensions. For larger teams (20 and 50), there is a significant improvement of the concurrent approaches, since there are enough agents



(a) 100 goals (b) 100 goals

(c) 200 goals (d) 200 goals

(e) 500 goals (f) 500 goals

Figure 4.7: Total time of the mission. Fig.(a)(c)(e) are for the BS in a extreme position. Fig.(b)(d)(f) are for the BS in the center of the scenario.

to extend several chains in different directions.

From the sequential methods, in general, the approaches $S1$,$S3$,$S6$ provide better results. All of them consider the distance between clusters including CD cost. $S3$ also forces the robots to visit first the clusters with smaller number of relays, giving priority to the clusters closer to the BS. $S6$, using CWD cost of eq.(4.5), penalizes the clusters with more dispersed goals or where the goals are within areas difficult to access. A common case in our scenario due to the obstacles distribution and the employed LoS communication model.

Regarding to the concurrent methods, the best methods to sort the clusters visits are RL and RDL, because they take into account the relays to reach the clusters, which implicitly include the distances from the BS. The better results are provided specially using LC heuristics, in $C1$,$C4$,$C7$. It is because these techniques deploy chains in different directions, but prioritize the clusters closer to the BS. We see a clear improvement of the best concurrent techniques with respect to the best sequential ones for 20 and 50 robots, being 30% and 21%, respectively in Fig.4.7(d). In Fig.4.7(f) it is 17% and 31%, for 20 and 50 robots respectively. The concurrent methods C3,C6,C9 (MP heuristics) extend the chains to visit the maximum number of primary goals. This causes a sequential and oscillatory behaviour, very counterproductive when there are many robots. The team extends only one chain and

the robots go from one side of the map to another.

We can conclude that the most reliable of the sequential methods for the different BS positions and for different number of goals is S1 (CD), which considers the distance between the clusters to order the visits. When the size of the robot team increases, the concurrence provides better results, particularly when the BS is centered, extending several chains to visit different clusters of goals. Being C1 (RL-LC) the method that, in mean, obtains the better results for all $\#robots/\#goals$ ratios.

In similar way to the technique of the previous chapter, in case of detecting alterations in the scenario, as appearance of new obstacles, the method can be relaunched in order to reactively respond to this situation. But this is possible only if no disconnections have occurred between the agents.

4.6.2 Computation time

The simulations were implemented on C++ and performed on a machine with Intel Core i7-4770 processor clocked at 3.4GHz with 8GB of RAM. Our algorithm proceeds in two steps: first, computing the positions of the relay goals, and then, the allocating of the visit order of the goals. The $\mathbf{x}_{mc}$ are computed to maximize the primary goals covered from each relay position, from the intersection of their communication areas. The time to compute the communication area of one primary goal is $8ms$. Then, the VP is computed from the BS to each $\mathbf{x}_{mc}$, requiring two FMM gradient computations, one from the obstacles ∇D_{obst} of Fig.4.3(c), and another from the BS, $\nabla D'$ of Fig.4.3(d). Both gradients require $50ms$ in total. Then, the path is obtained descending $\nabla D'$ in less than $1ms$. Note that the gradients are computed once for all the mission and the paths VP are computed for every $\mathbf{x}_{mc}$. Placing the relay goals over one VP in mean takes $7ms$.

The computation times to obtain the relay positions and to allocate the goals to the robots are shown in the Table.4.2. The proposed relay computation is faster than some works in the literature with similar deployment objectives. In [42], a solution to deploy 8 robots to visit 9 waypoints is obtained in minutes. In [26], 12 robots explore the environment. The objectives for the robots lay over the frontiers between explored and unexplored areas, being at most a couple of dozens of goals. The solution is found in several seconds. Our approach is able to obtain a solution for these instances of $\#robots/\#goals$, in less than a second in the worst case. The worst case would imply that each primary goal is within an independent cluster.

Table 4.2: Mean computation times of relay goals positions, and cluster computation and allocation, expressed in seconds

	Relay goals	**Allocation**			
Agents / Primary goals		**5**	**10**	**20**	**50**
100	1.72	0.23	0.62	0.8	1.52
200	2.67	0.4	1.01	1.28	2.23
500	5.67	0.96	2.14	2.6	4.1

4.7 Conclusions

In this chapter we have presented a method to deploy a team of robots to visit a high number of locations of interest and to transmit the information to a static base station, under connectivity constraints. It is a fast and scalable method to compute the relay positions to reach the goals with connectivity, which improves the computation time with respect to other techniques with similar objectives in the literature. The approach groups the goals into different clusters in order to obtain sub-optimal solutions of visit order for high number of goals. We have proposed and evaluated several sequential and concurrent heuristics to visit the clusters of the goals, in order to obtain those that result in the shortest times of the mission. We can conclude that the sequential approaches are more efficient for low ratios $\#robots/\#goals$ and for the BS located in extreme positions in the scenario. However, for high ratios the concurrent routines reduce the mission time, deploying different relay chains of robots. For future works we will generalize our method for both, intermittent and permanent connectivity. Since our method is focused on large teams of robots and many goals, we also want to include the bandwidth constraint as in [29], to avoid the latency problems in the real-world communication between the robots.

The method, as the technique developed in the previous chapter, it is able to respond to changes in the scenario, replanning in case of detecting new obstacles. However, this can be possible only in case of detecting these obstacles before a disconnection of some agents, but not after. Thus, in the next chapter we present a simple but efficient technique to tackle with the opposite situations. To reconnect the team after loosing connectivity between them due to the appearance of new obstacles.

Chapter 5

Multi-robot coordination for connectivity recovery after unpredictable environment changes

5.1 Introduction

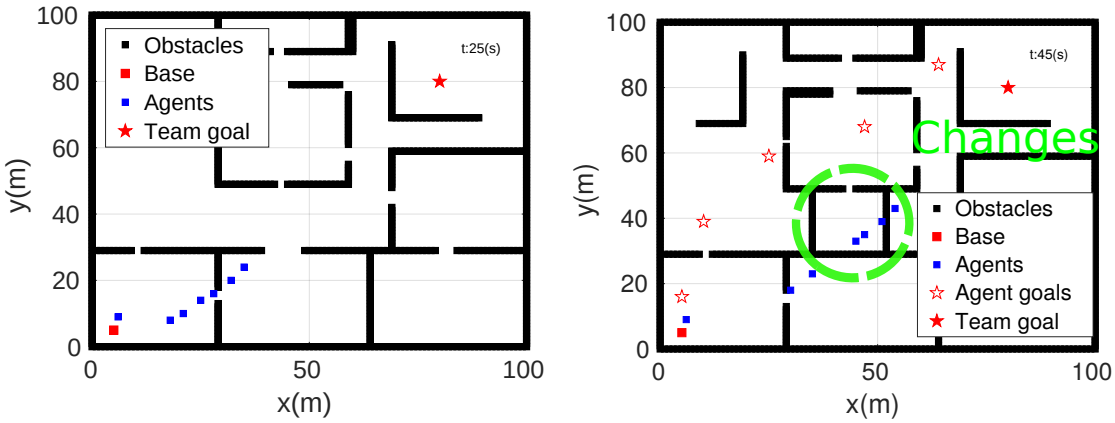
The communication in multi-robot teams is crucial, for exchange of data between the agents during their mission. In SLAM applications [46], the agents periodically meet each other in predefined points to share the mapped segments and reduce the localization error. In critical scenarios, such as exploration missions in disaster or robotized intervention scenarios, it is safer to keep permanent connectivity between the agents, [34]. These approaches ignore the failures that may be produced in the communication during the execution of the mission.

In Chapter 3, we have developed a method that was able to handle with the disconnections due to alterations of the scenario, such as new obstacles appearance. In the proposed approach the team is able to replan the mission at the moment of detecting the changes previously to the disconnection.

We consider the appearance of sporadic and unpredictable events during the mission that alter the environment and therefore, produce failures in communication, i.e. disconnection of robots in the team. We contemplate common and also critical situations for the robots, such as opening and closure of doors, or appearance of new obstacles. All these events may break the communication between the agents, and the team is split into different groups. So that, each group may be formed by a single or several agents. This fact makes a centralized solution unfeasible, because the communication between them is broken.

Therefore, we develop a distributed algorithm to recover the connectivity among the team in order to fulfill the mission. We focus on chain formations, developed in the previous Chapter 4. One robot has to reach a goal and the other robots act as relays forming a chain to maintain the connectivity with the base station, in a multi-hop data transmission.

An illustrated example is depicted in Fig.5.1. The agents dispose of an initial map of the scenario, and obtain a plan to reach an objective, starting the deployment mission, Fig.5.1(a). As it is common in real world, the environment may change. Some doors can be opened, others to be closed or new obstacles can suddenly appear, Fig.5.1(b). In this case, the team of agents is divided into three groups, do not having communication between groups. Every group of agents predicts the actions of the other groups based on the knowledge of



(a) Initial deployment

(b) Replanning after changes

Figure 5.1: Team deployment. In (b), the 3 groups replan the objectives after the changes.

their previous information, which may also have changed. They find out a new solution, if it exists, to simultaneously recover the communication and fulfill the mission of forming a chain. The new agent goals, in the chain, are depicted with red stars in Fig.5.1(b). The algorithm is decentralized from the point of view of the agents in the same group. Each agent can execute the algorithm, because they share the information in the group. The algorithm is distributed from the group perspective, since the information of each of them is different.

The rest of the chapter is organized as follows. The related works are presented in Section 5.2. In Section 5.3 we describe the problem of making predictions without full information of the entire environment. Section 5.4 presents the proposed distributed algorithm. In Section 5.4.1, we describe the chain formation and in Section 5.4.2 we develop our prediction based planner. In Section 5.5 we validate the proposed approach by means of simulations of random variations of the scenario, and discuss the results. And finally, the conclusions are presented in Section 5.6. The developed method is related to the publication [4].

5.2 Related Wwrks

In Chapter 3, we developed a communication-aware deployment planner for a multi-robot team. In that strategy, when the team is executing an exploration mission and detects alterations of the scenario, it relaunches the planner with the new environment information, avoiding the interruption of connectivity. This way, the agents never lose communication with the rest of the team. In the present work we consider the situation in which the communication between the team is interrupted, and the agents must restore the connectivity after appearance of new obstacles, in order to accomplish their mission. In [35], the authors deal with stochastic failures of the nodes by means of a control law that increases the connectivity of each agent in the graph of agents. This method also anticipates to the total failure of communication in the net of agents, but not after the failure as in the present work. Furthermore, the considered failures are tied to a total malfunction of some agents, but not to failures produced due to the environment changes, as the ones considered in this work.

When a group of agents is disconnected from others, it does not have any information

about them. So, it has two ways of actuation: either to establish some kind of policy when disconnection occurs, for example using some agent to search for the rest to reconnect the entire team; or to predict the possible actions of the rest of the groups, replanning the mission to simultaneously restore the connectivity. Here we develop a distributed planner which uses predictions about the other groups, comparing it with a reconnection policy adapted to our problem, described in Sect.5.5.

An example of policy establishment is proposed in [47], where a team of UAVs uses a bidding policy to allocate goals for a team of robots with communication faults. The authors of [48] study different behaviours for the robots to restore the communication, subject to some specific locations of the scenario. We consider impossible to properly choose a secure location where the connectivity can be restored when unpredictable alterations of the environment happen. The authors of [49] and [50] develop distributed methods to efficiently reconnect lost robots. Here the connected agents form a tree, acting as relays, to guide lost robots to a goal location with connectivity. In our method, all the robots are actively looking for recover connectivity by means of predictions about the plans of other groups, do not pre-defining specific locations to restore the connection.

The prediction of the actions of the robots implies planning under uncertainty of the actions of the other agents as in [51][52][53]. Several works have dealt with planning under uncertainty, as the Belief Roadmaps (BRM) in [54], or as the Belief Rapidly-exploring Random Trees (BRRT) in [55]. Both methods include the uncertainties of the positions of the agents over the paths, particularly useful for problems such as SLAM, analyzed in both articles. In [56], a probabilistic roadmap (PRM) is developed to obtain paths for the agents of a multi-robot team where they are able to reduce their localization uncertainty. In [57], the authors apply a POMDP technique to coordinate a team of bartender-waiters robots, using a probabilistic model for the uncertainty of the sensing devices.

In all the aforementioned methods the uncertainty is associated to the localization error, so that can be modelled. However, in our particular problem, the appearance of the obstacles is sporadic and unpredictable, thus no uncertainty models can be used. We propose a solution where the disconnected groups predict the behavior of the agents of the other groups, based on the information available at the instant of the disconnection. Thus, a policy to re-connect the groups in a chain from the base station is applied.

5.3 Problem definition

A set of N agents is denoted as $A = \{a_1, ..., a_N\}$. Again $\mathbf{x}_a = \{x_{a_1}, ..., x_{a_N}\}$ correspond to their positions. The mission of the team is to reach some goal location x_{goal} and establish a connectivity link with the Base Station placed at x_{BS}. At some moment, during the deployment to form the chain, the environment changes, causing disconnections between the agents of the team, Fig.5.1. So, the team is divided into different groups, expressed as g_i, that is $\mathbf{g} = \{g_1, .., g_M\}$, where M denotes the number of groups.

Each group is formed by one or several connected agents. We assume that every agent is equipped with a wireless antenna and a connectivity link between two agents can be established when they are within the communication range d_c of each other and there is *line-of-sight* between them. Moreover, the robots have a limited range of field of view d_v, thus only can observe changes within it. In each group, the agents share information between

them. We denote the information of a group g_i as I_i, that is basically the updated map of the environment observed by all the agents of the group. When the changes in the environment cause breakdown of communication, every group must compute a new plan to re-connect a chain. The plan of a group g_i is $\Omega(g_i|I_i) = \{\pi_1, \pi_2, ...\pi_N\}$, where π stands for the path of the agent from its current location to its corresponding assigned position in the chain. This plan computes the paths for the agents of the group and the predicted paths for the agents of the rest of groups.

At the moment of disconnection, the entire team knows the locations of all the agents. But from this point, every group observes a different scenario with the new modifications. So each group obtains its own plan taking into account the information I_i, and the predicted plans of the other groups. The plan for g_i can be formally expressed as:

$$\Omega(g_i) = \Omega\left(g_i|I_i, \Omega(g_j|I_j^i)\right), j = 1, .., M, i \neq j \tag{5.1}$$

where I_j^i denotes the information that g_i has about g_j, and $\Omega(g_j|I_j^i)$ is the plan of group g_j predicted by g_i, computed from the new observed scenario.

5.4 Prediction-based distributed coordination

In the statement of the problem we have established that the robots have to form a chain to connect the goal to the base station. In 5.4.1 we describe the algorithm that each group executes for the chain formation. In 5.4.2 we explain how each group computes its own plan jointly to the plans predicted for the other groups, according to eq.5.1.

5.4.1 Chain formation

Every group will execute the chain planner in order to predict the plans of other groups as well as to compute its own plan. The plan of some group g_i corresponds to form a chain, expressed as $\Omega(g_i|I_i)$ in eq.(5.1). The chain planner used in this chapter differs from the one proposed in Chapter 4. Here we do not obtain the Voronoi path as in the previous chapter. Instead, we use the direct path from the BS to the goal position. Alg.7 develops the procedure used by each group to compute the chain.

At first, the algorithm computes the shortest path from the base to the goal, expressed as π_{chain}, using *compute_chain_path* function in l.1. Again, FMM is used for the path computation. Descending the gradient from x_{goal} to x_{BS}, we obtain the path π_{chain}.

Similarly to the previous chapter, all the positions where the agents will act as relays lie on this path. The algorithm distributes the relay goals with an interval of the communication range of the agents (d_c), as well as where the agents have line-of-sight between them, with

Algorithm 7 Chain planner of group g_k

Require: Agent locations $\mathbf{x}_a$, Base station x_{BS}, Team goal x_{goal}, Information I

1: $\pi_{chain} \leftarrow compute\_chain\_path(x_{BS}, x_{goal}, I)$
2: $\mathbf{x}_{chain} \leftarrow compute\_chain\_goals(\pi_{chain}, I, d_c)$
3: $\langle \mathbf{x}_a, \mathbf{x}_{chain} \rangle \leftarrow hungarian(\mathbf{x}_a, \mathbf{x}_{chain}, I)$
4: $\Omega(g_k) \leftarrow compute\_paths(\langle \mathbf{x}_a, \mathbf{x}_{chain} \rangle, I)$
5: **return** $\Omega(g_k)$

compute_chain_goals function in l.2. Note that, $\mathbf{x}_{chain}$ represent the local goals for the team. Where one goal is the goal of the mission x_{goal} and the rest are relay goals. The difference with respect to the proposed approach of the previous chapter is that here we do not separate the relays from the obstacles. In the case of using the previous approach, does not affect to the performance of the proposed methodology in the present chapter.

The algorithm allocates the computed goals to each agent, with *hungarian* function in l.3. Again the FMM is used to compute the cost to reach each goal. Since, a unique gradient computation from an agent position provides the distances to all the goal locations, FMM is executed N times using Alg.21, one per agent, obtaining all the costs to the goals. Here we use eq.A.3 to stop the iteration of the wavefront propagation when the distance to all the goals position is obtained. This is because the gradient obtained by FMM may vary, in case of observe new obstacles. The goals are allocated to the agents, obtaining tuples of agent-goal $\langle \mathbf{x}_a, \mathbf{x}_{chain} \rangle$, using the Hungarian algorithm. As explained in the previous chapter, the sum of the distances that travel the entire team is the minimal. If the number of relay goals is lower than N, the Hungarian method excludes the agents which are not used in the chain. Finally, the paths are obtained with *compute_paths* function in l.4. The gradients of the allocated agents to some goal are used to obtain each path.

Note that the chain cannot be deployed if: i) a path π_{chain} does not exist from the base station to the goal; ii) there is not enough agents to place them over this path and reach the base, produced when $length(\pi_{chain}) > N * d_c$; iii) the obstacles distribution obstruct the line-of-sight between the relays over π_{chain}. In these cases, the algorithm will return that no solution exists and its reason.

5.4.2 Prediction-based planner

The predictions that each group must make about the other groups, defined in eq.(5.1), may become intractable if the team is composed by many agents and the number of groups is large. Every group $g_i \in \mathbf{g}$ recursively predicts the plans of the rest of the groups $g_j \in \mathbf{g}, i \neq j$, whose plans are also based in predictions. According to eq.(5.1), the number of predictions per group would be M^{M-1}. However, the complexity of the prediction is significantly reduced if we do not consider all the groups, but only the relevant ones for each of them, developed in Alg.8.

The first action of the group is to update the environment information, $update\_information$ function in l.1. Here, the subindex k stands for the group which is planning. The update refers to the new observed environment information from the agent sensors within the visual range d_v. In a chain formation, each agent depends on a unique agent and only one agent depends on him, its parent and descendant, respectively. This concept can be also applied to the groups. Therefore, the algorithm obtains the groups with its own information in l.2, where the groups are formed by connected agents. Then sorts the groups based on the proximity to the base station in l.3. This way, each group depends on the previous and the next group in the chain. The number of predictions that makes every group is reduced to $2*(M-2)+2$. Where $2*(M-2)$ predictions are for intermediate groups and 2 predictions for the extreme groups of the chain.

Then, the algorithm computes the plans for all the groups, in lines 4-7, using the chain planner of Alg. 7 and the information that the group believes that has every group g_i. At this point, all the plans of the rest of the groups are obtained. The algorithm predicts the

Algorithm 8 Group planner of g_k

Require: Agent locations $\mathbf{x}_a$, Base station x_{BS}, Team goal x_{goal}, Information I_k

1: $I_k \leftarrow update\_information(\mathbf{x}_{a_k}, I_k, d_v)$
2: $\mathbf{g} \leftarrow obtain\_groups(\mathbf{x}_a, I_k, d_c)$
3: $\mathbf{g} \leftarrow sort\_groups(x_{BS}, I_k)$
4: **for each** $g_i \in \mathbf{g}$ **do** ▷ $i = \{1, ..., M\}, i \neq k$
5: $I_i^k \leftarrow update\_information(\mathbf{x}_{a_k},\ I_i^k,\ d_v)$ ▷ $\mathbf{x}_{a_i}$ are locations of g_i
6: $\Omega(g_i) \leftarrow chain\_plan(\mathbf{x}_a, x_{BS}, x_{goal}, I_i^k)$ ▷ Alg.7
7: **end for**
8: $\Omega(g_k) \leftarrow prediction\_plan(\mathbf{x}_a, \Omega(g_i), I_i^k)$ ▷ Alg.9
9: **return** $\Omega(g_k)$

individual path of each agent of each group, l.8, described in Alg.9.

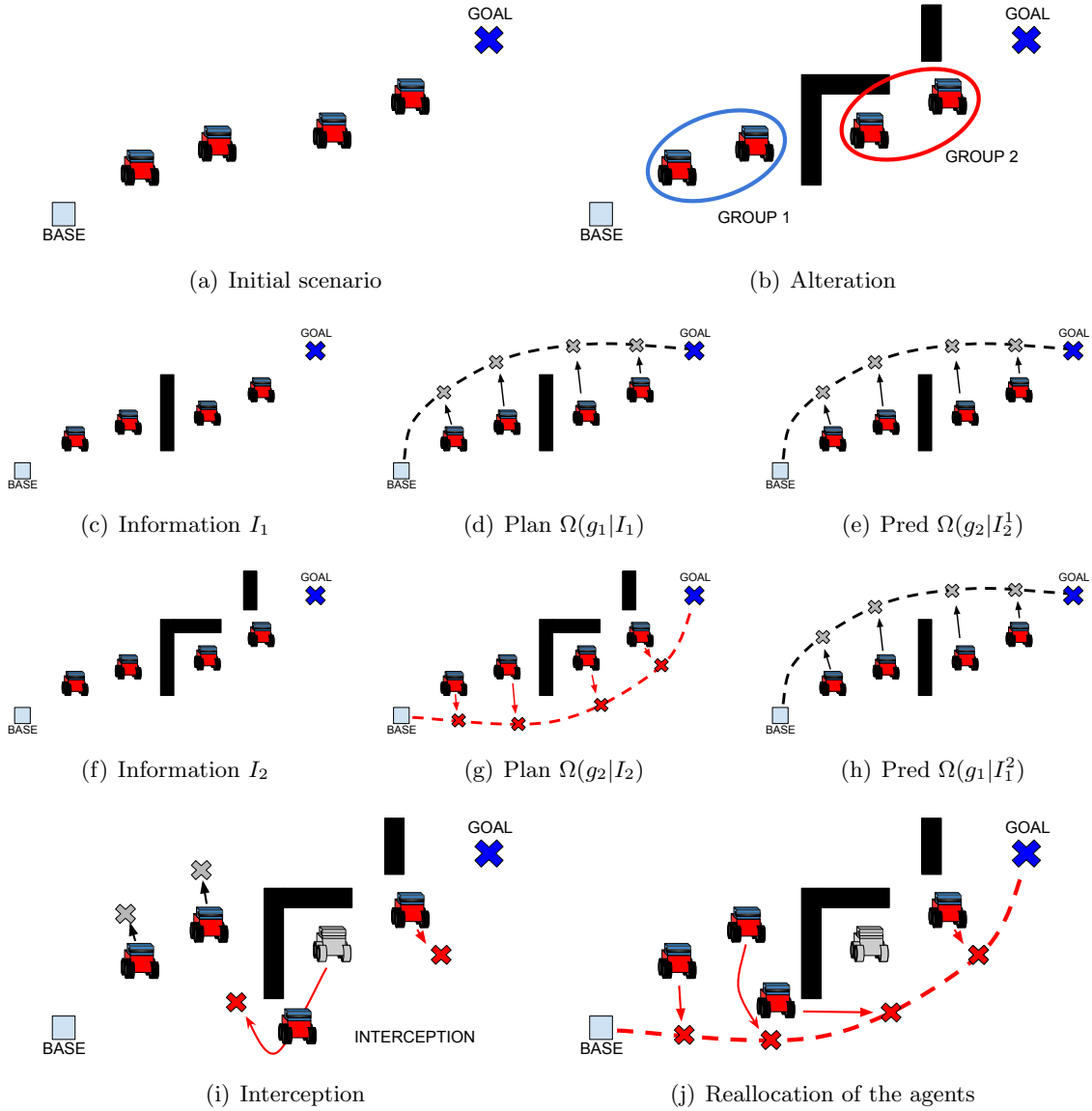
The Agent planner, in Alg.9, computes the paths for all the agents of the team, l.1. Since the groups are sorted in terms of proximity to the base station, all the agents are also sorted in the same way. So, each agent a_n must check how acts its link in the chain that pertains to another group. First, it checks the goal of its parent in the chain, a_{n-1} l.2. If the goal of the parent does not coincide according to the information of both groups, $I^k_{g(n-1)}$ and $I^k_{g(n)}$ l.3, the agent must intercept the parent to transmit the new information, l.4. This is because the parent does not have the correct goal and it is going to some position to provide connectivity to no one. This way, the group is able to re-plan its chain to connect both groups. The $intercept(\pi, x, I)$ function computes the path to the closest position of the path π from the position x using information I.

An illustrative example of this procedure is depicted in Fig.5.2. The team starts the deployment with the initial information, Fig.5.2(a). The appearance of new obstacles separates the team into two disconnected groups, Fig.5.2(b). Each group observes different environment variations, depending on its visual range d_v, Fig.5.2(c) and 5.2(f). Therefore,

Algorithm 9 Agent prediction planner

Require: Agents $\mathbf{x}_a$, Plans $\Omega(g_i)$, Information I_i^k

1: **for** $n = 1 : N$ **do** ▷ N agents
2: **if** $a_{n-1} \notin g(n)$ **then** ▷ $g(n)$ is the group of agent a_n
3: **if** $goal(a_{n-1}|I^k_{g(n-1)})! = goal(a_{n-1}|I^k_{g(n)})$ **then**
4: $\pi_n^- \leftarrow intercept(\pi_{n-1}, x_{a_n}, I^k_{g(n)})$ ▷ π_n: path of a_n
5: **end if**
6: **end if**
7: **if** $a_{n+1} \notin g(a_n))$ **then**
8: **if** $goal(a_{n+1}|I^k_{g(n+1)})! = goal(a_{n+1}|I^k_{g(n)})$ **then**
9: $\pi_n^+ \leftarrow intercept(\pi_{n+1}, x_{a_n}, I^k_{g(n)})$
10: **end if**
11: **end if**
12: $\Omega(g_k) \leftarrow \pi_n^- + \pi_n^+ + compute\_path(x_{a_n},\ x_{lg_n},\ I^k_{g(n)})$ ▷ Complete path of agent n
13: **end for**
14: **return** $\Omega(g_k)$



(a) Initial scenario (b) Alteration

(c) Information I_1 (d) Plan $\Omega(g_1|I_1)$ (e) Pred $\Omega(g_2|I_2^1)$

(f) Information I_2 (g) Plan $\Omega(g_2|I_2)$ (h) Pred $\Omega(g_1|I_1^2)$

(i) Interception (j) Reallocation of the agents

Figure 5.2: Explanation of the proposed approach

the groups have different initial plans, obtained with Alg.7, depicted in Fig.5.2(d) and 5.2(g). The plan of g_1 is $\Omega(g_1|I_1)$, depicted in Fig.5.2(d), and matches with its prediction of the plan for g_2 denoted as $\Omega(g_2|I_2^1)$, depicted in Fig.5.2(e). Thus, the agents of g_1 believe that their plan is correct and they will execute it. However, the prediction $\Omega(g_1|I_1^2)$made by g_2 in Fig.5.2(h), does not match with its plan $\Omega(g_2|I_2)$, as indicated l.3 in Alg.9. Therefore, g_2 intercepts the agent of g_1, Fig.5.2(i), executing *intercept* routine in l.4 of Alg.9. After interception, the entire team is a unique group and the agents go to their respective goals, in order to form the chain, Fig.5.2(j). In general, if agent a_n is able to intercept to a_{n-1} at some point on its path π_{n-1}, it goes to that location. If not, a_{n-1} will be at the end of the path π_{n-1}. Then a_n transmits to a_{n-1} the new information about the changed environment. The same procedure is repeated for agent whom a_n provides connectivity, the descendant

agent a_{n+1} in the chain, l.7-11. After informing the agents of other groups, a_{n-1} and a_{n+1}, the agent a_n finds the path to its own objective in the chain, l.12. Note that the paths of all the agents which pertain to g_k are the real paths that travel the agents of the group, and the rest of the paths are just predictions for agents of other groups.

When some agent attempts to access to some room, with a unique possible access where could be some teammate, and this access is blocked by a new obstacle, the robot executes a *trapped* routine. It consists in following all the walls of the room, trying to find a new possible access. This exploration procedure is necessary because the robots must ensure if it is possible or not to access to this part of the scenario. If it is not possible, the agents inside cannot be re-connected to the whole team.

Extending recursively the method to all the groups in **g** existing in a given moment, it converges to a chain solution connecting in a unique group all the robots of the team from the base station to the final goal, in the case that a solution exists. An obvious sufficient condition is that the frequency of changes in the environment is lower than the time required to find and execute the solution. Another sufficient condition is that at least one group will observe sometime the correct information of the whole scenario, to be transmitted to the others. Eventually, this group will be the closest one to the most altered area. But if these conditions are not met, the method will continue trying to find a solution.

5.5 Simulations and discussion

5.5.1 Environment

We have tested our method in the $100m \times 100m$ environment depicted in the Fig.5.1(a), for a team formed by 7 agents. We fix the visual range of the agents d_v to $30m$. The communication range d_c between the agents is $30m$ as well, and there must be line-of-sight between the agents to establish connectivity. We set the velocity for all the agents to $2m/s$. The mission of the agents is to form a chain, if it is possible, from $x_{BS} = [5, 5]$ to $x_{goal} = [80, 80]$.

We generate randomly the scenario variations, as well as the initial positions of the agents. The robots start from these positions with the initial map of the environment, Fig.5.1(a). All of them know the locations of the base station, the goal and the initial positions of the whole team, but not the variations. Three types of variations might occur: closures, appearance of small obstacles, and openings. We have run 100 trials, randomly producing 3 closures of doors, 5 new random obstacles and 5 openings of doors in each trial. Three reasons could prevent to obtain a solution: i) the base station and/or the goal are completely enclosed by obstacles; ii) there no exists a possible chain with the available agents, due to the new obstacles; or iii) some agents are trapped. Both situations are evaluated, with and without possible solution.

5.5.2 Comparisons

The proposed method is compared with two alternative solutions. The first one considers the ideal case in which all the agents have full knowledge of the new environment and are connected in the entire scenario, despite the obstacles. Therefore, if a solution exists, the method finds it and the agents directly form the chain.

The second one establishes a policy which guarantees the regrouping of all the agents. In

[48] four behaviours were proposed for communication re-establishment between multi-robot teams. The four approaches guide the robots to: i) closest open space; ii) stored waypoints during the mission; iii) nearby inclines; iv) last known position of the nearest teammate. All these strategies depend on specific locations of the environment, so we cannot consider them because, with random appearance of obstacles, some areas could be completely blocked. Hence, in our case, it is more appropriate to use some agents to reconnect the rest of the team. In [49] a group of robots is used to extend a tree and guide the disconnected agents to a goal position, the root. This method does not consider that the root robot can be completely isolated, a possible case in our problem. Moreover, the number of agents in the team could not be enough to reach all the areas of the scenario, due to the new obstacles.

Therefore, we consider that for the problem dealt with here, a searching strategy, with a searcher group, is the most proper policy to reconnect the entire team. The *searcher* group is selected to look for the rest of the agents, which wait at their initial positions until some searcher agent arrives. The selected searcher group is the closest, not trapped, group to the base station. One agent acts as leader and the rest follow him, within d_c, in order to avoid disconnections. It does not have any knowledge of the new environment beyond its visual range, so that it discovers the changes during the seeking. When new agents are reconnected, the method chooses a new agent to reach and a new leader by proximity. The distance between agents is computed with FMM. The searching strategy acts in the same way as the prediction planner in case of detecting a trap situation, described in Sect.5.4.2. This way it assures the regrouping, if it is possible.

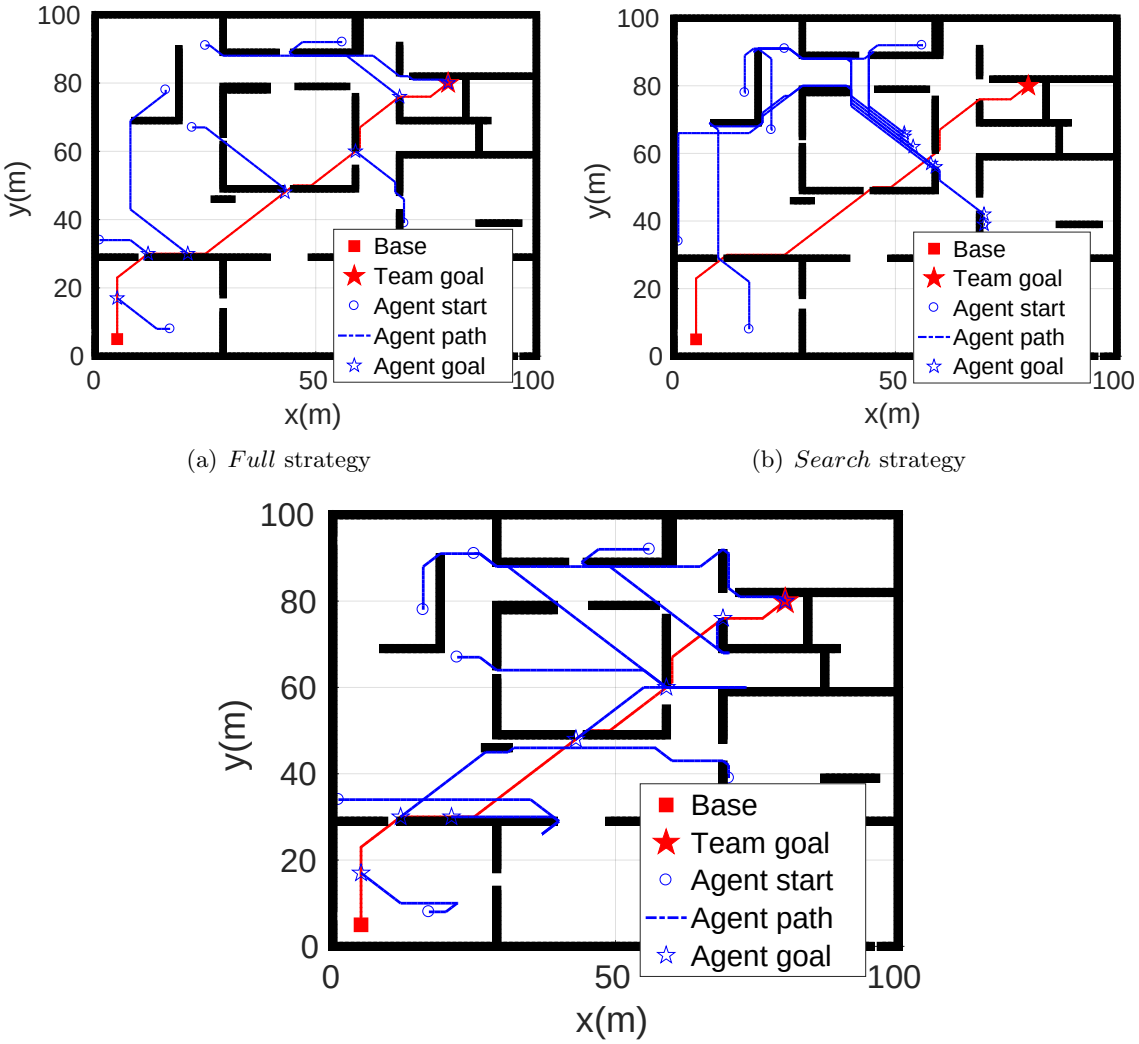
The results of the proposed comparisons are denoted as $Full$ when the agents have full information of the environment and connectivity, $Search$ for the solution using the searching strategy, and $Pred$ for our technique in which the robots only have partial information around their field of view and the information from the robots of its own group. The examples of these strategies in the altered scenario are depicted in Fig.5.3. With $Full$ in Fig.5.3(a), all the agents have the entire information of the changed environment and go directly to form the chain, depicted with red line. With $Search$ depicted in Fig.5.3(b), the agents are sequentially reconnected until form a unique group. Fig.5.3(c) depicts the trajectories of the agents with the proposed $Pred$ approach. The agents are reconnected, forming the chain, after discovering new obstacles in the scenario.

5.5.3 Results

We measure the difficulty of the scenario by means of the number of the groups in which the team is split when the scenario changes. The evaluated metrics are: i) the time to reconnect the entire team, ii) the time to fulfill the mission (if a solution exists), and iii) the total distance travelled by the team. The reconnection time is the time to merge the robots in a unique group. The mission time is the time when the agent, at the goal location, is connected to the base station through its teammates. The mission time includes the reconnection time. The distance is the sum of the distances travelled by all the agents.

After running 100 random trials, we have obtained 48 scenarios with a possible solution and 52 where there is not possible to fulfill the mission forming a chain. The video of some simulated scenarios can be found in the link[1].

[1] http://robots.unizar.es/data/videos/iav19yamar.mp4



(a) *Full* strategy

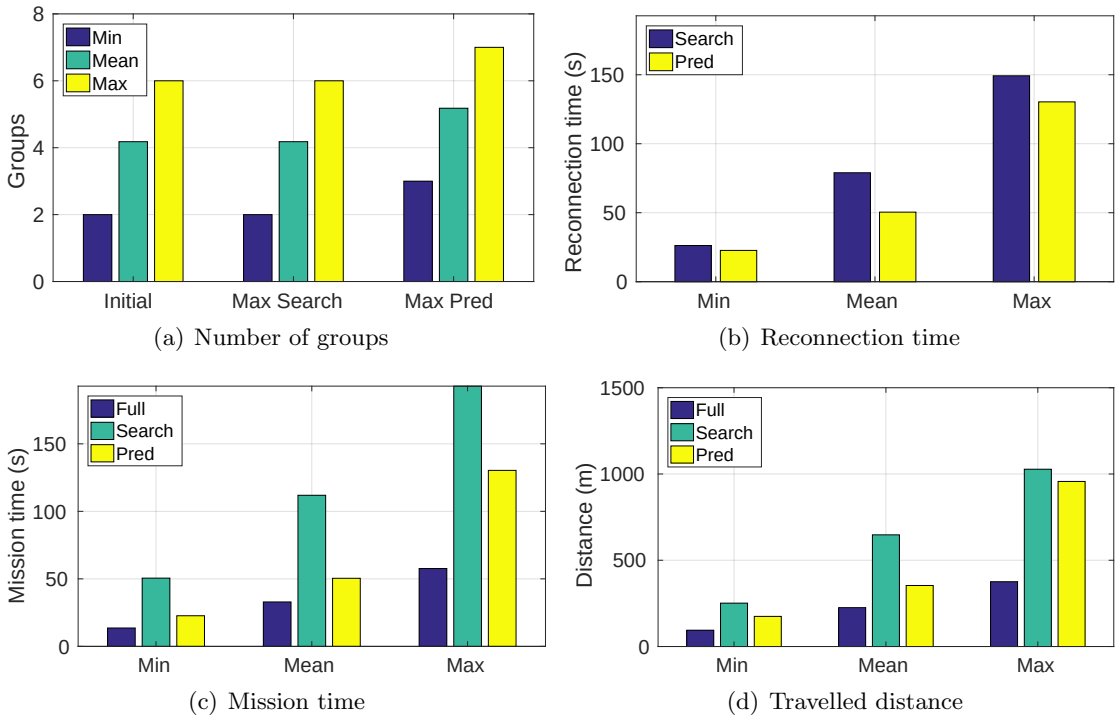
(b) *Search* strategy

(c) *Pred* strategy

Figure 5.3: Trajectories of the agents in the tested methods. In (b), only the paths until reconnecting the entire team are illustrated for clarity. After reconnection, the agents also form the chain.

Scenarios with solution.

The results for scenarios with solutions are depicted in Fig.5.4. In Fig.5.4(a), the number of groups are depicted. The first three bars correspond to the minimal, mean and maximum number of initial groups for the 48 scenarios. The other bars correspond to the maximum number of groups during the simulations obtained by *Search* and *Pred* algorithms, respectively. As can be observed, with *Pred* algorithm the agents are separated into more groups in order to search other groups. While in *Search* routine, when some group is reconnected, their agents are following the searcher agent and never get disconnected. So the values are exactly the same as for the initial groups. In both, *Search* and *Pred* strategies, the agents merge in a unique group.



(a) Number of groups (b) Reconnection time

(c) Mission time (d) Travelled distance

Figure 5.4: Results of scenarios with possible solution.

The times to reconnect the team are depicted in Fig.5.4(b). We show the minimal, mean and maximum times to reconnect the entire team for *Search* and *Pred* routines. Using predictions, all the agents are moving, attempting to reestablish communication with the rest of groups, so the reconnection occurs faster. *Full* routine is not included here because from the beginning all the agents have connectivity. Fig.5.4(c) shows the times to accomplish the mission (forming a chain) for the three routines. With *Full* routine the agents go directly to their new objectives to form a chain, so we use this time as a baseline for comparison. The values for the *Search* approach are much higher, because first all the groups recover the communication and, only then, the team forms the chain. In mean, it takes $111.9sec$ to fulfill the mission for *Search* strategy, against $32.91sec$ of *Full* routine. The proposed method *Pred*, fulfills the mission in $50.42sec$, in mean. Much lower than the sequential regrouping and chain formation, using *Search*. The maximum values of times denote that there are some scenarios which are highly altered, so *Search* and *Pred* approaches require more time to discover the new obstacles and to regroup. The minimum and mean times of *Full* and *Pred* do not differ significantly, because when the variation of the scenario does not affect to the initial plans of all the groups, the groups are reconnected with the first or few predictions, fulfilling the mission quickly. However, when the alterations are substantial, the groups require to travel more distance, discovering new obstacles, to find other groups. The worst scenario of *Pred* requires $130.3s$ against $57.66s$ of *Full*. As can be seen, the results of connectivity recovering and mission accomplishing times are exactly the same for *Pred*, because both tasks are accomplished simultaneously.

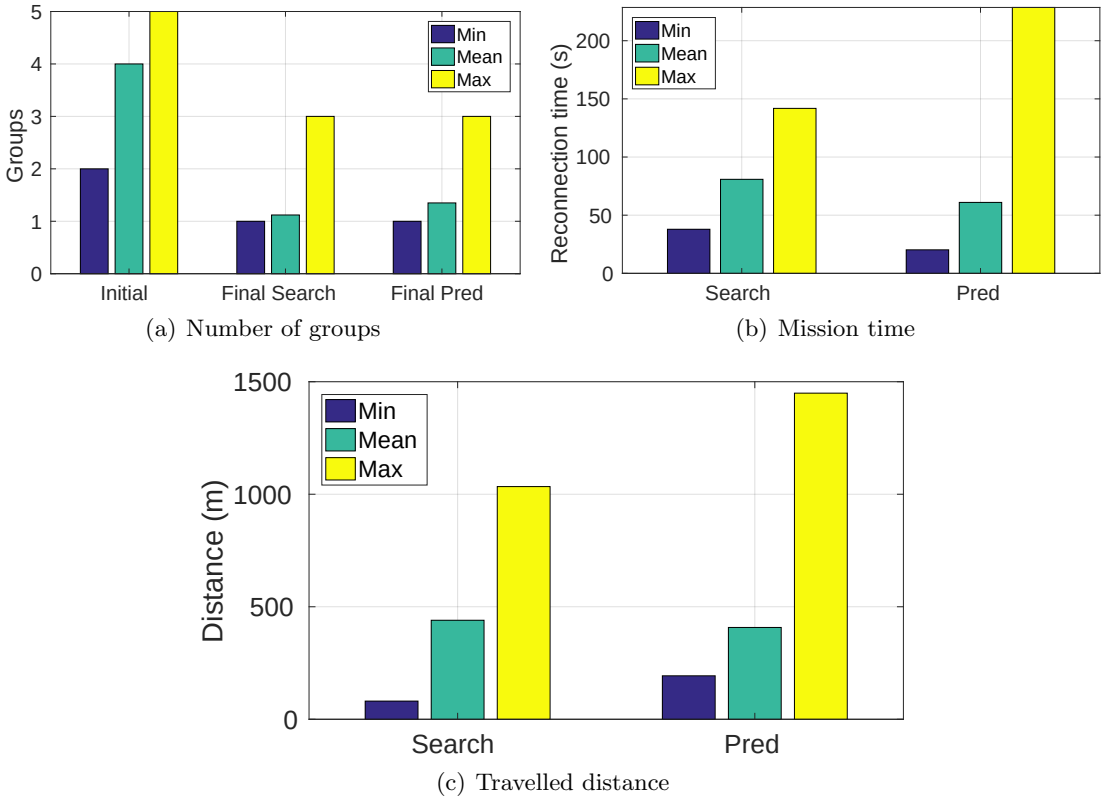
Fig.5.4(d) depicts the total distance travelled by all the team. In mean, using predictions,

the agents travel less distance than using the searching strategy, $353.8m$ against $647.4m$. This is logical because in *Search* strategy the agents regroup and form a chain sequentially. A leader searches to the disconnected agents and already connected agents are following him. Whilst, with the proposed *Pred*, the agents perform both tasks simultaneously, reconnecting the team and forming the chain. The same occurs if we observe the maximum distances, but the difference is not so significant, $956m$ for *Pred* against $1029m$ for *Search*. This happens because, in some scenarios with significant variations, there are many agents intercepting agents of other groups (Alg.9), because the latter ones have observed less variations. Once again the travelled distance for *Pred* is higher than for *Full*, since in the latter all the team has a full knowledge of the environment, and the agents go directly to the goals.

As can be observed, the deviation from *Full* strategy is more significant for *Search*. The mean travelled distance is 187% higher than for *Full*, against 57% of increase for the proposed *Pred*. In the same way, the increase of time is 240% for *Search* and 53% for *Pred*.

Scenarios with no solution

The results for scenarios with no solution are represented in Fig.5.5. Since the absence of solution could be caused by a confinement of some agents by obstacles, we evaluate the groups at the end of the mission. Here, the mission finishes when the agents realize that they are not able to form a chain to transmit the information to the base station. Again, the



(a) Number of groups

(b) Mission time

(c) Travelled distance

Figure 5.5: Results of scenarios with no solution.

first three bars correspond to the number of groups at the beginning of the mission. As we can observe, the minimum and maximum number of groups for *Search* and *Pred* routines are identical. The maximum values indicate that for the worst scenarios the proposed *Pred* planner is able to determine that it is not possible to reconnect more groups, so the mission is unattainable. The means of the groups are slightly higher for *Pred* than for *Search*, 1.35 groups against 1.12 respectively. It occurs because *Pred* has failed in four scenarios, due to a mismatch in goal allocation. The prediction planner, as opposed to searching routine, allows disconnections of the agents. So, when two agents of two different groups are allocated for going to search one to each other, they can be permanently exchanging their positions without merging in a unique group, appearing cyclic behaviours for these groups. This is also reflected in the maximum time and travelled distances. When this happens, it can be detected by the algorithm; a solution would be to fix the chain path planned by one of them, enforcing the other agent to move toward it, resolving the cyclic situations. In this kind of situations, our *Pred* method would be more like the *Search* method, reducing the maximum times for reaching the solution or for being aware that there is no solution. Despite this, the mean times of the proposed *Pred* planner outperforms the *Search* routine, $60.99sec$ against $80.84sec$ respectively, corresponding to 33% of improvement. In the same fashion, the mean distance is lower, but only 8%. The improvement of *Pred* over *Search* is not that significant than in scenarios with solution, where it is around 100% in both, time and distance. This is because there are more alterations in the scenario are more significant, and require more exploration to be discovered by the team.

5.6 Conclusions

In the present chapter we have presented a distributed method to coordinate a team of robots, in order to recover the connectivity after communication failures caused by unpredictable events, such as appearance of new obstacles or lost of line of sight. The proposed approach employs the predictions of actions of the different groups of agents to reconfigure the team.

With the proposed strategy, all the groups of disconnected agents seek for the rest of groups to restore the connectivity and also form the chain to reach the goal location. The method is compared against other two strategies: the ideal one, in which all the agents have full connectivity and information of all the modifications in the scenario (*Full*); and another in which a group of agents is used to search the rest and reconnect them forming a unique group (*Search*). Our approach outperforms *Search* strategy, since all the agents are moving seeking for the rest. Obviously, the results for our prediction-based planner are worse than for *Full* strategy, because it requires an extra time to discover the new obstacles and to obtain new paths for the whole team in the environment. But if it exists, a solution is found.

In the future, we want to generalize the proposed method for different deployment strategies. For instance, we will consider the tree formation used in the previous chapters 2 and 3. We are also working in a generalization of the proposed method for the dynamic obstacles. This new approach could be used in scenarios with human presence, which exhibit natural environment alterations from the point of view of the robots.

Part II

Data Gathering with periodic information delivery to a static Base Station

Chapter 6

Trajectory planning for time-constrained agent synchronization

6.1 Introduction

In the previous part of the document, the data acquisition was planned in order to establish a connectivity link with the Base Station. The robots are used in role of relay, retransmitting the information of their teammates when they visit the primary goals. However, the establishment of the link with the BS it is not always possible. This is mainly due to the following reasons: the distance to the goals, the dimensions of the scenario altogether with the position of the BS, and the limited ranges of the wireless devices and the number of the available agents in the team.

Therefore, the information of the measurements taken at the primary goals must be delivered going directly to the BS. At the same time, from the BS, new information from new locations of interest may be requested, generated them automatically or by some human operator.

Another approach may be devoting only some agents to go to the BS in order to retransmit the information of their teammates, that are only being used to reach the primary goals. The last ones, are called *worker* agents, that are used only for working purposes. That is, they are only visiting the goals and taking measurements. The first ones are what we call *collector* agents, which are travelling in the scenario receiving the data of the workers in order to retransmit it to the BS. The trajectories of these collector agents are known by the worker, being always the same and travelled at constant velocity. Hence, the workers know about the collectors position along the time and can decide where and when to share the data with a collector agent. An illustrative example of the application of the proposed trajectory planner is depicted in Fig.6.1.

The tasks of the mission are allocated to the workers in Fig.6.1(a). The trajectory of the collector agent is shared between the team. Then, the data gathering mission starts and the workers visit their goals, taking measures and deciding where and when to share the data with the collector Fig.6.1(b)-6.1(c). Obtaining the best trajectory according to the time to transmit all the gathered data and constrained to the trajectory of the collector. That is, before it returns to the BS for uploading, Fig.6.1(d). This is the problem solved in this

chapter, the planning of the trajectory for a working agent in order to exchange data with some collector teammate with a known trajectory.

We propose two methods to compute the trajectories:

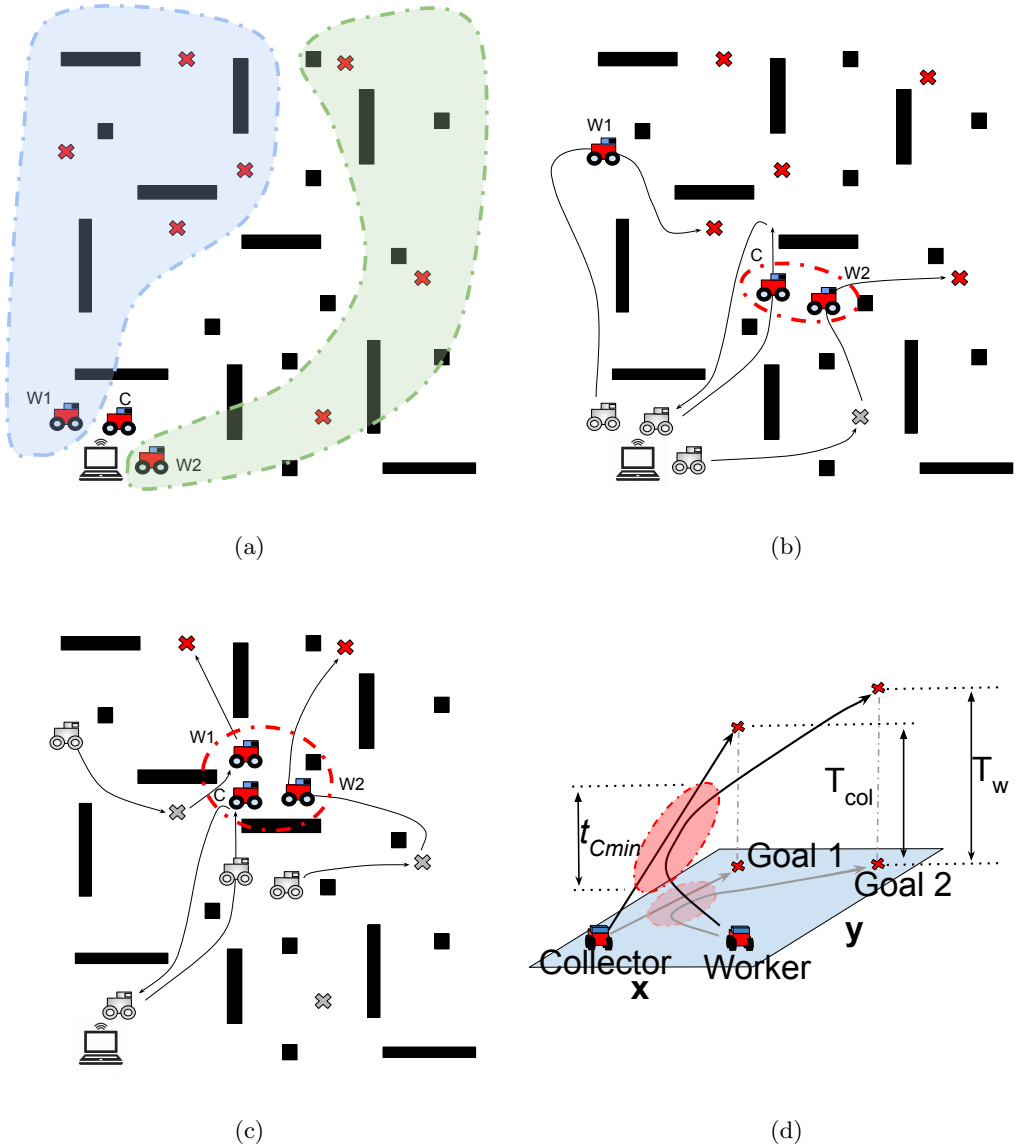


Figure 6.1: Illustrative explanation of the proposed method. In (a), 3 robots must make observations from goal locations (red crosses). Two agents are used as workers, W1 and W2, and one agent is used in role of collector, denoted with C. The Base Station allocates the tasks to each worker (the colored areas in (a). The trajectory of the collector agent is shared between the team. Fig.(b)-(c) represent the trajectories planned by the workers to meet the collector, during goals visit. Fig.(d) illustrates a planned trajectory for synchronization between worker-collector in the *x-y-time* space. It is computed according to the required time for the transmission $t_{c_{min}}$.

– A sampling-based approach: based on a random sampling of the space in order to compute a sub-optimal solution in short time.

– An optimal approach: based on the computation of the optimal trajectory but in some larger time.

The rest of the chapter is organized as follows. Section 6.2 summarizes the works of the literature. The problem of trajectory computation is presented in Section 6.3. In Section 6.4 we formally define the dynamic communication areas. The developed sampling-based trajectory planner and its results are presented in Section 6.5. The optimal version and the corresponding results are presented in Section 6.6. Finally, the conclusions of both techniques are summarized in Section 6.7. The sampling-based technique was presented in [5].

6.2 Related works

In this chapter we consider a distributed solution to the data gathering missions. Therefore, the centralized solvers as the proposed in the previous chapters are not feasible. The proposed methods for exploration missions [26][31][30] cannot solve our scenario either, since in their scenarios there no exist time constraints for synchronization between agents.

Our scenario shares more similarities with patrolling works [58][59][60]. The agents patrol a predefined paths, obtained from a environment graph, making observations and synchronize with their teammates just during the time to share data. However, the predefined paths are main drawback of these methods. It guarantees that at some moment the agents will meet each other. However, the robots never deviate from their paths, that could be inefficient in data gathering missions. In [13], the agents periodically meet each other in order to share information between them. But, the problem of plan paths to make concur large teams of agents may become intractable. In our approach, we propose the usage only of some members of the team that will travel a known path, the collector agents. So, the worker agents plan their trajectories, without disrupting the travelled paths of the collectors.

Furthermore, in all the aforementioned techniques a specific allocation algorithm for connectivity tasks is required. The presented trajectory planners do not need from the usage of this specific task allocator for connectivity, because how and where to achieve the synchronization are computed by both presented trajectory planners.

The trajectory computation involves the time in which the path is traversed, so the planner explores different temporary options to reach the goal. The runtime of optimal methods such as Dijkstra[61], FMM[37] or A*[62] scale with the number of dimensions. Therefore, the first proposed approach is based in the sampling-based methods, which does not require the grid discretization. The randomized multiple-query algorithms such as Probabilistic Roadmaps (PRM) [63], are still computationally heavy for our problem and do not really provide better results than other sampling-based techniques. Therefore, we employ as the base method the Rapidly-exploring Random Trees (RRT) [64]. Due to its single-query and randomized nature, it fits well to explore several trajectories, choosing the best one among them.

There are many variations of RRTs in the literature. The RRT-Connect [65] raises two trees, one from the initial position of the robot and another from the goal. This way, the environment is explored faster and it reduces the probability of getting trapped. The Dynamic-Domain RRTs (DD-RRT) [66] limit the sampling domain, generating random

nodes in areas which do not lead towards the obstacles and minimizing the execution time. The solution cost is improved by RRT* [67] and Informed RRT* [68]. RRT* improves the parent selection and includes a reconnection routine of neighboring nodes, reducing the cost of possible paths, with respect to the basic RRT. The Informed RRT* employs heuristics to delimit the sampling domain, finding out solutions faster and in more problematic scenarios where the randomized algorithms are not usually effective, as narrow passages. Both mentioned techniques increase the computation time with respect to the basic RRT.

The presented TC-RRT, described in Sect.6.5, takes into account some of the features of the cited methods. We delimit the sampling space to lead the worker robot through the communication area, as occurs with DD-RRT and Informed RRT*. We use a parent selection routine, similar to RRT*, not to reduce the solution cost, but to avoid deadlock in a local minima, so increasing the success rate of the algorithm.

The second proposed time-constrained planner is based on an optimal planner. We use the FMM, due its advantageous properties. On the contrary to the sampling-based techniques, FMM is slower, but it computes the optimal paths. So, if there exists a possible path that accomplishes the time constraints, the solution is found. Regarding optimal solvers, the Dijkstra's algorithm [61] evaluates all the possibilities from the starting point to every position of the grid and it obtains the best path to the goal, sharing these properties with FMM. However, the FMM is more efficient in terms of precision. The interpolation of the distance, provided in the Appendix A, is more accurate to the real distance. Another possibility is the A* algorithm [62], which rapidly guides the search of the path using heuristics. In [69], the algorithm was adapted to obtain paths that lead the robot to visit stationary signal sources. In the scenarios considered in our work, the goals to be reached by every robot are known, but the position where the mate is intercepted is ignored a priori. Thus, the needed subgoals for communication are not explicitly computed to be used for an A* path planning algorithm. This makes difficult to find out a good heuristic. Therefore, in the case of evaluating many options for exchanging information positions, it is necessary to execute the A* algorithm the same number of times as positions of all the connectivity areas, which is computationally inefficient.

6.3 Problem setup

The problem solved here is the computation of a trajectory to simultaneously reach some goal location and synchronize with a collector agent in movement. In other words, to find a trajectory to a goal location that traverses the dynamic communication area of a collector robot. This involves a spatio-temporal planning. Thus, let us define some location of the scenario as x. Each position is visited at some time t, so we can define a node $n = [x\ t]^T$ to the position and time in the scenario. Throughout this chapter $x(n)$ and $t(n)$ will denote position and time of some node n. Since the space is three-dimensional, the distance between a pair of nodes includes the difference between times, besides the Euclidean distance. Thus, the distance between any two nodes n_1 and n_2 is expressed as:

$$d(n_1, n_2) = \begin{bmatrix} \|x(n_1) - x(n_2)\| \\ |t(n_1) - t(n_2)| \end{bmatrix} \tag{6.1}$$

We denote τ as a trajectory travelled by an agent, that can be defined as a sequence of contiguous nodes $\tau = [n_0, n_1, ...n_N]$, where N denotes the number of nodes in the trajectory.

The expression $x(\tau)$ refers to each position of the trajectory and $t(\tau)$ to each of their respective times. We define $T(\tau)$ as the total time to travel the trajectory τ. The worker must communicate with a collector mate, which is assigned for relaying the information to the base station. The trajectory of the collector mate is expressed as τ_c, and it generates a dynamic communication area $A(\tau_c)$. This area is also composed by nodes n formed by positions and times. The details of computation of this area are explained in Section 6.4. An example with a scenario in presence of obstacles is depicted in Fig.6.2(a).

Knowing the size of the packages to transmit, the worker determines the minimum time to fulfill the transmission, $t_{c_{min}}$. So, the time that the trajectory of the worker must remain within $A(\tau_c)$, to fulfill the data transmission, is denoted with $t_c(\tau)$. In the case of having a primary goal to reach, the worker must traverse the communication dynamic area $A(\tau_c)$ and then reach the primary goal. In absence of a goal, the obtained trajectory only must remain within $A(\tau_c)$ during $t_{c_{min}}$. This can be formally expressed as:

$$\begin{aligned} \tau^* = & \underset{\tau}{\operatorname{argmin}}(J(\tau)) \\ & \text{subject to} \quad t_c(\tau) \geq t_{c_{min}} \end{aligned} \tag{6.2}$$

where the cost is computed using the normalized times and distances of the trajectories, expressed as:

$$J(\tau) = w_t\overline{t(\tau)} + w_d\overline{d(\tau)} \qquad \overline{(\cdot)} = \frac{(\cdot)}{max(\cdot)} \tag{6.3}$$

The parameters $w = (w_t, w_d)$ represent the weighting factors of the time and distance, respectively. In the present work we have considered only the time parameter for the proposed sampling-based planner, presented in Sect.6.5. So we set $w = (1, 0)$, for the proposed TC-RRT planner. Despite this, both parameters, distance and time, in eq.6.3 can be used without changes in the method.

6.4 Dynamic communication area

In order to synchronize with the collector, the worker has to know where and when to do so. The collector may be in movement or not, so instead of defining a different strategies for communication, we obtain its communication area. This way, it does not matter if the collector is moving or not. The planner only obtains a trajectory that traverses this area during $t_{c_{min}}$. In the case of moving, the collector drags its communication area through its trajectory, becoming dynamic. We consider that both, worker and collector, are equipped the same wireless antennas, that have a limited signal range. Therefore, we can define the communication area as all the nodes n of the scenario that are within the signal range of the antenna and with non-obstructed *line-of-sight*. Formally expressed as:

$$A(\tau_c) : \{n \mid dist(\tau_c, n) \wedge LoS(\tau_c, n)\} \tag{6.4}$$

where $LoS(\tau_c, n)$ is a boolean function that computes if exists *line-of-sight* between the trajectory and the node, using the Bresenham algorithm [70]. $dist(\tau_c, n)$ function is the condition of distance between the trajectory τ_c and the node n, formulated as:

$$dist(\tau_c, n) : \begin{bmatrix} \|x(\tau_c) - x(n)\| \leq d_{th} \\ |t(\tau_c) - t(n)| \leq \epsilon \end{bmatrix} \tag{6.5}$$

The distance threshold d_{th} is established based on the propagation parameters which assure communication, according to the communication model [71]:

$$P_{RX} = P_{TX} - 10\gamma log_{10}(d_{th}); \qquad d_{th} = 10^{\frac{P_{TX}-P_{RX}}{10\gamma}} \tag{6.6}$$

where P_{TX}, P_{RX} represent the transmitted/received power and γ is the path-loss exponent. The example of dynamic communication area of a mate is depicted in Fig.6.2(a).

Clearly not all the area is necessary, since the worker and the collector start their trajectories from different locations. So, some nodes of $A(\tau_c)$ are not reachable by the worker. Which means that including all the nodes of $A(\tau_c)$ only will increase the computation time to analyze the possible trajectories. Thus, we find out the feasible nodes where and when the worker could reach the collector at its trajectory:

$$A_{reach} : \{n \in A(\tau_c) \mid \|x_w - x(n)\|/v_w \leq t(n)\} \tag{6.7}$$

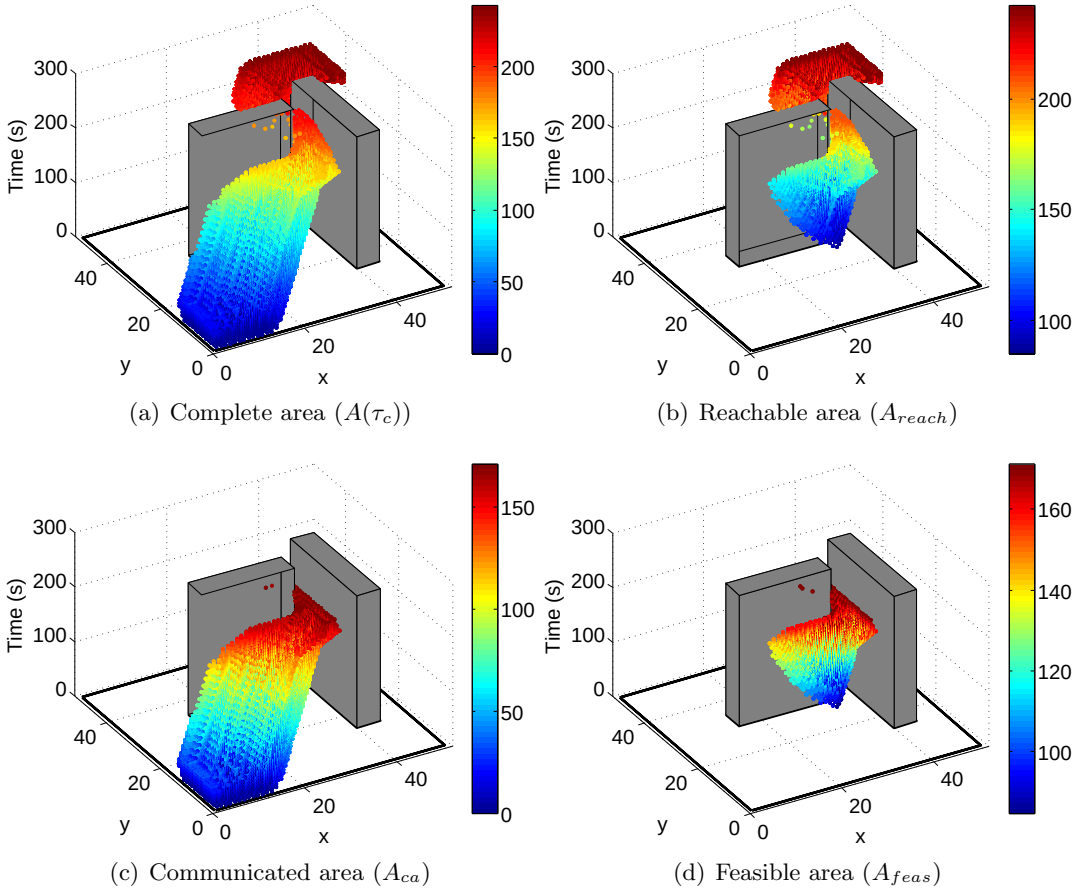


Figure 6.2: Communication area decomposition. Gray objects are obstacles.

where x_w is the initial position of the worker and v_w is its maximum attainable speed by the worker. The reachable area is illustrated in Fig.6.2(b).

The amount of information which the worker must share requires a minimum time for transmission ($t_{c_{min}}$). So only those nodes of the communication area, which will guarantee the transmission of the data, are considered by the planner. Thus, we define the communication assurance area (Fig.6.2(c)), that guarantees the complete information transmission as:

$$A_{ca} : \{n \in A(\tau_c) \mid t(A(\tau_c)) \leq T(\tau_c) - t_{c_{min}}\} \tag{6.8}$$

where $T(\tau_c)$ denotes the time of the trajectory of the collector. In other words, the last moment which the collector can exchange data.

In conclusion, the velocity of the worker and the required communication time constrain the area. Thus, we define the feasible area, as the area which must be reached in order to assure the communication, depicted in Fig.6.2(d):

$$A_{feas} : A_{reach} \cap A_{ca} \tag{6.9}$$

If $\exists n_{feas} \in A_{feas}$, it means that the mission may be accomplished if the location of $x(n_{feas})$ is reached no later than $t(n_{feas})$. However, if $A_{feas} = \emptyset$, there is no solution.

Therefore, the worker guides the search to the goal through this communication area, in order to synchronize with the relay. This procedure is explained for both, the sample-based and optimal planners in sections 6.5-6.6, respectively.

6.5 Sampling-based trajectory planner

6.5.1 Method description

The trajectory planner uses as base method the basic RRT. The proposed TC-RRT (Alg.10) has the same structure, but with different sampling and parent selection functions. First of all, let us define each node in the tree as $z = [x\ p\ t\ a\ t_c]^T$, where $x(z)$ and $p(z)$ are the position and the parent of z, $t(z)$ is the time to reach z, $a(z)$ is a boolean to indicate if z is within $A(\tau_c)$, and $t_c(z)$ is the communication time accumulated up to z in the tree. All these variables are computed when the node is inserted in the tree, with *InsertNode* procedure. In line 1 of the algorithm the tree $\mathcal{T}$ is initialized from the starting position of the worker denoted as x_{ini}. At each iteration our algorithm generates a random sample x_{sample} in the workspace outside the obstacles, in l.3, using *AreaSample* defined in Alg.10. The way to generate samples depends on if they are within or outside A_{feas}. This sample is connected to a parent in the tree by means of *AreaNearest* procedure in l.4, defined in Alg.13, selecting the node z_{near} that provides the fastest movement. As in the basic RRT, the *Steer* function cuts the stretch of the line between the generated sample (x_{sample}) and the position of the selected parent z_{near}, l.5. If the new branch is not obstructed by some obstacle, the node is inserted in the tree $\mathcal{T}$, l.6-8. After expanding K nodes, there may be several branches or trajectories which achieve the goal. All of them fulfill the condition of the communication time, thus we select the fastest trajectory, as defined in eq.6.2. The algorithm iterates until expanding the number of nodes specified by the user. This way, the environment is rapidly explored, although without the optimal solution, because of the randomized nature of the algorithm.

Two examples of tree computation and their respective trajectories are shown in Fig.6.3. Let us explain in more detail the different parts of the algorithm.

InsertNode routine (Alg.11) receives the position of a new node and the parent, and computes the time to reach the node from the parents position (l.1), where v represents the velocity of the worker. Then, in l.2, it computes the total time to reach it from the root or the initial position. It checks the presence in the area by means eq.6.1 in l.3, where d_ϵ represents a small value of distance. Note that this operation is quite lightweight, because Δt_{col} selects

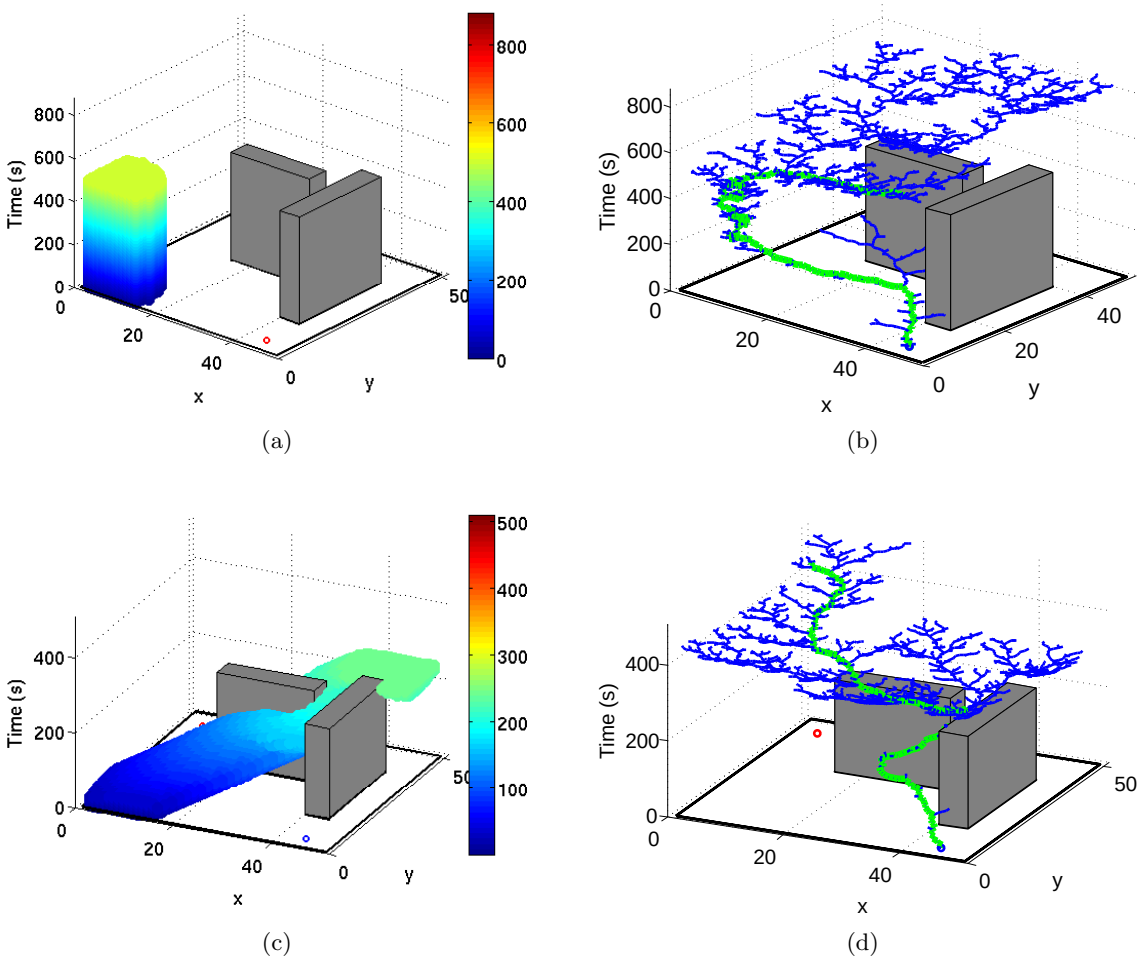


Figure 6.3: Tree (trajectory) computation for different scenarios. In (a), a worker that has collected data at [45,5] (red circle), must share information with a static collector at [5,5], then return to the initial position. In (b), the tree explores the environment, looking for the static communication area of the collector. When $A(\tau_c)$ is achieved, the worker remains within $A(\tau_c)$ until finishing the data exchange. Finally, $\mathcal{T}$ is expanded to the goal. The obtained trajectory is depicted in green. In (c), the collector is moving from [5,5] to [45,45], thus the area is dynamic. The worker must transmit data, before reaching the goal at [5,45]. In (d), $\mathcal{T}$ intercepts the communication trail area of the collector as fast as possible, remains in $A(\tau_c)$ until the end of communication, then achieves the primary goal.

Algorithm 10 TC-RRT

1: $\mathcal{T}$ ←InsertNode$(x_{ini}, 0)$
2: **for** i=1:K **do**
3: x_{sample} ← AreaSample
4: z_{near} ← AreaNearest(x_{sample})
5: x_{new} ← Steer$(x_{sample}, x(z_{near}))$
6: **if** ObstacleFree(x_{new}) **then**
7: $\mathcal{T}$ ←InsertNode(x_{new}, z_{near})
8: **end if**
9: **end for**
10: **return** $\mathcal{T}$

Algorithm 11 InsertNode(x_{new}, z_{near})

1: $\Delta t = \frac{\|x_{new} - x(z_{near})\|}{v(x_{new}, x(z_{near}))}$
2: $t_{new} = t(z_{near}) + \Delta t$
3: $a_{new} = \exists n \ : \ d([[x_{new} \ t_{new}]^T, n]) \leq [d_\epsilon \ \Delta t_{col}]^T, n \in A(\tau_c)$
4: **if** a_{new} & $a(z_{near})$ **then**
5: $t_{c_{new}} = t_c(z_{near}) + \Delta t$
6: **else**
7: $t_{c_{new}} = 0$
8: **end if**
9: $\mathcal{T} \leftarrow [x_{new} \ z_{near} \ t_{new} \ a_{new} \ t_{c_{new}}]^T$

only one horizontal slice of $A(\tau_c)$. If both, the parent and the node, are within $A(\tau_c)$, the time of l.1 is accumulated as communication time, l.4-8. And finally, all the obtained variables are inserted as a new node in the tree in l.9.

AreaSample (Alg.12) function delimits the sampling space and generates a new random sample. When some node is introduced into the communication area on time, i.e. in A_{feas} (l.1), and does not fulfill the condition of communication time of eq.(6.2) ($t_c(z) < t_{c_{min}}$), the tree is expanded through $A(\tau_c)$. The algorithm selects the node which accumulates the maximum communication time, z_a in l.2. Then, choosing a greater time of the collectors trajectory, using t_{min}, t_{max} in l.3, it delimits the sampling space by the distance d_{th}. The minimum and maximum bounds of the space where the random samples will be generated, are

Algorithm 12 AreaSample

1: **if** $\exists z : \{dist([x(z) \ t(z)]^T, n), n \in A_{feas} \wedge t_c(z) < t_{c_{min}}\}$ **then**
2: $z_a = \text{argmax}_z(t_c(z))$ ▷ Node which accumulates the maximum communication time
3: $x_m^- = x : \|x - x_m(t(z_a) + t_{min})\| \leq d_{th} \quad x_m^+ = x : \|x - x_m(t(z_a) + t_{max})\| \leq d_{th}$
4: $x_{sample} \leftarrow rand(x_m^-, x_m^+)$
5: **else**
6: $x_{sample} \leftarrow rand(x_{min}, x_{max})$ ▷ x_{min}, x_{max} are the limits of the scenario
7: **end if**
8: **return** x_{sample}

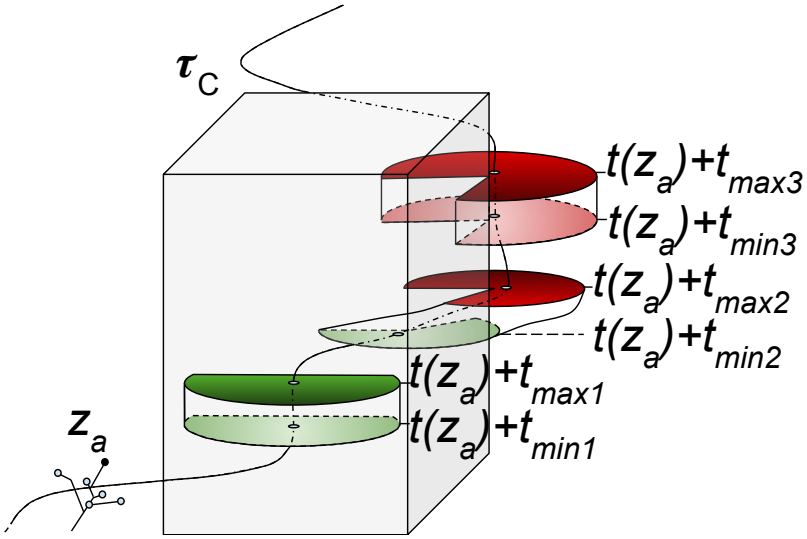


Figure 6.4: Sampling space within communication area $A(\tau_c)$. The maximum communication time is achieved with the node z_a, green and red slices depict the visible and obstructed parts of $A(\tau_c)$, respectively. The obstructed parts produce deadlock situations, because the tree attempts to expand a branch that collides with an obstacle.

x_m^- and x_m^+, respectively. The selection of time limits is made in accordance with the relative speed between the collector and the worker, setting $t_{min} = \Delta t_{col} v_{col}/v_r$ and $t_{max} = t_{min} + \Delta t_{col}$, see Fig. 6.4. In the opposite case, if there exist no nodes which have entered within the communication area, A_{feas}, or the communication time has already been accumulated, so that $t_c(z) \geq t_{c_{min}}$, the samples are generated outside the obstacles in all the scenario, l.5-7.

AreaNearest (Alg.13) finds out the best parent node z_{near} of the tree to connect the generated sample. The procedure is represented in Fig.6.5. If there are no nodes of the tree inside $A(\tau_c)$, l.1, the new sample is connected to parents that provide the fastest movement, l.2. In the case that some node is within $A(\tau_c)$, l.3, and accomplishes the condition of eq.6.2, l.4, it selects all the nodes accomplishing that condition, z_c in l.5, and chooses the parent of the fastest movement, in l.6. When there are no nodes in $A(\tau_c)$ which accomplish $t_c(z) \geq t_{c_{min}}$, l.7, the suitable candidates to parent are those that have accumulated the maximum communication time of the entire tree, l.8. To increase the number of suitable candidate parents, a relaxation time t_r is applied in l.9. It is computed as $t_r = n_{ts} d_{max}/v_{max}$, where n_{ts} is the number of timesteps and d_{max} and v_{max} are the maximum attainable step and speed of the worker, respectively. Finally, the chosen parent is the one which provides the minimal time, l.10. Therefore, the selected parent is a node of the tree that reduces the time to reach the generated random sample, keeping the tree within $A(\tau_c)$, see Fig.6.6.

6.5.2 Results

In this section we discuss the performance of the TC-RRT and present its limitations. We test the method in the scenario of Fig.6.3(c), where a worker has to capture a moving collector to transmit the collected data. The technique is evaluated for different system parameters. These parameters are the number of extended nodes, the communication time $t_{c_{min}}$ and the

Algorithm 13 AreaNearest(x_{sample})

if $\nexists z : a(z)$ **then**
  $z_{near} = \text{argmin}_z \left(\frac{\|x_{sample} - x(z)\|}{v(x_{sample}, x(z))} \right)$
else
  if $\exists z : t_c(z) \geq t_{c_{min}}$ **then**
    $z_c = \{z \mid t_c(z) \geq t_{c_{min}}\}$
    $z_{near} = \text{argmin}_{z \in z_c} \left(\frac{\|x_{sample} - x(z)\|}{v(x_{sample}, x(z))} \right)$
  else
    $z^* = \text{argmax}_z(t_c(z)) \ : \ a(z)$ ▷ Maximum communication time of the entire tree
    $z_c = \{z \in z^* \mid t_c(z) \geq t_c(z^*) - t_r\}$
    $z_{near} = \text{argmin}_{z \in z_c}(t(z))$
  end if
end if
return z_{near}

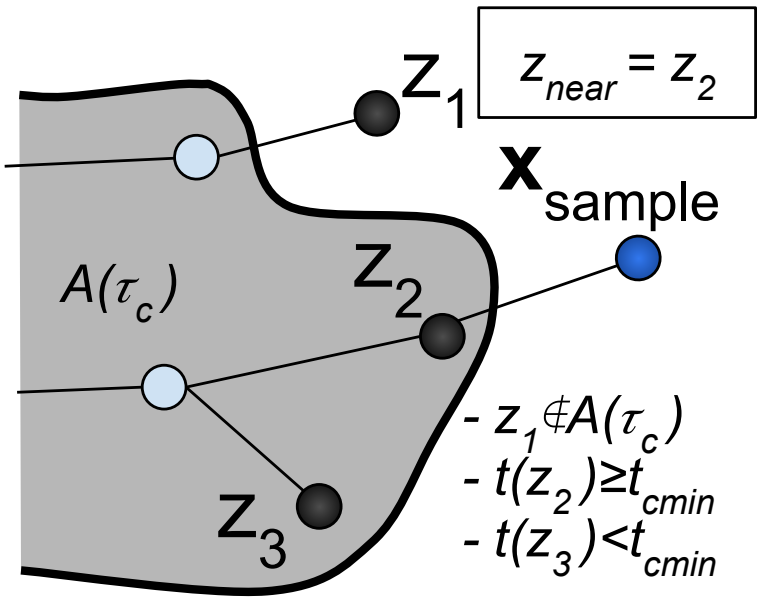


Figure 6.5: Selection of suitable parent candidates. In this case, the best one is z_2, because it accomplishes the condition of minimal communication time, of eq.(6.2).

relative velocity between the worker and the collector. The communication time is selected as the percentage of the time available to transmit data, that is, the time of A_{reach}. We select: (1000, 2000, 5000, 10000) nodes, (25%, 50%, 75% and 90%) of time of A_{reach} and (-10%, 0%, +10%) for the speed of the worker with respect to the collector's speed. Varying the relative speed, the slope of $A(\tau_c)$ changes. Consequently, the time of A_{reach} changes as well, so that, the times for the different relative speeds are (182, 163 and 148) seconds, respectively. Increasing the speed of the collector or reducing the speed of the worker, the slope is reduced,

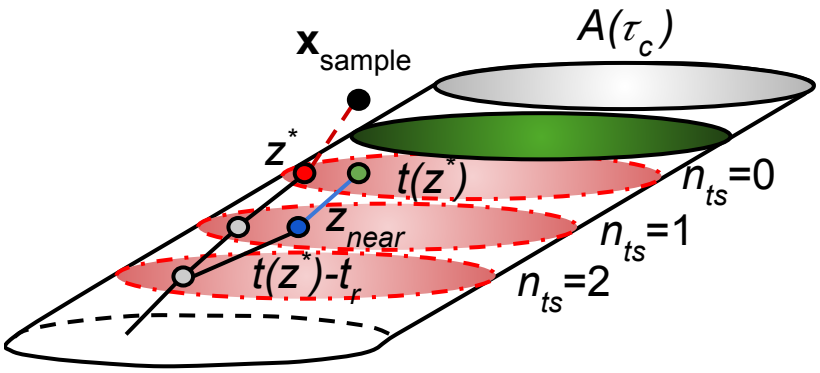


Figure 6.6: Refine parent selection with the relaxation time. x_{sample} is generated in the sampling space (green slice), but connecting to the best parent z^* (in red), the tree leaves $A(\tau_c)$. Using a relaxation time t_r, it connects to a suitable parent z_{near} (in blue), which keeps the tree within $A(\tau_c)$.

which means, that it is more difficult to remain synchronized. The performance of the method is tested in terms of the solution cost, i.e. total time for the trajectory to the goal ($t(\tau)$), success rate and the execution time. A successful trajectory is the one that fulfills the communication time condition of eq.6.2. The algorithm is implemented in MatLab and tested on a machine Intel Core i7 clocked at 3.40 Ghz and 8Gb of memory. The results are presented in Fig.6.7 and an example of tree expansion for the tested scenario can be found in the video [1].

The cost of the solution does not change significantly with the number of nodes, Fig.6.7(a),6.7(c),6.7(e). This is because the base method is the basic RRT, thus when some branch reaches A_{reach}, as well as the goal, it is unlikely that some other drastically different branch will reach the same positions. However, the success rate increases when some nodes threshold is exceeded, Fig.6.7(b)6.7(d)6.7(f). This is more remarkable in the transition from 1000 to 2000 nodes and for 90% of time of A_{reach}.

When the speed of the worker increases with respect to the collector, the success of the algorithm increases. Logically, the worker reaches the communication area and the goal faster, thereby reducing the solution cost. High values of $t_{c_{min}}$ reduce the time range to achieve the communication area on time. This fact, altogether with the randomized nature of the algorithm makes impossible to guarantee the communication. Even in the best case, 10% of extra speed, the success rate is just 60% for 90% of available A_{reach} (equivalent to an interval of 8 seconds).

Expanding over 2000 nodes for this scenario, the solution is stabilized, the cost and success rate remain practically constant. The medium values of execution times were (1.3, 2.25, 5.47, 11.76) seconds for (1000, 2000, 5000, 10000) nodes respectively. Considering that for this scenario the results are good enough with few nodes (2000), it takes about 2 seconds to

[1] http://robots.unizar.es/data/videos/robot17yamar.mp4

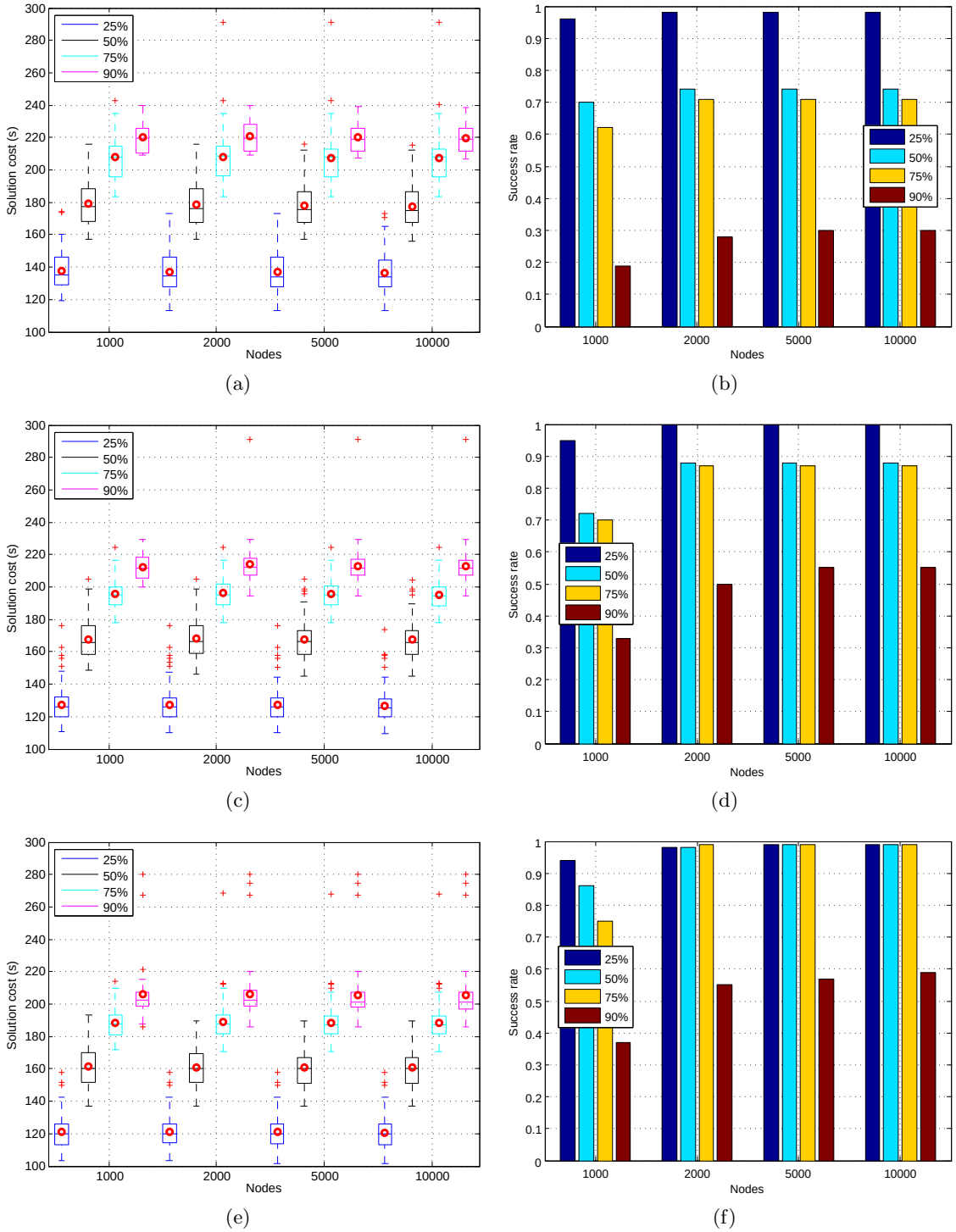


Figure 6.7: Results for 100 random trials in the scenario of Fig.6.3(c). Each row represents -10%, 0% and +10% of relative speed, with respect to the speed of the collector. The red circles in (a)(c)(e) represent the mean value of the costs of the trajectories.

obtain a fairly steady solution.

Table 6.1: Results for different timesteps, $n_{ts} = 0, 1, 2$.

	Execution time				**Success rate**				**Solution cost**			
n_{ts}	**1000**	**2000**	**5000**	**10000**	**1000**	**2000**	**5000**	**10000**	**1000**	**2000**	**5000**	**10000**
0	1.164	2.064	5.16	11.16	0.74	0.88	0.9	0.91	186.1	186.87	186.58	185.81
1	1.27	2.166	5.13	11.27	0.76	0.95	0.95	0.96	187.09	188.36	187.05	187.14
2	1.32	2.21	5.29	11.39	0.75	0.99	0.99	0.99	188.05	188.05	187.38	187.06

As described in Sect.6.5.1, we employ a relaxation time t_r to increase the success rate of the algorithm. We set the number of timesteps $n_{ts} = (0, 1, 2)$, and the results are depicted in Table 6.1. The solution cost does not vary significantly, it is slightly lower without adding t_r. Logically, increasing n_{ts}, the number of evaluated candidates increases, but the execution time do not raise significantly. At the same time, using the relaxation factor increases the success rate, because the tree expansion avoids getting stuck within coverage area. This means that it is worthwhile to use this parameter. The simulations of Fig.6.7, were performed with $n_{ts} = 2$.

The method works under the hypothesis that the global plan for the robots will be maintained. If the scenario or the strength of the signal change, it might produce variations in the planned trajectories executed by the agents. In this case, if the robot trajectories do not change drastically and only are deviated from the original trajectory, the technique could still work by extending the communication area to be explored around the original communication planned area. Otherwise, another more costly solution can be that the collector returns to a previous communicated position in its trajectory, relaunching the planner. This is obviously a problem that will be formally dealt in future work.

6.6 Optimal trajectory planner

6.6.1 Method description

The optimal trajectory planner has a similar behaviour to the previous TC-RRT. It obtains trajectories that traverse the dynamic communication area of a collector. This means that the suitable trajectories traverse A_{reach}. As described in the previous chapters, a unique gradient computation using the Fast Marching Method, provides all the possible paths to all the points of the scenario. Thus, we compute two gradients: one from the initial position of the worker expressed as $\nabla D(x_{ini})$ and another from the goal position defined by $\nabla D(x_{goal})$. This way, descending both gradients from any position that belongs to A_{reach}, allows building a path which traverses the communication area. For the gradient propagation we use Alg.21 and propagating the wavefront throughout the entire scenario using eq.A.2.

The communication area A_{reach} described in the Section 6.4, allows multiple communication possibilities. We consider three types of trajectories according to the minimal communication time ($t_{c_{min}}$), the distance to A_{reach}, and the relative velocities between the worker and the collectors.

– If the worker is faster or is close to A_{reach}, and $t_{c_{min}}$ is short, one of the options is to *intercept* the collector, reaching the area as fast as possible. The worker traverses it, exchanging the data, and proceeds to reach the goal. This procedure is illustrated in

Fig.6.8(a) and it attempts to minimize the time of the mission.

– If the priority of the mission is to reduce the travelled distance, the worker adopts a *lazy* approach. Reaching the area, *waiting* in some position transmitting the data, meanwhile the collector goes towards its goal. The example of this kind of trajectory is depicted in Fig.6.8(b). This situation and the previous one appear when exist some points where the interval, between the lower and upper limits of the communication area, is greater than $t_{c_{min}}$.

– When $t_{c_{min}}$ is a significant time, it is necessary to *follow* the collector, as illustrated in Fig6.8(c). Therefore, the worker finds out the optimal position where to intercept to the collector, follows it until complete the communication and, then it continues to its goal.

The three situations are analyzed using three routines: $Intercept$, $Wait$ and $Follow$. Each of them computes all the possible trajectories that accomplish the connectivity time constraint defined in eq.6.2. Then, the optimal trajectory is chosen based on the optimality criteria of eq.6.3.

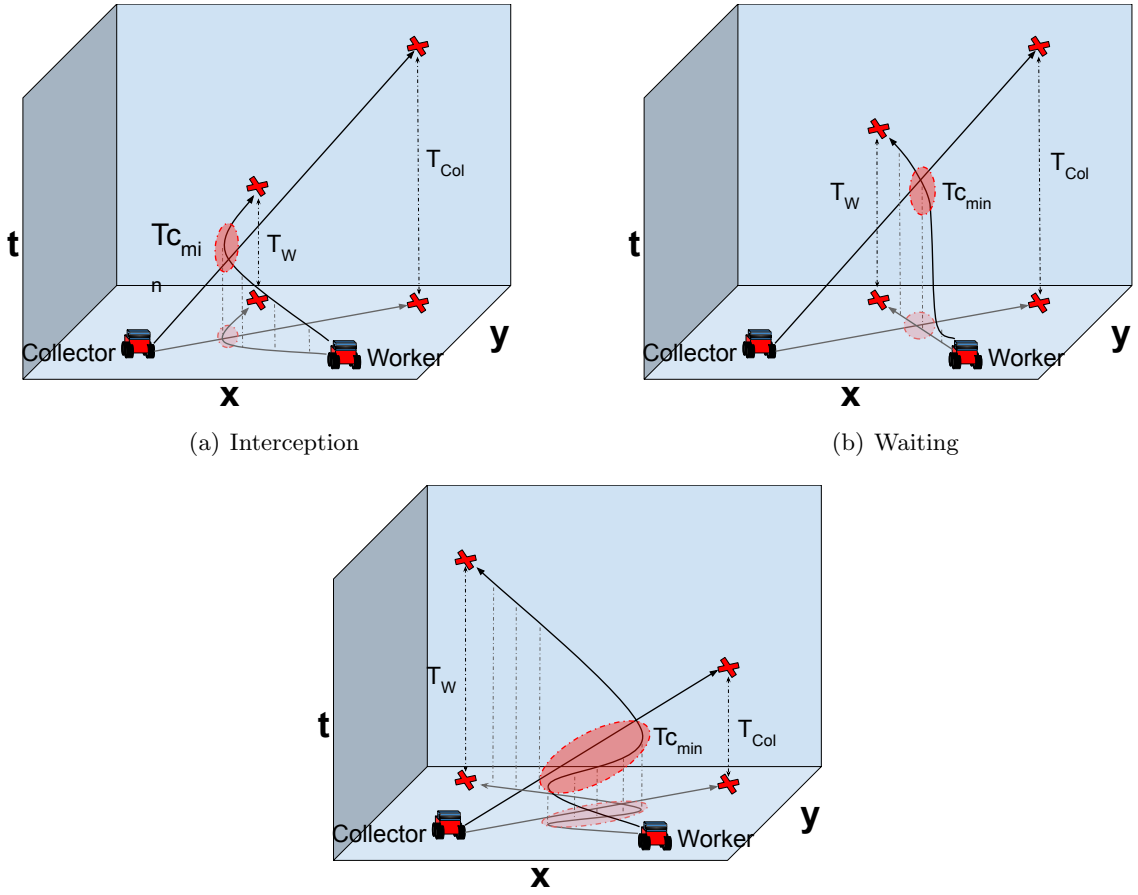
The first one, the $Intercept$ routine, is described in Alg.14. The algorithm evaluates all the positions of A_{reach} in l.2. The trajectory is built by descending the gradient $\nabla D(x_{ini})$, from each position of A_{reach}, in l.3. Then the method, obtains the first point where the worker achieves the area in l.4-5. Here, the worker waits to the first communication with the collector defined as $t_{min}(A_{reach})$, l.6-8. δt represents a temporal or vertical slice of the communication area. Finally, the remaining trajectory to the goal, which traverses the area A_{reach} to communicate with the collector, is built. Firstly, descending the gradient to the goal $\nabla D(x_{goal})$ in l.9. And then concatenating with τ_{ini}, in l.10.

The $Wait$ procedure, Alg.15, follows the same principle as the interception. The difference is that the trajectory remains in the same position until the collector leaves, dragging the communication area or until the communication time is high enough to fulfill the data transmission $t_{c_{min}}$, in l.4-7. In l.4 *end* stands for the last position of the trajectory. The rest of the procedure is the same.

Algorithm 14 Intercept routine

Require: $\nabla D(x_{ini}), \nabla D(x_{goal}), A_{reach}$

1: $\tau_{tot} \leftarrow \emptyset$
2: **for each** $n \in A_{reach}$ **do**
3: $\quad \tau_{ini} \leftarrow build\_traj(n, \nabla D(x_{ini}))$
4: $\quad n_{inter} \leftarrow \{n \mid x(\tau_{ini}) \cap x(A_{reach})\}$
5: $\quad n_i \leftarrow n_{inter}[1]$ ▷ First position of the intersection
6: $\quad$ **while** $t(n_i) < t_{min}(A_{reach})$ **do**
7: $\quad\quad \tau_{ini} \leftarrow [x(n_i) \;\; \Delta t]$ ▷ Wait until area arrives
8: $\quad$ **end while**
9: $\quad \tau_{goal} \leftarrow build\_traj(n, \nabla D(x_{goal}))$
10: $\quad \tau_{tot} \leftarrow \tau_{ini} \cup \tau_{goal}$ ▷ Concatenate paths
11: **end for**
12: **return** τ_{tot}



(a) Interception

(b) Waiting

(c) Following

Figure 6.8: Possible trajectories for synchronization, based on transmission time. In (a), the worker agent is faster, it intercepts its collector mate, exchanges data and moves to the goal. In (b), the worker waits to the collector, communicates the data and goes to the goal. In (c), the transmission time $t_{c_{min}}$ is large. The worker intercepts the collector, follows it while exchanges data and reaches the goal.

The *Follow* routine is summarized in Alg.16 and is somewhat more complex. As described above, in this case the minimum communication time for exchange requires to follow the collector. Thus, the trajectory necessarily requires to enter into A_{feas}, whose points assure a complete data exchange. So, the trajectories are obtained, remaining in the same point until first communication with the collector, as described in l.2-7. From now on the worker needs to move inside the communication area A_{reach}, so it is necessary to compute a new gradient for this purpose, and obtaining new trajectories inside this area. Since A_{feas} can have a large number of points, it is computationally expensive to calculate $|A_{feas}|$ gradients. Thus, we choose the optimal trajectories according to eq.6.2 in l.8. The gradient has to cover only the required area A_{reach}, thus the rest of the points are discarded, in l.9, in order to reduce the time of computation. The point where the trajectory leaves the communication area A_{reach} will be some point of its contour. So the gradient $\nabla D(\mathbf{x}_{inter})$ is computed only within A_{reach}

Algorithm 15 Wait routine

Require: $\nabla D(x_{ini}), \nabla D(x_{goal}), A_{reach}$

1: $\tau_{tot} \leftarrow \emptyset$
2: **for each** $n \in A_{reach}$ **do**
3: $\quad \tau_{ini} \leftarrow build\_traj(n, \nabla D(x_{ini}))$
4: $\quad n_e \leftarrow \tau_{ini}[end]$ ▷ Last position of τ_{ini}
5: $\quad$ **while** $t(n_e) < min(t_{max}(n), t(n_e) + t_{c_{min}})$ **do**
6: $\quad\quad \tau_{ini} \leftarrow [x(n_e)\ \Delta t]$ ▷ Wait until area leaves
7: $\quad$ **end while**
8: $\quad \tau_{goal} \leftarrow build\_traj(n, \nabla D(x_{goal}))$
9: $\quad \tau_{tot} \leftarrow \tau_{ini} \cup \tau_{goal}$ ▷ Concatenate paths
10: **end for**
11: **return** τ_{tot}

Algorithm 16 Follow routine

Require: $\nabla D(x_{ini}), \nabla D(x_{goal}), A_{reach}, A_{feas}, F, w$

1: $\tau_{tot} \leftarrow \emptyset$
2: **for each** $n \in A_{feas}$ **do**
3: $\quad \tau_{ini} \leftarrow build\_traj(n, \nabla D(x_{ini}))$
4: $\quad$ **while** $t(\tau_{ini}[end]) < t_{min}(A_{reach})$ **do**
5: $\quad\quad \tau_{ini} \leftarrow [x(\tau_{ini}[end])\ \Delta t]$ ▷ Wait until area arrives
6: $\quad$ **end while**
7: **end for**
8: $\tau^*_{ini} \leftarrow choose\_traj(\tau_{ini}, w)$ ▷ Using eq.6.2
9: $F_{area} \leftarrow F(!A_{reach}) = 0$ ▷ Only the points of the area
10: $\mathbf{x}_{inter} \leftarrow \tau^*_{ini}[end]$ ▷ Last positions of τ^*_{ini}
11: $\nabla D(\mathbf{x}_{inter}) \leftarrow gradient(\mathbf{x}_{inter}, F_{area})$
12: $\mathbf{n}_{cont} \leftarrow contour(A_{reach})$
13: **for each** $n \in \mathbf{n}_{cont}$ **do**
14: $\quad \tau_{inter} \leftarrow build\_traj(n, \nabla D(\mathbf{x}_{inter}))$
15: $\quad \tau_{goal} \leftarrow build\_traj(n, \nabla D(x_{goal}))$
16: $\quad \tau_{tot} \leftarrow \tau^*_{ini} \cup \tau_{inter} \cup \tau_{goal}$
17: **end for**
18: **return** τ_{tot}

in l.10-11, in order to obtain all the possible trajectories to the contour points obtained in l.12. Finally, the stretch of the trajectories for synchronization are obtained descending this gradient in l.14. And the remaining stretch to the goal in l.15.

The entire procedure to obtain the optimal trajectory is shown in Alg.17. The chosen trajectory is the one that minimizes the constrained criteria defined in eq.6.2. It is important to highlight that the expensive procedures as area computation and segmentation are executed once. This also occurs with the gradient computation, it is only executed three times: one for initial position, one for the goal position, and another one in Alg.16 to obtain the stretch within A_{reach} for *Follow* procedure. The time to descend the gradient with *build_traj* routine is much less time-consuming. This makes it possible to evaluate a large amount of possible

trajectories without a drastic increase of the computation time.

6.6.2 Results

In this section we present the simulated results of the proposed optimal planner. We consider two cases: one worker to synchronize with a single collector in the environment and one worker to synchronize with one collector from multiple present in the environment.

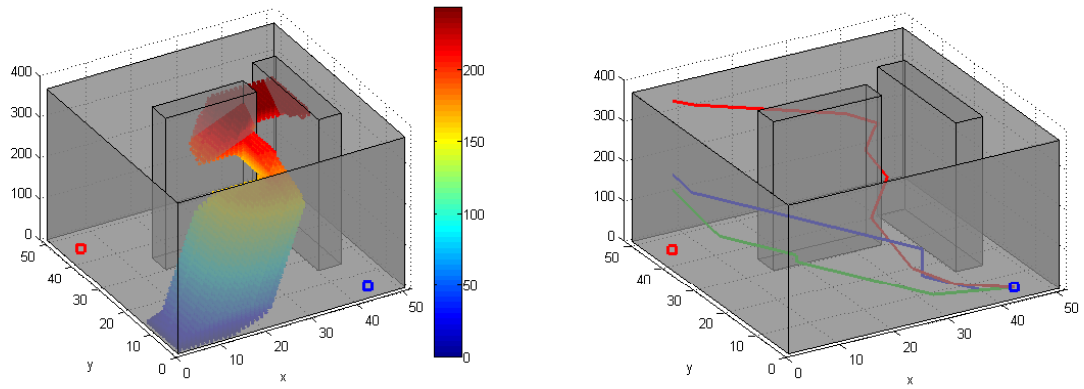
Single collector case

The first simple example serves to illustrate the execution of the developed optimal planner. Consider a simple scenario, in presence of obstacles, where two robots are exploring the

Algorithm 17 Complete routine

Require: $x_{ini}, x_{goal}, \tau_c, F, w$

1: $A(\tau_c) \leftarrow com\_area(\tau_c)$ ▷ Using eq.6.4
2: $[A_{reach}, A_{feas}] \leftarrow area\_parts(A(\tau_c))$ ▷ Using eq.6.7-6.9
3: $\nabla D(x_{ini}) \leftarrow gradient(x_{ini}, F)$ ▷ Using eq.3.2
4: $\nabla D(x_{goal}) \leftarrow gradient(x_{goal}, F)$ ▷ Using eq.3.2
5: $\tau_i \leftarrow intercept(\nabla D(x_{ini}), \nabla D(x_{goal}), A_{reach})$ ▷ Alg.14
6: $\tau_w \leftarrow wait(\nabla D(x_{ini}), \nabla D(x_{goal}), A_{reach})$ ▷ Alg.15
7: $\tau_f \leftarrow follow(\nabla D(x_{ini}), \nabla D(x_{goal}), A_{reach}, A_{feas}, F, w)$ ▷ Alg.16
8: $\tau^* \leftarrow choose\_traj(\{\tau_i, \tau_w, \tau_f\}, w)$ ▷ Using eq.6.2
9: **return** τ^*



(a) Communication area

(b) Obtained trajectories

Figure 6.9: Obtained trajectories in presence of one collector. In (a), the collector has gathered data at [5,5] and goes to [45,45] to transmit data to OP, creating a dynamic communication area $A(\tau_c)$. The worker, after having collected information in [45,5], goes towards [5,45]. In (b), the method obtains different trajectories according for different $t_{c_{min}}$ and robots velocities. The green, blue, and red lines represent the optimal trajectories obtained by interception, waiting, and following routines, respectively.

environment, depicted in Fig.6.9. The worker has gathered data and it must go to the next location to continue with this task. Meanwhile, a collector mate is travelling a constant trajectory and it proceeds to go to upload the information to the BS. From the initial plan, provided by BS, the worker knows which trajectory that travels the collector. Therefore, it obtains the time to transmit the data ($t_{c_{min}}$), and estimates the communication area ($A(\tau_c)$), according to the speed of the collector, Fig.6.9(a). We evaluate two situations: when the worker is faster and the amount of data is small, thus, $t_{c_{min}}$ is a little time, and when both agents have the same speed and $t_{c_{min}}$ is a long time.

The obtained trajectories are depicted in Fig.6.9(b), and results are shown in Table 6.2. The times are expressed in seconds and distances in meters. The Alg.17 is executed varying the weighting factor w of eq.6.3. Setting $w_t = 1$, the time must be minimized and the optimal trajectory (green line) is obtained by interception routine of Alg.14. However, establishing $w_d = 1$, the optimal trajectory (blue line) is provided by waiting routine of Alg.15, reducing the travelled distance, but increasing the time spent. In the case of increasing $t_{c_{min}}$, the optimal trajectory (red line) is obtained by $Follow$ method of Alg.16, because the others do not provide a solution, since they do not accomplish the constraint of eq. 6.2. We select $t_{c_{min}} = 165seg$, which represents almost 100% of time of A_{reach}, so that the worker deviates from its goal, following the communication area trace generated by the collector in movement.

Multiple collectors case

We also test our method for multi-robot scenarios, where there are multiple collectors in the environment. The simulations are performed in two scenarios, depicted in Fig.6.10. We consider all the collectors present in the environment as possible candidates to transmit the data. The worker has to obtain the optimal trajectory, choosing the best collector to share data with. We present the mean results based in 100 random collectors distributions for each scenario. That is, randomly generating the initial and goal locations of each collector. Obviously, such a scenario does not make sense in a real mission, because the collectors always have to go from and to the BS position. But, this way a more exhaustive analysis of the performance of the method is provided.

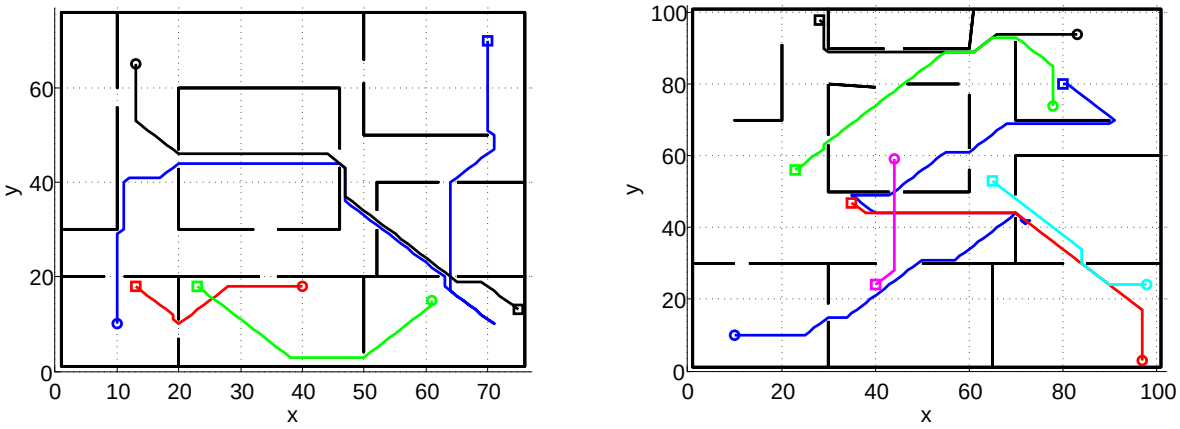
We evaluate the method performance for different communication necessities, so that, varying $t_{c_{min}}$. Although the algorithm uses all three routines: $Intercept$, $Wait$ and $Follow$, we want to assess them separately. So, we only use some of them, based on $t_{c_{min}}$. In order to assess $Intercept$ and $Wait$ routines, we choose 10 and 30 seconds for $t_{c_{min}}$. The routine $Follow$ obtains the optimal trajectories for large values of $t_{c_{min}}$, thus we select 75% and 100% of possible communication time, in other words, of A_{reach}. The chosen metric for the evaluation are the travelled distance and employed time, expressed in meters and seconds, respectively. We fix the speed of the worker to be twice the speed of the collectors.

As described in Sect.6.6.1, some procedures require a unique execution. Table 6.3 shows the times of these procedures for both scenarios. The time of computation of the gradients from start and goal position of the worker is around 4 sec. We also show the computation time to obtain the trajectories of the collectors (τ_c), form the communication area ($A(\tau_c)$) and decompose it, expressed in seconds. Obviously, for a larger scenario and a higher number of collectors in the team, the values are bigger.

In order to assess the optimality of the trajectories with time-constrained communication, we provide the values of distance and time of the direct trajectories, without connectivity

Table 6.2: Obtained results for scenario of Fig.6.9.

	Worker/Collector speed	$t_{c_{min}}$	$t(\tau)$	$d(\tau)$	$t_c(\tau)$
Intercept	2	30	147.68	65.36	33.94
Waiting	2	30	186.05	60.08	31.11
Following	1	165	369.14	92.28	166.23



(a) 3 team collectors.

(b) 5 team collectors.

Figure 6.10: Tested scenarios. Circles and squares represent initial and goal positions, respectively. Different color lines are trajectories of the collectors. The worker (blue circle) must reach a location (blue square), but previously, it must exchange information with some collector. In (a), the worker intercepts a collector, of black trajectory, and follows it until fulfills the data exchange. In (b), the worker follows the trajectory of the collector depicted with red line.

constraints, in Table Table 6.4. The distance and time of the direct path $d(\tau_{dir})$ and $t(\tau_{dir})$, respectively.

When we set $w_d = 1$, the distance of the trajectory must be minimized, so the optimal

Table 6.3: Computation times

Scenario	∇D (s)	τ_c $A(\tau_c)$	$A(\tau_c)$ segmentation
Fig.6.10(a)	3.81	18.29	15.42
Fig.6.10(b)	3.91	29.84	26.09

Table 6.4: Direct paths values

Scenario	$d(\tau_{dir})$ (m)	$t(\tau_{dir})$ (s)
Fig.6.10(a)	99.5	79.6
Fig.6.10(b)	122.75	98.2

trajectories are provided by $Wait$ routine. By contrast, if the chosen weight is $w_t = 1$, the trajectories come from $Intercept$ routine, at the expense of being longer. The mean results appear in Tables 6.5-6.6. The results are based on: the number of analyzed trajectories ($|\tau|$), time of computation (t_{comp}), the time $t(\tau^*)$, the distance $d(\tau^*)$ and the communication time $t_c(\tau^*)$ of the optimal trajectory, expressed as $t(\tau^*), d(\tau^*), t_c(\tau^*)$, respectively. The times are expressed in seconds and the distance in meters.

As can be seen, the trajectories provided by $Wait$ procedure have practically the same distance as the direct path $d(\tau_{dir})$ of Table 6.4. The times are reduced employing $Intercept$ routine, but it is difficult to obtain similar times to $t(\tau_{dir})$, because the worker considerably deviates from τ_{dir}. It is remarkable that the computational time of $Wait$ is smaller than for $Intercept$. This is due to the $Intercept$ method requires an extra time to intersect the trajectories with the area A_{reach}.

When the amount of data to transmit is large, $t_{c_{min}}$ is high. Therefore, the worker is forced to follow some collector until fulfill the data transmission. Obviously, the worker is drastically deviated from its original goal, but accomplishing the information exchange, as shown in Table 6.7. It is noteworthy that the computational time of $Follow$ is considerably lower than the other procedures. On the one hand, this is because the number of analyzed trajectories ($|\tau|$) is much lower, since the worker only must navigate within the communication area of some collector (A_{reach}), as described in Alg.16. And on the other hand, increasing the minimal communication time, leaves fewer trajectory possibilities. So, this fact also reduces the number of trajectories to analyze.

Table 6.5: Results for scenario of Fig.6.10(a).

$t_{c_{min}}$	**Routine**	$\|\tau\|$	t_{comp}	$t(\tau^*)$	$d(\tau^*)$	$t_c(\tau^*)$
10	Intercept	1677	24.01	105.82	100.9	17.57
	Wait	1677	12.7	115.13	99.7	13.58
30	Intercept	1677	23.94	121.57	102.75	32.4
	Wait	1677	12.7	134.86	102.75	32.09

Table 6.6: Results for scenario of Fig.6.10(b).

$t_{c_{min}}$	**Routine**	$\|\tau\|$	t_{comp}	$t(\tau^*)$	$d(\tau^*)$	$t_c(\tau^*)$
10	Intercept	3661	73.79	105.4	126.67	12.89
	Wait	3661	38.67	111.86	122.75	12.27
30	Intercept	3661	73.87	129.32	127.96	31.07
	Wait	3661	38.71	136.89	127.98	31.11

Table 6.7: Results of $Follow$ routine for scenario of Fig.6.10.

Scenario	$t_{c_{min}}$	$\|\tau\|$	t_{comp}	$t(\tau^*)$	$d(\tau^*)$	$t_c(\tau^*)$
Fig.6.10(a)	75%	945	5.82	148.14	117.22	73.76
	100%	234	1.8	149.06	121.28	77.84
Fig.6.10(b)	75%	1378	12.31	199.13	154.76	110.27
	100%	238	2.73	208.2	167.5	110.08

6.7 Conclusions

In the present chapter we have proposed two methods to compute trajectories to synchronize two agents in movement. The proposed planners, considering a known trajectory of a collector mate, obtain a trajectory for a worker agent in order to synchronize with the collector to fulfill the data exchange. Using either planners, the worker is able to obtain the best trajectory based on the amount of data to share, and does not require to use a specific task allocator for this purpose, since it is implicitly included in the proposed approaches.

Firstly, we have formally defined the dynamic communication areas of agents in movement. We have described the different parts of this area in order to obtain the useful part to compute the trajectories for synchronization.

Secondly, we have developed a sampling-based planner in order to obtain a fast solution, but obtaining suboptimal trajectories in terms of success rate and solution cost. As for all the sampling-based planners, the solution quality can be improved by increasing the number of extended nodes, but at the expense of requiring more time.

Finally, we have proposed a second approach to compute the optimal trajectories for synchronization. This method is based on the Fast Marching Method and by analyzing multiple possible solutions, it finds out the optimal trajectory, based on two different metrics: distance and time. It is able to obtain the optimal solution in much larger time than the proposed sampling-based technique. However, if the number of analyzed trajectories is not substantial, the computation time is not disproportionate. So, it is used in the following Chapter 7 for the worker agents in a complete data gathering mission, using one or more collectors.

As future work, we want to include the dynamic obstacles that can be present in the environment in both planners. For example, in scenarios with human presence, they must be included in the planning process, in order to plan collision free trajectories and at the same time to maintain the connectivity in order to not interrupt the synchronization.

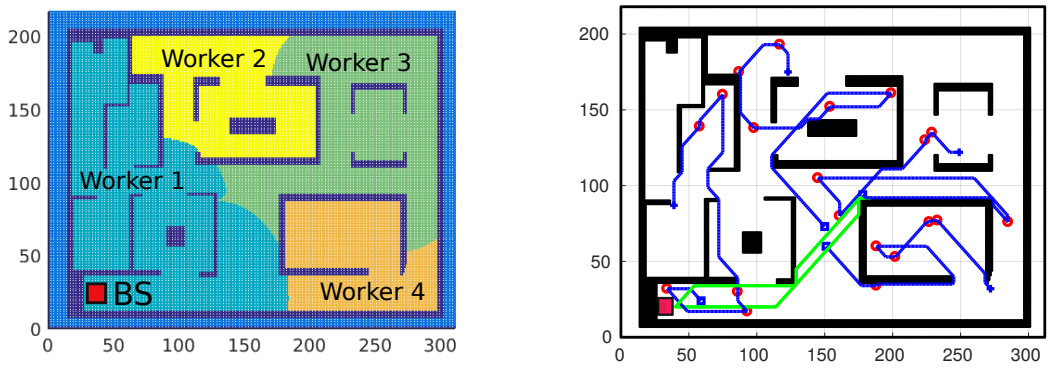
Chapter 7

Multi-agent coordination for on-demand data gathering with periodic information upload

7.1 Introduction

In this chapter we develop a method for planning and executing a multi-agent team deployment to gather data in some scenario. In every cycle of the the mission the robots have to reach different locations of interest, periodically requested by a static Base Station (BS), taking measurements and uploading the information to the BS. The BS will select new goals for the next cycle of request. As in the previous works, an absence of a wireless communication network, makes it impossible to upload the gathered information from the goals locations. Furthermore, the BS and the robots have a limited communication range, thus the robots will have to approach to the BS in order to upload the information. As described in the previous chapter, the idea is to use some agents in role of collectors and others in role of workers. Depending on times for transmission, for working, and for travelling, a balance between the number of workers and collectors has to be found to minimize the information refreshing time while maximizing the number of information packages delivered to the BS. To find a solution to this problem is the objective of the work developed here.

An assumption is that the goals will be every cycle uniformly distributed within the whole scenario, but changing from one cycle to another. We propose to divide the scenario into working zones for the agents, associating one per worker. These working zones are denoted as segments or partitions throughout this chapter. This way each worker will receive a number of goals approximately proportional to the size of the area. So the workload of the workers will be also proportional to that size. Although a first idea could be to segment the scenario in zones of similar area to balance the workload of the workers, it does not lead always to the best solution, as we will see later. The segments do not change during the mission, and the workers gather data from the goals of their associated segments, delivering the information either directly to the BS or to the closest moving collector, as illustrated in Fig.7.1. The workers use the optimal trajectory planner developed in the previous Chapter 6. This way, the team avoids meeting in static rendezvous points to redistribute the working areas of the agents. This would cause a deterioration of the refreshing time of the information. The collectors would need to wait for the workers at each cycle in order to communicate them the



(a) Segmentation

(b) Deployment

Figure 7.1: Example of data gathering mission, for 20 objectives per cycle, with 5 agents: 4 workers and 1 collector, 5 objectives/worker. In (a), the scenario is divided into 4 segments, 1 per worker. (b) depicts the trajectories of 4 workers (blue lines) visiting 5 objectives each (red circles) and going to transmit data. The collector trajectory is the green line.

distribution of the new segments as well as the next meeting point.

An example of these kind of missions is a fire monitoring, where the locations to monitor and also the frequency of their appearance can change over time. The proposed method can also fit with light changes in other applications such as: human-robot cooperation, where the workers are human operators performing some kind of task in some area and interact during a specified time with a moving robot; warehouse commissioning and logistics, in which the worker robots pick up items and bring them to some collector agents that deliver all the items to the depot point.

Therefore, here we develop a technique for planning and executing data gathering missions in connectivity constrained scenarios, working in three steps: (i) the scenario partition in several working areas or segments, each assigned to one worker agent; (ii) the computation of the number of collectors needed for the mission for that partition, their trajectories, and the assignment of workers to meet the collectors or directly to move towards the BS; (iii) the routing for each worker to visit the goals of its assigned area, and synchronize with its assigned collector in movement for the data exchange.

A centralized planner solves the points (i) and (ii), and this information is shared with all the agents. The routing and synchronization tasks of (iii) correspond to a distributed algorithm that employ the worker agents during the gathering mission.

For reaching the plan that estimates the best balance between the information refreshing time and the total number of information packages delivered at the BS, the planner evaluates different scenario configurations: three area partition criteria and different ratios of $\#collectors/\#workers$ in the team. The scenario configuration that provides the best balance is the one selected to be executed.

The remaining part of the chapter is organized as follows. Different works of the state of the art are presented in Section 7.2. The different parts of the data gathering missions are defined in Section7.3. Three partition methods used in this work are defined in Section 7.4. The entire planner of the data gathering mission is developed in 7.5. The planner that

use the worker agents to compute the tours to visit the goals of their segments is defined in Section 7.6. The evaluation of the method is discussed in Section 7.7 and the conclusions can be found in Section 7.8. The publications [6] and [7] correspond to the developed technique in this chapter.

7.2 Related works

As described in the previous chapter, the mission type of this part shares more similarities with patrolling missions [58][59]. Where the agents travel invariant paths through a precomputed graph, re-transmitting the data when they meet each other. Obviously, it will be inefficient to employ this approach for our problem, since the goals appear in different locations, needing to compute the graph with every request cycle. Furthermore, the refreshing time in the BS exponentially increases with each data re-transmission between agents. In our approach, we compute the destinations for the collectors, so they will persistently travel to and from these points and the workers will come to these positions to upload their data, only in the case of having some data to share. So that, only one retransmission is made. This approach is similar in spirit to [72], where an agent performs a persistent task, moving towards other agents to meet them for recharging at the best computed point of their trajectories. In our kind of missions, it is more effective that the collector agents travel invariant paths, being the workers who move to share the data with them, in order to preserve the time of collectors cycles since it delimits the refreshing time.

In [73], the authors develop a method where some workers, with buffer limitations, gather data and transmit it to dynamic collectors. In that work there exist a wireless communication network and their collectors are permanently connected to a central server in the entire scenario. They are the only ones capable to upload the data to the server without the need of travel to a unique static depot point, situation considered in our work. Obviously, the travels to a static depot point increase the refreshing time, especially in large scenarios. In our approach the workers remain in their working areas and only travel short paths to the collectors, not needing to go up to the BS.

Our method is more flexible in terms of agent concurrence. The workers are not enforced to concur with their collector in some fixed rendezvous points, as in classic agent meeting problem [74][75], since it may become computationally intractable for big fleets of agents. Instead, the workers decide when and where to share the data with the collector, so that, do not stopping the motion if it is not needed. In [76], the robots are enabled to transfer deliveries between each other to reach different delivery locations. However, their system to establish the meeting points is fully centralized, that cannot be directly applied here because our team acts in a distributed way during the execution of the mission.

7.3 Problem statement

The problem to solve is planning the deployment of a team of robots in a scenario with connectivity constraints due to obstacles or distances to the BS. The BS requests for data from different goal locations. The team of agents must coordinate in order to periodically reach the goals, acquire data and upload it to the BS. The grid representation of the environment is used again. The positions of the obstacles are defined as $\mathbf{x}_o$, the position of the BS is x_{BS}, and

the positions of the agents and goals are $\mathbf{x}_a$ and $\mathbf{x}_g$, respectively. The BS periodically requests information from M goal locations, and the team, composed by N robots, is coordinated to move towards the goals, then delivering the information to the BS. The robots can act either as workers (N_w) or as the collectors (N_c), being $N_c+N_w = N$. During the time of the mission, denoted as $T_{mission}$, M remains constant. That is, when BS receives the information from m goals ($m \leq M$), it generates m new goals. Our approach must compute the plan of the mission, previously to deploy the agents. To this end, it must: i) obtain the working areas for the worker agents, denoted as S_w; ii) find out the best balance between collector and workers; iii) pair the workers with the collectors or with the BS to transmit the data, expressed as P_{cw}; iv) compute the trajectories of each collector π_c, according to P_{cw} and S_w. Once the mission starts, each worker computes its path π_w, in order to visit the corresponding goals, $\mathbf{x}_{g_i} \in S_{w_i}$ and go to deliver the information to a collector or BS, according to the association P_{cw}.

Π_w and Π_c are the sets of worker's and collector's paths, respectively, being $\pi_w \in \Pi_w$ and $\pi_c \in \Pi_c$. The times for travelling a path π and paths Π are expressed as $t(\pi)$ and $t(\Pi)$, respectively. The refreshing period from the time in which BS requests information and receives it is $T_{refresh}$. This period is the mean time of the collectors T_c and of the workers that transmit directly to the BS. Naturally, it must be minimal, whilst the number m of goal information received has to be as large as possible. The problem of computation of trajectories of workers and the collectors can be formally expressed as:

$$\pi^*_{w_{ij}} = \underset{\pi_{w_i} \in \Pi_{w_i},\ \mathbf{x}_{g_i} \in \pi_{w_i}}{\operatorname{argmax}} (|\mathbf{x}_{g_i}|) \tag{7.1}$$

$$\pi^*_{c_j} = \underset{\pi_{c_j} \in \Pi_{c_j}}{\operatorname{argmin}} \left(t(\pi_{c_j}) - \overline{t(\Pi_{w_{ij}})} \right) \tag{7.2}$$

$$subject\ to\ t(\pi^*_{c_j}) - t(\pi^*_{w_{ij}}) \geq 0 \tag{7.3}$$

where c_j refers to collector j and w_i to worker i. Eq.(7.1) obtains the optimal route for workers to visit the maximum number of goals $\mathbf{x}_{g_i}$ assigned to the agent w_i from all the possibles Π_{w_i}. Eq.(7.2), computes the trajectory of the collector c_j that minimizes the time that the workers must wait for the collector arrival, once that they have visited the goals within their respective partitions. $\Pi_{w_{ij}}$ represents the paths of the workers w_i assigned to collector c_j, obtained with eq.(7.1), and $\overline{t(\Pi_{w_{ij}})}$ denotes the mean time of these paths. The constraint of eq.(7.3) enforces the workers to fulfill the maximum number of their assigned goals, with eq.(7.1), and to meet the collector when it arrives in the current cycle. When a worker and a collector meet each other, the distance between them must be lower than d_{com} and the line-of-sight must not be occluded to establish a connectivity link to share data. The three steps achieved by the planner are described in the following sections: i) scenario segmentation in Sect.7.4, ii) collector's trajectories and segment allocation to workers in Sect.7.5, and iii) workers routing in Sect.7.6.

7.4 Scenario segmentation

There exist many works of scenario partition or segmentation for different purposes. In [77], the authors use segment the scenario in order to have semantic information information of

(a) BAP (b) PAP (c) RAP

Figure 7.2: Employed segmentation algorithms.

the scenario for applying SLAM techniques. In [78], a cleaner robot uses the segmented environment for coverage purposes, in order to obtain the shortest paths to clean each room. In [79][80] the segmentation is used for robot navigation. In [79], the authors segment the scenario to simplify the robot navigation in semi-structured environments, using Generalized Voronoi graph (GVG). The authors of [81] propose a Voronoi partition for coverage purposes and including the uncertainty of the localization of the agents. In [80], the environment decomposition is applied for heterogeneous multi-robot navigation.

In [82], the authors made a survey on different techniques of environment partition. They have analyzed four techniques: morphological-based segmentation[83], distance transform-based segmentation[84], Voronoi graph-based segmentation[85], graph partition-based segmentation[86] and feature-based segmentation[87]. More recent works use learning techniques to segment the environments [88][89]. The precision of these methods is clearly greater and, obviously, the time to learn the parameters is the major drawback of these approaches. However, all the cited techniques are focused in splitting the scenario based on its rooms. In our work, the partition is used in order to fairly distribute the environment into working areas of the workers.

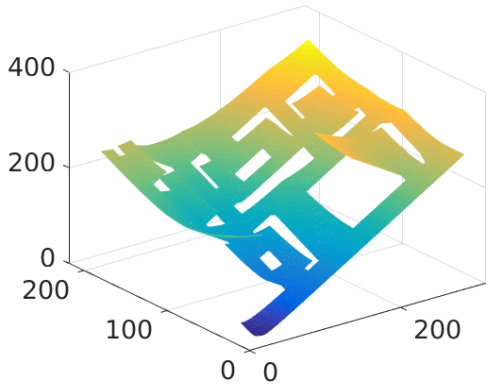
An interesting approach is proposed in [90], where a Voronoi graph of the environment is generated, and the tasks of the agents are allocated based on a graph partitioning. This way, the agents cover similar areas, reducing the revisiting of already explored areas. Nevertheless, in that approach each agent a priori knows exactly what goals it must visit.

In this work we develop and evaluate three segmentation algorithms: Balanced Area Partition (BAP), Polygonal-like Area Partition (PAP) and Room-like Area Partition (RAP). All of them use use the FMM as a base method. We encourage to visit the link [1], with the proposed segmentation processes in different scenarios.

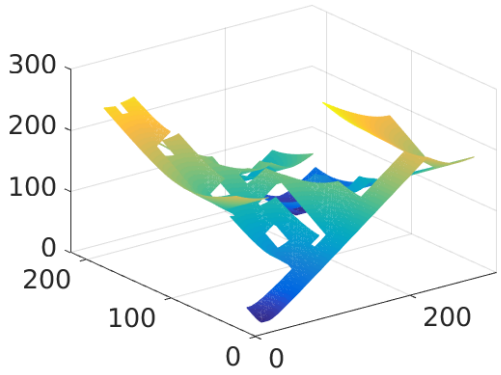
7.4.1 FMM for area partition

In the work developed in this chapter, the FMM, apart from the path planning for collectors and the time constrained trajectory planning, described in Sect.6.6 of the previous chapter, it is also used for the area partition for the workers and workers-collectors associations. Hence, here we describe only the properties employed for these purposes.

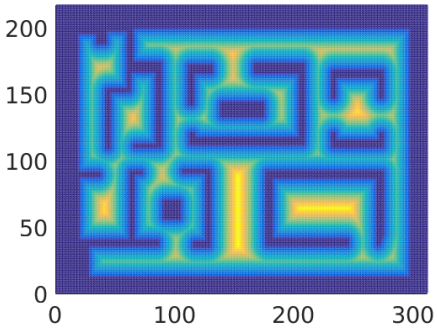
[1] http://robots.unizar.es/data/videos/paams19yamar/segmentations/



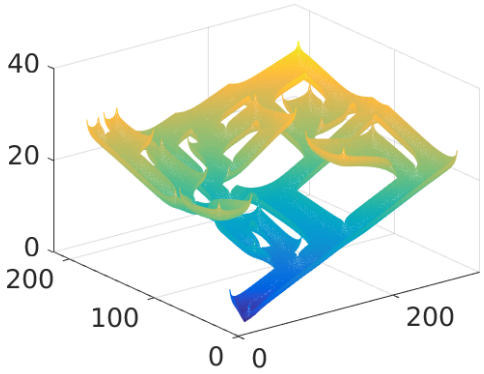
(a) One wavefront from (20,30)



(b) Two wavefronts: (20,30) and (250,180)



(c) Obstacles gradient ∇D_o



(d) One wavefront $F = \nabla D_o$

Figure 7.3: FMM gradients.

The simple gradient computation from a single source position provides the distances from the source to all the free cells of the grid, as depicted in Fig.7.3(a). Initializing several wavefronts from different positions, the resulting gradient ∇D will represent the distance to the closest origin, see Fig.7.3(b). This is used for the segmentation methods of Sect.7.4.3-7.4.4. The positions where the wavefronts collide, are the frontiers of the segments. As detailed in Chapter 4, varying the values of F, the propagation velocity becomes non-uniform. The gradient wavefront faster for higher values and slower for lower values of F. A faster propagation corresponds to lower values of the gradient, and the slower represents higher gradient values. An example is depicted in Fig.7.3(c)-7.3(d). The obstacles gradient ∇D_o of Fig.7.3(c), is obtained with FMM initializing the sources of wavefronts at the positions of the obstacles. ∇D_o is also known as the distance transform of the scenario. If now, we propagate the wavefront from (20,30) as in Fig.7.3(a), but setting $F = \nabla D_o$, the resultant gradient obtains lower values in points that are more distant from the obstacles, as illustrated in Fig.7.3(d). This property is used for the room-like segmentation of Sect.7.4.4 and for associations between collectors and workers in Sect.7.5.3, in order to favour the coverage of the wide open spaces.

7.4.2 Balanced Area Partition (BAP)

The main feature of this segmentation algorithm is that it obtains segments with balanced areas. A uniform FMM wavefront propagation obtains these areas. Firstly, the algorithm computes the gradient from the BS position ∇D_{BS}, which denotes the distance to the BS. Since the team employs N_w workers, the space is divided into N_w segments, with areas $A_{w_i}, i = 1, ..., N_w$, accomplishing $A = \sum A_{w_i}$. The optimal segment area is denoted by $A_{opt} = A/N_w$, measured as number of cells in the grid.

Initially FMM propagates a wavefront from BS, expanding A_{opt} cells. If the total area of expanded cells is higher or equal to $A_{opt}/2$, a heuristic threshold, and the number of already obtained segments is lower than N_w, these cells become a new segment. If not, the expanded cells are added to an adjacent segment of minimum area. Here, we use the condition A.4 in the algorithm of wavefront propagation presented in Alg.21.

The algorithm iterates until classify all the free space A, choosing as wavefront origin, the closest non-classified point to x_{BS} from ∇D_{BS}. This iterative procedure is illustrated in Fig.7.4. The figures 7.4(a)-7.4(c) depict the extended cells using the FMM wavefront propagation. And figures 7.4(d)-7.4(f) illustrate already classified space. The number of extended cells depends on the obstacle distribution in the scenario, since the algorithm requires more iterations to cover the entire area if there are any remaining non-classified small areas. This may also produce bigger segments than A_{opt}, obtaining a lower number of segments than N_w. In this case, the algorithm iteratively halves the biggest segments until

(a) Gradient at $it = 0$ (b) Gradient at $it = IT_{BAP}/2$ (c) Gradient at $it = IT_{BAP}$

(d) Segments at $it = 0$ (e) Segments at $it = IT_{BAP}/2$ (f) Segments at $it = IT_{BAP}$

Figure 7.4: *BAP* iterations. IT_{BAP} denotes the total number of iterations to finish the segmentation process of *BAP*.

obtain N_w. Fig.7.2(a) illustrates the segmentation for 10 segments. Although the shape of the segments is quite irregular, the areas are equitable, which a priori favours a balanced time for workers in all the segments.

7.4.3 Polygonal Area Partition (PAP)

This algorithm attempts to keep equitable distances between the centroids of the segments and their boundaries with other segments or obstacles. *PAP* algorithm is summarized in Alg.18. The algorithm consists in two main phases: centroid initialization and iteration. The gradients in both phases are using the condition A.2 in Alg.21 to extend the wavefront over all the free cells of the grid.

Firstly, the algorithm obtains the obstacles gradient, corresponding to the distance to them, l.1, as explained in Sect.7.4.1, see Fig.7.3(c). After that, it iteratively finds the maximum values of this gradient in l.2, that correspond to the largest free spaces, for example rooms. The position of the maximum is the initial location of the centroid of each segment. Since the value of the maximum represents the greatest distance to the closest obstacle, the algorithm removes all the points of the grid within the radius equal to the maximum. This way, it avoids to initialize new centroids of the segments in the same rooms. This procedure is repeated N_w times, one per segment/worker. An example of the initialization of the centroids is depicted in Fig.7.5.

Algorithm 18 Procedure for PAP and RAP

Require: Grid, $\mathbf{x}_o$, N_w, Partition type (PAP or RAP)

1: $\nabla D_o \leftarrow compute\_gradient(\mathbf{x}_o)$ ▷ Eq.(3.2)
2: $\mathbf{x}_c \leftarrow initialize\_centroids(N_w, \nabla D_o)$
3: **If** (PAP) $[S, \mathbf{x}_c] \leftarrow it\_part(\mathbf{x}_c, Grid)$ ▷ Alg.19
4: **If** (RAP) $[S, \mathbf{x}_c] \leftarrow it\_part(\mathbf{x}_c, \nabla D_o)$ ▷ Alg.19
5: **return** $S, \mathbf{x}_c$

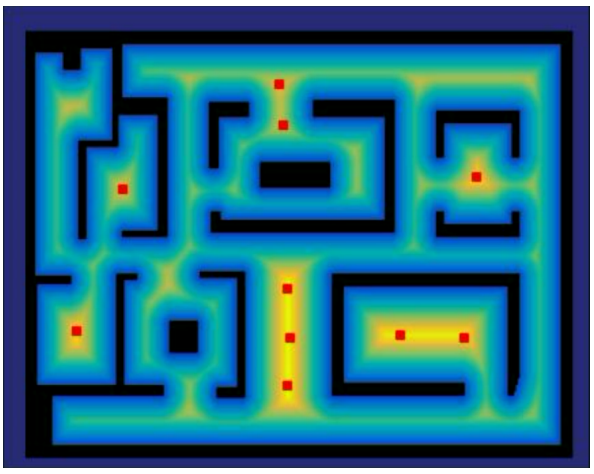
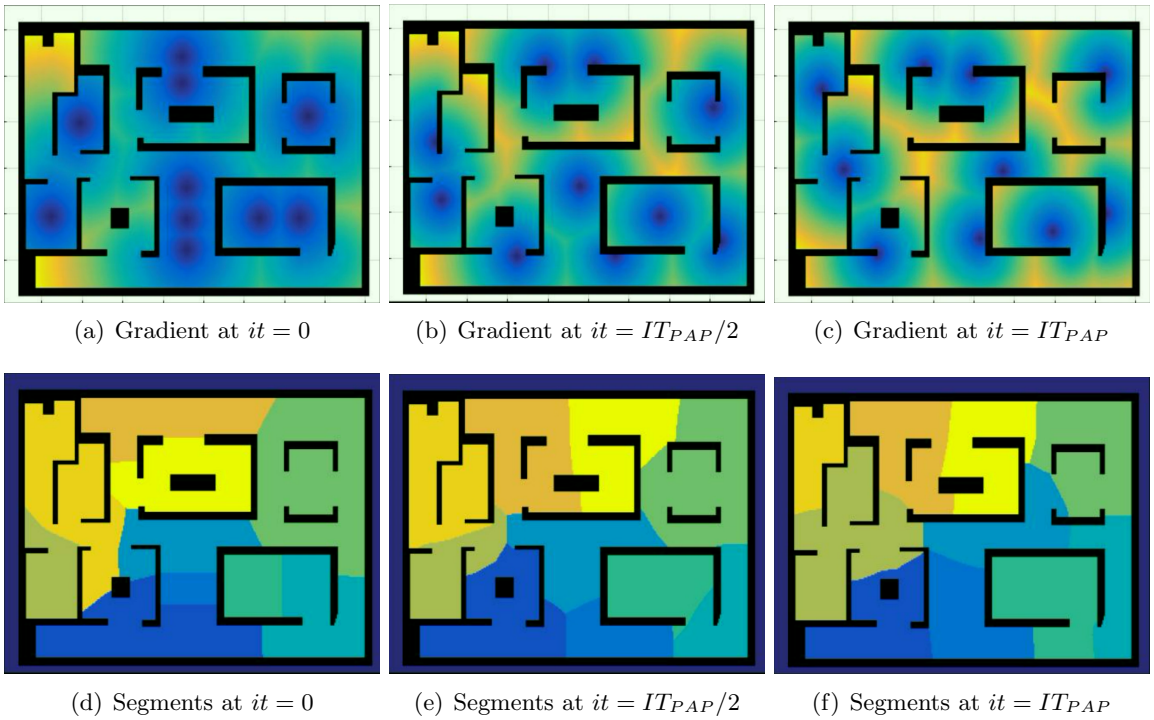


Figure 7.5: Centroids initialization over the distance transform of the scenario, for *PAP* and *RAP* methods. Yellow colour of the cells represents the farthest points from the closest obstacles. The centroids are depicted with red squares.



(a) Gradient at $it = 0$ (b) Gradient at $it = IT_{PAP}/2$ (c) Gradient at $it = IT_{PAP}$

(d) Segments at $it = 0$ (e) Segments at $it = IT_{PAP}/2$ (f) Segments at $it = IT_{PAP}$

Figure 7.6: *PAP* iterations. IT_{PAP} denotes the total number of iterations to finish the segmentation process of *PAP*. Darker blue colour cells in Fig.(a)-(c) denote the positions of the centroids, from where the wavefront is propagated.

After the initialization, the method iteratively moves the centroids until achieving the equilibrium between the distances of the centroids. This procedure is described in Alg.19 and illustrated in Fig.7.6. At first, the method computes the distance gradient from the centroids (l.2), using again FMM depicted in Fig.7.6(a). The variable *costmap* takes the values of the basic grid to propagate the wavefronts, so F is 0 or 1 in eq.(3.2). Therefore, N_w wavefronts are uniformly propagated in all the directions, one per segment. The positions where the wavefronts collide among them are the boundaries of the segments. The segments obtained from the wavefront collisions of Fig.7.6(a) are depicted in Fig.7.6(d). The highest value of the gradient of each partition is the farthest position from the centroid $\mathbf{x}_{p_f}$ in l.3. So we move every centroid in the direction of its farthest position in the segment: computing the paths, l.4, and moving the centroids along them, l.5. The algorithm iterates until achieving a balanced distance between the centroids of the segments as illustrated in Fig.7.6. The condition of the balanced distances is accomplished when the centroids repeat their positions, l.1 of Alg.19. We encourage the reader to watch the video for better understanding of this procedure. The resulting segments are depicted in Fig.7.2(b).

As we can see the obtained partitions are similar to the centroidal Voronoi partitions as used in [91], where a scenario without obstacles was segmented to maintain equitable distances between the agents. The same approach was applied to a scenario with obstacles in [92] and [93]. But the surrounding of the obstacles is not taken into account. In our work we have include the obstacles in order to obtain more precise partitions of the scenario.

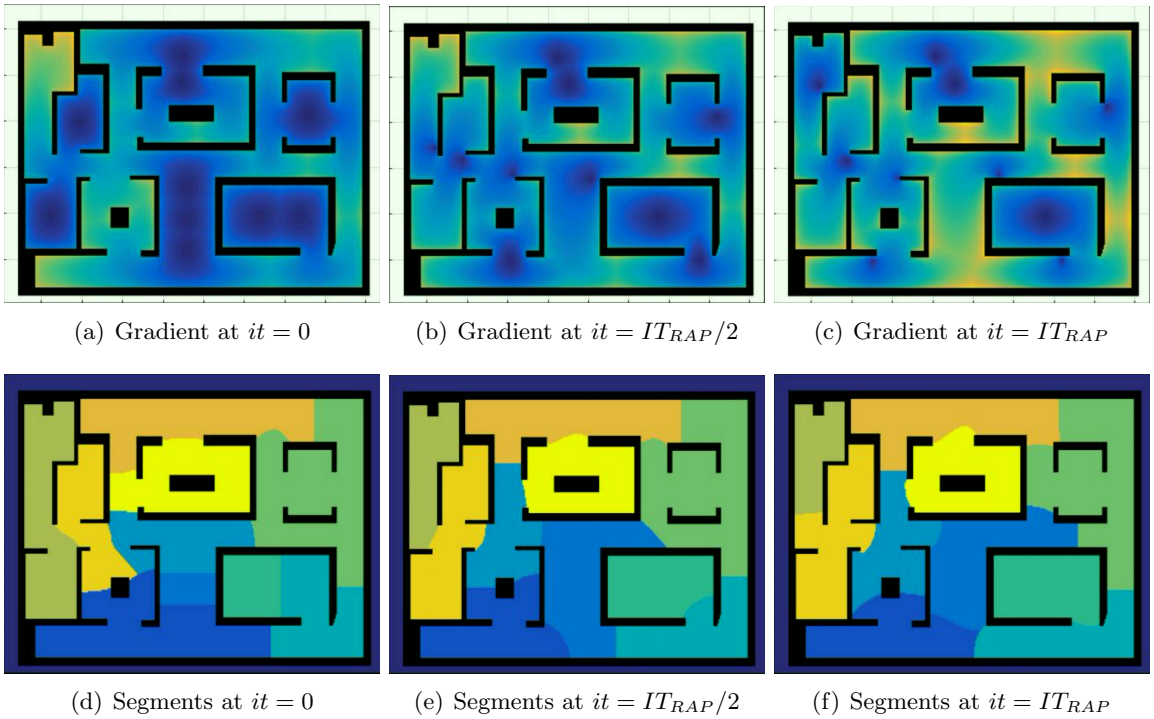
Algorithm 19 Iterative Partition (*it_part*)

Require: *Centroids* $(\mathbf{x}_c)$, *costmap*

1: **while** !*repeated_positions*$(\mathbf{x}_c)$ **do**
2: $[\nabla D_c, S] \leftarrow gradient\_and\_partitions(\mathbf{x}_c, costmap)$
3: $\mathbf{x}_{pf} \leftarrow compute\_farthest\_in\_partition(\nabla D_c, S)$
4: $\Pi_f \leftarrow gradient\_descent(\nabla D_c, \mathbf{x}_{pf})$
5: $\mathbf{x}_c \leftarrow move(\Pi_f)$
6: **end while**
7: **return** $S, \mathbf{x}_c$

7.4.4 Room-like Area Partition (RAP)

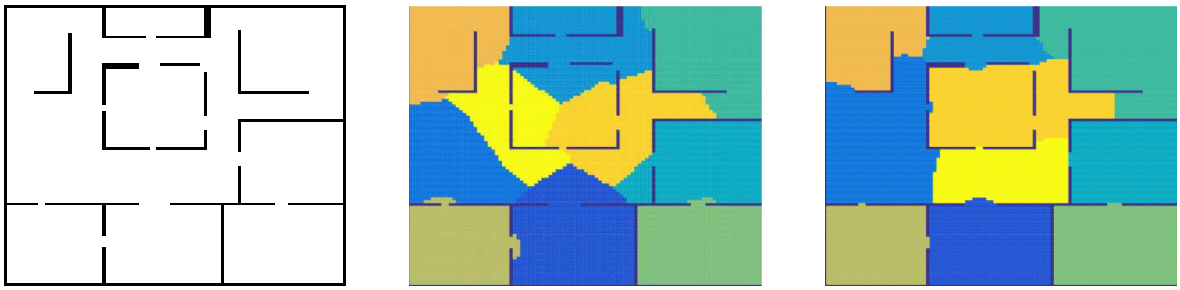
This area partition method employs the same procedure that iteratively moves the centroids as *PAP*. But instead of using the basic grid of the map, it uses the obstacles gradient ∇D_o computed in l.1, as described in l.4 of Alg.18. Then, *costmap* variable in Alg.19 takes values of ∇D_o, depicted in Fig.7.5. This changes the propagation of the wavefronts, becoming non-uniform. So that, the wavefronts cover faster the wide areas, such as rooms, and slow down when reaching tight spaces, commonly corresponding to doors, where the wavefronts collide. Using this property, the resulting segments tend to cover the rooms, as can be seen in Fig.7.2(c). Because of that, their areas differ from *PAP*'s algorithm. The iterative procedure



(a) Gradient at $it = 0$ (b) Gradient at $it = IT_{RAP}/2$ (c) Gradient at $it = IT_{RAP}$

(d) Segments at $it = 0$ (e) Segments at $it = IT_{RAP}/2$ (f) Segments at $it = IT_{RAP}$

Figure 7.7: *RAP* iterations. IT_{RAP} denotes total the number of iterations to finish the segmentation process of *RAP*. Darker blue colour cells in Fig.(a)-(c) denote the positions of the centroids, from where the wavefront is propagated.



(a) Scenario (b) *PAP* (c) *RAP*

Figure 7.8: Example of the differences between *PAP* and *PAP* segmentations. 10 segments.

is illustrated in Fig.7.7. As we can observe, in comparison with the initial segmentation of *PAP* in Fig.7.6(d), *RAP* method obtains also polygonal-like segments, but they are at the transitions between different spaces, corridors or rooms as seen in Fig.7.7(d).

The final obtained segments, depicted in Fig.7.6(f) do not obtain the exact shape of the rooms of the scenario. But this is because the number of required centroids differ from the number of rooms of the scenario. For example, in Fig.7.8, we can see how *RAP* method fits better the segments in the rooms, because the number of centroids is approximately equal to the number of rooms and the transitions between them (doors) are more notorious. While *PAP* method provides segments only with polygonal shapes.

7.5 Collectors trajectories and segment allocation

In this section we explain how the planner selects the best number of the collectors used for the mission, the computation of their trajectories and the association of the workers to the collectors to share data. All these procedures are based on the segments of the workers.

7.5.1 Working time estimation

The method computes the collectors trajectories based on the estimated working time of their associated workers. During the mission, the collectors persistently travel these paths without stopping, being the workers who move to share the gathered data with them.

Since the distribution of the goals will change every cycle, the planner has to estimate an averaged working time in each segment. Assuming that the goals are uniformly distributed, we consider the number of goals within each worker segment i will be approximately proportional to its area, obtained as $M_{s_i} = M * A_{s_i}/A$. In order to fairly estimate the distribution of the goals within a segment, the algorithm automatically places M_{s_i} goals (centroids) within segment S_i using the *PAP* procedure. Thus, the estimated goals are equidistant between them, considering the obstacles. The $NN + 2O$ procedure, explained in Sect.7.6.1, computes the shortest tour from each segment centroid to visit the M_{s_i} goals, estimating the working time each worker will spend in its segment.

Algorithm 20 General planning procedure

Require: $Grid$, M, N, x_{BS}

1: $\Pi_c^* \leftarrow \emptyset,\ P_{cw}^* \leftarrow \emptyset,\ S_w^* \leftarrow \emptyset,\quad \mathbb{T}_c \leftarrow \emptyset,\ \mathbb{M}_d \leftarrow \emptyset$
2: $\nabla D_{BS} \leftarrow compute\_gradient(x_{BS}, Grid)$
3: **for each** *segmentation* **do** ▷ *BAP, PAP, RAP*
4: **for** $N_c = 0 : N/2$ **do**
5: $N_w = N - N_c$
6: $[S_w, \mathbf{x}_{c_w}] \leftarrow segment(grid, N_w)$
7: $T_{work} \leftarrow estim\_work\_time(S_w)$ ▷ Sect.7.5.1
8: $[\Pi_c, P_{cw}, T_c, M_d] \leftarrow coll\_paths(Grid, N_c, x_{BS}, \nabla D_{BS}, \mathbf{x}_{c_w}, S_w, T_{work})$
9: $\mathbb{T}_c \leftarrow \mathbb{T}_c \cup \{T_c\}, \mathbb{M}_d \leftarrow \mathbb{M}_d \cup \{M_d\}$
10: **end for**
11: **end for**
12: $[\Pi_c^*, S_w^*, P_{cw}^*] \leftarrow best\_plan$ ▷ eq.(7.4)
13: **return** Π_c^*, S_w^*, P_{cw}^*

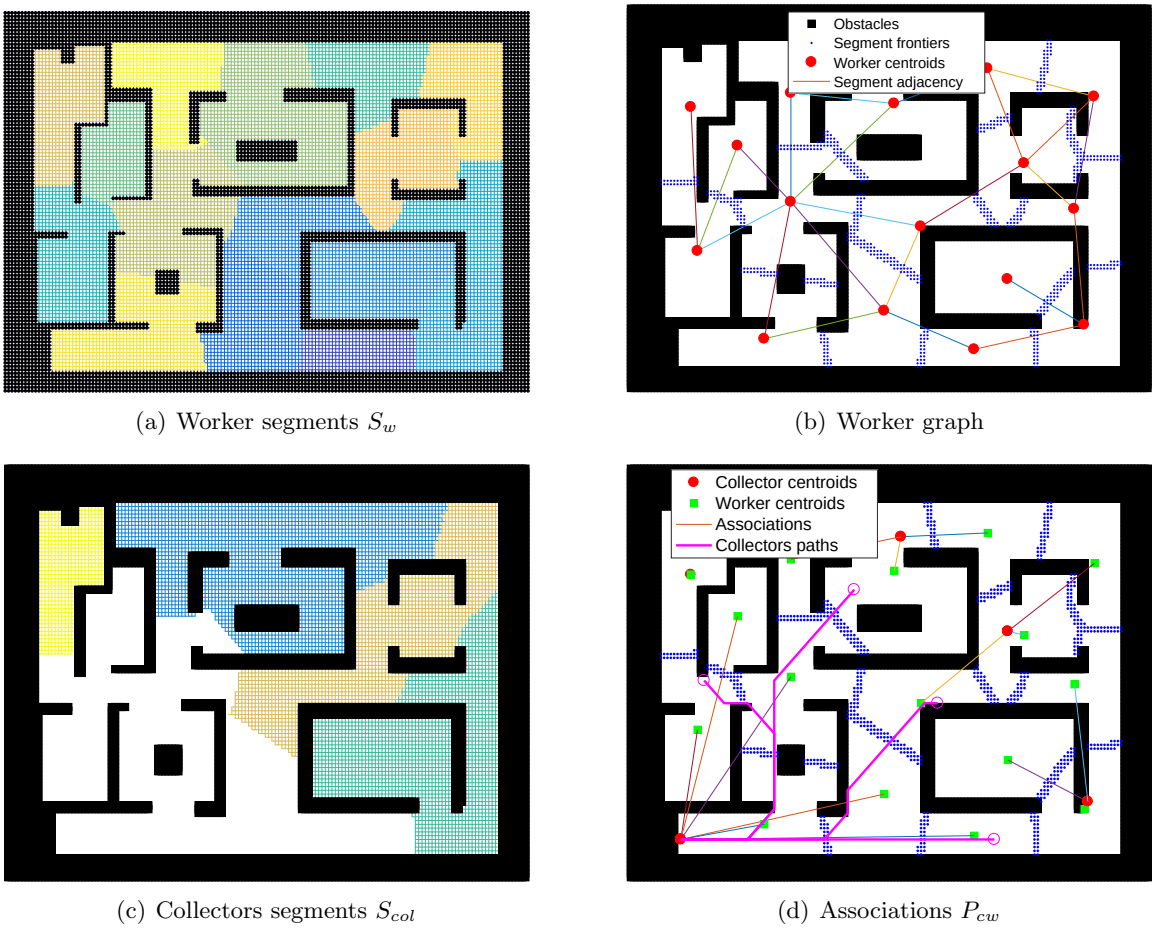
7.5.2 Planning procedure

Based on the average working time estimated for the segments, the plan procedure in Alg.20 obtains the needed collectors, their paths and, therefore their time periods T_c. The paths of collectors are computed from the gradient to the BS, in l.2. The algorithm evaluates the three scenario partitions, in l.3. The number of collectors to be evaluated is $N_c = 0...N/2$, l.4, because it makes no sense to devote more than one collector to a single worker. When the system adds a new collector, it renounces to a worker, l.5. This changes the working areas, so the algorithm segments the scenario every iteration, in l.6, and estimates the working times for the resulting segments, in l.7 as described in Sect.7.5.1. Then it computes the paths of the collectors (Π_c) and collector times (T_c), in l.8 with *coll_paths* function. This procedure also associates the workers with the collectors (P_{cw}), obtaining the goals that will be delivered to the BS (M_d), explained in Sect.7.5.3. A direct movement of workers to the BS, without using collectors, is also evaluated. It corresponds to the case $N_c = 0$. The times and the delivered goals are stored, in l.9, in order to choose the best plan from the different partitions and collectors in l.12, using the utility function:

$$U = max\big[\alpha * (1 - \mathbb{T}_c/max(\mathbb{T}_c)) + \beta * \mathbb{M}_d/max(\mathbb{M}_d)\big],\ \alpha + \beta = 1 \tag{7.4}$$

In this work, we set the values $\alpha = \beta = 0.5$, giving the same priority to the refreshing time and the number of deliveries. However, these values can be adjusted depending on the kind of mission, i.e. higher values for α in critical missions, such as fire monitoring, and higher values for β in simpler missions, such as surveillance.

The complexity of Alg.20 is $O(N/2(it_w + it_c + 1)nlogn)$, where: n is the number of free cells in the map, being $nlogn$ the complexity of FMM; it_w is the number of iterations to achieve the balance in the segmentation process for the workers ($it_w = 1$ for BAP); it_c are the iterations to balance the collector segments; the remaining $+1$ is the FMM to obtain the paths of the collectors.



(a) Worker segments S_w

(b) Worker graph

(c) Collectors segments S_{col}

(d) Associations P_{cw}

Figure 7.9: Collectors' path computation and association, for 16 segments (workers) and 4 collectors. In (c), white space are segments of workers that upload directly to the BS.

7.5.3 Collector trajectories and workers-collectors association

An illustrative example of collector computation is depicted in Fig.7.9. This method also employs the FMM-based *RAP* procedure to associate the collectors to the workers. It receives the segmented scenario for the workers, Fig.7.9(a), and obtains the graph of the worker segments, depicted in Fig.7.9(b). The vertices of the graph are the centroids of the segments and the edges link the adjacent vertices. The workers at the segment of the BS and the adjacent ones will upload data directly to the BS, thus their edges are removed from the graph.

The method iteratively computes the paths of the collectors, associates them with the workers, and estimates the mean refreshing time at the BS. First, it computes the connectivity of the graph, which is obtained as the number of edges that has every vertex. Second, it selects N_c vertices of the maximum connectivity, and initializes the centroid of the segments as the initial farthest position to be reached by the collector ($\mathbf{x}_{col}$), before coming back to the BS. Then, it iteratively moves these centroids, by propagating the gradient ∇D_{front} from the frontiers of the worker segments, depicted with blue points in Fig.7.9(b), using Alg.19

executing $[S_{col}, \mathbf{x}_{col}] \leftarrow it\_part(\mathbf{x}_{col}, \nabla D_{front})$. This way, the shape of the segments is taken into account, filling faster the smaller and uniform worker segments, and slower the largest and irregular ones. The final segments associated to the collectors (S_{col}), which group the worker segments associated to it, are depicted in Fig.7.9(c).

The workers of the segments grouped in a collector segment S_{col} share data with the collector that comes to $\mathbf{x}_{col}$, as illustrated in Fig.7.9(d). Then, the paths from x_{BS} to each $\mathbf{x}_{col}$ are obtained descending ∇D_{BS}. According to eq.(7.1)-(7.3), in order to find out a balance between the time of the collector T_c and time of workers to reach $\mathbf{x}_{col}$, the path of each collector is iteratively contracted until achieving the balance. As can be observed in Fig.7.9(c), the method associates not only adjacent workers to upload to BS, but also others corresponding to the next levels in the adjacency graph, until $T_{refresh}$ stops decreasing.

7.6 Workers trajectories

After obtaining the global plan of the mission, the workers know the distribution of their segments and the trajectories of their corresponding collectors. Here we develop the trajectory planner that execute the workers to visit the goals of their segments and to synchronize with the collector in movement.

7.6.1 Goals visit methods

Each worker has to visit the maximum number of goals in its segment, in such a way that it will be able to reach the collector to share the data in some point of its trajectory. In other words, there exist a time window, starting from the instant of receiving a request of new goals to the next collectors cycle, in which the worker has to visit the maximum number of possible goals. In most of the works of robotics field that consider time windows, as [94][95], the time constraints are related to the time of the task. That is, a goal has a specified time to be reached by some agent. In our case, the goals does not have a time window, but the final destination of the worker is constrained by the collector trajectory.

The workers employ different routines based on who the agent has to communicate with. The workers associated to upload data directly to the BS does not have any time window, so that they can visit all the goals. However, the workers associated to share data with a collector are constrained by a time window, defined by the collector cycle. The costs to reach the goals are evaluated from the FMM gradient. The distances are the values of the gradient at the goals positions and the times to reach the goals are obtained dividing these distances by the speed of the workers, forming the cost matrix. The three main routines to obtain the worker's route are:

- *Brute Force (BF)*: obtains the optimal solution testing all the possible routes.

- *Nearest Neighbor with 2-opt Improvement (NN+2O)*: a first route is initialized with the *Nearest Neighbor (NN)* procedure. Then, the route is improved by means of a local optimization using 2-opt method [45], that swaps every two edges of the route, goals in our case, checking if the new route outperforms the previous one. This method is able to obtain the routes in real-time (milliseconds), against classic orienteering problem methods, which require minutes to find a solution [96].

– *NN+2O with Time Window (NN+2O-TW)* and *BF-TW*: with the same structure as the basic *NN+2O* and *BF*, but taking into account a time condition to meet with the collector at time, formally expressed as:

$$T_c - t_{c_{min}} >= t_{accum} + t_{g_j} + t_{g_j - c}, j = 1, ..., K \tag{7.5}$$

where T_c is cycle time for the collector, $t_{c_{min}}$ is the time to transmit the data of already collected information, t_{accum} is the accumulated time of the trajectory, t_{g_j} is the time to reach the next non-visited goal j, $t_{g_j - c}$ is the time from this goal to the collector and K is the number of goals for the worker. When the condition of eq.(7.5) is not accomplished, the algorithm stops iterating.

The workers that upload data directly to BS use *BF* to obtain a solution in real-time for instances up to 12 goals (about 50ms), whilst *NN+2O* is used to obtain a suboptimal solution for more than 12 goals. The workers that transmit to a collector employ *BF-TW* for less than 12 goals, and otherwise use *NN+2O-TW*. This way the workers upload to the collector every cycle some or all the tasks allocated for them.

7.6.2 Trajectories for synchronization

In the present work we use the optimal trajectory planner, developed in the previous chapter in Sect.6.6. We choose this approach because it provides the optimal solutions. Therefore, the agents will be able to gather data from more goals. Here, instead of allowing the trajectory planner to choose from the possible collectors to communicate with, the association algorithm developed in the previous section 7.5 is used. Therefore, each worker only stores the trajectory of its associated collector.

By contrast, the workers autonomously compute their trajectory based on the amount of the data to share. This value will vary from one cycle to another. For example, if in one cycle an agent it is not able to reach some goal location, it will have more time for the next collector cycle to reach most of the goals and, consequently, gather more data.

The same planner is used for the agents that transmit their data to the BS. In this case, the mate to share data with is static, because the BS in our case is not dynamic. So each agent projects the BS coverage area in time, to the instant that the gathering mission finishes. Notice that all these agents will visit all the goals, since they do not have time constraints. However, their transmission time also varies, because it depends on the amount of goals to reach and they change during the mission.

The complexity of the computation process of the complete trajectories for a worker i at each cycle is: $O((K_i + 3)nlogn)$. Where $K_i + 3$ FMM executions are required. K_i is the number of goals of the agent i. This part corresponds to the route computation to visit the goals of Sect.7.6.1. The remaining 3 gradient computations, using FMM, correspond to the synchronization part. As described in the previous chapter in Sect.6.6, the optimal planner obtains 3 gradients: from the agent position, within the communication area, and to the next goal location. The evaluation of the possible trajectories is not included, because in our scenarios the number of evaluated trajectories is very low and the time to obtain one of them is negligible.

7.7 Results

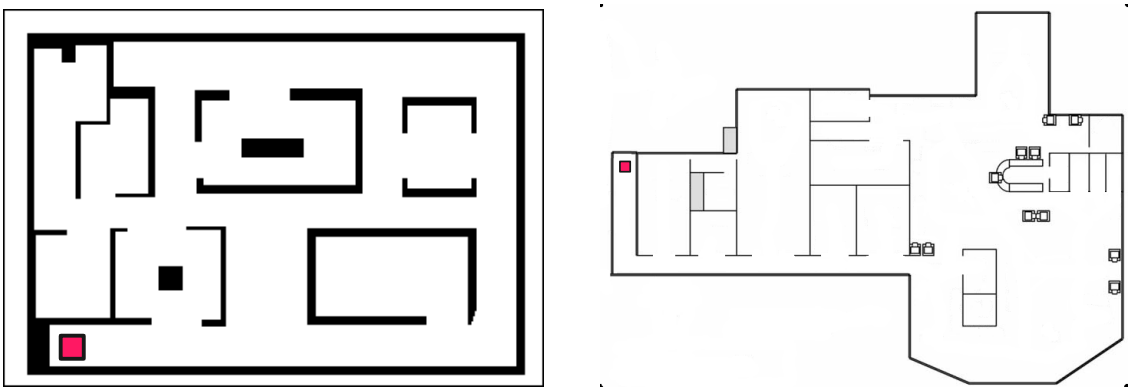
The method was implemented in C++ and the simulations were performed in a computer with i7 CPU clocked at 3.4GHz with 8GB of RAM. We evaluate it in the scenarios of Fig.7.10, extracted from [58]. We evaluate our approach for a team of 20 agents and for M=100 of requested goals. The communication range between the agents is set to d_{com}=10 cells and a constant velocity for the workers and collectors of 2 cell/sec. The time to gather data at one goal is 5 seconds and to transmit this data is 1 second. The plan execution is evaluated based on 20 trials of randomly generated goals in a 1000 seconds mission. The evaluated metrics are: the refreshing time $T_{refresh}$, that is the time elapsed to deliver the requested information , and the number of delivered information packages, that is number of goals. The mission starts with the agents at BS, the plan is obtained with Alg.20 and shared between the agents. Then, in a initialization phase, the workers go to their respective segments and start gathering the first batch of requested goals. The results analysis shows that the mean execution refreshing times and number of packages delivered at BS for different ratios workers-collectors and the three partition methods, are close to the ones estimated by the planner. Examples of deployments are available in the link[2].

The results are depicted in Fig.7.11-7.12. Both scenarios present different layouts and difficulties for the team deployment. The mean time to obtain the plan with Alg.20, is less than one minute in both scenarios. The algorithm tests different number of collectors from 0 (baseline for comparison) to $N/2$. As we can see in the results only up to 8 collectors were tested. This is because, as explained in Sect.7.5, the workers of segments adjacent to the BS are associated to communicate with it. So, more than 8 collectors are never required.

According to the utilities of Fig.7.13, obtained with eq.(7.4), a first clear result is that using collectors is better than not using them. The planner estimates that the best plan for the scenario of Fig.7.10(a) is BAP using 2 collectors, although 1 or 3 collectors provide similar utilities. For the scenario of Fig.7.10(b) the best configuration is using PAP method and employ 2 collectors for the mission. The computed utilities for the plan execution are close to the planned ones: the best result in Fig.7.13(a) is BAP with 1 collector, and for Fig.7.13(b) the execution values match with the estimated ones. Regarding Fig.7.11 and 7.12, in mean the plan execution get worse $T_{refresh}$ only in 9 seconds and delivering 10 less goals. The little differences found between planned and execution mean values are due to the fact that the estimation considers that the number of goals within each segment is proportional to its area. However, this does not always occur with real goals distribution. It can be concluded that the use of 1-3 collectors provide the best results in the tested scenarios.

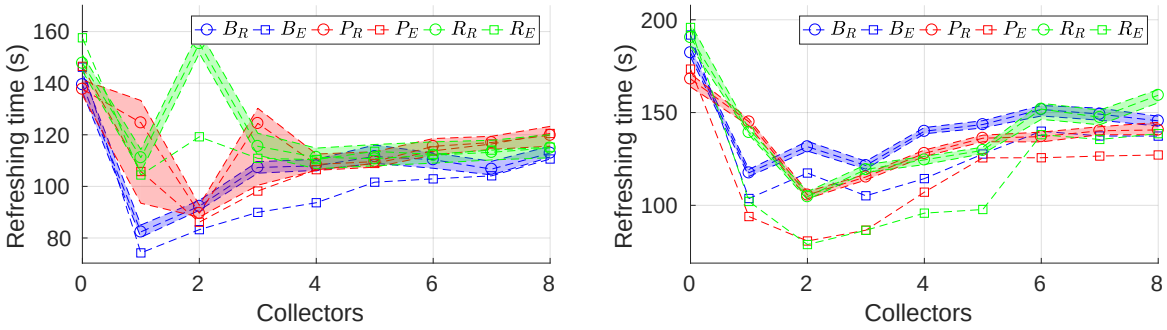
Regarding the kind of segmentation, the BAP method that splits the scenario in segments of approximately similar area, works better in the first scenario in which a more homogeneous obstacle distribution is found. The PAP method, which splits the scenario in polygonal segments, works better in the second scenario, where the obstacles are not homogeneously distributed, having large diaphanous areas and narrow corridors. The polygonal segmentation fits better the cleared areas than the other segmentation methods. Anyway, seeing the refreshing time, the number of goals delivered, and the utilities, it can be said that the PAP segmentation provides a good solution for both scenarios.

[2]http://robots.unizar.es/data/videos/paams19yamar/simulations/



(a) 156x110 (b) 172x125

Figure 7.10: Tested scenarios. The red squares illustrate the BS location.



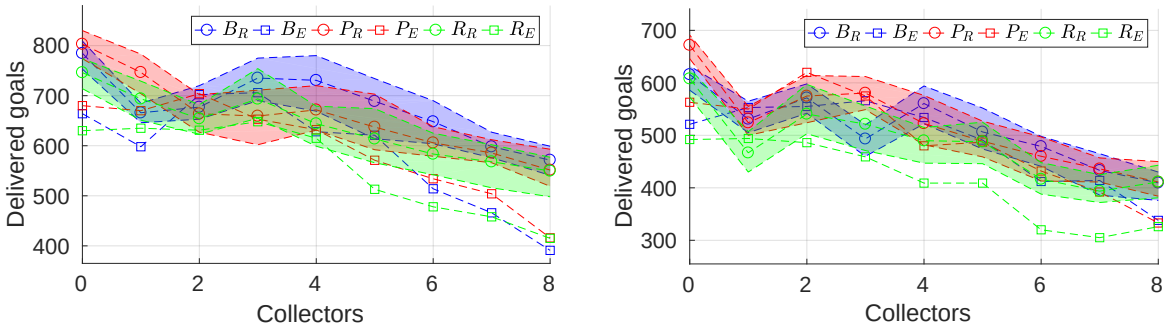
(a) Scenario of Fig.7.10(a) (b) Scenario of Fig.7.10(b)

Figure 7.11: Refreshing times. The letters B, P, R in the legends refer to BAP, PAP, RAP methods respectively. Sub-index R and E, denote real and estimated values, respectively. The squares represent estimations and the circles the real values. The coloured bands are delimited by the maximum and minimum of the real values.

7.8 Conclusions

In this chapter we have presented a method to plan the deployment of a team of agents to periodically gather information on demand from some a priori unknown goal locations, delivering them to a static Base Station. The use of collectors for uploading the information at the Base Station is more useful that directly moving all the robots to the base, from the point of view of the balance between the refreshing time and the number of delivered goals. We have tested three area partition algorithms, concluding that the PAP segmentation, which splits the scenarios in polygonal areas that fit well the free workspaces, provides good results for one to three collectors in the tested scenarios.

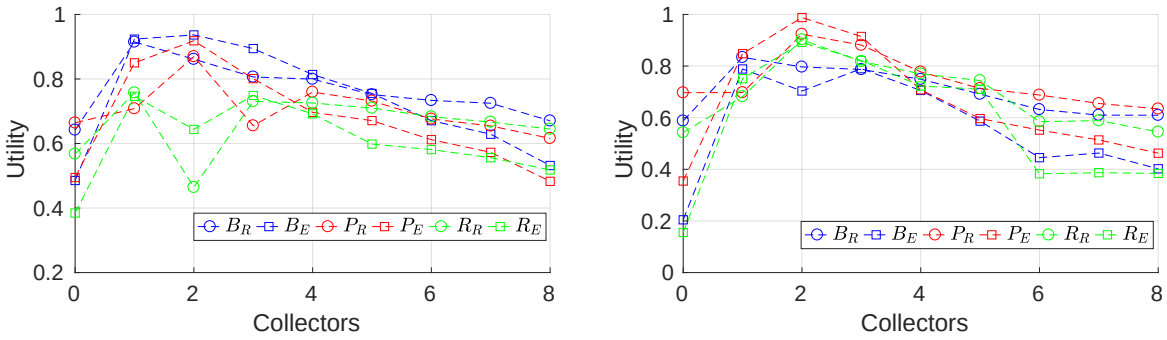
As future works, we want to generalize our method for dynamic environments, and where the uncertainty in the agents trajectories must be taken into account. We want to use a training phase to estimate the workload of the workers within their segments. Generating



(a) Scenario of Fig.7.10(a)

(b) Scenario of Fig.7.10(b)

Figure 7.12: Delivered goals.



(a) Scenario of Fig.7.10(a)

(b) Scenario of Fig.7.10(b)

Figure 7.13: Utilities, using eq.(7.4).

large instances of randomly distributed goals, in order to adjust the trajectories of the collectors. In this work we have focused in missions where the goals are uniformly distributed in the scenario. As a next step, we are adapting the proposed approach for missions where the goals are condensed in some specific areas and the areas vary during the evolution of the mission. In fact we are currently working in the same approach, but instead of splitting the environment, we propose to directly distribute the goals between the available agents. The current sequence of the algorithms is: to find the best partitions for the workers and then find out the trajectories of the collectors to serve to the workers. We want to test our algorithm for the opposite case: firstly splitting the scenario for the collectors and, then splitting each of these segments for the workers.

Chapter 8

Conclusions

8.1 Conclusions

In the present work we have proposed multiple methods to plan the deployment of multi-robot teams in connectivity constrained environments. The planning of the deployment is done in order to monitor an environment, reaching some locations of interest and delivering the acquired data to a static Base Station. Due to the common communication constraints, as limited communication ranges of the wireless sensors and the obstacles, a direct communication of the robots from the goal locations with the base is impossible. Therefore, the team of agents must coordinate autonomously in order to be able to deliver the acquired data to the Base Station. We have focused in two main research lines, associated to the type of connectivity between the agents and the base station during the deployment mission.

For the first communication type, developed in Part I, we have considered that the robots must establish a connectivity link with the Base Station from the goals locations.

In the Chapter 2, we have proposed a method to compute the optimal positions for the robots that will be used in role of relay, retransmitting the information of their teammates, from the primary goals, to the Base Station during the deployment mission. In the proposed approach, the obtained relay goals have a tree topology to interconnect the agents, when they are visiting the relay and primary goals. We have proposed and analyzed three criteria to compute the tree: connectivity, exclusiveness and distance. The approach was evaluated based on two metrics: the number of agents used in role of relay during the deployment and the estimated distances to the goals. We have observed that both criteria related to the number of relays, connectivity and exclusiveness, or a combination of them, provides a fair balance between the required relays and the distances.

In Chapter 3, we have developed two techniques, based on a sequence of algorithms to plan the deployment mission of a team of agents: computation of the relay positions, from Chapter 2, a path planner that considers the communication between the agents and a method to allocate the agents for both tasks, relay and primary. The first technique maximizes the connectivity during the deployment to fulfill the mission as fast as possible. The second technique plans the deployment mission in order to fulfill it with the minimum number of agents. It keeps a similar connectivity level as the first approach, by focusing the occupation on specified agents for relay tasks. However, the time to finish the mission is grater with respect to the first approach.

In the Chapter 4, we have generalized the deployment planners, developed in the previous

chapters, for all the instances of agents and goals. Firstly, we have proposed a simple and fast approach to compute the relay positions, by means of using relay chains. The relay chains can be seen as the tree branches of the Chapter 2. With this method, we are able to reduce the time to compute the relay goals, by substantially reducing the number of potential candidates for this purpose. The aforementioned relay chains, altogether with the primary goals connected by them, form *clusters* of goals. We have proposed and evaluated several heuristic methods to visit the *clusters* in two ways: sequentially and in parallel. Furthermore, we have developed a simple method to allocate the goals to the agents within the clusters, taking into account the goal type, primary or relay. The results have shown that in some situations the sequential methods have a better performance, which seems being not so obvious. This is due to the type of scenario. Enlarged and stretched scenarios where the Base Station is placed in some extreme position, allows very few possibilities to extend several chains of robots, which favours the sequentialization of visit of the clusters. On the contrary, in more wide scenarios with open spaces where the base is placed in the middle, it is possible to extend several chains in different directions. Therefore, concurrent methods obtain better results. The sequence of the proposed techniques allows to plan a deployment in very short time and for all the instances of goals and agents.

In Chapter 5, we have proposed a simple method to reconnect a group of agents in changeable environments with the appearance of new obstacles, doors closing, etc. The mission of the agents, on the contrary to other works in the literature, is two-fold: regroup the team and at the same time to form a specified formation. In this work we have considered the chain formations. As we have shown, the proposed approach is able to regroup the agents in a chain formation in simple scenarios. We have compared our method to another technique that sequentially reconnects the agents and forms the chain, always assuring the regrouping. Compared to this technique, the developed method fulfills the mission in shorter time.

For the second research line in Part II, we have considered that after reaching some primary goals locations to gather data, some agents must go to the Base Station in order to deliver the collected information.

For this purpose, in the Chapter 6, we have developed a method to synchronize two agents in movement. Firstly, we have formally defined the dynamic communication area and its different parts for synchronization in movement. We have developed two techniques to plan trajectories considering a time-constrained communication between the agents. In the proposed approaches, an agent must plan a trajectory in order to synchronize with a mate in movement, whose trajectory is known. The first method, so-called TC-RRT, consists in a random sampling of the space to obtain a feasible trajectory. Therefore, it obtains the solution in a short time, but being this solution sub-optimal. The second proposed method it is based on the Fast Marching Method, that obtains optimal solutions. However the computation requires a greater time, with respect to the sampling-based technique. Apart from obtaining the optimal solution, this method is able to choose the best solution for different criteria: distance and/or time. At the same time, the method can optimally choose the best mate to exchange data, from several mates present in the scenario, without any allocation procedure for this purpose. Being this, the second advantage with respect to the proposed sampling-based method.

Finally, in the Chapter 7, we have proposed a strategy, based on a sequence of algorithms, to plan the data gathering mission from an environment. From the Base Station, information from some locations of interest is requested periodically. The agents must plan the deployment

to, periodically, reach the goal locations, gather data and deliver the collected information to the base. We have proposed the usage of the agents in two roles: as workers, which gather data from the goal positions, and as collectors, which are travelling constant trajectories collecting the data of the workers in order to retransmit it to the Base Station. We have developed and evaluated three methods, based on the FMM, to split the scenario into working areas for worker agents. An iterative method plans the gathering mission, before deploying the agents: obtaining the best partition of the scenario, the best balance between the number of agents to be used in role of workers and in role of collectors, the associations between them to transmit the gathered data and the computation of the trajectories of the collectors, associated to the workers partitions. The algorithm adjusts the aforementioned parameters based on the refreshing time of the information at the Base Station and the amount of delivered information packages. The results show that, in the tested scenarios, the partition method that provides the best results is the one that splits the scenario in polygonal areas, for both terms, the delivered data and the refreshing time. It is also the method that provides the most reliable results, since the estimated values do not differ substantially from the obtained in the real data gathering mission when the plan is executed. As expected, the results have shown that there exist a limit for the number of agents employed as collectors. It is counterproductive to use many collectors at the expense of sacrificing agents for working tasks. Therefore, it is more profitable to devote one collector to serve to several workers.

8.1.1 Future work

Along the present document, we have used as planning tools two generic path planners: RRT and FMM. The most of the works use the Fast Marching Method, due to its advantages presented in the respective chapters. We have observed that the recent Fast Marching Trees (FMT) [97], could improve the results of the basic FMM for path planning. The FMTs are a perfect balance between the RRTs and FMM. They combine the efficiency of the FMM and the speed of the RRTs.

The FMM have also been used with the purpose of segmenting the scenario and to obtain the costs in terms of distances used in the allocation methods. In this sense, the FMTs cannot be used because they omit too many points of the scenario due to their random sampling nature. Because of that, we want to employ some variations of the basic FMM, as Group Marching Method (GMM) [98] or Untidy FMM (UFMM) [99], in order to reduce the complexity from $O(nlogn)$ to $O(n)$, without the loss of precision of the basic FMM.

This thesis has been devoted to develop different algorithms for multi-robot deployment planning, analyzing the results by means of simulations. This allows to evaluate the methods in large scenarios and using large teams of robots. But, we also want to carry out the experimentation in real-world scenarios, mainly to observe the influence of the variation of the signals in a real environment over the performance of the methods. Mainly in the case in which the conditions differ from the ones considered in the planning phase, for instance new obstacles or not foreseen disconnections. This is also interesting because of the usage of an intermittent communication in the present work. It requires a some time to establish the connectivity between the agents, in order to start the data transmission. Thus, due to the periodic connections and disconnections between the agents, we will need to include this time in the proposed planners.

In the first part of this thesis, the chain or tree formations, the agents were free to

navigate between goals without connectivity constraints. They only have to establish the communication links from the base to the agents when they are reaching the goals. As future work, we want to generalize the proposed methods in order to maintain a constant communication, that is, during the entire deployment mission. In the same way, we want to include dynamism in the proposed approaches. This means planning in environments with dynamic obstacles, which, in similar fashion as the connectivity constraints, will increase the time to fulfill the mission.

In the second part of this work, the agents plan their trajectories in order to synchronize in movement. As mentioned in the conclusions of the respective chapters, we want to adapt the proposed methods to environments with dynamic obstacles. For instance, any environment with human presence, as some factory. An immediate consequence is the necessity to include the predictions of the possible movements of the moving obstacles. In the case of humans, it would require to model and predict the possible movements of the person, within a temporal window. This, obviously, could include an uncertainty in the planned trajectories, increasing the time of synchronization. Therefore, the refreshing times as well as the number of the delivered information could be degraded. Thus, we will include this uncertainty in our algorithms in order to avoid to plan trajectories through areas where many dynamism is observed during the mission.

8.2 Conclusiones

A lo largo de la presente tesis, se han propuesto varios métodos para la planificación de despliegues de equipos multi-robot en entornos con restricciones de comunicaciones. La planificación de los despliegues se realiza para monitorizar un entorno, alcanzando determinados puntos de interés y entregando esa información a una estación base estática. Nos hemos centrado en dos líneas de investigación principales, asociadas sobre todo con el tipo de comunicación que deberán mantener los agentes durante el despliegue con la base.

Para la primera se ha considerado que a la hora de visitar los objetivos primarios, los agentes deberían transmitir la información a la base.

En el Capítulo 2, se ha propuesto un método para el cálculo de las posiciones óptimas para el posicionamiento de los robots que se utilizan en rol de relé, para retransmitir la información desde los objetivos primarios a la estación base a medida que estos se vayan visitado a lo largo de la misión de despliegue. La técnica propuesta obtiene las posiciones basándose en una topología del tipo árbol para interconectar a los agentes, que se encuentran en las posiciones de los objetivos primarios así como de relé. Se han propuesto tres criterios para el cálculo del árbol: conectividad, exclusividad y distancia. El método se ha evaluado en base a dos métricas: la cantidad de agentes relé utilizados para el despliegue y las distancias estimadas a los objetivos. Se ha comprobado que los dos criterios correspondientes al número de relés, o una combinación de ellos, proporcionan el mejor balance entre los enlaces necesarios para la misión y las distancias.

En el Capítulo 3, hemos desarrollado dos técnicas basadas en una secuencia de algoritmos para planificar el despliegue del equipo de agentes: el cálculo de las posiciones de los relés, del capítulo anterior, un planificador de caminos considerando comunicaciones y un método de asignación de los objetivos, de relé y primarios. El primer planificador de despliegues maximiza la conectividad durante el despliegue realizando la misión en el menor tiempo

posible. El segundo planifica la misión para realizar el despliegue con el menor número de agentes posibles, manteniendo un nivel de conectividad semejante al anterior, mediante la concentración de la ocupación en tareas de conectividad en determinados agentes. Con lo cual, incrementa el tiempo de la misión con respecto al primer planificador.

En el Capítulo 4, se ha generalizado el método de planificación de despliegues de los capítulos anteriores, para cualquier cantidad de agentes y objetivos a alcanzar. Primeramente, hemos propuesto un método sencillo pero rápido para el cálculo de las posiciones de relés, mediante el uso de cadenas de relés, que no son más que ramas del árbol que se proponía en el capítulo 2. Con este método se ha reducido el tiempo de cómputo de las posiciones de relés reduciendo sustancialmente el número de candidatos potenciales donde posicionarlos. Dichas cadenas de relés, junto con los objetivos primarios que conectan, forman *clusters* de objetivos. Hemos propuesto y evaluado distintos métodos heurísticos para la visita de los *clusters* de dos maneras: secuencial y en paralelo. Además se ha desarrollado un método para la asignación de los objetivos dentro de los propios *clusters*, teniendo en cuenta el tipo de objetivo, relé o primario. Los resultados han mostrado que en determinadas situaciones los métodos secuenciales tienen un mejor comportamiento, lo cual resulta poco evidente. Esto se debe al tipo de escenario. Los escenarios alargados, estrechos y en los cuales la estación base se encuentra en un extremo, dejan pocas posibilidades de desplegar varias cadenas, lo cuál favorece la secuencialización. Por el contrario, escenarios más anchos y en los cuales la base se sitúa en el centro, permite la extensión de varias cadenas en distintas direcciones, con lo cual métodos paralelos obtienen mejores resultados. El conjunto de estas técnicas permite una planificación rápida para cualquier cantidad de agentes y objetivos.

En el Capítulo 5, se ha propuesto un método sencillo para reconectar un grupo de agentes en entornos cambiantes ante la aparición de nuevos obstáculos, cierres de puertas, etc. El objetivo de los agentes, por el contrario que en otros trabajos, es doble: conseguir la reagrupación a la vez que adoptan una formación determinada. En este trabajo nos hemos centrado en la formación de cadenas. Se ha demostrado que en entornos sencillos, en la mayoría de ocasiones, el método es capaz de reagrupar a los agentes en una formación de cadena. Hemos comparado nuestro método con otra técnica que reconecta a los agentes y forma la cadena de manera secuencial, siempre asegurando la reconexión del equipo. Comparado con esta técnica, nuestro método completa la misón en menor tiempo.

En la segunda linea de investigación, se ha considerado que tras alcanzar los objetivos para la recopilación de datos, algunos de los agentes deben ir a la base para entregar la información.

Para ello, en el Capítulo 6, se ha desarrollado un método para la sincronización de dos agentes en movimiento. Primeramente se ha definido formalmente el área de comunicaciones dinámica y sus diferentes partes para la sincronización en movimiento. Se han desarrollado dos métodos para la planificación de trayectorias considerando una comunicación con restricciones temporales. En ambos métodos, un agente debe planificar su trayectoria para sincronizarse con otro compañero en movimiento, cuya trayectoria es conocida. El primer método, llamado TC-RRT, consiste en el muestreo aleatorio del espacio para el cálculo de las trayectorias. Por ello obtiene una solución en un intervalo corto de tiempo a cambio de que la misma sea subóptima. El segundo método propuesto se basa en el FMM, con lo cual la solución obtenida es óptima. Sin embargo el cálculo requiere más tiempo. Además de obtener la solución óptima, con este método se puede elegir el criterio para la elección de la mejor solución: distancia y/o tiempo. Al mismo tiempo, el método es capaz de elegir de

manera eficiente al mejor compañero para la sincronización de entre varios. Siendo esta, la segunda ventaja frente al TC-RRT.

Finalmente, en el Capítulo 7, se ha propuesto una estrategia, basada en una secuencia de algoritmos, para planificar una misión de recopilación de datos en un entorno. Desde la estación base periódicamente se solicita información de unos puntos de interés determinados. Los agentes deben planificar la misión para, de manera periódica, alcanzar los objetivos, tomar muestras y entregárselos a la base. Hemos propuesto el uso de los agentes en dos roles: trabajadores, que recolectan datos del entorno, y colectores, que realizan trayectorias constantes recolectando la información de los trabajadores para posteriormente retransmitirla a la estación base. Hemos desarrollado y analizado tres métodos, basados en el FMM, para partir el escenario en áreas de trabajo para los trabajadores. Se ha propuesto un método iterativo para la planificación de la misión, antes de desplegar a los agentes: obteniendo la mejor partición del entorno, el mejor balance entre la cantidad de agentes que se usará a modo de trabajadores y colectores, las asociaciones entre ellos para comunicarse y el cálculo de las trayectorias de los colectores, asociadas a las particiones de los trabajadores. Los resultados han mostrado que, en los escenarios testeados, el método de partición en áreas poligonales es el que proporciona los mejores resultados, tanto en términos de cantidad de información recolectada y su tiempo de *refresco*, así como en términos de fiabilidad. Es decir, las estimaciones realizadas previamente a la ejecución de la misión, se han acercado mucho a los resultados obtenidos durante el despliegue. Como era de esperar, los resultados han mostrado que el uso de demasiados colectores es contraproducente ya que se dispone de menor cantidad de trabajadores. Resultando más rentable dedicar un colector a recopilar datos de varios trabajadores.

8.2.1 Trabajo futuro

A lo largo de la presente tesis, se han utilizado 2 planificadores de caminos de base: RRT y FMM. En gran mayoría de trabajos ha sido el FMM, debido a las ventajas que se han presentado y las cuales se han podido observar en los distintos métodos desarrollados. Hemos visto que la reciente aparición de los FMTs (Fast Marching Trees) [97] podría mejorar los resultados que obtenemos con FMM en cuanto al cálculo de caminos. Es un balance entre RRT y FMM, mezcla la eficiencia de FMM y la rapidez de los RRT.

El FMM también ha sido utilizado para la segmentación de escenarios, a la vez que para el cálculo de los costes de distancias usados en por los métodos de asignación. En este sentido, los FMT no podrían servir para reducir tiempo de cálculo, ya que omiten demasiados puntos del escenario debido a su naturaleza aleatoria. Por ello, se deben utilizar métodos que consideren el grid completo. Es decir, que podríamos utilizar alguna variación del FMM básico, tales como Group Marching Method (GMM) [98] o Untidy FMM (UFMM) [99], para, de esta manera reducir la complejidad de $O(nlogn)$ a $O(n)$, sin la pérdida de precisión del método básico.

Deseamos realizar una experimentación real, para poder observar la variación de la señal en entornos reales. Sobre todo resultaría interesante, por el hecho de haber utilizado una comunicación intermitente a lo largo de todo el trabajo realizado en la presente tesis. El establecimiento de conectividad entre dos agentes para poder comenzar a transmitir datos, requiere un cierto tiempo. Por tanto, debido a las conexiones y desconexiones periódicas entre agentes, se deberá incluir ese tiempo para el establecimiento de comunicación en los

métodos desarrollados.

En la primera parte de la tesis, la formación de las cadenas, se le permite a los agentes moverse libremente entre los objetivos. Tan solo se requeire que se pueda establecer comunicación desde la Estación Base hasta los agentes, al alcanzar los objetivos. Como un trabajo futuro queremos generalizar los métodos propuestos para mantener una comunicación continua, es decir, a lo largo de todo el trayecto que realizan los robots. Esto, muy posiblemente, incrementará el tiempo de la misión, puesto que los movimientos de los agentes se verán drásticamente restringidos. Del mismo modo, queremos incluir dinamismo en los métodos propuestos. Es decir, planificar en entornos con obstáculos dinámicos. Lo cual, al igual que las restricciones de conectividad incrementará el tiempo para finalizar la misión.

En la segunda parte de este trabajo, los agentes planifican trayectorias para la sincronización en movimiento. Tal y como ya se ha mencionado en las conclusiones de los capítulos correspondientes, deseamos adaptar los métodos propuestos a entornos con presencia de obstáculos dinámicos. Por ejemplo, un entorno cualquiera con presencia de personas. La consecuencia inmediata, sería la necesidad de incluir alguna predicción de los posibles movimientos de los obstáculos. En el caso de seres humanos, sería modelar y predecir los posibles movimientos que hacen las personas, dentro de una ventana temporal. Esto, obviamente, añadiría una cierta incertidumbre en las trayectorias, ya que si un obstáculo desconocido afecta las trayectorias de los robots, el tiempo para la sincronización aumentaría. Como consecuencia, los tiempos de refresco así como la cantidad de datos entregados, evaluados en el presente trabajo, se verían degradados. Por tanto, queremos incluir dicha incertidumbre en nuestros algoritmos, con el fin de, por ejemplo, poder evitar planificar caminos en zonas en las cuales se ha observado mucho dinamismo a lo largo de la misión.

Appendix A

Fast Marching Method (FMM)

The Fast Marching Method was originally proposed by Sethian in [37], as a solution to the Eikonal equation. A simple way to describe the method is as the propagation of a wavefront from some source position over all the map, computing the distance gradient from the source to each position of the map.

The computation of the gradient distance is obtained by solving the following equation:

$$|\nabla D|F = 1 \tag{A.1}$$

where F is the propagation speed and D is the distance value of the gradient.

The algorithm used in the present work to propagate the wavefront computing the distance gradient is described in algorithm 21.

The algorithm receives the source location or locations $\mathbf{x}_{source}$ and the grid with the speed of the wavefront F. In the case of using a uniform propagation of the wavefront, F takes the values of the grid: $F = 1$ for cells that represent free space and $F = 0$ for cells that contain an obstacle. The wavefront has two variables: the positions reached by the wavefront

Algorithm 21 Gradient computation

Require: $\mathbf{x}_{source}, F$

$\mathbf{x}_w \leftarrow \mathbf{x}_{source}, \quad \mathbf{c}_w \leftarrow 0$ ▷ Initialize wavefront
$D \leftarrow \infty$ ▷ Initialize gradient
$D(\mathbf{x}_w) = 0$
while *condition* **do**
  $x \leftarrow \mathbf{x}_w(0), c \leftarrow \mathbf{c}_w(0)$ ▷ First value of the wavefront
  $\mathbf{x}_{neigh} \leftarrow neighbours(x)$ ▷ Neighbours of x
  for each $x_i \in \mathbf{x}_{neigh}$ **do**
    if $F(x_i)$ & $D(x_i) == \infty$ **then** ▷ Non obstacle and already non computed
      $c_i \leftarrow compute\_cost(x_i)$ ▷ Interpolation with costs of neighbours of x_i
      $[\mathbf{x}_w, \mathbf{c}_w] \leftarrow insert\_in\_wavefront(x_i, c_i)$
      $D(x_i) = c_i$
    end if
  end for
end while
return D

$\mathbf{x}_w$ and the distance from the source to these positions $\mathbf{c}_w$. The wavefront is initialized with the positions of the source or sources and the cost is zero for all these positions, in line 1 of the algorithm. All the positions of the gradient are initialized with infinity, because these positions have not already been computed, line 2. The distance to positions that represent the origin of the wavefront or wavefronts are set to zero, line 3. The algorithm iterates until accomplishing some condition, specified by the user defines in line 4. The different stoppage conditions are described in the following section A.1.

At each iteration the first element of the wavefront is extracted from the list, in line 5. This is because the wavefront is a sorted list of positions and distances that is arranged in ascending order of the distance to the source. So that, the first element is always pointing to the closest position of the wavefront to the source. Then, the neighbours of the point are obtained in line 6. These points are the positions to open and compute the distance.

We check that the neighbor cell does not contain an obstacle and is not already computed, in lines 7-8. The, the method computes the distance as described in section A.2, line 9. This procedure consists in selecting the best neighbours to interpolate the distance and the approximation of the eq.(A.1). The new point is inserted into the wavefront in line 10, sorted with the distance in ascending order. The obtained cost is the distance gradient at this position in line 11. The algorithm iterates until some of the conditions of the section A.1 is not accomplished.

A.1 Stoppage conditions

In the present work we have used the Fast Marching Method for different purposes. Therefore, we have established different wavefront propagation conditions, based on the type of the task. We have employed the FMM for three purposes: (1) to cover the entire map of the scenario; (2) to reach some specified position or positions; (3) to extend a specified number of cells.The different stoppage conditions, referred to line 4 of Alg.21, are:

1. Covering all the free cells of the scenario. The distance gradient to all the free positions of the grid is obtained. This is accomplished when the wavefront has covered all the grid positions and it is empty. We employ this condition mainly for some segmentation procedures proposed in this work. This is formally defined as:

$$condition = |\mathbf{x}_w| > 0 \tag{A.2}$$

2. Reaching position/s. The wavefront is extended until reaching some desired positions $\mathbf{x}_i$. So, when these positions are inserted into the wavefront, the distance to these locations have been already obtained and the algorithm stops iterating. This condition is used when we want to obtain the costs to reach one or several goal locations and we do not need to extend the wavefront over the entire grid. It is formally expressed as:

$$condition = \{|D(\mathbf{x}_i)| == \infty \parallel |\mathbf{x}_w| > 0\} \tag{A.3}$$

   Some positions of the scenario may be isolated from the source of the wavefront. So, in order to assure the stoppage, we include the condition A.2, that checks if the wavefront has covered the region of the source.

3. Expand specified number of cells. This condition is only used by the proposed *BAP* segmentation method, described in Chapter 7. The algorithm iterates, propagating the wavefront until the number of opened cells is lower than a specified number of cells by the user K_{cells}:

$$condition = \{|D(\mathbf{x}) < \infty| \leq K_{cells} \parallel |\mathbf{x}_w| > 0\} \tag{A.4}$$

Similarly to the previous condition, the wavefront may start from some small area with fewer free cells than K_{cells}. So, we include the condition A.2 to avoid infinite loops.

A.2 Distance Gradient Approximation

We take the formulation of the FMM presented in [100] for two dimensional case. So the approximation of the distance defined in eq.(A.1) can be rewritten as:

$$max(T_{ij}^{-x}D, -T_{ij}^{+x}D, 0) + max(T_{ij}^{-y}D, -T_{ij}^{+y}D, 0) = \frac{1}{F_{ij}^2} \tag{A.5}$$

where ij correspond to coordinates in the grid, F_{ij}^2 is the propagation velocity at this coordinates, T_{ij}^{-x} is the finite difference operator along negative x at ij and D is the distance function to be computed.

Thus, eq.(A.5) is the equation to be solved with *compute_cost* function in l.9 of Alg.21. This function selects the best neighbours for the interpolation to compute the cost of x_i in the algorithm. The selection is based in the selection of up to two neighbours, each one from different axes. As developed in [100], there are two best neighbours for interpolation A and B, so that $D_A \leq D_B$. A denotes the neighbour of the horizontal axis and B of the the vertical. Therefore, the expression (A.5) is reformulated as:

$$(D - D_A)^2 + (D - D_B)^2 = \frac{h^2}{F^2} \Leftrightarrow \begin{cases} D = d_A = d_B \\ (d_A - D_A)^2 + (d_B - D_B)^2 = \frac{h^2}{F^2} \end{cases} \tag{A.6}$$

The two parameters d_A and d_B are interpreted as the axes of a Cartesian coordinate frame. The solutions for eq.(A.6) are found at the intersections between the diagonal $d_A = d_B$ and a circle of radius h/F centered at (D_A, D_B)

Developing this formulas, the final expression of the gradient distance is obtained by the expression (A.7):

$$D = \begin{cases} D_A + h/F & \leftarrow \quad D_B - D_A \geq h/F \\ \frac{(\beta + \sqrt{\beta^2 - 4\gamma})}{2} & otherwise \end{cases} \tag{A.7}$$

and where

$$\beta = -(T_A + T_B) \qquad \gamma = \frac{T_A^2 + T_B^2 - h^2/F^2}{2} \tag{A.8}$$

Appendix B

Bibliography

[1] Y. Marchukov and L. Montano, "Multi-robot optimal deployment planning under communication constraints," in *Robot 2015: Second Iberian Robotics Conference* (L. P. Reis, A. P. Moreira, P. U. Lima, L. Montano, and V. Muñoz-Martinez, eds.), (Cham), pp. 677–690, Springer International Publishing, 2016.

[2] Y. Marchukov and L. Montano, "Communication-aware planning for robot teams deployment," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 6875 – 6881, 2017. 20th IFAC World Congress.

[3] Y. Marchukov and L. Montano, "Fast and scalable multi-robot deployment planning under connectivity constraints," in *2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, April 2019.

[4] Y. Marchukov and L. Montano, "Multi-robot coordination for connectivity recovery after unpredictable environment changes," *IFAC-PapersOnLine*, 2019. 10th IFAC Symposium on Intelligent Autonomous Vehicles.

[5] Y. Marchukov and L. Montano, "Trajectory planning under time-constrained communication," in *ROBOT 2017: Third Iberian Robotics Conference*, (Cham), pp. 794–806, Springer International Publishing, 2018.

[6] Y. Marchukov and L. Montano, "Multi-agent coordination for on-demand data gathering with periodic information upload," in *17th International Conference on Practical Applications of Agents and Multi-Agent Systems*, 26-28 June 2019.

[7] Y. Marchukov and L. Montano, "Multi-agent coordination for data gathering with periodic requests and deliveries," in *17th International Conference on Practical Applications of Agents and Multi-Agent Systems*, 26-28 June 2019.

[8] M.Ji and M.Egenrstedt, "Distributed coordination control of multiagent systems while preserving conectedness," *IEEE Transactions on Robotics*, vol. 23, no. 4, 2007.

[9] M. M. Zavlanos, M. B. Egerstedt, and G. J. Pappas, "Graph-theoretic connectivity control of mobile robot networks," *Proceedings of the IEEE*, vol. 99, no. 9, pp. 1525–1540, 2011.

[10] P.Urcola, L.Riazuelo, M.T.Lázaro, and L.Montano, "Cooperative navigation using environment compliant robot formations," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008.

[11] P.Urcola and L.Montano, "Adapting robot team behavior from interaction with a group of people," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011.

[12] M.N.Rooker and A.Birk, "Multi-robot exploration under the constraints of wireless networking," *Control Engineering Practice*, vol. 15, no. 4, pp. 435–445, 2007.

[13] G. A. Hollinger and S. Singh, "Multirobot coordination with periodic connectivity: Theory and experiments," *IEEE Transactions on Robotics*, vol. 28, pp. 967–973, Aug 2012.

[14] D.Tardioli, A.R.Mosteo, L.Riazuelo, J.L.Villaroel, and L.Montano, "Enforcing network connectivity in robot team missions," *The International Journal of Robotics Research*, vol. 29, no. 4, pp. 460—-480, 2010.

[15] J.Fink, A.Ribeiro, and V.Kumar, "Robust control of mobility and communications in autonomous robot teams," *IEEE Access*, vol. 1, 2013.

[16] Z. Mi, Y. Yang, and G. Liu, "Coverage enhancement of mobile multi-agent networks while preserving global connectivity," in *2011 IEEE International Conference on Robotics and Automation*, pp. 5381–5386, May 2011.

[17] N. Majcherczyk, A. Jayabalan, G. Beltrame, and C. Pinciroli, "Decentralized connectivity-preserving deployment of large-scale robot swarms," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4295–4302, Oct 2018.

[18] C. Rizzo, D. Tardioli, D. Sicignano, L. Riazuelo, J. L. Villarroel, and L. Montano, "Signal based deployment planning for robot teams in tunnel-like fading environments," *The International Journal of Robotics Research*, 2013.

[19] Y. Yan and Y. Mostofi, "Robotic router formation in realistic communication environments," *IEEE Transactions on Robotics*, vol. 28, pp. 810–827, Aug 2012.

[20] P. K. Penumarthi, A. Q. Li, J. Banfi, N. Basilico, F. Amigoni, J. O'Kane, I. Rekleitis, and S. Nelakuditi, "Multirobot exploration for building communication maps with prior from communication models," in *2017 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, pp. 90–96, Dec 2017.

[21] P. Mirowski, T. K. Ho, and P. Whiting, "Building optimal radio-frequency signal maps," in *2014 22nd International Conference on Pattern Recognition*, pp. 978–983, Aug 2014.

[22] M. Lindhé and K. H. Johansson, "Exploiting multipath fading with a mobile robot," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1363–1380, 2013.

[23] S.O.Anderson, R.Simmons, and D.Goldberg, "Maintaining line-of-sight communications networks between planetary rovers," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2266–2272, 2003.

[24] O. Burdakov, P. Doherty, K. Holmberg, J. Kvarnström, and P.-M. Olsson, "Relay positioning for unmanned aerial vehicle surveillance*," *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 1069–1087, 2010.

[25] E. F. Flushing, L. M. Gambardella, and G. A. D. Caro, "On using mobile robotic relays for adaptive communication in search and rescue missions," in *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pp. 370–377, Oct 2016.

[26] J. Banfi, A. Q. Li, N. Basilico, I. Rekleitis, and F. Amigoni, "Asynchronous multirobot exploration under recurrent connectivity constraints," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5491–5498, May 2016.

[27] J. Banfi, A. Quattrini Li, I. Rekleitis, F. Amigoni, and N. Basilico, "Strategies for coordinated multirobot exploration with recurrent connectivity constraints," *Autonomous Robots*, vol. 42, pp. 875–894, Apr 2018.

[28] Y. Pei and M. W. Mutka, "Steiner traveler: Relay deployment for remote sensing in heterogeneous multi-robot exploration," in *2012 IEEE International Conference on Robotics and Automation*, pp. 1551–1556, May 2012.

[29] Y. Pei, M. W. Mutka, and N. Xi, "Coordinated multi-robot real-time exploration with connectivity and bandwidth awareness," in *2010 IEEE International Conference on Robotics and Automation*, pp. 5460–5465, May 2010.

[30] P. Mukhija, K. M. Krishna, and V. Krishna, "A two phase recursive tree propagation based multi-robotic exploration framework with fixed base station constraint," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4806–4811, Oct 2010.

[31] N. Majcherczyk, A. Jayabalan, G. Beltrame, and C. Pinciroli, "Decentralized connectivity-preserving deployment of large-scale robot swarms," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4295–4302, Oct 2018.

[32] M.Ficco, C.Esposito, and A.Napolitano, "Calibrating indoor positioning systems with low efforts," *IEEE Transactions on Mobile Computing*, vol. 13, pp. 737–751, 2014.

[33] M. Lott and I. Forkel, "A multi-wall-and-floor model for indoor radio propagation," in *Vehicular Technology Conference, 2001. VTC 2001 Spring. IEEE VTS 53rd*, vol. 1, pp. 464–468 vol.1, 2001.

[34] D. Tardioli, D. Sicignano, L. Riazuelo, A. Romeo, J. L. Villarroel, and L. Montano, "Robot teams for intervention in confined and structured environments," *Journal of Field Robotics*, vol. 33, no. 6, pp. 765–801, 2016.

[35] C. Ghedini, C. H. C. Ribeiro, and L. Sabattini, *A Decentralized Control Strategy for Resilient Connectivity Maintenance in Multi-robot Systems Subject to Failures*, pp. 89–102. Cham: Springer International Publishing, 2018.

[36] E. F. Flushing, M. Kudelski, L. M. Gambardella, and G. A. D. Caro, "Spatial prediction of wireless links and its application to the path control of mobile robots," in *Proceedings of the 9th IEEE International Symposium on Industrial Embedded Systems (SIES 2014)*, pp. 218–227, June 2014.

[37] J. A. Sethian, "A fast marching level set method for monotonically advancing fronts," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 93, pp. 1591–1595, Dec. 1996.

[38] J. Twigg, J. Fink, P. Yu, and B. Sadler, "Efficient base station connectivity area discovery," *The International Journal of Robotics Research*, vol. 32, pp. 1398–1410, 2013.

[39] M. S. Hassouna and A. A. Farag, "Multistencils fast marching methods: A highly accurate solution to the eikonal equation on cartesian domains," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, pp. 1563–1574, Sep. 2007.

[40] J. Sethian, "Evolution, implementation, and application of level set and fast marching methods for advancing fronts," *Journal of Computational Physics*, vol. 169, no. 2, pp. 503 – 555, 2001.

[41] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[42] E. Stump, N. Michael, V. Kumar, and V. Isler, "Visibility-based deployment of robot formations for communication maintenance," in *2011 IEEE International Conference on Robotics and Automation*, pp. 4498–4505, May 2011.

[43] Y. Solana, M. Furci, J. Cortés, and A. Franchi, "Multi-robot path planning with maintenance of generalized connectivity," in *2017 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, pp. 63–70, Dec 2017.

[44] P. Vansteenwegen, W. Souffriau, and D. V. Oudheusden, "The orienteering problem: A survey," *European Journal of Operational Research*, vol. 209, no. 1, pp. 1 – 10, 2011.

[45] G.A.Croes, "A method for solving traveling-salesman problems," *Operations Research*, vol. 6, Dec 1958.

[46] C. S. W. Burgard, M. Moors and F. E. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on Robotics*, vol. 21, pp. 376–386, 2005.

[47] P. B. Sujit and J. B. Sousa, "Multi-uav task allocation with communication faults," in *2012 American Control Conference (ACC)*, pp. 3724–3729, June 2012.

[48] P. Ulam and R. C. Arkin, "When good communication go bad: communications recovery for multi-robot teams," in *IEEE International Conference on Robotics and Automation, 2004.*, vol. 4, pp. 3727–3734 Vol.4, April 2004.

[49] G. Habibi and J. McLurkin, "Maximum-leaf spanning trees for efficient multi-robot recovery with connectivity guarantees," in *Distributed Autonomous Robotic Systems*, pp. 275–289, Springer Berlin Heidelberg, 2014.

[50] G. Habibi, L. Schmidt, M. Jellins, and J. McLurkin, "K-redundant trees for safe and efficient multi-robot recovery in complex environments," in *Robotics Research: The 16th International Symposium ISRR*, (Cham), Springer International Publishing, 2016.

[51] J. M. M. Filho, E. Lucet, and D. Filliat, "Real-time distributed receding horizon motion planning and control for mobile multi-robot dynamic systems," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 657–663, May 2017.

[52] A. M. F. Stulp, W. Koska and M. Beetz, "Seamless execution of action sequences," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3687–3692, 2007.

[53] A. D. Dragan, K. C. T. Lee, and S. S. Srinivasa, "Legibility and predictability of robot motion," in *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 301–308, March 2013.

[54] S. Prentice and N. Roy, "The belief roadmap: Efficient planning in belief space by factoring the covariance," *The International Journal of Robotics Research*, vol. 28, no. 11-12, pp. 1448–1465, 2009.

[55] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *2011 IEEE International Conference on Robotics and Automation*, pp. 723–730, May 2011.

[56] V. Indelman, "Cooperative multi-robot belief space planning for autonomous navigation in unknown environments," *Autonomous Robots*, vol. 42, pp. 353–373, 2018.

[57] A. A. G. C. J. H. C. Amato, G. Konidaris and L. Kaelbling, "Policy search for multi-robot coordination under uncertainty," *International Journal of Robotics Research*, vol. 35, pp. 1760–1778, 2017.

[58] A. Farinelli, L. Iocchi, and D. Nardi, "Distributed on-line dynamic task assignment for multi-robot patrolling," *Autonomous Robots*, vol. 41, pp. 1321–1345, Aug 2017.

[59] D. Portugal and R. Rocha, "Msp algorithm: Multi-robot patrolling based on territory allocation using balanced graph partitioning," in *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, (New York, NY, USA), pp. 1271–1276, ACM, 2010.

[60] J. M. Díaz-Bánez, L. E. Caraballo, M. A. Lopez, S. Bereg, I. Maza, and A. Ollero, "A general framework for synchronizing a team of robots under communication constraints," *IEEE Transactions on Robotics*, vol. 33, pp. 748–755, June 2017.

[61] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, pp. 269–271, Dec. 1959.

[62] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, pp. 100–107, July 1968.

[63] N. M. Amato and Y. Wu, "A randomized roadmap method for path and manipulation planning," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 1, pp. 113–120 vol.1, Apr 1996.

[64] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," tech. rep., Department of Computer Science, Iowa State University, 1998.

[65] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 2, pp. 995–1001 vol.2, April 2000.

[66] A. Yershova, L. Jaillet, T. Simeon, and S. M. LaValle, "Dynamic-domain rrts: Efficient exploration by controlling the sampling domain," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 3856–3861, April 2005.

[67] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[68] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2997–3004, Sept 2014.

[69] F. H. Tseng, T. T. Liang, C. H. Lee, L. D. Chou, and H. C. Chao, "A star search algorithm for civil uav path planning with 3g communication," in *2014 Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pp. 942–945, Aug 2014.

[70] K. I. Joy, "Breshenham's algorithm. in visualization and graphics research group," in ., December 1999.

[71] J. Goldhirsh and W. Vogel, "Handbook of propagation effects for vehicular and personal mobile satellite systems: Overview of experimental and modelling results," in ., December 1998.

[72] N. Mathew, S. L. Smith, and S. L. Waslander, "A graph-based approach to multi-robot rendezvous for recharging in persistent tasks," in *2013 IEEE International Conference on Robotics and Automation*, pp. 3497–3502, May 2013.

[73] M. Guo and M. M. Zavlanos, "Distributed data gathering with buffer constraints and intermittent communication," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 279–284, May 2017.

[74] M. Meghjani and G. Dudek, "Combining multi-robot exploration and rendezvous," in *2011 Canadian Conference on Computer and Robot Vision*, pp. 80–85, May 2011.

[75] M. Meghjani, S. Manjanna, and G. Dudek, "Fast and efficient rendezvous in street networks," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1887–1893, Oct 2016.

[76] B. Coltin and M. Veloso, "Online pickup and delivery planning with transfers for mobile robots," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5786–5791, May 2014.

[77] L. Fermin-Leon, J. Neira, and J. A. Castellanos, "Incremental contour-based topological segmentation for robot exploration," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2554–2561, May 2017.

[78] A. Kleiner, R. Baravalle, A. Kolling, P. Pilotti, and M. Munich, "A solution to room-by-room coverage for autonomous cleaning robots," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5346–5352, Sep. 2017.

[79] M. Liu, F. Colas, L. Oth, and R. Siegwart, "Incremental topological segmentation for semi-structured environments using discretized gvg," *Autonomous Robots*, vol. 38, pp. 143–160, Feb 2015.

[80] M. Kallmann, "Shortest paths with arbitrary clearance from navigation meshes," in *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '10, (Goslar Germany, Germany), pp. 159–168, Eurographics Association, 2010.

[81] M. Tzes, S. Papatheodorou, and A. Tzes, "Visual area coverage by heterogeneous aerial agents under imprecise localization," *IEEE Control Systems Letters*, vol. 2, pp. 623–628, Oct 2018.

[82] R. Bormann, F. Jordan, W. Li, J. Hampp, and M. Hägele, "Room segmentation: Survey, implementation, and analysis," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1019–1026, May 2016.

[83] E. Fabrizi and A. Saffiotti, "Augmenting topology-based maps with geometric information," *Robotics and Autonomous Systems*, vol. 40, no. 2, pp. 91 – 97, 2002. Intelligent Autonomous Systems - IAS -6.

[84] A. Diosi, G. Taylor, and L. Kleeman, "Interactive slam using laser and advanced sonar," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 1103–1108, April 2005.

[85] P. Beeson, N. K. Jong, and B. Kuipers, "Towards autonomous topological place detection using the extended voronoi graph," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 4373–4379, April 2005.

[86] E. Brunskill, T. Kollar, and N. Roy, "Topological mapping using spectral clustering and classification," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3491–3496, Oct 2007.

[87] K. Sjöö, "Semantic map segmentation using function-based energy maximization," in *2012 IEEE International Conference on Robotics and Automation*, pp. 4066–4073, May 2012.

[88] K. Zheng, A. Pronobis, and R. Rao, "Learning graph-structured sum-product networks for probabilistic semantic maps," in *AAAI Conference on Artificial Intelligence*, 2018.

[89] M. Luperto and F. Amigoni, "Predicting the global structure of indoor environments: A constructive machine learning approach," *Autonomous Robots*, vol. 43, pp. 813–835, Apr 2019.

[90] K. M. Wurm, C. Stachniss, and W. Burgard, "Coordinated multi-robot exploration using a segmentation of the environment," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1160–1165, Sept 2008.

[91] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, pp. 243–255, April 2004.

[92] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, pp. 243–255, April 2004.

[93] 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 *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1321–1327, May 2017.

[94] E. Nunes and M. Gini, "Multi-robot auctions for allocation of tasks with temporal constraints," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pp. 2110–2216, AAAI Press, 2015.

[95] Z. Ma, K. Yin, L. Liu, and G. S. Sukhatme, "A spatio-temporal representation for the orienteering problem with time-varying profits," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6785–6792, Sep. 2017.

[96] P. Vansteenwegen, W. Souffriau, and D. V. Oudheusden, "The orienteering problem: A survey," *European Journal of Operational Research*, vol. 209, no. 1, pp. 1 – 10, 2011.

[97] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions," *The International Journal of Robotics Research*, vol. 34, no. 7, pp. 883–921, 2015. PMID: 27003958.

[98] S. Kim and D. Folie, *The group marching method: An O(N) level set eikonal solver*, pp. 2297–2300. ., 2005.

[99] L. Yatziv, A. Bartesaghi, and G. Sapiro, "O(n) implementation of the fast marching algorithm," *Journal of Computational Physics*, vol. 212, no. 2, pp. 393 – 399, 2006.

[100] R. Philippsen and R. Siegwart, "An interpolated dynamic navigation function," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 3782–3789, April 2005.