POLITECNICO DI MILANO
Master of Science in Computer Science and Engineering
Dipartimento di Elettronica, Informazione e Bioingegneria

POLITECNICO
MILANO 1863

An Analysis of Coordination Mechanisms for Multi-Robot Exploration of Indoor Environments

Thesis advisor:
Prof. Francesco Amigoni
Co-advisors:
Eng. Alessandro Riva
Dr. Jacopo Banfi

Candidate:
Marco Cattaneo, 842188

Academic Year 2016/2017

Ai miei genitori...

Abstract

In this thesis, we consider the problem of coordinating a team of multiple robots to explore an initially unknown environment. The goal of the robots is to explore the environment in order to build its map in the shortest possible time, managing the interferences between them. To coordinate the agents, we propose four new coordination mechanisms, which are variant that improve existing mechanisms. The mechanisms belong to two different families: *online* and *offline*. Online mechanisms allocate the robots by dinamically taking into account the current decisions made by all team members to assign a goal location to an agent, while offline mechanisms define static roles before the exploration starts and, thanks to them, drive the actions of the single agents. To properly compare the mechanisms a formal framework is proposed. The framework involves three main concepts: the *states* through which the exploration evolves, the *coordination mechanism* that allocates target locations to the agents, and the evaluation criteria related to the measures of *interference* and *availability*. The interference measure is here formally defined as the average distance agents have to each other, while the availability measures the average distance agents have to their assigned locations. The four coordination mechanisms that we propose are tested against some benchmark ones. The experiments are performed in 2D simulated environments and the mechanisms are tested over different types of environments. Results show how offline coordination mechanisms outperform the online ones in the initial stages of the exploration, while the situation is reverse as the exploration proceeds. Moreover, offline mechanisms work better in open environments, compared to online mechanisms. As a conclusion, results suggest that a combination of the online and offline mechanisms could provide better performance over a wide range of environments.

Sommario

In questa tesi, consideriamo il problema di coordinare un team composto da pi robot al fine di explorare un ambiente inizialmente sconosciuto. L'obiettivo dei robot di costruire una mappa dell'ambiettente, minimizzando il tempo di esplorazione e gestendo le loro possibili interferenze. Per coordinare gli agenti, proponiamo quattro nuovi meccanismi di coordinazione, i quali sono varianti migliorative di meccanismi gi esistenti. I meccanismi appartengono a due famiglie: online e offline. I meccanismi online allocano i robot considerando dinamente le azioni in cui tutti i membri del team sono correntemente coinvolti, mente i meccanismi offline basano la coordinazione sulla definizione di ruoli, assignati agli agenti prima dell'inizio dell'esplorazione. Per confrontare adeguatamente i meccanismi. proponiamo un framework definitio formalmente per descrivere il problema della coordinazione multi-robot. Il framework si basa su tre concetti fondamentali: gli *stati* attraverso cui evolve l'esplorazione, i meccanismi di coordinazione che allocato i robot ai loro obiettivi, e le misure di interferenze e disponibilit. L'interferenza formalmente definita come la distanza media tra gli agenti, mentre la disponibilit corrisponde alla distanza media tra gli agenti e gli obiettivi a loro assegnati. I quattro meccanismi di coordinazione proposti sono stati testati in opposizione ad altri gi esistenti, presi come riferimento. Gli esperimenti sono stati condotti su ambienti bidimensionali e i meccanismi sono stati testati su diversi tipi di ambienti. I risultati mostrano come i meccanismi offline si comporitno megli di quelli offline negli stadi iniziali dell'esplorazione, mentre la situazione si capovolge mano a mano che l'esplorazione procede. Inoltre, i meccanismi offline lavorano meglio, rispetto a quelli online, su ambienti aperti. Concludendo, i risultati suggeriscono che una combinazione dei meccanismi online ed offline potrebbe portare alle migliori performance su una vasta gamma di ambienti.

Contents

Chapter 1

Introduction

Mobile robots currently play an important role during different activities, either through teleoperation or autonomous decision making. Robots are deployed in scenarios where the direct intervention of human operators is dangerous or not possible, ranging from volcano interiors to the aftermath of a natural disaster. In these cases the teleoperation control is frequently difficult because of the extreme conditions and, for this reason, an autonomous robot system is often preferable. In search and rescue or mapping settings robots are called to explore an initially unknown environment in order to build a representation of space, called map. The deployment of multiple robots in such settings is known to bring several benefits to the exploration [13, 19], including an increased fault-tolerance and a higher efficiency of the system. However, considering a set of robots poses different challenges. Robots, indeed, have to merge their local maps into a common and coherent view and they must be able to locate themselves on such a map. Moreover, team members should cooperate to avoid collisions between them and to increase the overall exploration performances.

This thesis is about multi-robot exploration of unknown environments. In particular, the work is related to coordination mechanisms deployed by the team of robots to manage the interference between them and to reduce the exploration time of an initially unknown environment. The main aspect considered in this work is the difference between online and offline coordination mechanisms. Online mechanisms allocate the robots by dinamically taking into account the current decisions made by all team members to choose the goal location for an agent, while offline methods define static roles before the exploration starts and, thanks to them, drive the actions of the single agents.

The aim of the thesis is twofold: propose a framework to describe the coordination of multiple robots and test different coordination mechanisms under the framework proposed. To properly compare strategies, indeed, the need of a unified framework to represent the basic features of a large set of approaches to the problem emerges. Such a framework is presented and used to derive precise evaluation criteria suitable to analyze a priori (without the need of experiments) the possible behaviour of different coordination mechanisms on a

given environment. To assess the effectiveness of the framework and the performance of the proposed mechanisms, several experiments are performed over different, simulated, 2D environments.

Offline mechanisms appeare to be more effective in the early stages of the exploration, while online mechanisms usually maintain the agents close to each other. However, as the exploration goes on, many offline mechanisms tend to leave the agents to explore independently, free to pursue individual payoffs.

Before going on with the discussion, it is important to clarify what are the main elements characterizing the problem under analysis. A multi-robot exploration problem involves different aspects and can be informally defined as: given a set of robots and an environment unknown to them, design a method which makes the robots explore the environment in the shortest possible time. Therefore, a method should take the environment already perceived by the robots and their locations as input, put them together in a common map, and decide how to move the agents in the environment to minimize the overall exploration time. We are now faced with three problems of figuring out how the robots have moved with respect to their map, of extending the map based on new sensor data acquired by all team members, and of taking decisions based on the robots' poses estimates and the map just built in order to plan the paths of the agents. The former two problems are known as the Simultaneous Localisation And Mapping (SLAM) problem and are well studied in the literature, both in the single and multi-robot settings [7] [21] [54] [41]. In the multi-robot case, besides mapping for themselves, robots have to merge their local maps into a common one and must be able to locate their poses on the common representation. In [11] a solution to the multi-robot SLAM problem is given, providing an approach to merge the local maps when a couple of robots rendezvous and establish their relative pose.

The latter problem, which can be referred to as EAC (Exploration and Coordination) problem, deals with the decision making about *where to go next* and *who goes where* given the partial map of the environment and the pose estimate of each agent. EAC is, then, related to two different issues regarding the exploration of an environment: *exploration strategies* which decide where to go [2] and *coordination mechanisms* which basically answer the question: who goes where? [26].

The goal of an exploration strategy is, in other words, to find the best locations on the partially known environment that robots should reach to go on with their task, assuming the solution of the SLAM problem. Coordination mechanisms, instead, take the candidate locations identified in the environment as input and decide how to assign them to the team of robots in such a way that the interference between the agents is managed and the overall exploration time is minimized.

Coordination is well known to bring several benefits in relation to the exploration time [13, 19, 57, 58] because, by coordinating, robots can simultaneously explore different parts of the environment. Moreover, agents can manage the interference between them, avoiding crashes and improving the general stability of the system. In particular, in [20], the relative effect of exploration and

coordination mechanisms on the exploration is studied. Exploration strategies appeared dominant with respect to coordination mechanisms on well structured environments while the situation is reversed on less structured ones.

Coordination mechanisms can be classified in two main categories: online and offline mechanisms. Online mechanisms implement coordination algorithms based on the current state of the exploration to decide how to assign each agent to its goal. This means that, at each step of the exploration, the current action of every team member is considered to assign an agent to its next location. An example of an online coordination mechanism is given in [58]. In this case, whenever a target point is assigned to a specific robot, the utility of the target is reduced: the allocation of robots to frontiers depends on the dynamic changes in the location utilities which are related to the current action of every team member. Thanks to this approach, robots are encouraged to spread around the space, parallelizing the exploration of the environment as much as possible.

Differently, offline mechanisms are not strictly related to the current actions of all the agents. Usually, roles are statically assigned to team members before the beginning of the exploration and the coordination deployed during the task depends directly on those roles. In [29] three offline coordination mechanisms are proposed. As an example, the *reserve* mechanism defines two roles for the agents: active agents, which immediately start to move, and reserve ones which wait to be called by active agents when needed. Once all the agents have an active role, they behave in a non-coordinated way, individually selecting goal locations on the basis of their individual payoffs. This type of coordination is static because it is defined before the effective exploration starts and does not consider the current decisions made by the agents.

The taxonomy introduced is briefly represented in Figure 1.1. This thesis is focused on the analysis of the coordination mechanisms adopted by the team. The aim of the work is to compare different mechanisms from the two families of coordination mechanisms just introduced to assess their relative effectiveness and efficiency in the multi-robot exploration context. Particular attention is payed to how the structure of the environment can affect the performance of online and offline coordination mechanisms and to how these influences could be identified a priori.

To compare various and heterogeneous mechanisms, a formal framework is proposed. It is, indeed, hard to find in the literature a unified framework to describe how the coordination mechanisms are involved in the exploration process, because most of the studies are focused on analyzing the performance of particular mechanisms on well defined domains and environments. As a result, it is often difficult to compare and analyze a priori the goodness of a mechanism with respect to others belonging to different settings and deployed in different environments. Generalizing the behaviour of a mechanism is often complicated, if not impossible, and this makes it really hard to predict the performance of a procedure on new scenarios. The framework proposed has the goal of enabling a comparison a priori between the most commonly adopted approaches. A priori here means without the need of performing simulated or real world experiments. The framework is inspired by the taxonomy, called MRTA (Multi-Robot Task

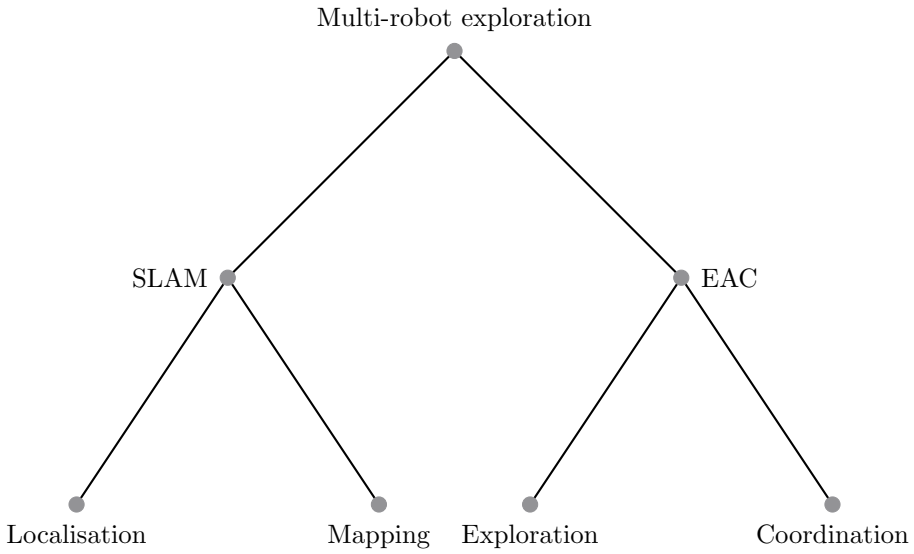


Figure 1.1: Taxonomy of multi-robot exploration problems.

Allocation), provided in [26]. In the formalization given here the tasks are discrete (e.g. explore point x, y on the map) and do not require the usage of multiple robots to fulfill them (each task is a Single-Robot task). On the other side, each robot is capable of addressing a single task at a time because it cannot be assigned to two goals simultaneously (Single-Task robot). The allocations of robots to their goals is instantaneous, meaning that no planning for future allocations is made because of the impossibility to predict how the exploration could evolve beyond the known environment.

In the framework proposed here, a coordination mechanism is seen as the algorithmic procedure that, taking the candidate locations as input, provides the allocations of agents to locations as output. Together with the exploration strategy used to compute the candidate locations on the map, a coordination mechanism solves an exploration problem.

Once a general framework for representing the coordination mechanisms has been defined, formal evaluation criteria is proposed. Since it is true that the goal of the exploration is to reduce the time spent by the team to complete it, to compare mechanisms depending on their exploration time is a good approach if data from the environment have already been collected. It is, indeed, really complex to state that a generic aspect of the exploration would reduce or not the overall exploration time and it is even more difficult to determine that a mechanism could behave better than another one without running a large amount of experiments in the first place. Moreover, a mechanism which minimizes the overall exploration time cannot be statically defined. The environment, indeed, is discovered step-by-step and almost no reasoning can be made on future allo-

cations. Therefore, all the coordination mechanisms considered here act greedily to allocate the agents, making more difficult to precisely evaluate their performance without testing them. For these reasons some evaluation criteria are proposed as a mean to compare mechanisms a priori, analyzing their effects without necessary having to test them. The criteria are related to a formal definition of the interference and availability measures, informally introduced in [29] and here described as the average distance among the agents and the average distance of an agent to a target location, respectively.

The formulated framework describing the coordination mechanisms, their interference, and their availability helps in designing different types of mechanisms and in assessing their possible performance without running experiments. However, to evaluate the effectiveness of the framework in the description of the coordination and verify the true performance of the mechanisms proposed, we performed some experiments on several 2D simulated environments, thanks to MRESim. MRESim is a simulator for multi-robot exploration of two-dimensional environments that allowed us to test the mechanisms over spaces provided as PNG images. The experiments runs share the same structure:

- A problem is defined to be solved by a mechanism, specifying the environment and team configuration to be deployed.
- A coordination mechanism is proposed to solve the problem.
- Hypotheses are made on the possible behaviour of the mechanism with respect to others, thanks to the framework introduced.
- Results are shown to validate the hypothesis made and to asses the effectiveness of the framework in evaluating the possible performance.

In particular, eight mechanisms implementing online and offline approaches are tested on six environments with ten different team configurations. This means that each mechanism is deployed on several problem instances, one for each environment-team configuration pair. The experiments resulted useful not only to test and understand the merits and drawbacks of online and offline coordination mechanisms for exploration but also to validate the framework proposed. The framework resulted a good, yet not complete, way of describing the exploration and allowed to derive rough estimates of the possible behaviour of the mechanisms, basing on the a priori analysis of their design choices in terms of interference and availability. In particular, offline mechanisms showed to be effective in the early stages of the task because the definition of roles before the beginning of the exploration helps in immediately spread the robots around the environment. Online mechanisms, instead, present a higher interference in the first stages of the exploration because they, knowing a little part of the environment, cannot exploit the online information before performing some allocations. However, offline mechanisms usually loose their effectiveness as the exploration goes on with respect to online ones, because they leave the robots free to explore independently in a non-coordinated way. On the other hand, online mechanisms do not stop coordinating the agents during the task and, in this sense, perform a

constant coordination. The results suggest that an optimal, greedy, mechanism could be obtained by combining online and offline settings.

This Thesis is structured as follows. In the second chapter, after this introduction, the state-of-the-art related to different aspects of the exploration is reported. The aim of this section is to provide a global, yet not complete, view of the contributions made in the field to underline what are the basis moving this study. The topics treated here range from the SLAM problem, to the exploration strategies, to the coordination mechanisms proposed in the literature. Particular attention is payed to the studies related to multi-robot cooperative behaviour, from the benefits brought, to the differences between online and offline approaches, to the work made on the formal modelling of the coordination mechanisms.

In the third chapter, the design choices made in this thesis are presented in terms of how the environments are represented, of which exploration strategy is adopted, and of how a team is configured in the framework and in the simulated experiments. Since this work is focused on coordination, no different exploration strategies are tested and the candidate locations are always identified according to a frontier-based strategy.

Once the setting is defined, the fourth chapter presents the formal coordination framework which is here developed to describe the coordination process. The framework is organized around three elements: the exploration state, the coordination mechanism, and the evaluation criteria, already introduced when speaking of interference and availability.

In the fifth chapter, we present the coordination algorithms deployed in the experiments. The aim of this chapter is to show how the framework proposed can be effectively translated into several coordination mechanisms.

The experiment chapter (the sixth one) illustrates the experimental setting, in terms of environments and simulation details. Moreover, for each mechanism, this chapter presents and comments the results obtained on the different environments tested, underling the good aspects and the drawbacks in relation to the types of environments that are explored.

As a conclusion, in the seventh chapter, a final comparison between online and offline mechanisms is given. Furthermore, the goodness of the framework is evaluated in describing the coordination mechanisms involved in the exploration. This chapters ends with some possible future work.

Chapter 2

State of the art

Three of the main issues that the robots face to autonomously explore an unknown environment are: localize their pose on a partial map, identify where to go next, and, in the case of multiple robots exploration, decide who goes where.

The self-localization of a mobile robot is widely recognized as one of the most important problems of autonomous navigation [14, 23, 54]. While such a task can be performed pretty well when the environment is a priori known, robot localization becomes much harder when a map of the environment is not available in the first place [55]. Considering an initially unknown environment, each robot is asked to incrementally build its own map (by acquiring step-by-step sensor observations) and to localise itself on the resulting partial representation. Moreover, in a multi-robot scenario, the local maps built by the single agents have to be merged in a common representation for the entire team. This additional complexity implies that robots should be able to localize themselves on the common map even if the various parts of the environment are not directly visited by them. The problem of simultaneous localization and mapping is generally referred to as SLAM.

Choosing where to go next corresponds to identifying candidate locations on the partial map of the environment, which is either directly provided by the SLAM solution or obtained as a higher-level representation from the SLAM output. This is a problem faced both in single and multi-robot exploration [5,15]. The algorithm, which computes the goal locations on the map, answering the question where to go next, is here referred as *exploration strategy* [2].

Deciding who goes where, instead, is only related to multi-robot scenarios in which, after identifying the candidate locations on the environment, each robot has to decide which location to choose. The algorithm that allocates robots to target locations is named *coordination mechanism* and has the aim of minimizing the exploration time while managing the interference among the team members [26].

The first section of this chapter is mainly focused on the basic setting of the SLAM regarding single robots and static maps to provide a guideline to understand the problem, intended as a pre-requisite for robots to explore and

coordinate. The rest of the chapter, instead, presents different state-of-the-art paradigms for both exploration strategies and coordination mechanisms along with a concise description of the Multi-Robot Task Allocation framework, which is an important theoretical tool used as a base for the model proposed in this work. The coordination mechanisms introduced in the following sections are used as a comparison benchmark for the new approach proposed in this thesis.

2.1 Simultaneous Localization and Mapping

The SLAM problem is mainly addressed in a probabilistic framework [7, 41, 47] in which an agent's pose at time t is defined as a vector composed of two translational and one rotational components (for robots forced to the ground):

$$l_t = [x_t, y_t, \phi_t^T] \tag{2.1}$$

and of three translational and three rotational components for robots moving in a 3D space:

$$l_t = [x_t, y_t, z_t, \phi_{t,p}^T, \phi_{t,r}^T, \phi_{t,k}^T] \tag{2.2}$$

Therefore, the path L_T followed by a robot is identified by a sequence of poses $[l_0, ..., l_T]$ where T is a generic exploration end time. A path is drawn on a map m which is here intended as the *rea* environment. Calling U_T the set of odometry measurements of the agent and Z_T the set of sensors' measurements collected up to T, the SLAM problem can be formulated as: recover a model of m and the sequence L_T given odometry and sense measurements (U_T and Z_T, respectively).

SLAM can be proposed in two formulations: an offline formulation in which all measurement data are processed in batch and the entire sequence of poses is computed at once and an online formulation which updates the poses of the robot dynamically as the exploration goes on. These two approaches are formally translated in computing the posterior over the entire robot path (offline):

$$p(L_T, m|U_T, Z_T) \tag{2.3}$$

and in computing the current robot location, instead of the entire path (online):

$$p(l_t, m|U_t, Z_t) \tag{2.4}$$

2.1.1 SLAM solution

Different paradigms exist to solve the SLAM problem in its various forms: from the online, to offline formulations, to single and multi-robot settings. A solution of the SLAM problem is here intended as a procedure capable of estimating the pose of each agent while simultaneously building a representation of the environment, consisting of a map.

Graphical methods are increasing their popularity as a solution of the offline

SLAM problem [16, 21, 39]. These methods are based on the construction of a graph to represent the environment.

Usually, the observations and the robot's poses are represented as nodes of the graph while the links between them are of two types: one connecting two poses and representing the odometry measurements and the other connecting the poses and the observations and representing the sensors measurements.

An example graph built following these principles is summarized in Figure 2.1. Here the observation of the environment m_1 is taken from both locations l_1 and l_2 while observation m_2 is only visible from l_2.

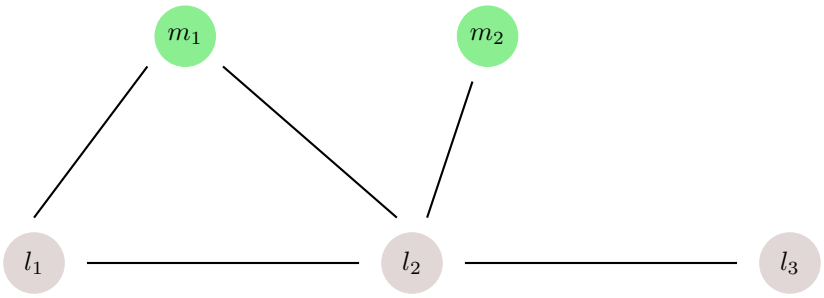


Figure 2.1: Graphical SLAM representation.

In [55], the graph is thought as a spring-mass model [27] and computing the SLAM solution is considered equivalent to computing the state of minimal energy of the model. The authors note that the graph corresponds to the log-posterior of the SLAM problem:

$$\log p(L_T, m|U_T, Z_T) \tag{2.5}$$

In [55] this logarithm is shown to be of the form:

$$\begin{gathered}\log p(L_T, m|U_T, Z_T) = \\ K + \sum_t \log p(l_t|l_{t-1}, u_t) + \sum_t \log p(z_t|l_t, m)\end{gathered} \tag{2.6}$$

Where $\sum_t \log p(l_t|l_{t-1}, u_t)$ is related to the arcs representing odometry measurements while $\sum_t \log p(z_t|l_t, m)$ is related to the ones representing sensor measurements.

The solution of the offline SLAM problem is, therefore, derived as the mode of the following equation:

$$L_T^*, m^* = \underset{L_T, m}{\operatorname{argmax}} \log p(L_T, m|U_T, Z_T) \tag{2.7}$$

Considering, instead, the online SLAM formulation, one of the most common methods to solve it is based on particle filters [35, 40]. A particle is intended as a guess about the true state of the environment and a particle filter selects

representative samples among the set of particles. These approaches make use of the particles filters to deal with the huge set of possible guesses obtained during exploration.

The FastSLAM algorithm proposed in [40] is often preferred over the Extended Kalman Filter approach which is limited by its computational complexity [8,25]. At any point of the exploration the FastSLAM maintains K particles of the form:

$$L_t^k, (\mu_{t,1}^k, ..., \mu_{t,N}^k), (\Sigma_{t,1}^k, ..., \Sigma_{t,N}^k) \tag{2.8}$$

Where L_t^k is the guess with index k about the path of an agent at time t while μ and Σ represent the means and variances of N two-dimensional Gaussians, one for each landmark of the map.

The algorithm initializes L_t to the known starting pose of the robot and starts with no Gaussians representing the landmarks, meaning that the map is still completely not available. After the initialization, two things can happen: either an odometry or a sensor measure is received.

- *Odometry* measure u_t: The algorithm updates the pose estimate for all the particles:
$$l_t^k \sim p(l_t | l_{t-1}^k, u_t) \tag{2.9}$$
- *Sensor* measure z_t: The algorithm firstly computes for each particle the probability of sensing z_t, being n the sensed landmark:
$$w_t^k := \mathcal{N}(z_t | l_t^k, \mu_{t,n}^k, \Sigma_{t,n}^k) \tag{2.10}$$
The factor w_t^k is called importance weight and represents how important is the particle k after the new measure came in. The next step of the algorithm is called *resampling* and is based on the importance weights of the particles. The particles are, indeed, re-drawn and the probability of a particle to be inserted in the new sample is its normalized importance weight.

When dealing with multiple robots additional challenges emerge. The team must, indeed, be able to merge the local maps into a global representation of the environment and robots should localise themselves on such a joint map. A solution to the multi-robot SLAM is provided in [11], in which the authors rely on the single robot SLAM, generalizing it to the multi-robot scenario. The algorithm proposed is divided in three phases. Firstly, each robot behaves independently solving its local SLAM problem. Then, when two robots meet during the exploration, their local maps are merged taking advantage of robot-to-robot relative range and bearing measurements. Once the maps are joint, the robots start over the independent navigation.The algorithm is based on the M-Space representation [34] of the linear features of the environment. Indeed, the method proposed is mostly suitable for environments that can be well represented through their linear features such as indoor environments.

2.2 Exploration strategies

An exploration strategy is an algorithmic procedure that, invoked at each step of the exploration, provides the candidate locations on the partial map that robots should reach in order to maximize their knowledge of the environment. Different approaches have been proposed to solve this problem [4, 38, 51, 53]. In many cases, an approach is characterized both by the way in which the environment is represented and by how the candidate locations are intended on such a representation.

It is important to underline that the environment representation mentioned here does not necessarily correspond to the output map provided by the SLAM but, instead, it could be a higher level representation of the environment. The representations deployed by exploration strategies, indeed, range from graph structures to occupancy gridmaps and constitute an abstract view of the environment, usually more suitable for path-planning with respect to direct maps used for localization.

The choice of the environment representation strongly affects, not only how a candidate location is computed, but also how it is defined. For instance, on a graph model, a candidate location is defined as a vertex of the graph, while on an occupancy gridmap a candidate location is one of the grid cells. Although some other paradigms exist, the strategies presented in this sections are grouped in three categories: *topological*, *information-gain* and *frontier-based* strategies, depending on their interpretation of both environment and target points. These classes constitute a good sample of methodologies which well describe the state-of-the-art in terms of exploration strategies.

2.2.1 Topological strategies

Topological strategies describe the environment through a graph. This representation lacks of metric information and is only focused on how the environment is connected. Such a representation is suitable, for example, when the distances are not directly computable due to robots limited sensing capabilities [24]. In topological strategies, vertices are the candidate locations that should be assigned to the robots and are discovered thanks to sensor measurements as the exploration proceeds.

Three main distinctions should be made on the types of graph structures built by different topological strategies in the literature:

- Undirected graphs with uniquely identifiable vertices.
- Undirected graphs with anonymous vertices.
- Directed graphs.

The first type of undirected graphs presents identifiable vertices meaning that each vertex of the graph can be recognized when revisited. This assumption is realistic when the sensing capabilitites of the agents allow them to derive

distinguishable landmarks on the map and to recognize them when they are seen again [6, 17, 43, 44]. In [42] an example of graph with identifiable vertices is given. Here the edges are said to be opaque, meaning that the link between two locations is known only when it is physically explored. Depth First Search algorithm is employed on such a graphical environment, incrementally obtained during the exploration.

An undirected graph with anonymous vertices doesn't make any assumption on the capabilities of the agents to recognize already seen portions of environments and, for this reason, the vertices of the graph are not recognizable in an easy way. In this context, markers can be dynamically placed and moved on the target locations to distinguish between explored and unexplored areas [9,24,31,33]. In [24] the need for markers is demonstrated and an example algorithm is proposed to solve the exploration on this type of graphs.

Taking into account directed graphs, the situation changes significantly, as it is not always possible to go back through edges and DFS is not applicable [1,10,59]. In [10] an algorithm is proposed to make two cooperative agents learn the directed graph. The algorithm makes the robots learn the graph and the homing sequence simultaneously by actively wandering through the graph. The work in [59], instead, shows that the exploration of a directed graph presents an upper and a lower bound in the number of edge traversals. The algorithm proposed in [1] tries to explore new edges that have not been visited so far. That is, starting at some visited node x with unvisited outgoing edges, the robot explores new edges until it gets stuck at a node y, i.e., it reaches y on an unvisited incoming edge and y has no unvisited outgoing edge. Since the robot is not allowed to traverse edges in the reverse direction, an adversary can always force the robot to visit unvisited nodes until it finally gets stuck at a visited node.

2.2.2 Information-gain strategies

Usually, information-gain strategies rely on occupancy grids to model the environment. In this case, the map is represented by a grid which cells can have one of three values $\{Clear, Obstacle, Unknown\}$. An example of occupancy grids is given in Figure 2.2. Clear cells are colored by green, obtacles are in black, and unkown cells are the white ones.

The measure of information in a probability distribution $p(x)$ is the entropy $H_p(x)$ and is defined as follows:

$$H_p(x) = -\int p(x) \log p(x) dx \tag{2.11}$$

H_p is a measure of uncertainty related to a random variable and is maximum when p is a uniform distribution in which all the outcomes are equally likely. On the other hand, H_p decreases as the random variable tends to be certain. The information gain is, therefore, defined as the decrease in the entropy between successive measurements.

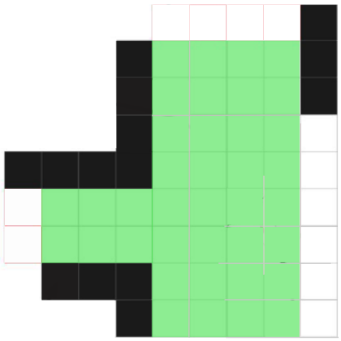


Figure 2.2: Information gain occupancy grid.

In the exploration context, let us call $p(m_i)$ the probability of the cell $i \in m$ to be known (either clear or obstacle) and $H(p(m_i))$ its uncertainty. The information-gain is, then, defined as the decrease in the map uncertainty due to a sensor observation z_t. Formally, it can be described as in [50]:

$$I(m_i|z_t) = H(p(m_i)) - H(p'(m_i|z_t)) \tag{2.12}$$

Where:

- m_i: is the cell with index i on the occupancy grid
- z_t: is the sensor measure obtained at time t
- $p(m_i)$: is the distribution for cell m_i
- $p'(m_i|z_t)$: is the updated distribution for cell m_i after z_t has been perceived.

Intuitively, when evaluating a clear cell against an unknown one, the information gain will be higher for the unknown cell because the entropy of the visited one has been updated at least once. Updating the entropy every time a new measure is obtained, the estimate of how much information the robots could gather from the available locations is provided at each step of the exploration.

Generally, the information gain measure is used to evaluate the different candidate locations and to detect the next best points that the agents should reach on the environment. This probabilistic approach offers a great versatility and has been used in different ways in the literature. In [3] the estimate of the information that could be gathered from a given location is combined with the distance of that location from the agent. This combined measure is used to establish the utility of the given location for the exploring agent. In [28], to compute the next best locations, the algorithm presented first generates a set of potential candidate locations. Next, it evaluates each candidate location according to: the expected information gain that will be obtained by sensing at

the location, the needed overlap between the two partial layout models (to ensure good alignment), and the motion cost required to move to the new position. In [52], instead, the communication success is embedded in the information-gain computation. This means that a target location is evaluated both by considering the information it could provide and by taking into account the communication possibilities robots have from that location.

As introduced, the information-gain strategy allows to evaluate the candidate locations on an occupancy grid but not to identify them. A good way to identify the locations to be evaluated is to consider the limits of the known space. This is the basic concept at the base of frontier strategies, presented in the next section.

2.2.3 Frontier strategies

Frontier-based strategies make use of occupancy grids to represent the environment but they differ from information-gain approaches in the definition of candidate locations. Frontiers strategies are not based on a probabilistic setting (even if probabilistic extensions are provided in the literature [22]) but, instead, rely on geometrical considerations about the partially known environment.

The basic idea behind the approach is to identify the boundary regions between the already visited and the unexplored space. These boundaries are called *frontiers*. They were first proposed in [60] for the single robot scenario and in [61] for the multi-robot exploration. Frontier strategies have later been revisited in many other studies [37], [46], [36].

Being the cells of one of three types $\{Clear, Obstacle, Unknown\}$, a frontier is a cluster of clear cells adjacent to unknown ones in the occupancy grid. An example is given in Figure 2.3 where frontiers are depicted in red. The set of

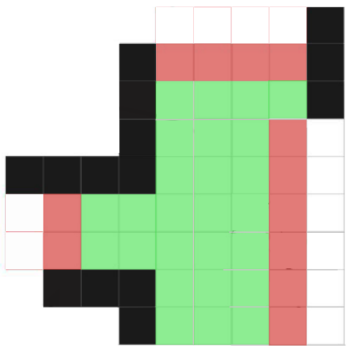


Figure 2.3: Frontier based occupancy grid.

frontiers represents the set of candidate locations that agents should reach at each step of the exploration.

Plain frontier-based strategies do not make any assumption on the information an agent could get by moving to a target location, because no probabilistic

reasoning is made on what is expected after visiting the frontier. Frontier strategies are often combined with information-gain strategies to, not only detect the frontiers discovered, but also evaluate them.

Frontier strategies are probably the most common strategy when dealing with exploration because of its good performance in practice and its relative ease in the implementation. In this thesis we focus on coordination rather than on exploration strategies, and we assume the exploration to be based on frontiers both in theoretical and experimental sections.

2.3 Coordination mechanisms

A coordination mechanism is the algorithm that, in a multi-robot scenario, allocates the team members to some target locations.

In the literature, coordination mechanisms are mainly addressed from a practical point of view [12, 29, 32, 49, 58]. Different algorithms are proposed along with several settings and environments in which the algorithms are tested.

Two main categories of coordination mechanisms have been identified in the past: online and offline mechanisms. Online mechanisms allocate robots by constantly considering the different actions taken by all team members [49, 57, 58]. Offline mechanisms, instead, tend to assign roles to robots before the exploration starts and agents take actions according to those predefined roles [12, 18, 29, 48].

Even if online mechanisms provide a stronger coordination than offline ones, they require to communicate more often with respect to offline mechanisms. Following an online mechanism, indeed, an agent has to communicate its pose and target location at each step of the exploration and base its decisions on the actions of its team members. Offline mechanisms, instead, leave robots more free during the exploration and agents can also behave as independent units in certain moments, reducing the communication need.

Coordination is well known to bring several benefits in relation to the exploration time [19] [13] because, by coordinating, robots can simultaneously explore different parts of the environment in parallel. Furthermore, agents can manage the interference between them, avoiding crashes and improving the general usability of the system. In particular, in [20], the relative impact of exploration strategies and coordination mechanisms on the team performances is studied. Exploration strategies appear dominant with respect to coordination mechanisms on well structured environments while the situation being reversed on less structured ones.

This chapter presents some state-of-the-art algorithms deployed for coordination to which the algorithms originally proposed in this work are either compared or inspired. Moreover, the chapter covers a little part of the theoretical studies made in the field of coordination, which serve as a basis for the formal model proposed in the following to describe the coordination mechanisms implemented.

2.3.1 Online coordination

Online coordination mechanisms allocate robots to locations, by dinamically taking into account the decisions all team members make. This means that, at each step of the exploration, an agent is assigned to a target by leveraging the communications coming from its teammates.

An example of online coordination mechanisms is given in [58] and [57]. In this cases, whenever a target location is assigned to a specific robot, the utility of the target is reduced: the allocation of robots to locations is related both to the utility and to the cost of reaching the location. This means that the coordination depends on the dynamic changes in the location utilities which are related to the current actions of every team member. This strategy will be referred to as *utility based* and relies on frontiers. Another online example is given in [49] and is based on an information-gain approach for candidate locations identification. The algorithm proposed tries to maximize the overall utility by minimizing the potential overlap in information gain among the various robots.

The *utility based* mechanism will be compared, as a sample of online approaches, with a modified version we introduce in later chapters, following the same online principles and named *direct optimization*. The mechanism proposed in this work differs from the one in [58] in the management of the initial steps of the exploration.

2.3.2 Offline coordination

Offline coordination mechanisms assign a robot to a goal point thanks to the definition of roles. The roles are established before the exploration starts and can be modified during it. After a good amount of steps the agents could also leave those roles to behave independently as single units. The basic idea is to reduce the need of communication by coordinating the robots through the roles defined, without the need for an online mechanism to constantly monitor the team activities.

Three offline mechanisms are proposed in [29], that is the main study on which we base the analysis of the results of the mechanisms proposed in the following. The *reserve* mechanism holds a partition of the team in idle and active robots. Idle robots are left waiting at their starting poses while active agents start the exploration. When an active agent encounters a branch point and needs to decide which direction to take, it calls a reserve agent to split the tasks. Once all agents are in active state, they behave in a non-coordinated way as single entities.

Similarly to the *reserve* strategy, the *buddy system* mechanism divides the team in pairs and maintains a partition in idle and active pairs. Active pairs start the exploration and, when they discover a branching point, they split up in single agents. When a single agent finds another branch, it calls a reserve pair to help. Once all agents are single and active, the team behaves as a set of independent agents.

The last mechanism implemented in [29] is named *divide and conquer*. The

team members start the exploration as a group with a leader agent. The leader selects its goal and all other teammates follow it, navigating to the same location. When a branching point is detected, the team splits in two groups made by half of the agents each. A new leader is named for the second group and the exploration starts over with the same structure. When all the robots become leaders and no groups are anymore present, the agents continue as single-robot explorers.

All these mechanisms explore through frontier-based strategies and will be compared later on with three variants proposed in this study. Two algorithms called *proactive reserve* and *proactive buddy system* will be tested as evolutions of *reserve* and *ruddy System*, in which idle robots do not wait at their starting positions but proactively move to a convenient waiting location computed as the barycenter of the polygon made by active agents.

Another mechanism, named *side follower*, is based on the same grouping idea as *divide and conquer* but, instead of forming a whole set which eventually splits during the exploration, it builds small subsets of three agents which never divide. The leader of the group is ecouraged to follow a straight path while the two followers are pushed to reach side frontiers: one of them is assigned to frontiers at the left side of the leader while the other one to the right side goals.

2.3.3 Multi-Robot Task Allocation taxonomy

Before formulating and testing the mechanisms, this thesis builds a formal framework for the comparison of the coordination mechanisms. The goal of the formalization is to precisely define both how a mechanism works and how two mechanisms can be compared.
This section discusses the taxonomy presented in [26] which is the base on which the formulation provided here is built.

Different models have been proposed to represent specific problems and situations dealing with multiple-robots coordination. The frameworks presented in the literature range from game theory approaches [30, 56] to the architecture called ALLIANCE introduced in [45] which defines a model of coordination based on motivational behaviour. However, these models are too specific for the settings adopted by the various studies, while the work proposed in [26] lend itself to be adapted to different scenarios.

The work in [26] is focused on Multi-Robot Task Allocation problems. In particular, a domain-independent taxonomy of MRTA problems is given. A task is defined as a subgoal which is needed to accomplish the overall goal of the system. A task can be discrete (e.g., deliver this package to room 101) or continuous (e.g., monitor the building entrance for intruders) and can vary in many other ways. The taxonomy is based on distinguishing among different types of tasks and robots:

- **Single-task robots (ST) vs. multi-task robot (MT)**: a single-task robot is a robot that can be assigned only to one task at a time, as opposite to multi-task robots that can handle different tasks at once.

- **Single-robot tasks (SR) vs. multi-robot task (MR)**: a single-robot task is a task that needs one robot to be fulfilled while a multi-robot task needs the work of more than one robot.
- **Instantaneous assignment (IA) vs. time-extended assignment (TE)**: an instantaneous assignment of a task to a robot is only related to the current allocation, no planning for future allocations is made. On the other side, in a time-extended assignment, more information is available such as the list of tasks that could be the matter of future allocations. Basing on this information, the current allocation can be made to take into account also future assignments.

This taxonomy is usefull when defining the model proposed in the following chapters. In the formulation given here, the tasks are discrete (e.g., explore a point on the map) and do not require the usage of multiple robots to fulfill them (each task is a *single-robot* task). On the other hand, each robot is capable of a single task at a time because it cannot be assigned to two goals simultaneously (*single-task* robot). Moreover, the assignment of robots to their goals is instantaneous, meaning that no planning for future allocations is made because of the impossibility to predict how the exploration could evolve besides the known environment. Accordingly to the MRTA taxonomy, the coordination problem considered in the context of unknown environment exploration can be modeled as a ST-SR-IA problem.

Chapter 3

Problem setting and definitions

The problem we address deals with multi-robot exploration of an unknown environment. It can be informally presented as: given a set of robots and an environment initially unknown to them, define an exploration strategy and a coordination mechanism to make robots map the environment in the minimum possible time.

The issues robots face during the exploration are related to simultaneously mapping and locating their poses in the environment, to build an abstract representation of the map in which is easier to perform decision making and path planning, to decide where to go and to decide who goes where to minimize the exploration time.

This work is focused on coordination mechanisms that provide an algorithmic procedure to allocate robots to the identified target locations, basically answering the question who goes where. For this reason, both the SLAM problem and the exploration strategies are studied in depth here.

The SLAM problem is considered solved by the approach proposed in [11] for multi-robot scenarios. Agents build their local maps and, when they cross paths, they establish their relative pose and merge the local maps into a global representation. The map derived is here abstracted in an occupancy grid from which relevant features of the environment can easily emerge. This representation facilitates the usage of frontier-based strategies to identify the candidate cells on the grid. Indeed, the coordination mechanisms proposed during this work rely on frontiers identification to decide where agents should go at each step of the exploration [61]. The frontiers detected are, then, assigned to robots thanks to four different coordination mechanism which are tested in several simulated experiments against the ones proposed in [29] and [58]. The coordination mechanisms follow both online and offline approaches and rely on the settings given in this section with respect to the SLAM solution and to the exploration strategy adopted.

The problem addressed in this thesis, from now on called *exploration problem EP*, can be formally defined as a 4-ple $\langle A, E, P_0, T \rangle$:

- A: set of robots used during the exploration.
- E: environment to be explored.
- P_0: initial poses of the team.
- T: termination criteria.

These four elements characterize the specific instance on which the exploration is deployed, setting the context of the experiments. The experiments are performed in MRESim, a Java-built 2D simulator for multi-robot environment exploration. An experiment in MRESim is characterized by the elements composing the EP: the environment (provided as a PNG image), the set of robots (given through a configuration file), their initial poses (given through the same team configuration file) and the termination criteria (expressed in our case as a percentage of area explored). Moreover, agents are provided with an exploration strategy and a coordination mechanism to be tested. Agents are placed at their starting poses and begin to take actions driven by the specified strategy and mechanism. In particular, MRESim allocates a thread to each agent, to simulate a distributed and parallel computation, typical of real-world scenarios. For each agent and step of the exploration, MRESim runs the exploration strategy and the coordination mechanism provided, until the termination criteria is met.

The following sections describe with more detail the choices made in terms of:

- How agents are represented.
- How the environment is represented in an occupancy gridmap.
- What are the relevant features of the environments in such a representation.
- How the frontiers are detected in the deployed exploration strategy.

Next chapters will be focused on using the setting provided here to model, implement, and test the coordination mechanisms.

3.1 Team configuration

The first aspect tackled in this chapter is how agents are represented. Besides setting the environment representation and the exploration strategy adopted, the way in which an agent $a \in A$ is characterized is important to define the context of this study.

An agent is here identified by an ID, a type, possibly a role, and a location on the map. Since the map is represented as a 2D grid, the location of an agent is of the form:

$$l_t = [x_t, y_t, \phi_t^T] \tag{3.1}$$

Where x_t and y_t are the coordinates of the grid cell occupied by the robot while ϕ_t is its rotational component.

The type of an agent is intended as the set of features of the robot: its speed, sensing capabilities, communication range, etc. The robots are here supposed to perceive the environment through a laser scan which is sufficient for the purposes of this work. The laser beam defines the sensing range of the robot. For simplicity, robots are considered forced to the ground, moving on the environment thanks to their wheels. The robots deployed in the simulated experiments are set with the default MRESim values for both sensing range and speed while communication range and battery are unlimited. The reason for the communication range to be infinite is that this thesis is not focused on how communication is performed among the agents but, instead, on what agents can do assuming that the communication is granted. The role is used by offline coordination mechanisms to select the step-by-step actions to be assigned to the agents and characterizes the agents in the different mechanisms.

The agents need a centralized coordinator, called *base station*, to hold the common map available to the agents, their poses and, possibly, their roles. The base station is configured as an agent but it is not actively involved in the exploration. It just serves as a central entity that, communicating with the agents, provides to them the information needed for the coordination. Its pose is fixed during the exploration and is given before its beginning. As for the agents, the communication range and the battery life of the base station are supposed unlimited.

3.2 Environments

In this thesis we consider indoor environments. This choice is motivated by the application contexts, which make indoor spaces more common to be explored than outdoor ones.

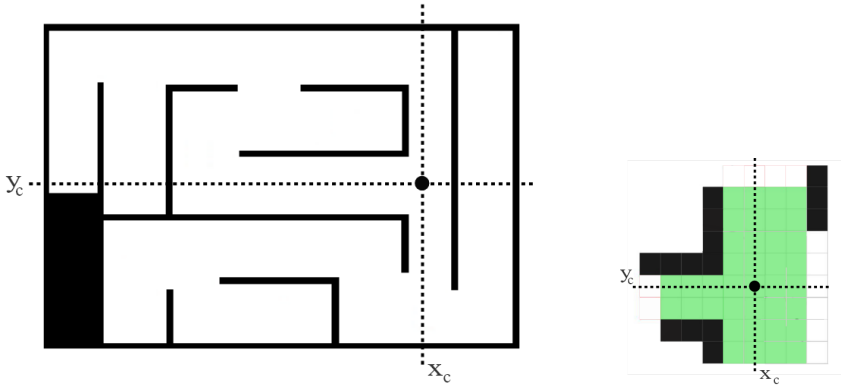
An environment, as intended from now on, consists of an abstraction of the map provided as output by the SLAM algorithm. In this work, agents hold a high level representation of their workspace in the form of a two-dimensional occupancy grid. This abstraction allows robots to make decisions with relative ease with respect to planning on the direct map and it is sufficient to represent the environment for the tasks this thesis is focused on.

Besides defining how the environment is represented during the exploration, this section provides some preliminary definitions about how the workspaces can be classified.

3.2.1 Occupancy grids

Occupancy grids are given as 2D grids which cells represent the areas of the environment. A cell is, then, a set of points characterized by its center. The center is a point on the environment, expressed thanks to its Cartesian coordinates. An example gridmap is shown in Figure 3.1a. The coordinates (x_c, y_c)

identify the center of the currently underlined cell in the figure.



(a) Coarse-grained occupancy grid (b) Fine-grained occupancy grid.

Figure 3.1: Occupancy grid shown with two different granularities.

A cell can have one of three values $\{Clear, Obstacle, Unknown\}$ depending on its occupancy probability. A *clear* cell is an area that was already visited by the team and does not contain any obstacle. An *obstacle* cell is recognized thanks to the sensor measurements of a robot, while *unknown* cells are the ones that were not reached yet by the team and remain still uncovered.

In Figure 3.1b a fine-grained view of the grid is given in which squares represent cells and are identified by their coordinates. Green squares are clear cells, white ones represent unknown cells, and black squares are the obstacles detected during the exploration. The grid is initialized to have all its cells set to unknown. Sensor data is added to the grid as the exploration goes on by setting the cells in which each laser beam ends as obstacles while giving a clear value to the cells along the straight line between the obstacles and the robot position. An instance of *EP*, made by the agents, the gridmap representing the environment, and the agents' initial poses is depicted in Figure 4.4.
Depending on the structure of the workspace, which is related to obstacle and clear cells positions, the environments can be classified according to different principles.

3.2.2 Environments classification

In this thesis we consider indoor environments, for which different parameters can be proposed to identify the features of a workspace. The main elements considered during this work to classify the environments are:

- **Small (S) vs. large-size (L)** environments: the area of the environment is the first element which affects the exploration. Smaller environments, obviously, will require a little effort with respect to larger ones either in terms of agents deployed or time taken during the task.

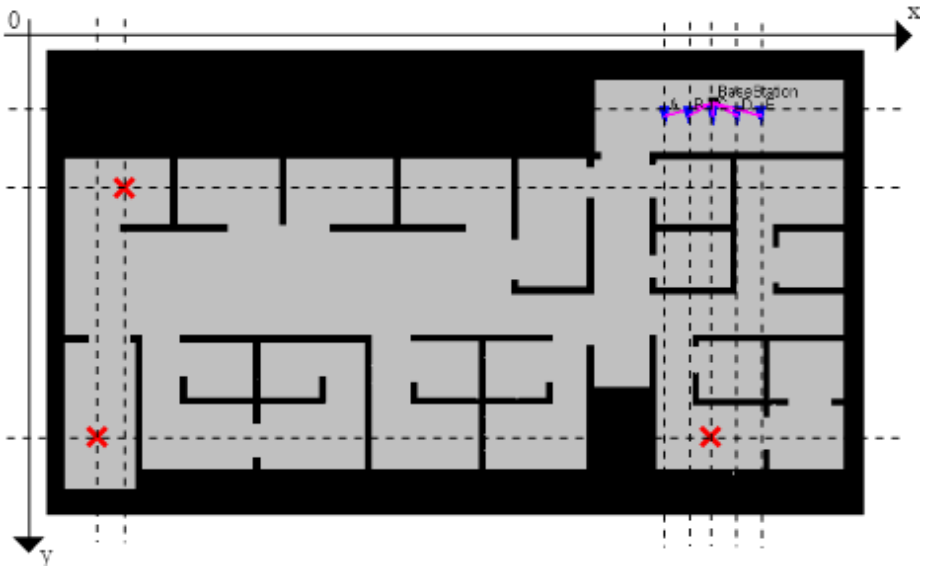
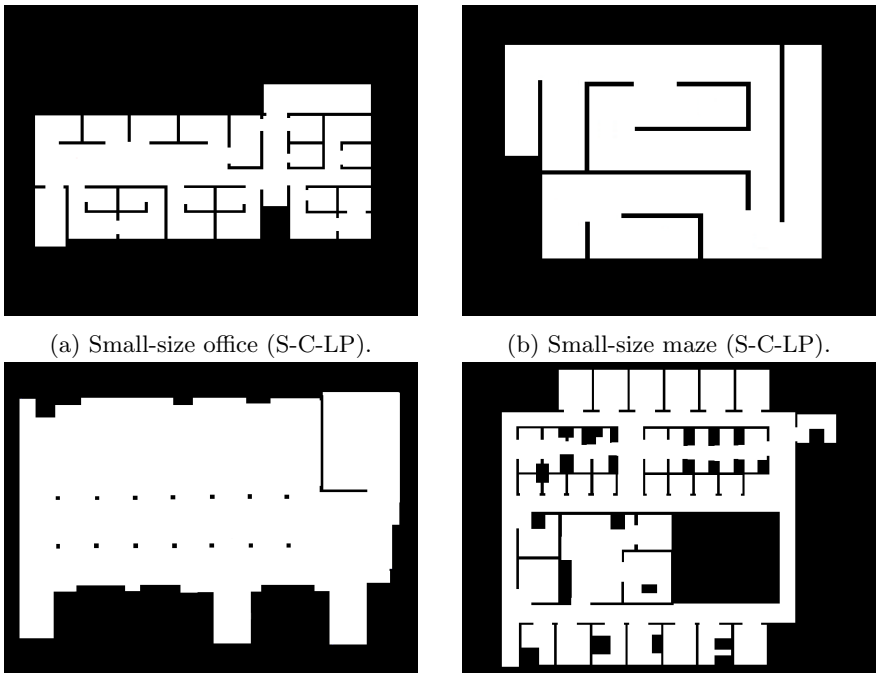


Figure 3.2: Instance of EP representing an exploration problem on the RoomsEnv in MRESim.

- **Open (O) vs. cluttered (C)** environments: the type of structure strongly affects the performances of the exploration. Open spaces present a less structured map that allows an obstacle-free exploration. On the other hand, cluttered workspaces are populated by many obstacles or rooms which provide landmarks on the map to drive the paths of the agents.
- **Highly (HP) vs. lightly (LP) parallelizable** environments: related to the area and structure of the environments, the parallelizability degree is probably the most import element when dealing with coordination mechanisms. A highly parallelizable space has target locations dispersed on the environment which results in a high branching factor. Lightly parallelizable spaces, instead, provide a few close candidate locations visible at once and this forces the robots to follow similar paths.

In predicting and analyzing the results obtained by a certain approach in a given workspace, it is crucial to distinguish between the different classes of environments: strategies could, for instance, be suitable for a L-C-HP space while behaving bad on a L-O-HP. In Figure 3.3 four example spaces are provided. The small-size office in Figure 3.3a can be considered a S-C-LP environment even if the degree of parallelizability is even lower in Figure 3.3b. The space in Figure 3.3b is a simple maze with a low branching factor. Robots will go on discovering the frontiers in front of them and there are little possibilities to split and parallelilze the exploration.



(a) Small-size office (S-C-LP).

(b) Small-size maze (S-C-LP).

(c) Open space (L-O-HP).

(d) Large-size office (L-C-HP).

Figure 3.3: Different types of environment.

Environments like the one depicted in Figure 3.3c, instead, are open and do not provide many landmarks to drive the exploration. The target locations available to the agents will tend to be sparse and this will allow a high parallelization of the exploration. Generally, L-O environemnts are also HP, while small and cluttered ones are probably LP because the movements of the robots result particularly constrained.

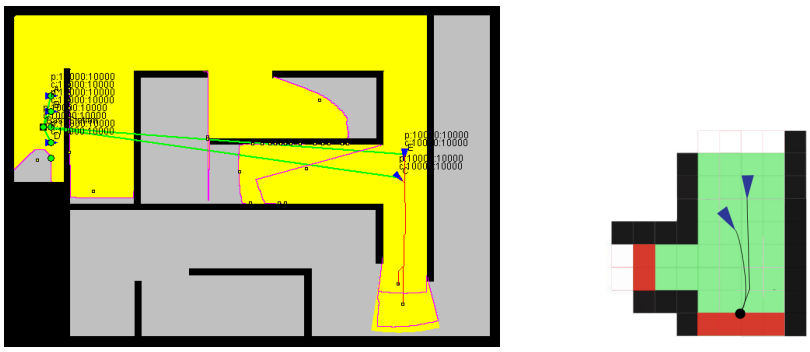
Finally, the environment proposed in Figure 3.3d is an instance of an L-C-HP space. Even if its structure is cluttered, represenitng an office, the size of the map allows robots to spread enough on it and to parallelize their tasks.

In other words, the different types of environments affect the way in which target locations are detected as the exploration goes on. In an open environment, unknown locations will be discovered with a concentric form centered in the initial poses because the boundary region between known and unknown space tends to expand in all directions, not being constrained by obstacles. In a cluttered environment, instead, new points will be non-uniform on the space because of the presence of obstacles, and possibly rooms, that forces the agents to expand the boundary region in defined directions.
More details on how the features of the environments affect the performances of coordination mechanisms will be given when dealing with the experimental phase.

3.3 Frontier-based strategies

To identify the candidate locations on the map, the mechanisms implemented rely on frontier-based strategies, first introduced in [61].

The key idea behind frontier strategies is to gain as much information as possible, moving robots to the boundary regions between the clear and the unknown space. These regions are called frontiers and are defined on the occupancy grid as those cells which are marked as clear and are adjacent to cells set to unknown. To be considered a frontier, the adjacent cells must have a



(a) Coarse-grained occupancy grid with frontiers in purple.

(b) Fine-grained occupancy grid with frontiers in red.

Figure 3.4: Occupancy grid with frontiers depicted shown in two different granularities.

minimum dimension (roughly the size of robots). In the same way, a vary large set of adjacent clear cells that could belong to a single frontier is split in two separate regions if its dimension is larger than a certain limit. Navigation goals, as represented in Figure 3.4, are here defined as the centers of known frontiers and constitute the output of the exploration strategy.

Chapter 4

Coordination framework

This thesis is focused on coordination mechanisms, intended as those algorithms that, taking the candidate locations on the map as input, allocate the team members to them in such a way that the exploration time is minimized and the possible interference among the robots is managed. The coordination problem can be seen, according to the Multi-Robot Task Allocation taxonomy in [26], as a SR-ST-IA problem. A task is here defined as the reach of a candidate location on a frontier on the map which is allocated to one of the team members.

The aim of this chapter is to provide a model for the general concept of coordination mechanism in the exploration context. The need for such a model emerged from the absence of a framework to formally compare a variety of strategies and approaches to the problem. Analyzing the goodness of a strategy with respect to another is often a hard task and a theoretical model in the background would definitely help in approaching comparisons.

To represent a coordination mechanism, the formal model proposed here is organized in two basic definitions: the *states* through which the exploration evolves and the *mechanism* itself, focused on the coordination algorithm driving the agents. The concept of state is needed to collect the information on which a coordination mechanism is deployed and to express the way in which the mechanism represents the environment.

4.1 Exploration state

An exploration state collects the relevant knowledge robots have about their workspace at each step of the exploration. It is a high level representation of the environment derived from the currently known occupancy grid. A minimum state definition should, at least, hold information on the agents' poses and on the portion of space already visited. Thanks to this information, the known environment can be distinguished from the unknown part and candidate locations on the map can be computed and allocated to agents on the basis of their poses. Many mechanisms can be developed relying only on these elements. However,

online mechanisms taking into account the actions in which all the team members are currently involved, would need also to know where each agent is going. For this reason, the state can be enriched by recording the current allocation of robots to their targets on the map.

From the above, an *exploration state* (ES) can be formally defined as a triple $\langle K_t, P_t, G_t \rangle$:

- K_t: environment already explored at time t.
- P_t: agents' poses at time t.
- G_t: agents' goals at time t.

K_t is the set of clear and obstacle cells discovered at time t which constitute the known occupancy grid. The size of K_t obviously increases as the exploration goes on and new parts of environment are uncovered. P_t, instead, is the vector containing the poses of the agents at time t, one entry for each agent ($P_t \in \mathbb{R}^n$). A pose is defined in Section 3.1 as the vector containing the Cartesian coordinates of the agents and its rotational component:

$$l_t = [x_t, y_t, \phi_t^T] \tag{4.1}$$

Similarly, $G_t \in \mathbb{R}^n$ is the set of candidate locations currently assigned to each robot and represents the task allocated to the robot at time t. Considering a frontier-based strategy for the exploration, the vector G_t would contain the frontiers centers that have been allocated to the various agents.

Figure 4.1 shows an example of exploration state at time t. K_t is denoted in yellow, agents are the blue triangles and their poses on the grid are known. Purple lines identify the path of the agents to their currently assigned target locations in G_t.

The state depicted in Figure 4.1 can be written as follows (the rotational componente in the agents' poses is omitted for simplicity):

- K_t: $\{(x, y) \mid (x, y)\ is\ Clear \vee (x, y)\ is\ Obstacle\}$.
- P_t:$\{(10, 200), (20, 200), (30, 200), (40, 200)\}$.
- G_t: $\{(10, 200), (20, 200), (30, 200), (40, 200)\}$.

4.1.1 State update and re-planning

As introduced, a state is strictly related to a time instant. In other words, agents refer a state for each exploration step. A step is defined from now on as the smallest, indivisible time unit. In this work, indeed, the time is considered discrete and the exploration results divided in a series of steps.

A mechanism allocates robots to goals at each time step, updating the state with the new information coming from agents observations. The state is, then, modified at every $t \in [t_0, t_1, ..., t_T]$ where t_T is the time at which the exploration can be considered concluded. However, usually a goal is not reached in a single

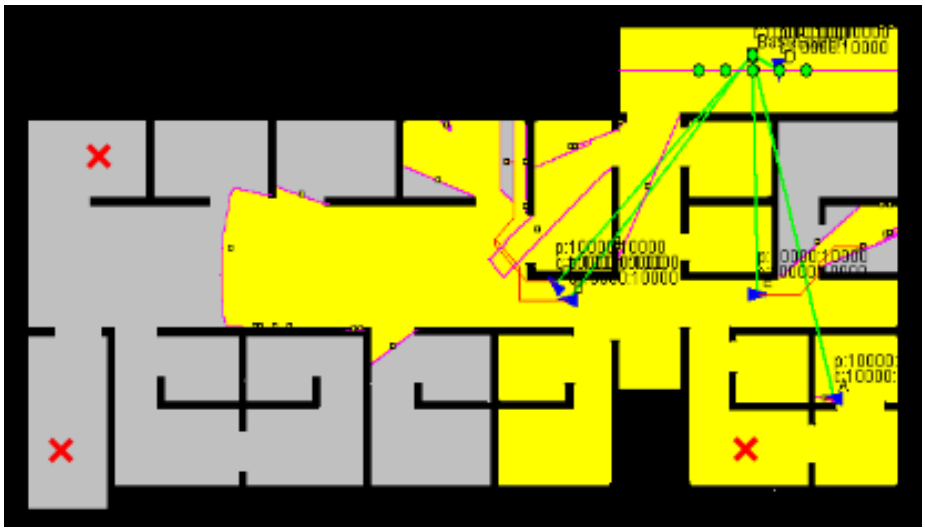


Figure 4.1: Example of exploration state during a MRESim exploration.

time step. If a robot $a \in A$ is assigned to $G_{t+1}(a)$ at time t and, when the mechanism is called to re-allocate the robot at the next step, it is still following its path towards $G_{t+1}(a)$, the goal assigned to a is not modified: $G_{t+2}(a) = G_{t+1}(a)$. Calling t_s the time at which $a \in A$ is assigned to the target $G_{t_s+1}(a)$ and t_g the time at which a reaches its goal, the interval $\Delta = t_g - t_a$ identifies the time taken by the robot to complete its assignment. In some cases, especially when the environment is big enough, Δ could be very large. If this happens, during Δ the knowledge robots have about the environment could vary significantly and better target options could be available for $a \in A$. A mechanism that does not update the target of a until it reaches its current target location could allocate the robots in an inefficient way during the path. For this reason, the state is usually updated after a time limit is passed. This concept is known as *re-planning* and forces a strategy to update the next goal of a robot after a certain amount of time, called Δ_l.

If $\Delta < \Delta_l$, then the robot reaches its goal and then receives a new assignment. If, instead, $\Delta > \Delta_l$, the robot can be forced to re-plan and get a new assignment. This avoids to ignore the new information coming in as the environment gets discovered by the team members and provides a way to correct the actions of each robot if better options become available.

4.2 Coordination mechanism

A coordination mechanism is defined as the algorithmic procedure that allocates the robots to the candidate locations on the gridmap at each step of the exploration. Along with the exploration strategy and with the other parts of the

exploration architecture, a coordination mechanism solves an instance of *EP*. The logical exploration architecture as presented so far is briefly summarized in Figure 4.2 and is needed to define the context in which the coordination mechanism is formalized.

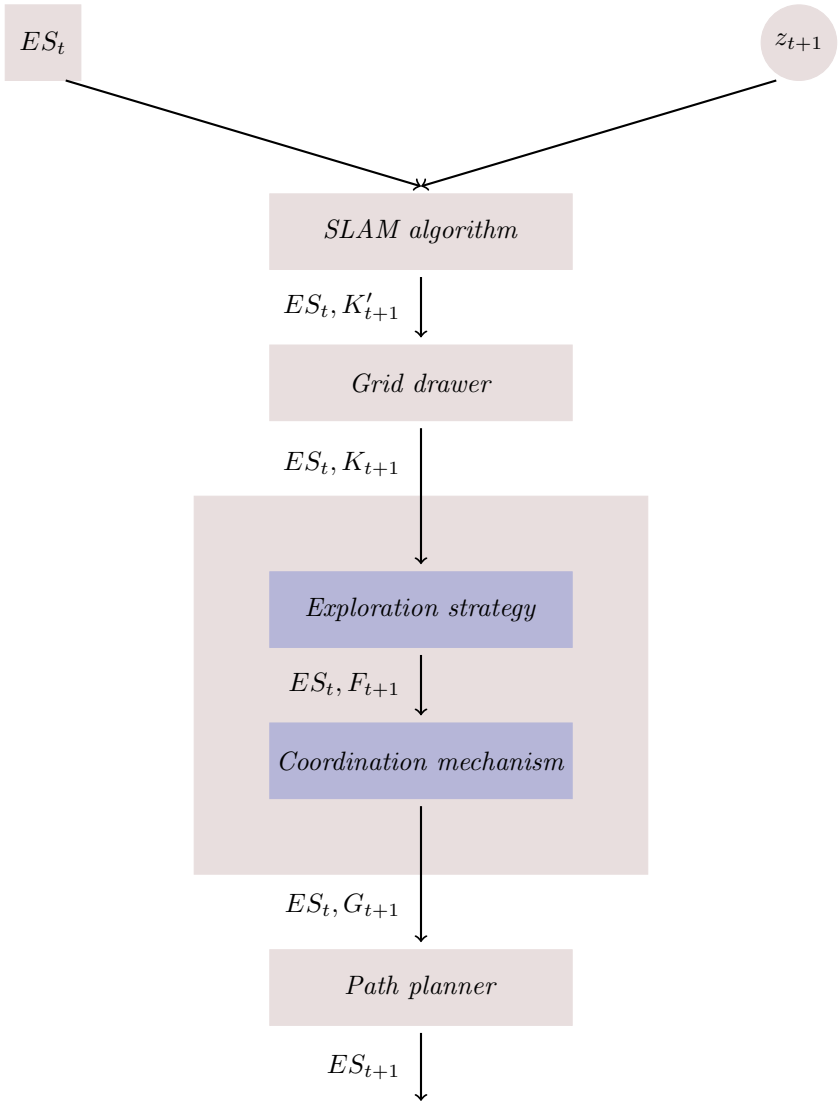


Figure 4.2: Logical exploration architecture.

$z_{(t+1),a}$ are the sensor observations directly taken by an agent $a \in A$ at time $t+1$. z_{t+1} is, then, the set of observations coming from all the agents. They are given as input to the online SLAM algorithm, along with the previous state

ES_t. The state ES_t is always available to all team members. This is possible because the mechanism holds an internal representation of the state, at each step of the exploration. The state ES_0 is initialized as follows: K_0 is empty because the environment is unknown, P_0 is given by the EP instance provided to the agents at the beginning of the exploration and G_0 is empty because no agent is allocated before the beginning of the task. The state is, then, updated acoording to what presented in Section 4.1.1.

The SLAM algorithm provides the partial map (K'_{t+1}) as output and verifies the locations of the agents on such a map. These elements are processed by a grid drawer which transforms both the raw map and the poses into an occupancy gridmap. The exploration strategy, which is here assumed to follow a frontier-based approach, takes ES_t and K_{t+1} as input to compute the frontiers that will be available at time $t+1$. Once obtained, the frontiers F_{t+1} are provided to the coordination mechanism, along with the previous state ES_t from which the current agents' poses (P_t) and goals (G_t) are extracted. The coordination algorithm allocates the team members to their new targets, giving G_{t+1} as output. The last step missing to update the state of the exploration, is to compute P_{t+1}. This element is written by the path planner module which, in turn, takes P_t from the previous state and G_{t+1} as computed by the coordination mechanism and provides a path between them. P_{t+1} is set to be the first cell belonging to the new paths the agents have to follow and its attainment will be verified by the SLAM algorithm during the next run of the exploration. If something goes wrong and the poses of the agents P_{t+1} are not verified at time $t+1$, the coordination mechanism is forced to replan to provide the agents that are in a corrupted state with a new allocation. If the robots are stuck and the poses result incorrect also after the replanning, the robots are driven by a random allocation for some subsequent steps of the exploration. Then, a new replan procedure is executed.

In this context, our aim is to provide a general formulation for the cooperation mechanism so that the various mechanisms implemented and tested later on result as an instance of the general model built here. The basic idea behind the representation proposed is to describe the allocation by means of a utility estimate. In other words, a coordination mechanism is seen as a function that assigns a utility to every agent-location pair, depending on the specific design purposes of the mechanism itself. A utility is intended as a numeric indicator of the goodness of an agent-location allocation which is related to the mechanism design choices. For instance, the reserve mechanism presented in [29] maintains a subset of idle robots at their initial locations while moving the other robots to discover the unexplored space. An active agent is always allocated to its closest frontier on the map and, when it detects a branching point, it calls a reserve agent which is turned into active state and goes to that branching frontier. In terms of utility estimate, this approach is translated in assigning a utility based on the distance to each active agent-frontier pair while increasing the utility of their starting locations for idle robots. When an active agent wants to call a reserve one to a branching frontier, it just needs to increase the utility of that frontier for the idle robots, so that the closest agent is activated.

Therefore, a coordination mechanism can be defined as the algorithm that, given the previous state ES_t and pre-computed elements such as F_{t+1}, provides G_{t+1} as output by defining a utility for every agent-location pair. A utility can, in turn, be specified as a function of the form:

$$u : (a \in A, l \in L) \rightarrow \mathbb{R} \tag{4.2}$$

Where L is the set of target locations that, in frontier methods, is normally considered equivalent to F. However, as stated for the reserve example, sometimes a mechanism needs to assign robots to other points of the map and not necessary to one of the frontiers (in the reserve case, the mechanism keeps idle robots at their starting location at the beginning of the exploration). To provide more flexibility to the model, L (a set of generic candidate locations) is used instead of F to represent the utility function also for frontier strategies.

The algorithmic procedure that, at each step, updates the utilities of the agent-location pairs, namely the coordination mechanism, is defined in Algorithm 1. Where:

- A is the set of agents globally known during the whole exploration process.
- U is the matrix of agents utilities, internally maintained by the mechanism. Each row corresponds to a robot and each column to a frontier. The cells are initialized to 0.
- *computeUtility()* is the function that, given an agent-location pair and the state of the exploration, computes the utility for that pair.
- *handleFreeLocations()* is the function that makes operations on the locations that an agent discards, possibly updating the internal utility matrix for the rest of the team.

The first part of the algorithm, composed by the first two *for* loops, takes every agent-location pair and computes its utility, producing the utility matrix U as output. In the second part, instead, the utilities are exploited to allocate each robot to the frontier with the maximum utility for it, among the available ones. The order in which the agents are allocated can depend on their roles, if they are specified in the given mechanism. Otherwise, if the roles are not assigned, the team members are allocated following a best-first approach: the agent with the highest agent-location utility is selected first and so on until every agent is assigned to the frontier with the maximum utility for it. Particular attention must be paid to the *handleFreeLocations()* function. This function handles the locations that are not allocated to the currently considered agent and, possibly, updates the utility estimates for the rest of the team. For instance, in the reserve mechanism, idle agents are called by active ones that, if discovering more than one frontier at the same time, and after selecting their best option, update the utilities related to the discarded frontiers for their idle teammates. In this way, the calling procedure is implemented thanks to the *handleFreeLocations()* function.

Data: ES_t,F_{t+1}
Result: vector G_{t+1}, containing the next allocation.
initialize the agent-frontier utility matrix $\{U := |A| \times |F_{t+1}|\}$
foreach $a \in A$ **do**
  initialize the current agent utility vector $\{U_a := |F_{t+1}|\}$
  foreach $f \in F$ **do**
    compute the current agent-frontier utility
    $\{U_a[f] := computeUtility(a, f, ES_t)\}$
  end
  update agent-frontier utility matrix with the obtained agent utility vector $\{U[a] := U_a\}$
end
foreach $a \in A$ **do**
  select the frontier with the maximum utility for the agent
  $\{goal := \text{argmax}_f\, U[a]\}$
  set that frontier as the next goal for the agent $\{G_{t+1}[a] := goal\}$
  update the utility matrix of the rest of the team
  $\{U := handleFreeLocations(a, (F_{t+1} \setminus goal), U)\}$
end
return G_{t+1}

Algorithm 1: Coordination mechanism

A coordination mechanism is, therefore, represented by the two functions *computeUtility()* and *handleFreeLocations()*. Implementing these two functions, or just one of them, a custom coordination mechanism can be instantiated. This structure is capable of generalizing various mechanisms, both in online and in offline settings. Examples will be given in the next sections, presenting the specific mechanisms implemented in this study.

A coordination mechanism completes the architecture and enables the exploration to evolve through its states. This state-update process is well represented on a graph which can be built by enriching the structure proposed in [**?**]. A graph following the given principles is built in Figure 4.3.

The nodes are of two types: state and measure nodes. The former nodes represent the state of the exploration, as defined in Figure 4.1. Measure nodes, instead, identify the sensor input agents perceive during the task. The edges are of two different types as well: a link between a sensor node (z_{t+1}) and a state node (ES_t) exists if the observation z_{t+1} was taken from the state ES_t while a link between two state nodes exists if the exploration evolves from the former to the latter state. The state-state edge is labeled by the coordination mechanism for simplicity but it stands for the entire exploration process which leads to the state update.

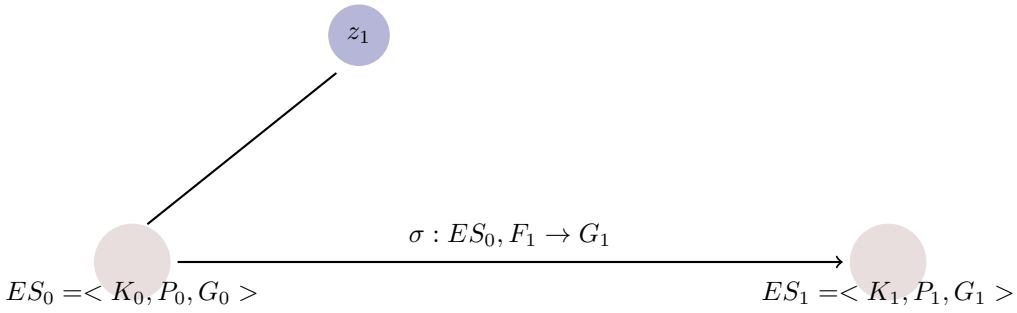


Figure 4.3: Graph representation of the state evolution driven by a mechanism.

4.2.1 Online vs. offline coordination

An online coordination mechanism is intended as an algorithm that allocates robots to frontiers by constantly taking into account the decisions made by their team members. An offline mechanism, instead, usually relies on the definition of predetermined roles, that can change during the exploration. The definition of roles allows to consider a little part of the decisions of the team members to allocate a specific agent. The online coordination is, generally, stronger because it is based on dynamic information and is constant for the entire exploration, meaning that the robots are coordinated at the same way during the entire exploration process. On the other hand, offline mechanisms are less complex and present a little communication need among the agents, compared to online approaches (in which every agent must be aware of the assignments of its teammates). The offline approach results weaker in terms of coordination because it is not able to precisely capture what is dynamically changing in the environment, basing on roles that are defined before the beginning of the exploration. Moreover, in many cases, the roles are abandoned by the team in later steps and robots are left more free to explore independently.

To clarify the difference, let's consider the online mechanism proposed in [58] and referred as *utility based* mechanism. This mechanism allocates agents by dynamically computing the utility of a frontier in relation to the number of robots assigned to it or to one of the frontiers nearby. In particular, the utility of an agent-frontier pair is obtained as combination of both the cost that the agent pays to reach the frontiers (intended as the distance from them) and the number of agents already allocated to the same (or close) frontiers. In this context, to compute the utility of a pair there is the need to know any currently available allocation.

Considering an offline mechanism, like the one called *reserve* and already presented, the usage of roles allows to reduce the knowledge agents must own to allocate frontiers. An active agent explores in a non-coordinated way and doesn't need any information coming from its teammates. An idle agent, instead, is allocated only once in a dependent-way. Indeed, only the first allocation

of a reserve robot depends on the decision made by an active one because, once turned into active state, the reserve robot starts a non-coordinated behaviour as well. In this scenario, to compute the utility of a pair there is the need of communication only during the first reserve agent allocation.

How are these approaches related to the model proposed in this section? As already emerged from the two examples, the main difference lies in the way in which the mechanisms compute the utility matrix U and, thus, in the implementation of *computeUtility()*. In particular, the key point is how the two mechanisms use the knowledge about the exploration state to implement some coordinated behaviour among the agents. While online mechanisms strongly rely on the previous allocation G_t and on the partially computed next allocation G_{t+1} to coordinate a robot with the team, offline mechanisms are related to the roles directly attributed to the agents and are based on those roles to exploit coordination possibilities. Being based on such roles, an offline mechanism should require less communication related to what other agents are doing because this computation is reduced by the role itself (e.g., a reserve agent must only be aware of its calling teammate state instead of the whole team allocation and position).

4.2.2 A practical example

To provide an example of how the exploration evolves accordingly to the model proposed, let us consider the *EP* instance given in Figure 4.4 and summarized as follows:

- A: $\{A, B, C, D, E\}$
- E: $\{(x, y) \mid (x, y) \in RoomsEnv\}$
- P_0: $\{(560, 180), (580, 180), (600, 180), (620, 180), (640, 180)\}$
- T: $\{(140, 150), (120, 490), (600, 490)\}$

Where the termination criteria is here referred to checkpoints to be discovered on the map. In other words, the exploration ends when all the cells occupied by the checkpoints belong to the explored space:

$$ES_t = ES_T \Leftrightarrow T \subset K_t \tag{4.3}$$

Assuming the exploration is driven by the reserve coordination mechanism, the initial state of the exploration can be represented as:

- K_0: $\emptyset$.
- P_0: $\{(560, 180), (580, 180), (600, 180), (620, 180), (640, 180)\}$.
- G_0: $\emptyset$.

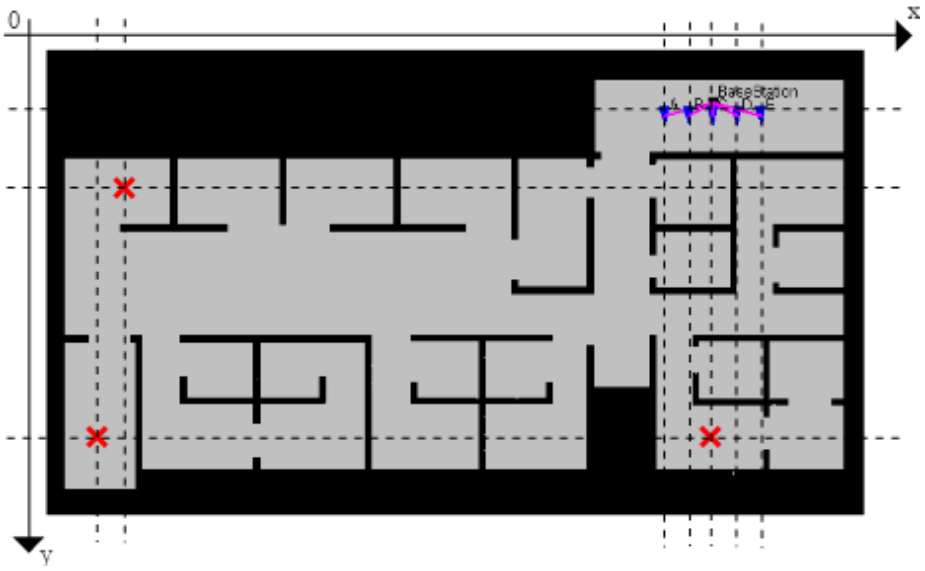
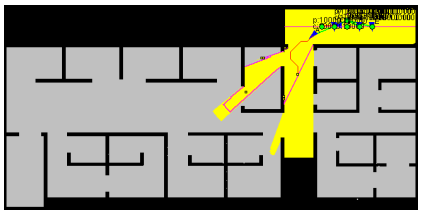


Figure 4.4: Instance of EP representing an exploration problem on the RoomsEnv in MRESim.

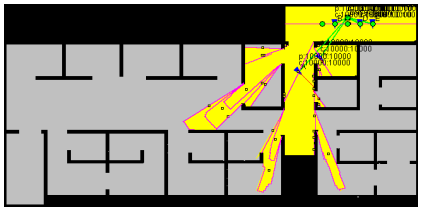
If, at time $t = 1$, the system gets the observation z_t coming from all the team members, the exploration architecture starts by solving the SLAM problem. Once the map is abstracted and frontiers are computed on it, the set F_1 is available for the coordination algorithm to allocate the robots. The reserve mechanism moves the first robot to start the exploration, leaving the three remaining robots in a reserve state at their starting poses. This situation is depicted in Figure 4.5a. When the active robot detects a branching point, it calls an idle agent to help. In Figure 4.5b, the second robot became active and was assigned to the first room. Three robots remain in the pool. After a while, all team robots are active (state represented in Figure 4.5c) and they start to behave independently by choosing the closest frontiers among the available ones. The exploration ends when the termination criteria is met by a state of the exploration. One of the possible final states of the exploration is shown in Figure 4.5d.

4.3 Optimal and dominant mechanisms

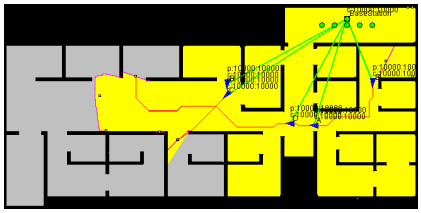
Generally, a model is formulated to allow humans to: evaluate, predict and control a complex system. Once a model for the coordination mechanisms has been defined, the aim of this section is to provide a formal evaluation criteria used to compare the different mechanisms, evaluating their performances without necessarily running a large amount of experiments. The concepts of optimal and dominant mechanisms are here defined to provide a rough a priori estimate



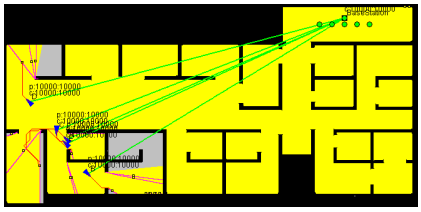
(a) ES_1: First state of the exploration on the RoomsEnv in MRESim.



(b) $ES_{m,1}$: Intermediate state of the exploration on the RoomsEnv in MRESim.



(c) $ES_{m,2}$: Intermediate state of the exploration on the RoomsEnv in MRESim.



(d) ES_T: Final state of the exploration on the RoomsEnv in MRESim.

Figure 4.5: Different states of the exploration.

about the goodness of the algorithms proposed in the next chapter to coordinate the robots during the exploration. These concepts are presented in the context of *Pareto optimality* and *Pareto dominance*, that are theoretically introduced and adapted to the exploration scenario.

4.3.1 Mechanism evaluation

The goal of a mechanism is to minimize the time spent by the team to reach the final state of the exploration. Therefore, it should be reasonable to consider the exploration time as a metric to evaluate the various algorithms. If such a metric seems perfect to compare the strategies afterwards, to state a priori that a coordination mechanism will take less time than another one to explore a given workspace, is pretty hard. Indeed, there are many elements affecting the exploration, starting from the type of environment under analysis. If a mechanism performs better than another on a S-O-LP environment, it is not guaranteed to be the best for all the possible maps. The mechanism could, indeed, be implemented specifically for those types of environments and could reveal strong drawbacks when dealing with a L-C-HP space. In addition, the high variability of the environment and the probabilistic nature of the pose estimates makes it difficult to generalize the results obtained from a specific *EP* instance.

To overcome this problem, the following sections are aimed to identify some elements that directly affect the exploration time but that, given a strategy, are easier to analyze without the need of practical experiments. The basic idea behind the formulation of these metrics is to decouple the performance evaluation

from the specific instance of *EP* under analysis, binding it to the structure of the mechanism. In other words, instead of estimating the performance measures basing on the dynamic behaviour of the algorithms, the estimate relies on the structure of the algorithms themselves. The two elements here introduced to characterize a mechanism are the *interference* and *availability* measures that are related to how a coordination mechanism tends to spread the robots on the environment, independently of the specific space type. These two measures have been, firstly, outlined in [29] but without a formal definition.

4.3.1.1 Interference and availability

Interference measures the distance robots have between each other, at each step of the exploration. A mechanism, during the exploration, can either force the agents to spread around the environment or maintain them close to each other. Usually, a coordination mechanism encourages the agents to fun out because it helps in parallelizing the exploration. The parallelization of the tasks of the agents, indeed, significantly reduces the exploration time and is one of the guiding principles of the coordination deployment. Moreover, keeping a low distance between the team members causes a high crash risk and increases the difficulties in the management of the system. For these reasons, a mechanism with a low interference distance is considered worse than one with a high interference indicator.

Interference, specified as λ from now on, can be expressed as follows, $\forall a \in A, t \in [0, ..., t_T]$:

$$\lambda_t(a) = \frac{1}{N-1} \sum\nolimits_{a' \in A | a' \neq a} d(P_t(a), P_t(a')) \tag{4.4}$$

Where $d(p_1, p_2)$ is the function that computes the distance on the $2D$ environment between two points.

$\lambda_t(a)$ is strictly related to a specific agent $a \in A$ and a specific time step $t \in [0, ..., t_T]$. However, the interference concept can be generalized to the state at time t as the average interference of single robots at t:

$$\lambda_t = \frac{1}{N} \sum\nolimits_{a \in A} \lambda_t(a) \tag{4.5}$$

Since a strategy is evaluated on the entire exploration, it makes sense to formalize also an interference measure for the global set of states, representing the whole exploration. The exploration interference is defined as the average interference over the states:

$$\lambda = \frac{1}{(T+1)} \sum\nolimits_{t \in [0, ..., t_T]} \lambda_t \tag{4.6}$$

Availability measures the distance each robot has to its currently assigned frontier. This metric is useful in determining how the mechanism tends to allocate the agents: either to close or to far targets. Mechanisms that assign

the robots to their closer frontiers allow them to re-plan often and to have an updated view of the environment. For this reason, algorithms with low availability distance measures are preferable over high-availability ones.

As in the interference case, also the availability measure can be expressed for a single agent, for a given state, and for the entire exploration. The availability, identified as α, of an agent $a \in A$ is defined, $\forall a \in A, t \in [0, ..., t_T]$, as:

$$\alpha_t(a) = d(P_t(a), G_t(a)) \tag{4.7}$$

Following the same reasoning did for interference, the state availability is the average availability of all the agents at t:

$$\alpha_t = \frac{1}{N} \sum\nolimits_{a \in A} \alpha_t(a) \tag{4.8}$$

Finally, the global availability can be written as follows:

$$\alpha = \frac{1}{(T+1)} \sum\nolimits_{t \in [0,\ldots,t_T]} \alpha_t \tag{4.9}$$

4.3.1.2 Evaluation criteria

An evaluation criterion can be proposed as a combination of the interference and availability metrics. The advantage brought by adopting this criterion is the ease in the estimation. With respect to predicting the exploration time taken by a strategy, its interference and availability are easier to estimate a priori (without running experiments in the first place) because they are based on the structure of the algorithm implemented, which is known without the need of an experimental campaign.

As introduced before, an ideal mechanism would maintain both a high interference measure, corresponding to a high average distance between the team members, and a low availability measure, representing a low average allocation distance. The evaluation criteria, called ϵ, used to compare mechanisms a priori should, then, be expressed as an objective function made by interference and availability contributions:

$$\epsilon = \lambda - \alpha \tag{4.10}$$

The negative sign to α is given to capture its real contribution: increasing α is not good for the exploration performance while increasing λ is. Naturally, the ϵ formulation can be expressed both for the single agent and for the single state.

4.3.2 Pareto Efficiency and optimal mechanisms

An optimal coordination mechanism would maximize the overall ϵ, obtaining the highest interference with the lowest possible availability over the entire exploration. The problem of finding the best coordination mechanism for a given environment can, therefore, be formalized as an optimization problem over ϵ:

$$\begin{aligned} \max_{P_t(a), G_t(a)} \quad & \epsilon = \lambda - \alpha \\ \text{s.t.} \quad & P_t(a) = Clear, \quad t = 0, \dots, t_T, a \in A, \\ & G_t(a) = Clear, \quad t = 0, \dots, t_T, a \in A \end{aligned} \tag{4.11}$$

However, the frontiers discovered during future steps of the exploration are not known, along with the next actions of the agents. This makes it impossible to design a mechanism to directly maximize ϵ because the dynamic information coming from the exploration will only be available at run time. ϵ can, instead, be roughly estimated before performing the experiments by considering the algorithmic structure of the mechanisms.

Therefore, the optimal mechanisms can only be characterized and, for this purpose, we introduce the concept of *Pareto optimality*. Pareto optimality is a measure of efficiency and, for this reason, is also called *Pareto efficiency*. In game theory, an outcome of a game is Pareto efficient if there is no other outcome that makes every player at least as well off and at least one player strictly better off. That is, a Pareto optimal outcome cannot be improved upon without hurting at least one player. In the coordination context, which represents the game, an outcome is an allocation of robots to frontiers during the entire path (or at the considered state, depending on the granularity). An optimal mechanism would provide Pareto optimal allocations at each step of the exploration, ensuring the ϵ maximization.

By restricting attention to the set of choices that are Pareto-efficient, a mechanism can be designed to make tradeoffs within this set, rather than considering the full range of every parameter. This idea is at the base of the *direct optimization* mechanism proposed in the next chapter.

4.3.3 Pareto Dominance and dominant strategies

When an allocation is not Pareto optimal, it can be improved by at least one change in the allocation of robots to frontiers. This change is named Pareto improvement and increases the gain of at least one agent in one of the states of the exploration. The Pareto improved allocation is said to dominate the former one. In game theory, an outcome of a game is *Pareto dominated* if some other outcome would make at least one player better off without hurting any other player. That is, some other outcome is weakly preferred by all players and strictly preferred by at least one player. If an outcome is not Pareto dominated by any other, than it is Pareto optimal.

When dealing with different mechanisms, recognizing dominant allocations can help in determining a priori the best approach, without the need for a large amount of experiments. For instance, the reserve mechanism, already introduced, is expected to provide lacking allocations in terms of availability in the initial allocations. For this reason, in the next chapter the *proactive reserve* mechanism is proposed to overcome this issue by reducing the availability distance of the initial allocations without affecting the overall interference. Thus,

the allocations made by the proactive reserve mechanism are supposed to dominate the ones provided by the reserve approach. In this case, the identification of the Pareto improvement of the allocations made by the reserve mechanism guided the design of its proactive version, helping in the definition of a better mechanism without requiring experiments in the first place.

Chapter 5

Coordination Algorithms

This chapter shows the details related to the coordination mechanisms implemented and tested in this thesis.

The coordination mechanisms proposed here are both online and offline and are divided in two sections. In the first section, four existing and representative mechanisms are introduced to provide a base line to which our results can be compared and evaluated. In the second section we introduce four coordination mechanisms that are intended as an evolution of the benchmark ones.

The algorithms are, first of all, intuitively presented and, then, formally described through the specification of the pseudocode for both the *computeUtility()* and the *handleFreeLocations()* function, in Algorithm 1. In particular, the choices made by the mechanisms are analyzed a priori (before running the experiments) in terms of interference, availability, and communication needs. Furthermore, for each mechanism, a priori evaluations are given for the different environments.

5.1 Benchmark coordination mechanisms

Our algorithms are compared with three offline mechanisms and an online one.

The offline mechanisms have been proposed in [29] and are called *reserve*, *buddy system* and *divide and conquer*. They have been already introduced in Section 2.3.2. These mechanisms share the same idea: to coordinate the robots, they assign a role to them. The role is statically defined before the beginning of the exploration and agents are assumed to be aware of it from t_0 on. The reserve mechanism separates the team members in active and idle robots, while the divide and conquer approach considers leader, and follower agents. The buddy system, finally, mixes the concepts of idle, leader and followers to tradeoff the benefits of the former mechanisms.

The single online mechanism presented here is called *utility based* and has been introduced in [57] and refined in [58]. This approach is dynamic and, instead of basing the coordination on the roles, computes the utility of a frontier

relying on the actions in which the team members are currently involved.

5.1.1 Reserve

As already mentioned, the reserve mechanism considers two roles: agents are either idle or active. An active agent moves by choosing the closest frontier among the available ones. When it detects a *branching point* (i.e., when it detects more frontiers at a time) on the environment, the active agent selects one of the possible directions (the closest) and calls the idle teammates, if any, to explore the other frontiers. A branching point can be composed by multiple frontiers as long as they have a minimum distance among them.

An idle agent, also called reserve agent, wait at its starting pose until it is called to frontier. When a branching point is detected the idle agents are assigned to the branching frontiers depending on their distance. The reserve mechanisms tries to minimize the availability distance in the activation of the idle agents.

The *computeUtility()* function driving the reserve mechanism is presented in Algorithm 2.

Data: a,l,ES_t,k
Result: returns the utility of the (a,l) pair
if *a is idle* **then**
  if *a.location* = l **then**
    return *1*
  else
    return *0*
  end
else
  return $k \cdot \frac{1}{d(a.location,l)}$
end

Algorithm 2: Reserve computeUtility() function

The input data is made by the agent a, the target location l (for which the utility must be computed), and the current state ES_t. For each active agent-location pair, the utility is defined by the inverse of the distance between the agent and the location, multiplied by a constant k. The value of k should be big enough to make the utility higher then 1. Idle agents, instead, are encouraged to maintain their initial poses by giving to those locations a value of 1 against the 0 attributed to any other candidate location. An idle agent is turned into active state by an already exploring teammate that, when it detects more than one frontier, calls the closest reserve robot thanks to the *handleFreelocations()* function provided in Algorithm 3.

This function receives the set of free locations deriving from a branching point and the currently available utility matrix. For each location provided, the *handleFreeLocations()* function selects the closer reserve agent and properly updates its utility matrix. This algorithm assumes that the roles of the agents

are known during the entire exploration process. Thus, the distances of the reserve agents to a given location $l \in L_{t+1}$ can be computed as $d(reserveAgents, l)$ and the closer agent can be easily identified. The utility corresponding to agent $a \in A$ is set to the inverse of the distance between a and the current location, multiplied by k. Since this utility was previously set to 1 for the the cell occupied by a and to 0 for the others, the update given by the distance will make the current location preferable on the alternatives if k is big enough to guarantee an utility higher than 1.

Data: L_{t+1},U
Result: updates the utility matrix of the reserve agents
for $i \in \{1, ..., |L_{t+1}|\}$ **do**
  get the currently considered location $\{l := L_{t+1}[i]\}$
  get the closer reserve agent to the location
    $\{a' = \text{argmin}_a \, d(reserveAgents, l)\}$
  update the utility for the given location for the agent
    $\{U[a'][l] := k \cdot \frac{1}{d(a'.location, l)}\}$
end
return U

Algorithm 3: Reserve handleFreeLocations() function

This mechanism tries to increase the interference distance between robots by distributing their calls over time. In this way robots will spread on the environment, mostly in the initial parts of the exploration, with a low communication need among them. However, one may notice two drawbacks in such a mechanism: the availability measure will probably be high for the reserve agents (which are called to frontiers discovered by active, and possibly far, teammates) and the offline nature of the role assignment used by this mechanism, along with its specific design, will leave robots more free as the exploration goes on (causing a non-cooperative and inefficient behaviour in later steps). Indeed, once all the agents are active, they behave in a non-coordinated way. An active agent selects its goal considering only its distance as a cost and does not take into account the target locations the other agents are moving to.

Given the above considerations, the reserve mechanism should have good performance when capable of reaching a good distribution of robots in the first steps of the exploration. Indeed, if the idle agents are employed in immediate sequence and without spreading too much over the workspace, they will start to explore as independent units when relatively close to each other, forming a sort of group. This will likely cause overlapped allocations. As a consequence, the reserve mechanism is expected to perform better on open and highly parallelizable environments (O-HP), which allow an immediate dispersion of the robots. The openness of the environment, indeed, favors the spread of the team members that are not constrained by the presence of obstacles.

5.1.2 Divide and Conquer

In this mechanism, agents are split in different groups and each group is composed by a single leader and some followers. The leader agent selects its goal frontier, which becomes the goal for all the followers too. Initially, all the agents belong to a single group and the leader is arbitrary elected. Whenever a group discovers a branching point, half of the followers form a new group (in which, again, the leader is arbitrary elected). The new group gets the free frontiers (the ones not selected by the original leader) and, if needed, is split again. The groups stop splitting when formed by a single agent. At this point the exploration proceeds in a non-cooperative manner and the agents take the closest frontiers as goals.

The *computeUtility()* function is presented in Algorithm 4. In this case, the follower agents assign a utility 1 to the current goal of their leaders and utility 0 to the remaining locations. In this way, these agents are forced to follow their leader. A leader agent, instead, picks the closer frontier and, thus, sets the utility of a location to the inverse of the distance between itself and the location.

Data: a,l,ES_t
Result: returns the utility of the (a, l) pair
if *a is follower* **then**
  get the agent which is leading the current follower
  $\{leader := leaderAgents[a]\}$
  if l = *leader's goal* **then**
    return *1*
  else
    return *0*
  end
else
  return $k \cdot \frac{1}{d(a.location,l)}$
end

Algorithm 4: Divide and Conquer computeUtility() function

The set of leader agents must be internally maintained by the mechanism and is here accessed as *leaderAgents*. When a leader detects a branching point, the free frontiers are provided to the new leader obtained by splitting the group in half. This behaviour is implemented in the *handleUnallocatedFrontiers()* in Algorithm 5.
Also in this case some information is required: the mechanism needs to store the new elected leader and its followers at each split. The closest frontier to the new leader, among the branching ones, is selected for the new group and both the utility vector of the leader and of the followers are updated.

While the reserve mechanism tries to deal with interference, the divide and conquer is focused on the availability distance. This mechanism is aimed at reducing the availability measure during the entire exploration. The agents are

Data: L_{t+1},U
Result: updates the utility matrix of the new leader and followers
get the new elected leader $\{leader := newLeader\}$
get the new assigned followers $\{followers := newFollowers[leader]\}$
get the closer frontier to the leader $\{l := \text{argmin}_l\, d(leader, L_{t+1})\}$
update the utility vector of the new leader
$\{U[leader][l] := k \cdot \frac{1}{d(leader.location,l)}\}$
for $i \in \{1, ..., |followers|\}$ **do**
  get the currently considered follower $\{a := followers[i]\}$
  update the follower utility vector $\{U[a][l] := k \cdot \frac{1}{d(leader.location,l)}\}$
end

Algorithm 5: Divide and Conquer handleFreeLocations() function

always allocated to one of the closer available frontiers, regardless their role. On the other hand, the main drawback concerns precisely the interference measure, which is very low when the robots move in large groups. As in the reserve case, if robots are not spread quickly over the environment, they will proceed greedily when still close to each other. In this sense, a strongly closed environment would be the worst case: robots would rapidly find different branching points determined by the many obstacles and, probably, would move as independent units too soon in the exploration. When dealing with open environments, this mechanism behaves similarly to the reserve approach. The ideal environment for this mechanism would be represented by a binary tree environment, in which a node represents a location on the map. In this case, starting at the root all the group splits happen when branching out the current node going down in the tree.

5.1.3 Buddy System

The buddy system coordination mechanism tries to combine the two previously presented approaches. The agents are grouped in pairs, with a leader and a follower. In addition, active and idle roles are assigned to each group. An idle pair waits at its starting location while active pairs begin the exploration. When a branching point is discovered by an active pair, the two agents split and start two independent explorations. If, in turn, an active independent agent gets to a branch, it calls a idle pair which starts to move. When all the idle pairs have been turned into active state and have split, the agents make individual decisions considering only their distance from the candidate locations (as in the previous mechanisms).

The buddy system exploits the interference and availability optimizations observed with the reserve and the divide and conquer mechanisms. Agents, indeed, can have one of the four roles already introduced: idle, active, leader, or followers. The basic idea is to manage the interference as done by the reserve approach, reducing the availability problem underlined for the reserve mecha-

Data: a,l,ES_t
Result: returns the utility of the (a, l) pair
if *a is idle* **then**
  if *a.location* $= l$ **then**
    return *1*
  else
    return *0*
  end
else
  if *a is follower* **then**
    get the agent which is leading the current follower
    $\{leader := leaderAgents[a]\}$
    if $l =$ *leader's goal* **then**
      return *1*
    else
      return *0*
    end
  else
    return $k \cdot \frac{1}{d(a.location,l)}$
  end
end

Algorithm 6: Buddy system computeUtility() function

nism. Indeed, the follower of a pair can be seen as an idle agent but, instead of waiting at its initial pose, it moves with its leader. In this way, when an active agent finds a branch, a reserve agent, that is, the follower in the pair, is already there. This mechanism (like the others offline mechanisms presented so far) is limited by the team size as it stops coordinating the robots when all the pairs have split and no more idle agents are available in the reserve pool. For this reason, the problems related to the non-cooperation in later steps remains. Indeed, the buddy system mechanism seems more suitable for O-HP environments which allow for a fast spread of the team members, similarly to the reserve and divide and conquer mechanisms.

The *computeUtility()* function describing the buddy system mechanism is presented in Algorithm 6, while Algorithm 7 shows the outline of *handleFreeLocations()*. If this mechanism detects a branching point, generating free locations that can be assigned to other agents, it first checks if the buddy of the active agent detecting the branch is either still a follower or if it has already been split. If the buddy agent is a follower, the system splits the pair assigning the follower to one of the free locations. Otherwise, if the detecting agent is alone, the system calls a reserve pair to help. If there are multiple free locations, they are all provided to the idle pairs and their allocation is based on minimizing the availability distance, as explained for the reserve case.

Data: a,L_{t+1},U
Result: updates the utility matrix of reserve pairs, new leaders and followers
get the buddy of the agent $\{buddy := buddies[a]\}$
if *buddy is follower* **then**
  get the closer free location to the follower
  $\{l := \text{argmin}_l\, d(b.location, L_{t+1})\}$
  split the pair, updating the follower utility
  $\{U[buddy][l] := k \cdot \frac{1}{d(buddy.location,l)}\}$
else
  for $l \in L_{t+1}$ **do**
    get the closer reserve agent to the location
    $\{a' := \text{argmin}_a\, d(reserveAgents, l)\}$
    update the utility of the agent $\{U[a'][l] := k \cdot \frac{1}{d(a'.location,l)}\}$
    update the uttility of the buddy of the agent
    $\{U[buddies[a]][l] := k \cdot \frac{1}{d(buddies[a].location,l)}\}$
  end
end

Algorithm 7: Buddy system handleFreeLocations() function

5.1.4 Utility Based

The utility based mechanism is the only example of online mechanism considered as benchmark in this work. The main idea behind this approach is to express an utility measure for every possible agent-frontier allocation as a combination of the distance of the agent from the frontier and of the number of robots assigned to it. In particular, whenever a robot is allocated to a frontier $f_1 \in F$, the utility of that frontier and of the ones close to it, is decreased by the observation probability. Being $f_2 \in F$ another available frontier, the observation probability is intended as the probability to observe f_2 from f_1. It is indicated as $P(\parallel f_2 - f_1 \parallel)$ and have been described in Section 2.3.1.

The *computeUtility()* function driving the utility-based mechanism is presented in Algorithm 8. The *handleFreeLocations()* function, instead, is structured as in Algorithm 9, where $P(visibility)$ is the probability of the current location to be visible from the location assigned to a, as previously defined.

Data: a,l,ES_t
Result: returns the utility of the (a, l) pair
compute the cost for the agent to reach the location
$\{c := d(a.location, l)\}$
get the current utility of the location $\{u := U[a][l]\}$
return $u - c$

Algorithm 8: Utility-based computeUtility() function

This mechanism has the clear advantage to be independent by the environment

on which it is deployed. It, indeed, is online and dynamically adapts the future allocations basing on the current ones, instead of relying on predetermined roles. The utility proposed takes into account both the interference and the availability management. The decrease in the utility depending on the currently known allocations encourages the agents to spread on the environment without directly coordinating them and controls the interference distance, which is implicitly maximized. Considering the distance to the target location in the utility computation, instead, the mechanism can maintain a good availability for the team members. This mechanism is supposed to behave well on many different types of environments. However, in the initial part of the exploration the utilities are all set to a default value and the allocation is made almost in a non-coordinated way. The cost of being this versatile is the high communication frequency required to the agents. To maintain an updated utility matrix, indeed, at every step the mechanism has to: compute the all the agent-frontier distances, keep track of the current goals communicated and update the frontier utilities at every allocation.

Data: a,L_{t+1},U
Result: updates the utility matrix of the visible frontiers
for $l \in L_{t+1}$ **do**
  if l *is visible* **then**
    update the utility, for all the agents, of the location
    $\{U[A][l] := U[A][l] - P(visibility)\}$
  end
end

Algorithm 9: Utility-based handleFreeLocations() function

5.2 Proposed coordination mechanisms

The following coordination mechanisms are proposed to improve the performance of the benchmark approaches already introduced, solving some of the problems previously underlined. Three offline mechanisms are formulated as a modified version of the ones in [29], namely the *proactive reserve*, the *proactive buddy system*, and the *side follower* mechanisms. They are all aimed at improving the way in which their corresponding benchmark versions manage either the interference among the robots or their availability.

Finally, a novel online mechanism is proposed to directly maximize the interference measure while minimizing the availability distance (as the utility based strategy does) but limiting the communication requirements and improving the initial performance of the team. This mechanism actually combines online and offline principles because a minimum roles definition is provided to the agents to coordinate themselves at the beginning of the exploration and to reduce the communication needs.

5.2.1 Proactive Reserve

The problem regarding the reserve mechanism underlined in the previous section is related to the availability of idle robots when turned into active state. In the reserve mechanism, each idle agent waits at its starting pose and, once called in action, it often has to run a long path to reach its assigned frontier. This results in a high overall availability measure. The proactive reserve algorithm presented here overcomes this limitation by assigning a proactive waiting location to all the reserve (idle) agents. The waiting location is chosen to reduce the distance between idle and active robots and is, then, computed as the barycenter of the polygon whose vertices are the locations of the active agents.

The proactive reserve modifies only the *computeUtility()* function with respect to the plain reserve mechanism, while maintaining the same *handleFreeLocations()* implementation. The proactive reserve *computeUtility()* function is shown in Algorithm 10.

Moving the robots to a waiting location makes it possible to reduce the average distance a reserve agent has to travel when an active one call it to explore a branching point. This mechanism, indeed, has the advantage of maintaining a low availability for reserve agents but shares with the plain reserve approach the drawback related to the offline nature of the mechanism design. Once active, the robots are not coordinated anymore: they choose their closest frontiers without leveraging the information coming from the rest of the team.

Data: a,l,ES_t
Result: returns the utility of the (a, l) pair
if *a is idle* **then**
  compute active agents barycenter
   $\{barycenter := computeBarycenter(A)\}$
  if $barycenter = l$ **then**
    return *1*
  else
    return *0*
  end
else
  return $k \cdot \frac{1}{d(a.location,l)}$
end

Algorithm 10: Proactive reserve computeUtility() function

The openness and the high parallelizability of the environment surely help the performance of this mechanism because, jointly with the other offline approaches, the distribution possibilities provided by the environment are crucial in the good behaviour of the team.

5.2.2 Proactive Buddy System

The same proactivity concept presented for the proactive reserve mechanism is adapted to the buddy system case. The resulting mechanism, called proactive buddy system, moves the waiting locations of the idle pairs onto the barycenter of the polygon whose vertices are the positions of the active agents (belonging to an active pair).

The pairs, when turned to an active state, are likely closer to the assigned frontiers compared to the plain buddy system approach and this helps in reducing the average availability measure of the mechanism. Even if the buddy system is an improvement of the reserve mechanism in terms of availability, the proactivity provided by this modification could significantly contribute to an even higher decrease in the average availability distance.

As in the proactive reserve case, even the *computeUtility()* function driving the proactive buddy system is modified to take into account the barycenter instead of the current agent location, as a waiting position. On the other hand, the *handleFreeLocations()* is the same of the plain buddy system mechanism. The pseudocode for the *computeUtility()* function is provided in ALgorithm 11.

Data: a,l,ES_t
Result: returns the utility of the (a, l) pair
if *a is idle* **then**
  compute active agents barycenter
   $\{barycenter := computeBarycenter(A)\}$
  if *a.location = barycenter* **then**
    return *1*
  else
    return *0*
  end
else
  if *a is follower* **then**
    get the agent which is leading the current follower
     $\{leader := leaderAgents[a]\}$
    if *l = leader's goal* **then**
      return *1*
    else
      return *0*
    end
  else
    return $k \cdot \frac{1}{d(a.location,l)}$
  end
end

Algorithm 11: Proactive buddy system computeUtility() function

5.2.3 Side Follower

Similarly to the divide and conquer mechanism, the side follower mechanism divides the agents in groups. However, instead of moving in a single group, the team members are initially organized in groups of three. The roles assigned to the three robots belonging to a subgroup are: leader, left follower, and right follower. The subgroups start the exploration all together, being distributed as much as possible on the currently available frontiers. The basic idea behind this mechanism is to never split the group and to never change the predefined roles. The coordination is, then, completely offline and related to the directions the agents should follow. The leader agent is encouraged to follow frontiers in front of it (along the direction of movement) while the side followers explore the remaining frontiers: the left follower will be allocated to frontiers that are discovered on the left of the leader and the right follower will be assigned to the right ones. Both the leader and the followers select the closest frontier among those available in their assigned directions.

This mechanism is aimed at optimizing the interference and availability between robots by statically determining their preference criteria for a frontier: the robots belonging to a subgroup will spread on the environment because they are allocated to geometrically distributed locations while maintaining a high availability driven by the distance of the assigned frontiers. The agents never leave their roles and the coordination is guaranteed for the entire exploration. The ideal environment for this mechanism would be indoor with a corridor with rooms at both sides: the leader agent would explore the corridor while assigning the followers to the various, uncovered, rooms. The key point when using strong, predefined, roles is that agents will require very little communication, since the coordination is almost completely fulfilled offline. In this case, indeed, the only thing a follower needs to know is the direction of its leader.

The *computeUtility()* function driving the side follower mechanism is presented in Algorithm 12.

Straight, left, and right frontiers are identified by considering the direction of the leader agent. Taking the first point in the path of the agent and its current position, a vector can be defined to represent its current direction.

5.2.4 Direct Optimization

The direct optimization mechanism combines the concepts mentioned so far in a mechanism that exploits both online and offline coordination. The core of the algorithm is the online computation of the utilities of the available frontiers equivalent to the one proposed in the utility based mechanism. The communication and computational needs are here tackled by providing an offline distinction of roles. The offline approach deployed in the direct optimization is based on the proactive reserve mechanism. Robots are divided in idle and active and the exploration is started as in the proactive reserve case. The difference, which makes this mechanism an online approach, is in the coordination robots deploy after all the team members are turned into active state. Instead of behaving

```
Data: a,l,ES_t
Result: returns the utility of the (a, l) pair
switch agent role do
    case leader do
        compute the straight frontiers subset {straightFrontiers}
        if l ∈ straightFrontiers then
            return 1 / d(a.location,l)
        else
            return 0
        end
    end
    case left follower do
        compute the left frontiers subset {leftFrontiers}
        if l ∈ leftFrontiers then
            return 1 / d(a.location,l)
        else
            return 0
        end
    end
    case right follower do
        compute the right frontiers subset {rightFrontiers}
        if l ∈ rightFrontiers then
            return 1 / d(a.location,l)
        else
            return 0
        end
    end
end
```

Algorithm 12: Side Follower computeUtility() function

independently, the agents dynamically coordinate their actions by combining the cost of reaching a frontier to the number of robots already assigned to it or to one of the frontiers nearby.

This mechanism is proposed to trade-off the benefits and drawbacks offered by online and offline coordination. The aim of this mechanism is to directly optimize the interference and availability measures both in the initial and in the final steps of the exploration, to always provide the robots with a coordinated management of their tasks. The online extension should allow the proactive reserve approach to behave better on C-HP environments than the corresponding offline formulation. At the same time, starting the exploration with an offline setting, allows the robots to deploy coordination even in the early stages of the task, in which the utility based mechanism could have difficulties to properly distribute the robots. Moreover, the offline approach allows the direct optimization mechanism to reduce the communication needs of the robots at the

beginning of the exploration, when the offline mechanism is sufficient to manage the interference and the availability.

In this case, the algorithms representing the *computeUtility()* and the *handleFreeLocations()* are directly derived as union of the reserve and of the utility-based mechanisms and, for this reason, it is not needed to report them here. In particular, the algorithms combined are the Algorithm 2 and the Algorithm 8 for the *computeUtility()* function and the Algorithm 3 and the Algorithm9 for the *handleFreeLocations()* function.

Chapter 6

Experiments and results

This section is aimed at presenting the experiments performed to evaluate the coordination mechanisms described in the previous chapter. The experiments are run in MRESim on simulated 2D environments. The environments have been selected to represent different classes: from open, to cluttered, to highly parallelizable spaces. Moreover, the experiments involve different team configurations in terms of team size. MRESim offers a non-deterministic exploration, meaning that when running multiple experiments over the same mechanism, the same environment, and the same team configuration, the results are not guaranteed to coincide. For this reason, each mechanism-environment-team configuration triple is tested 10 times and the results are presented here as average on the outcomes. To give an idea of the variability of the exploration time of the mechanisms, its variance obtained in the multiple runs is provided.

The next section describes in detail the EP instances over which the experiments are performed. In the second and last section, instead, the results obtained on the different instances are presented, for each mechanism proposed in this thesis.

6.1 Exploration instances

Recalling what presented in the previous chapters, an EP instance is characterized by the 4-ple $\langle A, E, P_0, T \rangle$:

- A: set of robots used during the exploration.
- E: environment to be explored.
- P_0: initial poses of the team.
- T: termination criteria.

In this section we present the team configuration, along with the environments in which the team is deployed. For each environment, the initial poses of the

agents are represented and the termination criteria is related to the percentage of free space explored. In particular, an experiment is considered concluded if the agents discover the 95% of the free space of the environment.

6.1.1 Team configuration

A mechanism is tested by deploying teams with different sizes. Here, we performed experiments with teams composed from two to ten agents. In MRESim a team is configured through a file which specifies the ID and names of the agents, their initial poses, orientations, types and, possibly their roles. The type of the robot is characterized by its sensing and communication ranges and its battery life.

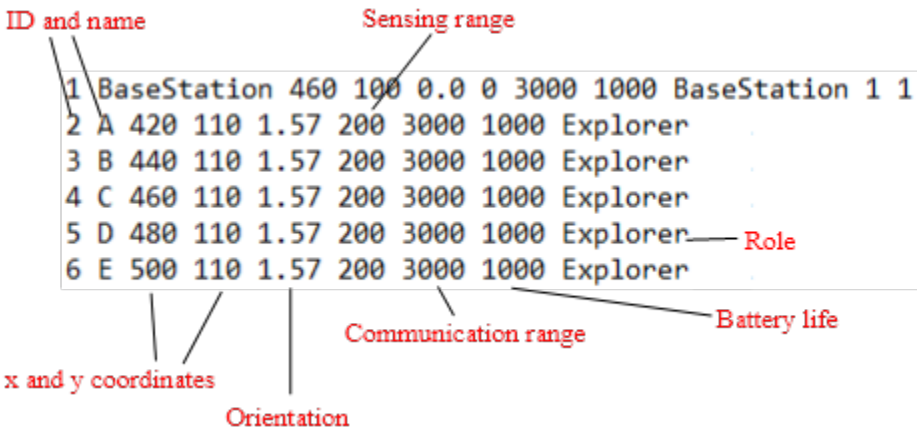


Figure 6.1: Team configuration file in MRESim.

In Figure 6.1 an example of team configuration file is given. The base station is not an agent but a base point to which agents communicate the information that the mechanisms require to coordinate themselves. As already introduced, this thesis in not focused on how the communication is built among the agents and, for this reason, the communication range is set to a arbitrary (high) value: This guarantees the communication possibility during the entire exploration. The same can be said for the battery life, set to an infinite value. The sensing range, instead, is left at the default MRESim value.

The team configuration file, along with the mechanism and the environment to be explored, compose the specific experiment to be run. The environments tested are presented in the following section.

6.1.2 Environments

The environments that we consider in this thesis are all indoor environments and are shown in Figure 6.2. The choice of indoor environments to test the mechanisms is motivated by the application contexts, which make indoor spaces

more common to be explored than outdoor ones. Thinking to a mapping or a search-and-rescue setting, indeed, the typical environments faced by a team of robots are houses, or offices.

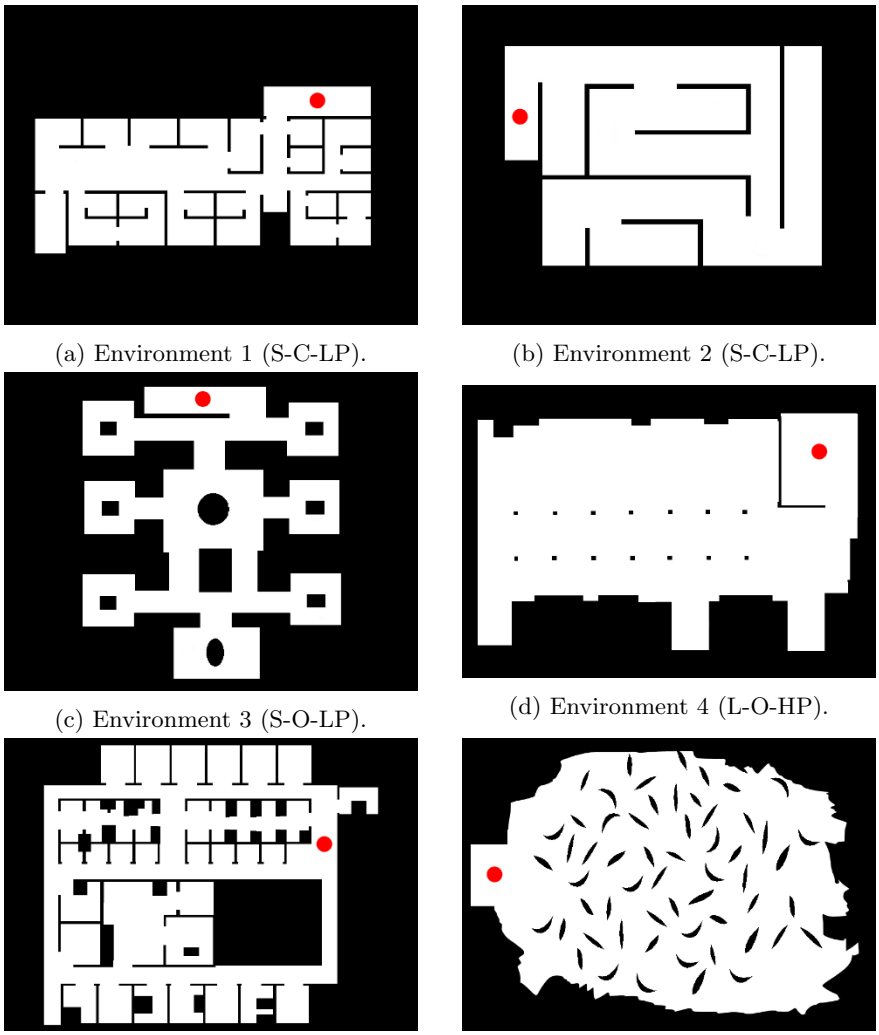
The first environment (environment 1 in Figure 6.2a) is a small, cluttered, and lightly parallelizable environment. It could represent a small office with average-size rooms. The space is fragmented but organized around a central (horizontal) corridor. The initial poses of the agents (red point) are located at the top-right corner. The structure of the obstacles and of the rooms will force the team to follow two main directions, making the environment lightly parallelizable. Indeed, some agent will be directed to the bottom-right part of the environment, while the others will move to the left, through the main corridor.

The environment 2 in Figure 6.2b is the one that offers less branching points with respect to the others. Indeed, the space representing a small and simple maze is organized in subsequent corridors. The initial poses of the agents are placed at the beginning of the maze, on the left. The agents will follow the corridors, possibly splitting when meeting one of the few branching points. The environment is, then, described as a S-C-LP space.
The third environment could represent a bunker with a large, central, corridor connecting six open rooms. Obstacles are well organized but the space is small enough, compared to the other environments in Figure 6.2. The initial poses of the team members are at the top of the bunker. The agents will follow the main corridor, discovering the branching points in correspondence of the rooms. The space is not much fragmented but the structure offers a little range of directions, making the environment slightly parallelizable.

The environment in Figure 6.2d is the first large environment of the series. It could represent the hall of a hotel or of a big palace. Robots start the exploration from the top-right corner. Once got out of the single room in the environment, the agents face an open space with small obstacles. The boundary region between known and unknown areas will expand concentrically with the agents poses as a center, at each step of the exploration. This makes it possible to recognize frontiers in many directions at the same time because the laser scan of each agent will not hit by any obstacle. The agents have the possibility to simultaneously explore different areas of the environment, which is, therefore, highly parallelizable.

In Figure 6.2e, the environment represents a big-size office with lots of rooms and corridors, branching out in many directions. The space is fragmented, due to the large quantity of walls and rooms all over the environment. The rooms have different sizes and are connected by corridors. The agents are initially placed at the top right of the workspace from where they have three directions initially available. The large size of the environment makes it highly parallelizable even if the space is fragmented. Agents, indeed, will be able to simultaneously discover frontiers along the many corridors and rooms and to spread enough on the environment, even in the early stage of the exploration.

The last environment, in Figure 6.2f is a large, cluttered, and highly parallelizable environment. This space could represent, for instance, the intern of an



(a) Environment 1 (S-C-LP).

(b) Environment 2 (S-C-LP).

(c) Environment 3 (S-O-LP).

(d) Environment 4 (L-O-HP).

(e) Environment 5 (L-C-HP).

(f) Environment 6 (L-C-HP).

Figure 6.2: Environments considered.

abandoned mine where obstacles are randomly placed on the ground. The many obstacles cover the entire space and no specific structure can be recognized for this environment. The boundary region will expand in a fragmented way in all directions, making the environment HP. The frontiers, then, will be dispersed at every step of the exploration, allowing the team members to spread with relative ease on the space.

A coordination mechanism should be able to generalize as much as possible its behaviour on these types of environments, exploiting the different features

of them. The results of the original mechanisms introduced in this thesis and of the benchmark ones are presented and commented in the next section.

6.2 Mechanisms results

The mechanisms tested in this section have been originally presented in Section 5.2. They are: *proactive reserve*, *proactive buddy system*, *side follower* and *direct optimization*.

The mechanisms are here compared in terms of exploration time, interference, availability, and decision time. The exploration time is the main evaluation criteria to analyze the results coming from the experiments. On the other hand, the interference and availability measures of the mechanisms have been theoretically considered in the previous chapter without considering the actual performance of the mechanisms. We here present the interference and availability measures obtained during the experiments to confirm the qualitative evaluation given in the previous chapter and validate the model proposed. Finally, the decision time indicates the time taken by the mechanism to allocate a single agent. This measure allows to analyze which mechanism is the fastest one in allocating the agents.

All the results shown in this section are compared against the benchmark mechanisms already introduced in Section .

6.2.1 Proactive Reserve

The proactive reserve mechanism is intended as a dominant mechanism with respect to its corresponding plain formulation, called reserve. The results obtained on the first three environments, for all the team configurations, are depicted in Figure 6.3, Figure 6.4, and Figure6.5, repsectively. These environments are the small ones. The proactive reserve mechanism is shown to generally outperform the benchmark mechanisms with almost any team configuration. However, its performance is worse or equal to the others in the second environment, the small maze. This is probably due to the limited branching possibilities offered by the maze. Independently of the mechanism and of the team size, indeed, this environment provides only four points in which the agents can split. Thus, the benefits brought by the coordination mechanism and by increasing the number of agents deployed are not exploited on this environment. On the other hand, the proactive reserve seems to have a good impact on the first environment, which is S-C-LP too. This can be due to the fragmentation in the calls of the agents. In other words, the environment is small and closed enough to discover few frontiers at a time, maintaining the idle agents in the reserve pool for a long time. Since the proactive reserve mechanism tries to increase the availability for the idle agents, it performs better than the reserve mechanism on environments that maintain agents in an idle state for long time. The third environment is open and allows offline mechanisms to perform well, and possibly better, than online ones. Indeed, the proactive reserve is shown to outperform the utility

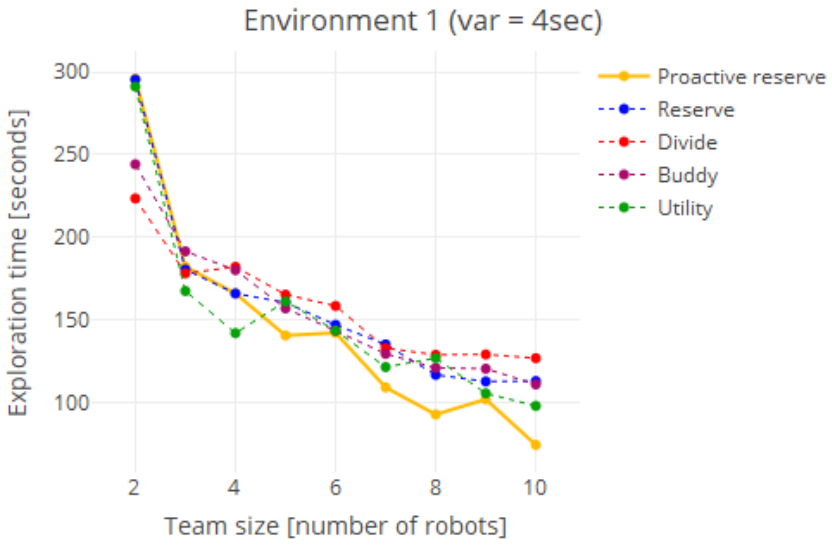


Figure 6.3: Proactive reserve exploration time on environment 1.

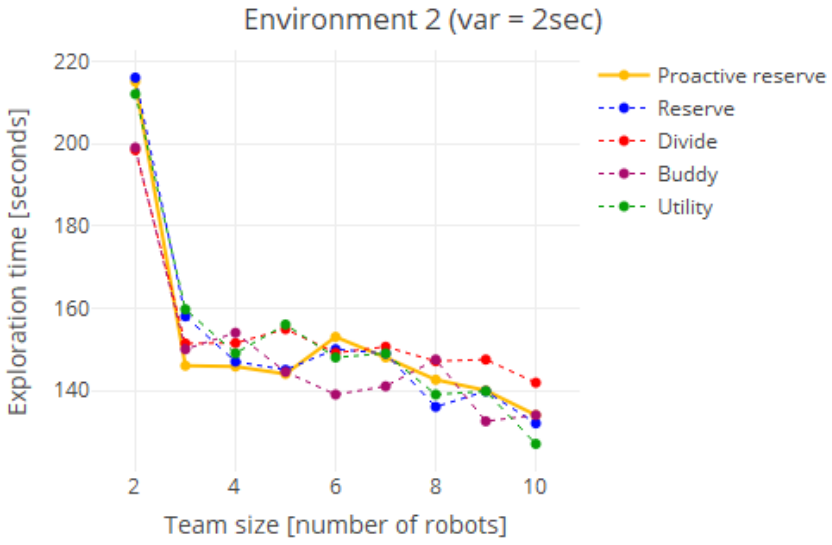


Figure 6.4: Proactive reserve exploration time on environment 2.

based, online, mechanism on the third environment. However, its performance is similar to the one of the reserve mechanism. Following the reasoning just introduced, in the third environment the agents discover more frontiers at a

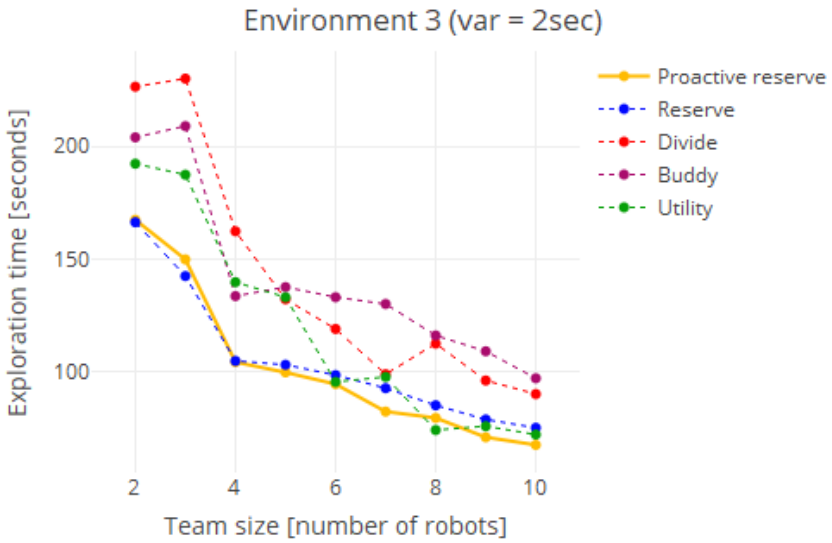


Figure 6.5: Proactive reserve exploration time on environment 3.

time from the early stage of the exploration and, as a consequence, the idle agents are almost immediately turned into active state. Therefore, the benefits introduced by the proactivity are not particularly effective on this environment, which activate the agents already in the initial part of the exploration.

In Figure 6.6, Figure 6.7, and Figure 6.8 the results obtained by the mechanism in the large environments, the last three are presented. The fourth environment is open and, as already stated, this is expected to favor the offline approaches over the online ones. Indeed, the proactive reserve mechanism resulted significantly better than the utility based and, on this environment, outperformed any other. The worst mechanism tested on this environment seems by far the divide and conquer, along with the buddy system. In an open environment, the division of agents in groups appears inefficient. This is because the agents have available frontiers in all the possible directions and allocating all the team members, or part of them, to the same direction can result in a waste of time. On the other hand, the proactive reserve mechanism initially spreads the agents as much as possible thanks to the openness of the space. The agents are not called in the immediate sequence, as in the case of the third environment, because the frontiers, even thought distributed over different directions, are not many. Not being constrained by obstacles, the frontiers belong to a single and extended boundary region and are not split into small pieces. The fifth environment is probably the most complicated to explore. All the mechanisms expose a similar behaviour, except the divide and conquer. The division into a single group seems not efficient also for extremely cluttered environments. The splits that the group makes during the exploration happen very fast in the early

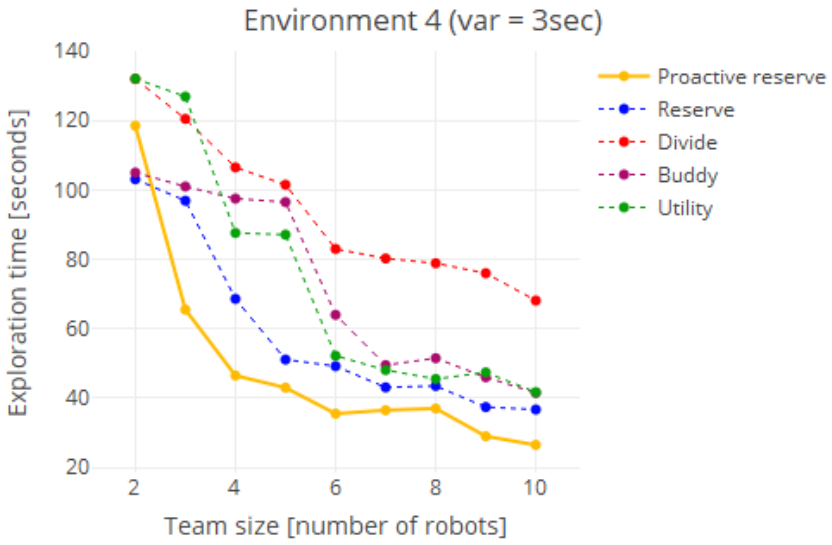


Figure 6.6: Proactive reserve exploration time on environment 4.

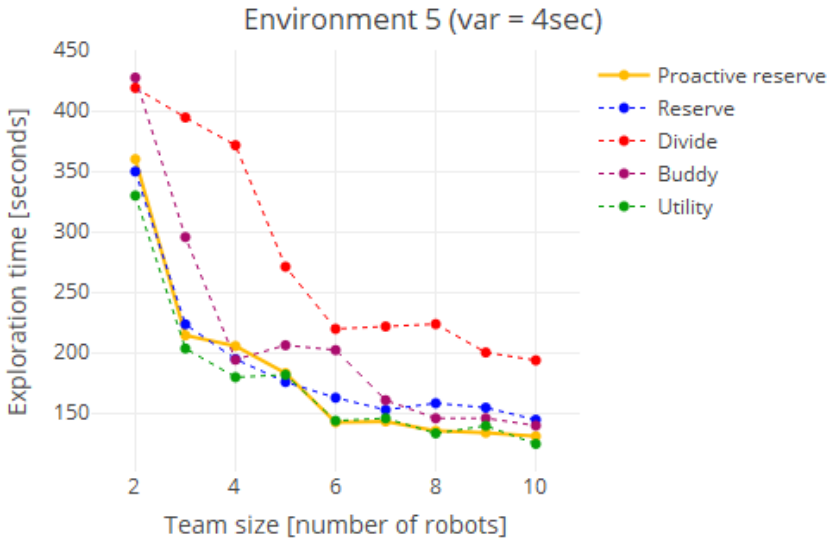


Figure 6.7: Proactive reserve exploration time on environment 5.

steps of the task, causing the agents of the group to be close to each other when they start to explore independently. This can cause the weak performance of the divide and conquer, that maintains a high interference in the initial steps.

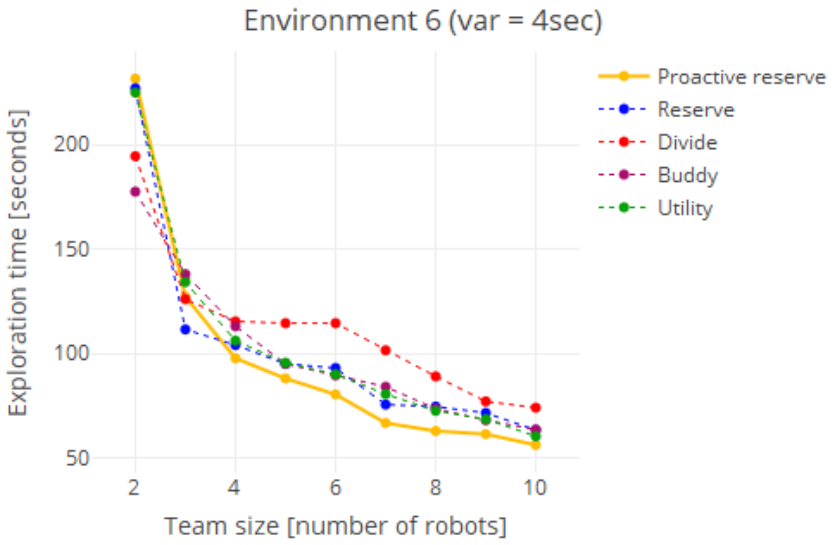


Figure 6.8: Proactive reserve exploration time on environment 6.

Instead, the utility based mechanism benefits from its online principle because, being the environment cluttered and with many branching points, the offline approaches tend to loose their coordination effect soon enough. Finally, the results related to the last environment show a similar behaviour from all the mechanisms. The proactive reserve seems slightly better than the rest of the algorithms, very close to its reserve ancestor. Like in the previous case, many frontiers are discovered pretty fast, due to the many, randomly-placed, obstacles. This decreases the time idle agents remain in the reserve pool, weaken the benefits coming from the proactivity of the pool. Dealing with interference and availability, this mechanism was expected to improve the availability measure while not affecting the interference with respect to the reserve mechanism, considered as reference. The interference and availability measure of the proactive reserve mechanism are compared with the ones of the reserve in Figure 6.9 and Figure 6.10. The results obtained in terms of interference and availability confirmed what assumed in the previous chapter. The interference is not strongly affected by the proactive mechanism. Since the interference metric measure the distance between the agents, it tends to be higher on the open environments (the third and fourth). The mechanism, indeed, when moving the agents towards the barycenter, maintains them with a certain distance and the resulting interference is similar, or better, to the one obtained when the agents wait at their initial poses. On the other hand, the availability measure is significantly improved by the proactive reserve on the open environments. On average, the idle agents are closer to their frontiers when they are turned into active state and this helps in reducing the availability distance. Finally, the decision time is

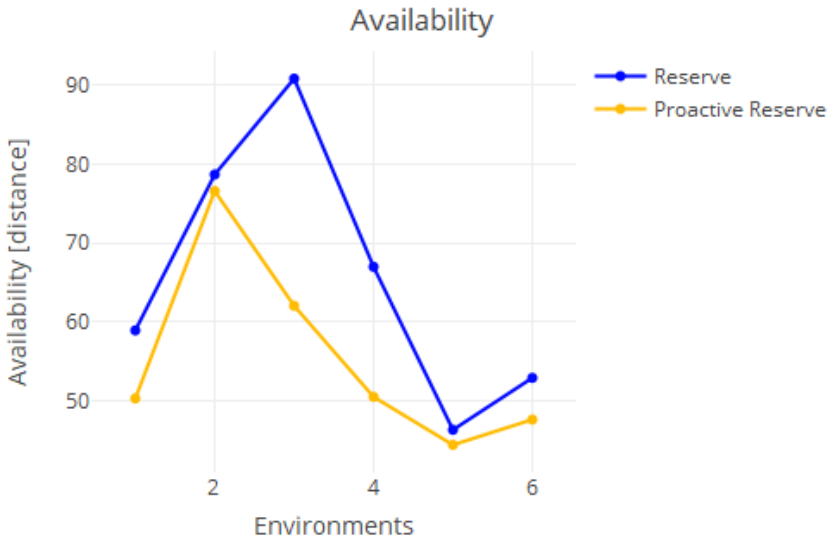


Figure 6.9: Proactive reserve vs. reserve availability.

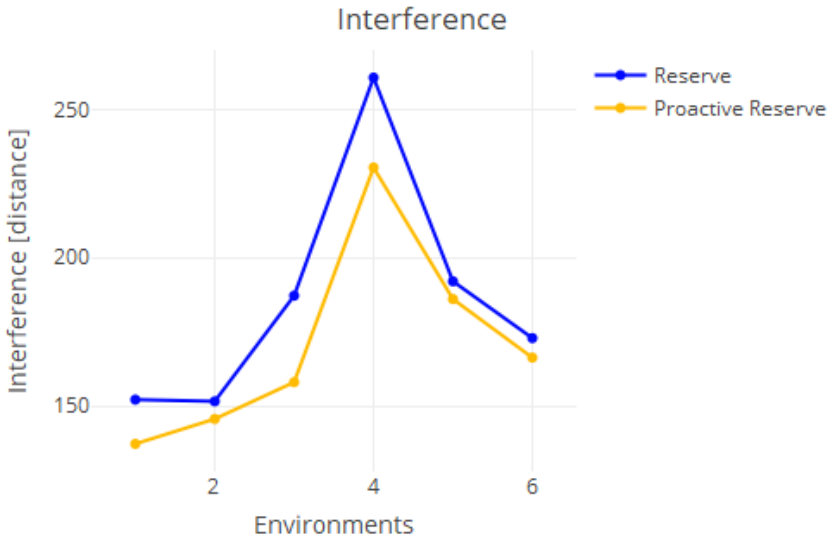


Figure 6.10: Proactive reserve vs. reserve interference.

depicted in Figure 6.11.

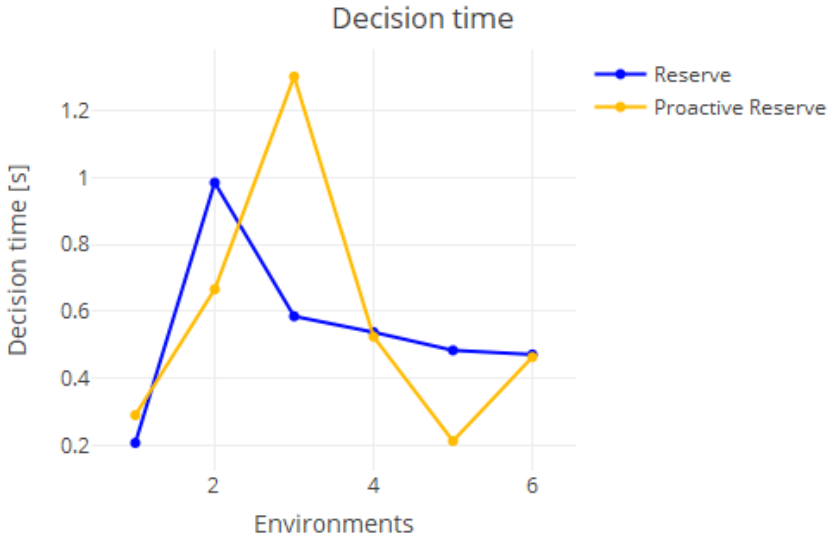


Figure 6.11: Proactive reserve vs. reserve decision time.

6.2.2 Proactive Buddy System

The proactive buddy system, as the previously analyzed proactive reserve, is aimed at improving one of the benchmark mechanisms. In particular, the proactive buddy system is expected to decrease the average availability distance resulted from the buddy system mechanism. In Figure **??** and **??**, the exploration time obtained by the proactive buddy mechanism show how the proactive version outperforms the benchmark one mostly in the open environments. As explained for the proactive reserve case, the freagmentation of the environments force the reserve agents (in this case, the reserve pairs) to be activated in the early stages of the exploration. The proactive modification, thus, does not help very much the exploration time on fragmented environment, as they do on open ones. These considerations can be seen especially in Figure 6.12 and in Figure 6.12 which represent fragemented environments. The buddy system already managed the availability of the agents as explained in the previous chapter. Thus, the proactivity added here, improves further the availability distance but the effects are not particularly evident. The proactive buddy system is shown to be worse, in general, than the proactive reserve. This can be due to the availability metric that, in the proactive reserve mechanism, is reduced without deteriorating the interference measure. On the other hand, the interference of the proactive buddy mechanism is higher because in the initial steps the agents move next to each other, being split in pairs instead of single units. Thus, considering the interference and availability measure of the two proactive mechanisms, the proactive buddy is expected to decrease the availability metric

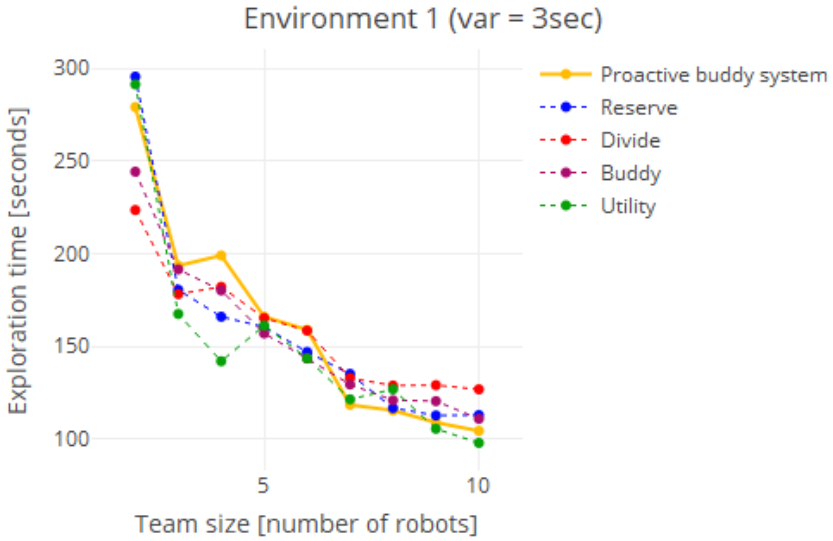


Figure 6.12: Proactive buddy exploration time on environment 1.

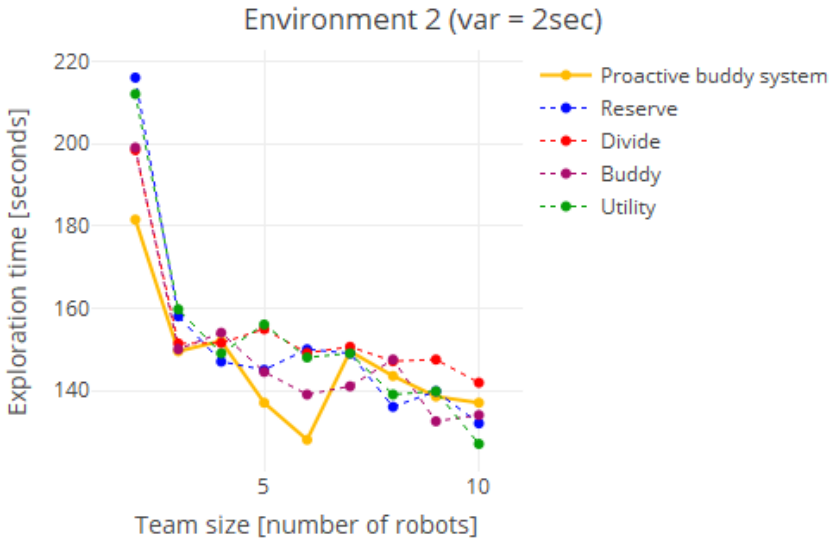


Figure 6.13: Proactive buddy exploration time on environment 2.

but to decrease also the interference measure (the proactive buddy maintains the agents closer to each other). The metrics computed during the experiments and presented in Figure 6.18 and in Figure 6.19 confirm what assumed. From

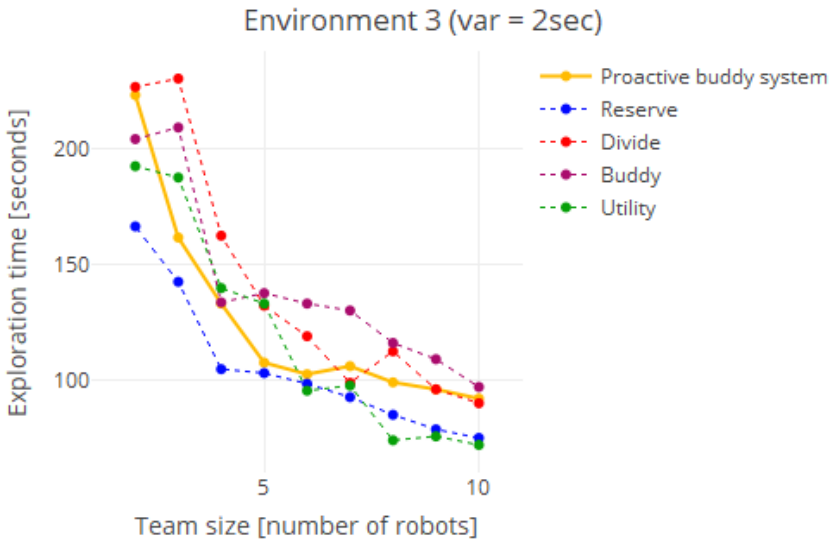


Figure 6.14: Proactive buddy exploration time on environment 3.

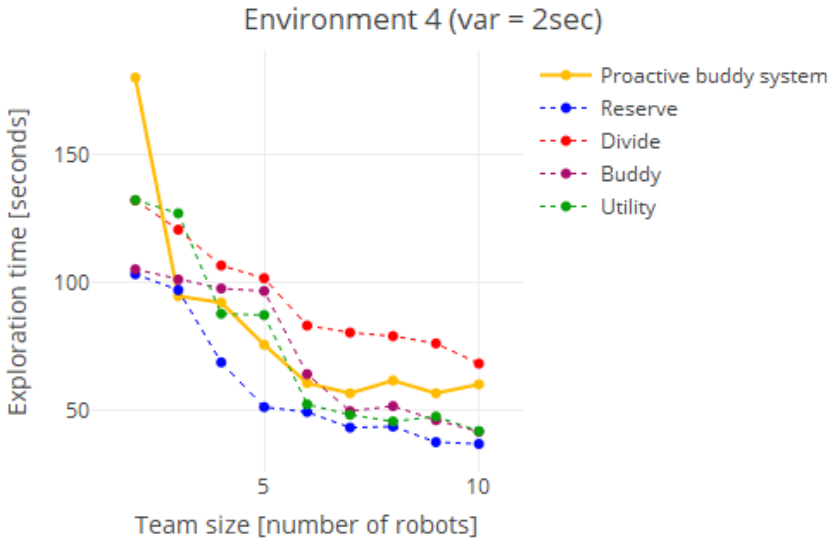


Figure 6.15: Proactive buddy exploration time on environment 4.

what seen in the results up to now, the interference measure seems affecting more the performance of a mechanism, compared to the availability. In other words, spreading the agents appears more impacting than assigning them to

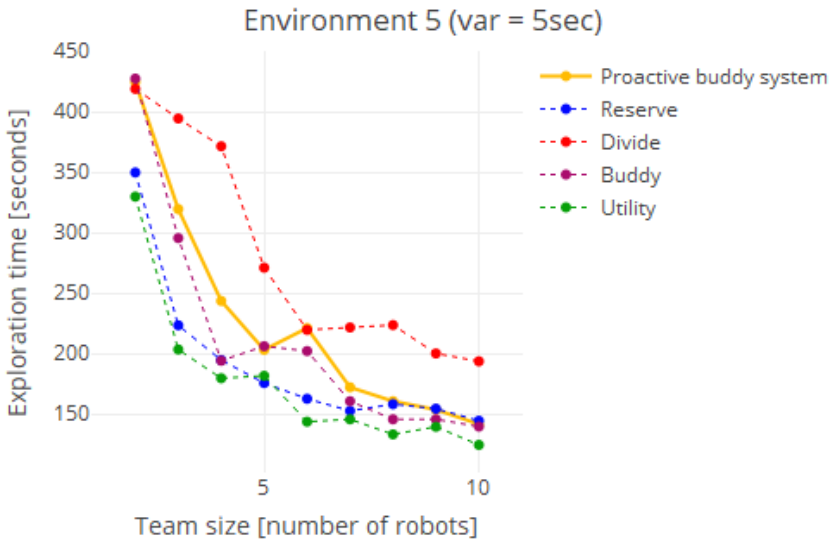


Figure 6.16: Proactive buddy exploration time on environment 5.

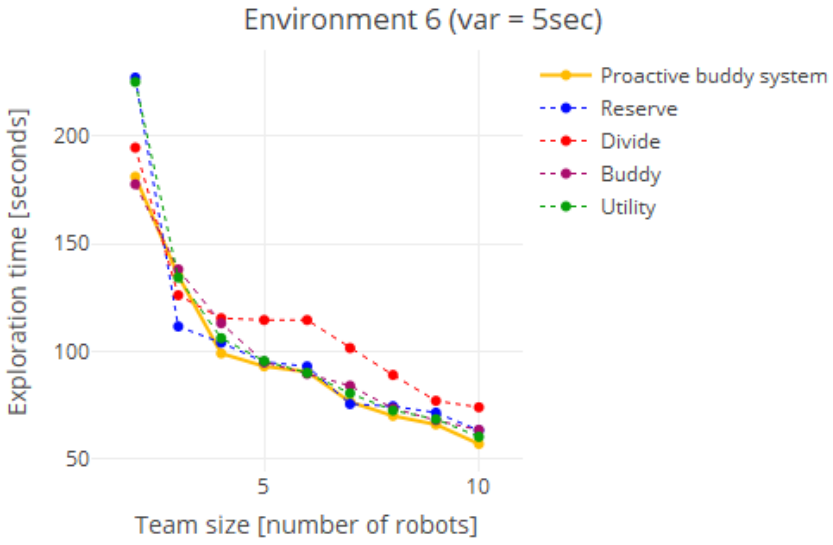


Figure 6.17: Proactive buddy exploration time on environment 6.

close target locations, at each step of the exploration.

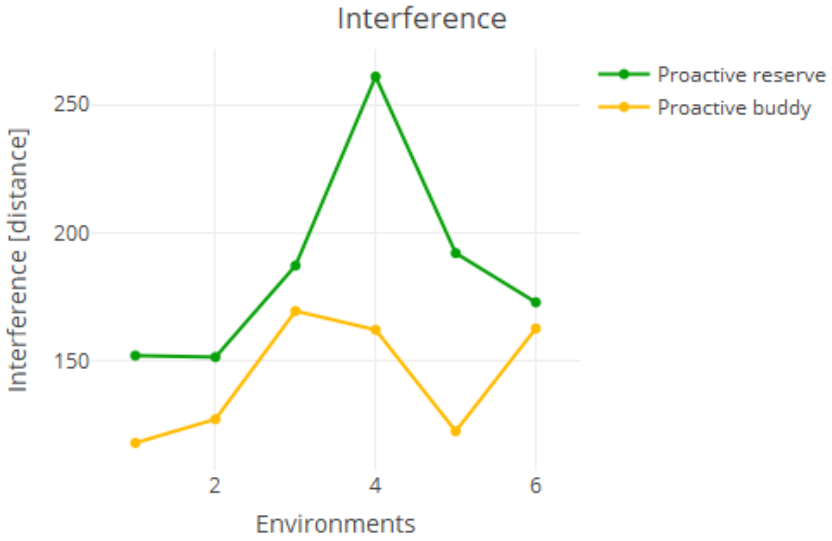


Figure 6.18: Proactive buddy vs. proactive reserve interference.

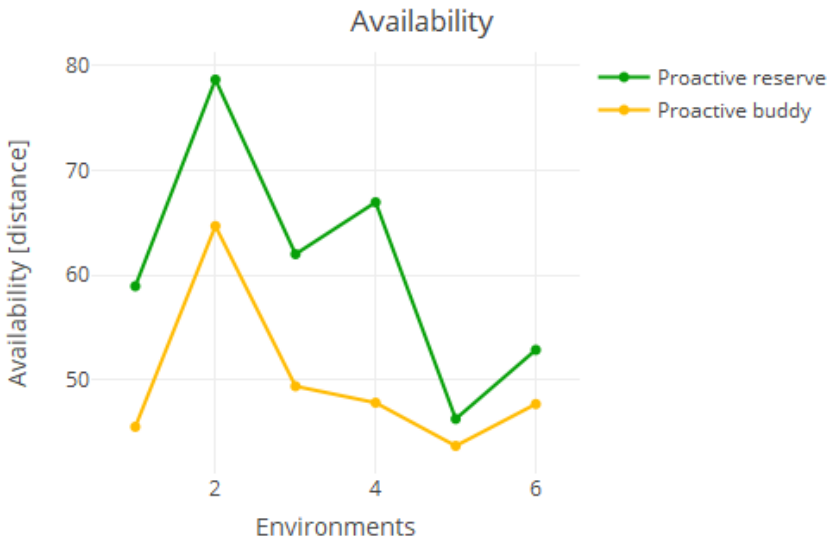


Figure 6.19: Proactive buddy vs. proactive reserve availability.

6.2.3 Side Follower

The side follower mechanism is aimed at leveraging particular features of the environments. The mechanism, indeed, is expected to perform well on environments that uncover the frontiers in predefined directions. Being the robots assigned to straight, left and right frontiers, the spaces that are organized around a central corridor are privileged by this mechanism. Such an environment is close to the first and third environments provided in this chapter. The results

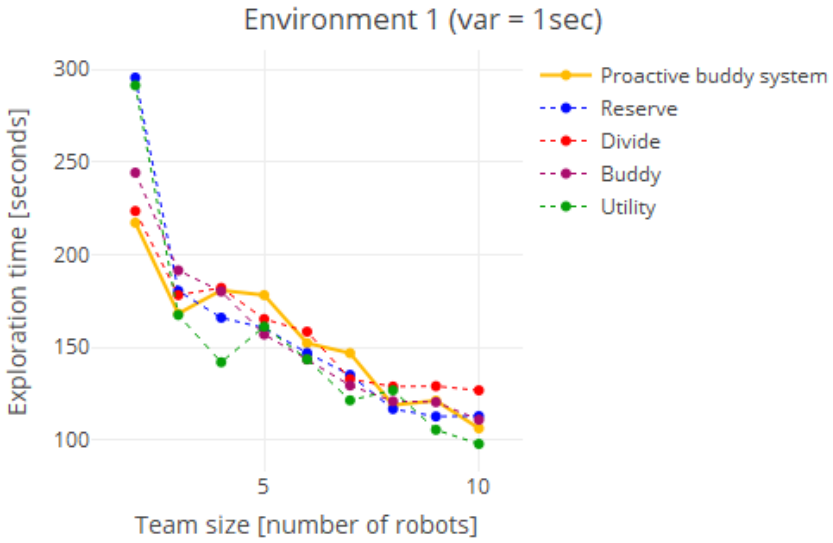


Figure 6.20: Side follower exploration time on environment 1.

in Figure 6.20, Figure 6.21, and Figure 6.22 are related to the small environments. Of particular interest is to compare the performance of the side follower mechanism with the one of the divide and conquer algorithm, since they share the same grouping principle. As introduced, the third environments is the one on which the side follower performed significantly better than the divide and conquer, obtaining similar results with respect to the reserve and proactive reserve mechanisms (that are the best mechanisms presented so far on the third space). The third environment, indeed, is very close to the ideal situation for this mechanism. The six rooms of the bunker expose their entry points around the central, large, corridor. The leader agent of each group will likely be initially assigned to the central corridor while spreading the followers to left and right rooms. The mechanism, probably, did not perform better than the reserve and proactive reserve mechanisms due to imprecise allocations of leader agents. The size of the central corridor, indeed, leads the agents (which should follow a straight line) to uncover more than one frontier inside the corridor itself. This brings the leaders to change direction while exploring the central part, increas-

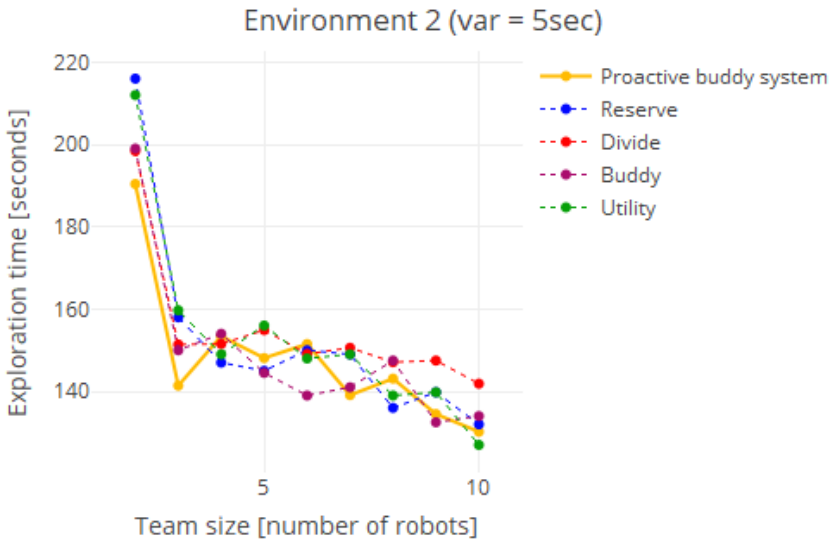


Figure 6.21: Side follower exploration time on environment 2.

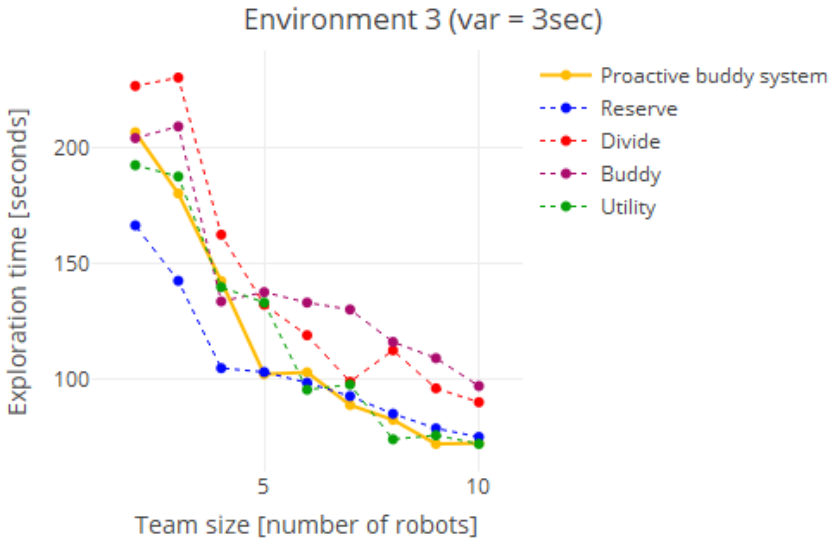


Figure 6.22: Side follower exploration time on environment 3.

ing the probability of being assigned to one of the rooms at some point of the exploration. The divide and conquer grouping mechanism has already been presented as lacking on almost all the environments. As a consequence, the side

follower modifies the grouping logic by never splitting the groups. The results show how this choice make it possible to outperform the divide mechanism in many different spaces. The exploration time of the side follower on large en-

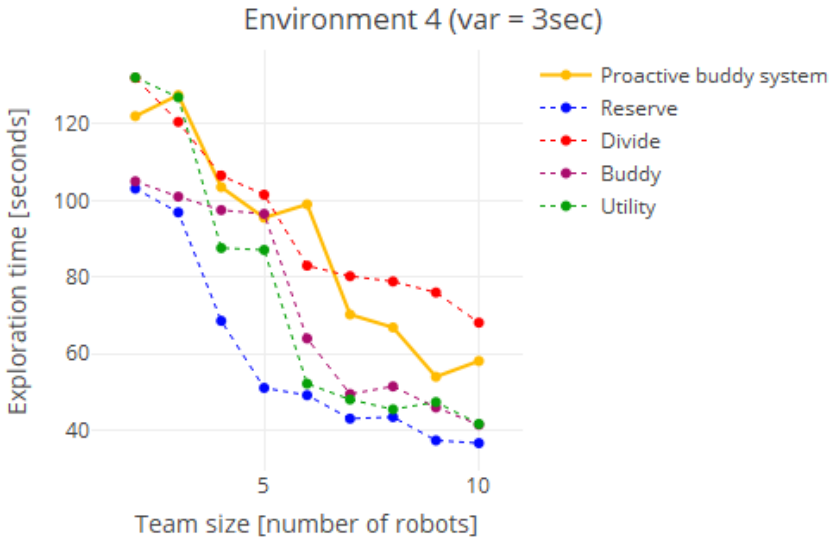


Figure 6.23: Side follower exploration time on environment 4.

vironments is shown in Figure Figure 6.23, Figure 6.24, and Figure 6.25. The outcome of the experiments over the fifth environment should be noticed. Even thought the environment structure is organized in different corridors, which reveal rooms in many directions, the side follower is capable of outperforming the divide and conquer mechanism on this HP space. This is because the side follower generalizes better over the different environments types, compared to the divide approach. Even if the environment is not ideal for the mechanism, indeed, the side follower never splits the groups, coordinating them from the beginning to the end of the exploration. The results in terms of interference and availability are provided in Figure 6.26 and in Figure 6.27. The side follower mechanism is expected to improve the management of the interference (over the divide and conquer) in the initial steps, while resulting worst in the later stages. After a while, indeed, the agents driven by the divide mechanism are independent and, possibly, spread over the workspace. On the other hand, the side follower shows a smaller availability measure, in particular in open environments. The interference of this mechanism is closer to the one of the divide in the third environment and is even better in the fifth. The situation is reverse in the availability on those environments, that are the environments on which the side follower outperformed the divide and conquer mechanism.

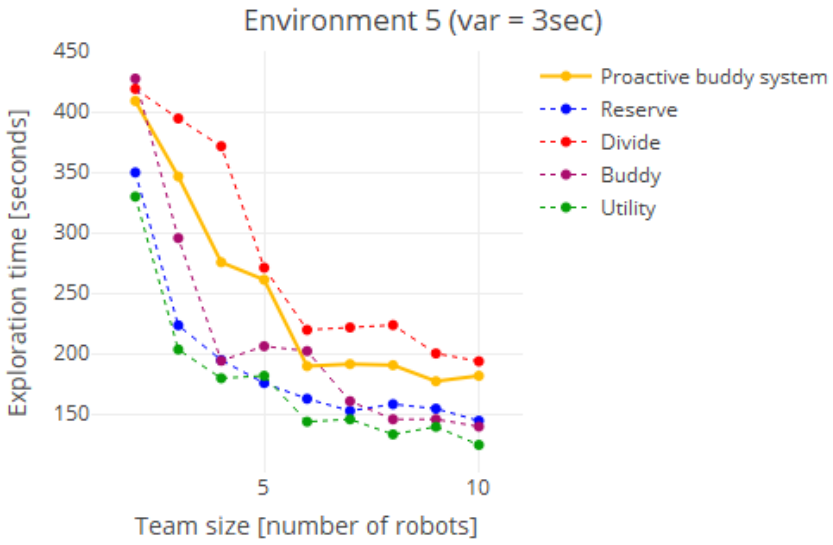


Figure 6.24: Side follower exploration time on environment 5.

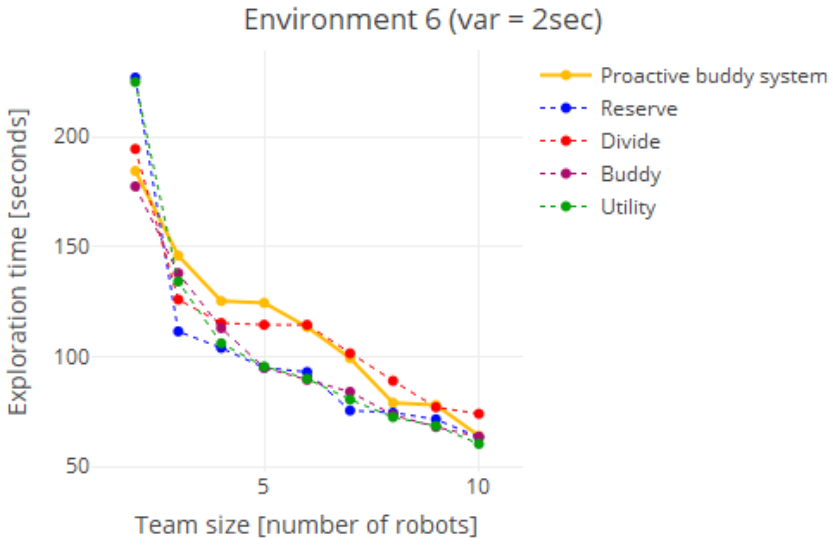


Figure 6.25: Side follower exploration time on environment 6.

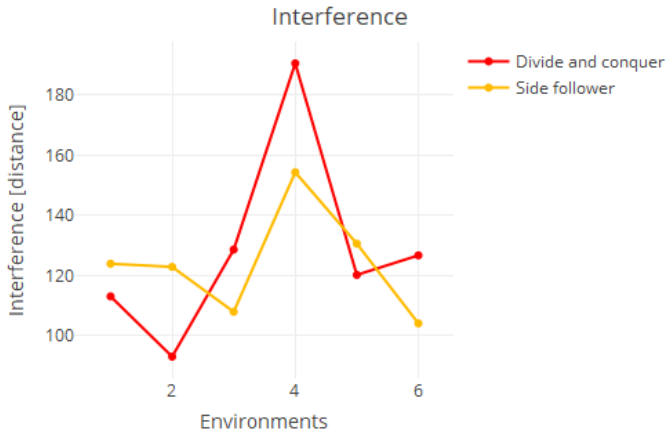


Figure 6.26: Side follower vs. divide and conquer interference.

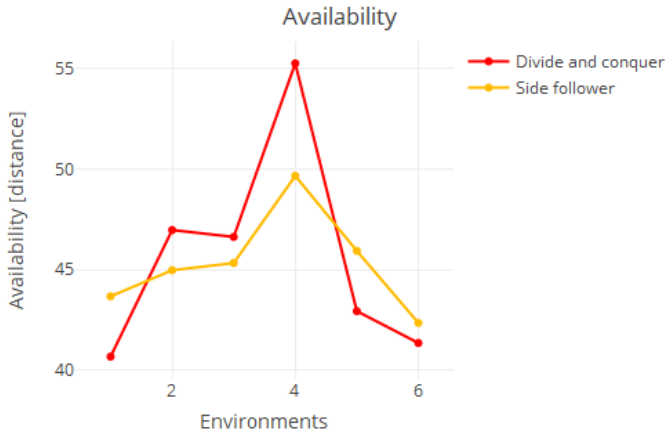
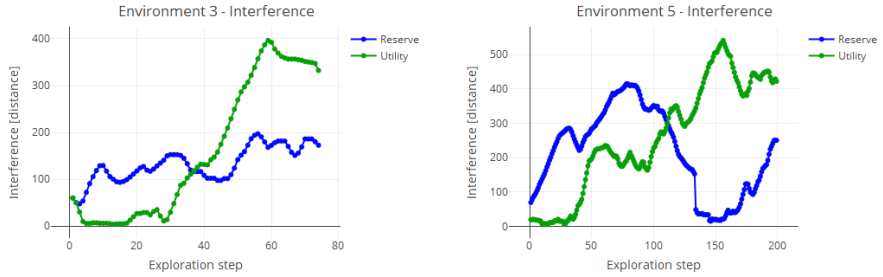


Figure 6.27: Side follower vs. divide and conquer availability.

6.2.4 Direct Optimization

The direct optimization mechanism combines online and offline mechanisms. The offline mechanisms are expected to perform better in the initial steps of the exploration, while lacking coordination in the later stages. Let's compare, for instance, the interference measures of the reserve and utility based benchmark mechanisms in Figure 6.28. The diagrams show the measures for two sample environments (the third, which is S-O-LP and the fifth, which is L-C-HP) in time and are averaged over the number of experiments and agents. The online utility based mechanism maintains a lower interference distance (i.e., agents are



(a) Utility based vs. reserve interference in environment 3.

(b) Utility based vs. reserve interference in environment 5.

Figure 6.28: Utility based vs. reserve interference in time.

close to each other) in the initial step of the exploration, compared to the reserve (offline) mechanism. The situation, instead, is reversed in later stages, when the reserve mechanism leave agents to explore independently. The basic idea of the

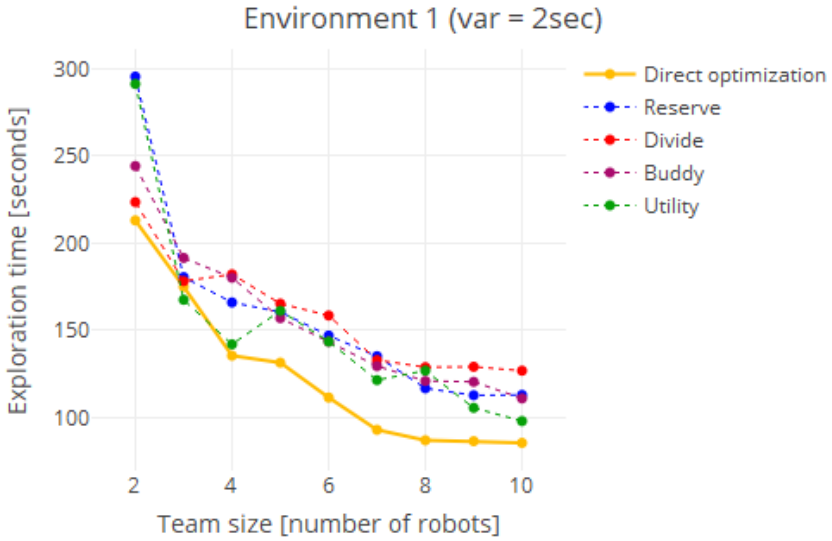


Figure 6.29: Direct optimization exploration time on environment 1.

direct optimization is to leverage the benefits of both online and offline mechanism by combining the proactive reserve approach (the best among the tested offline mechanisms) and the utility based mechanism. The results show how the direct optimization perform, in general, well compared to the benchmark mechanisms. Its performances are, in some cases, very close to the reserve and proactive reserve results. This means that the reserve mechanism is efficient in

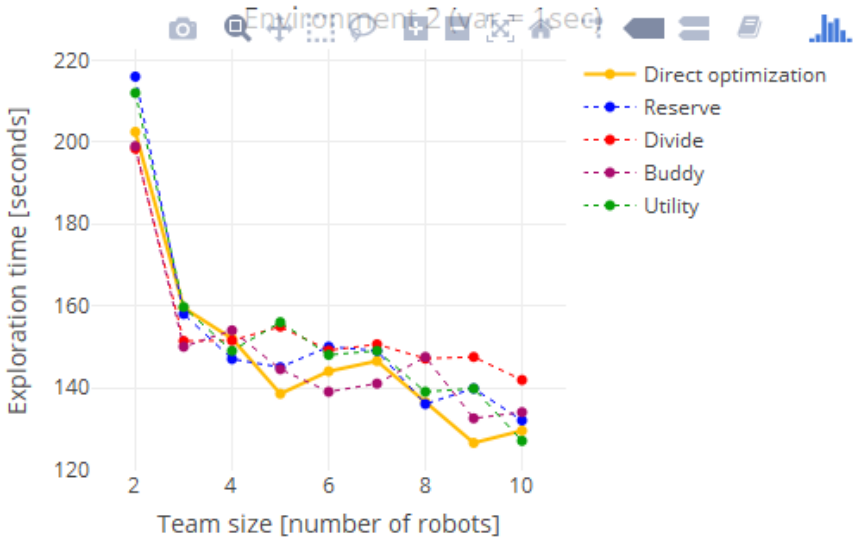


Figure 6.30: Direct optimization exploration time on environment 2.

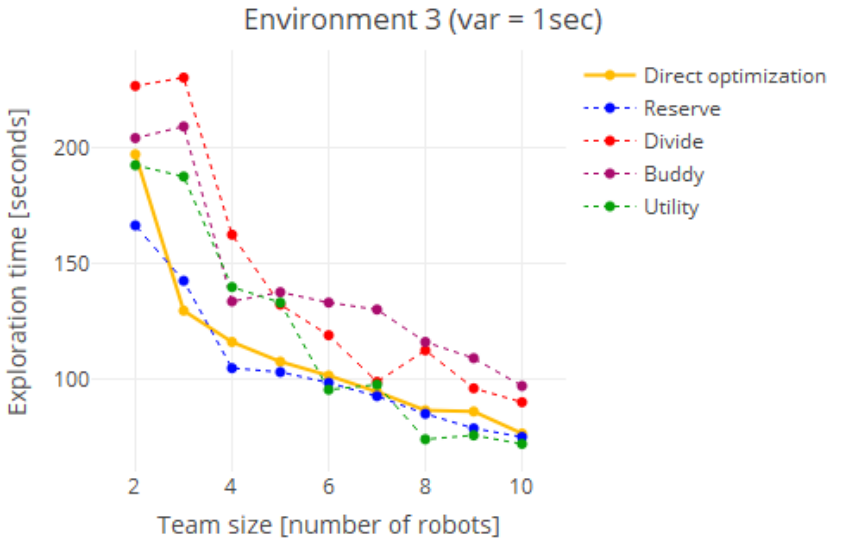


Figure 6.31: Direct optimization exploration time on environment 3.

spreading the agents over the environments and so is its proactive version. However, the improvement in the interference in the later stages of the exploration, allows the agents driven by the direct mechanism to decrease their overall explo-

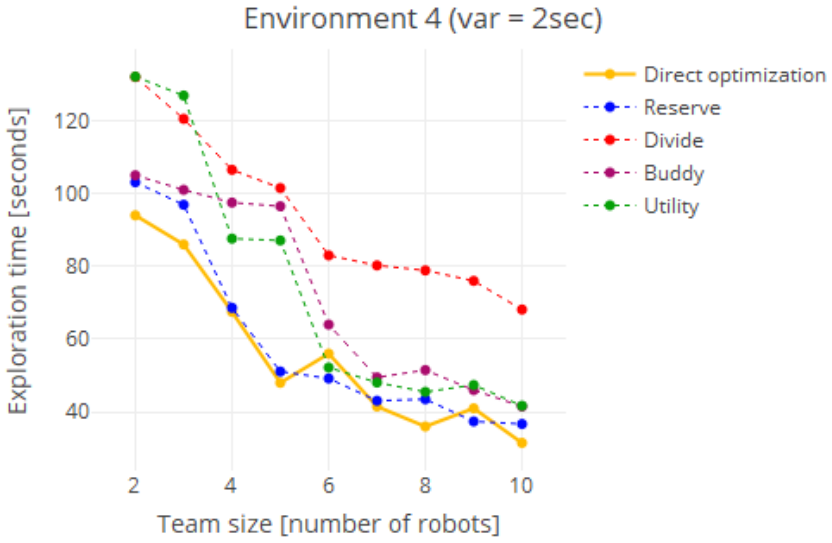


Figure 6.32: Direct optimization exploration time on environment 4.

ration time, in particular over cluttered environments (like the first one). As

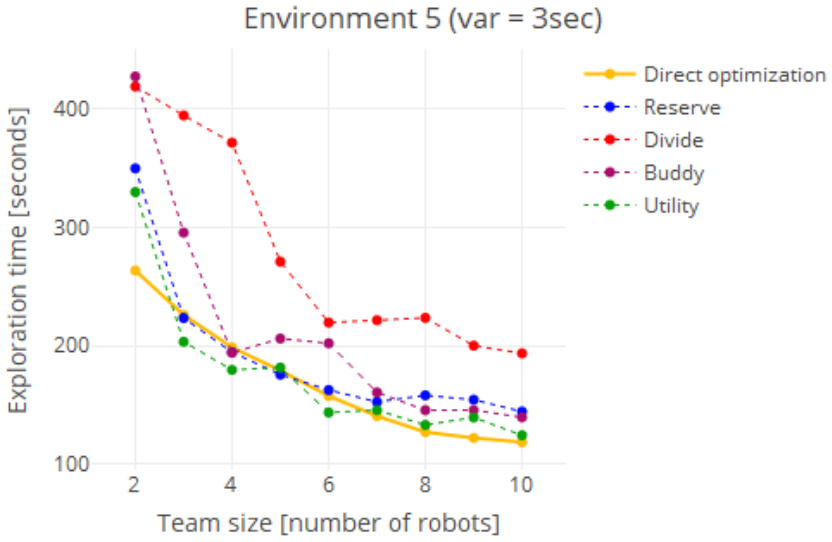


Figure 6.33: Direct optimization exploration time on environment 5.

already introduced, indeed, the problem in the management of the interference

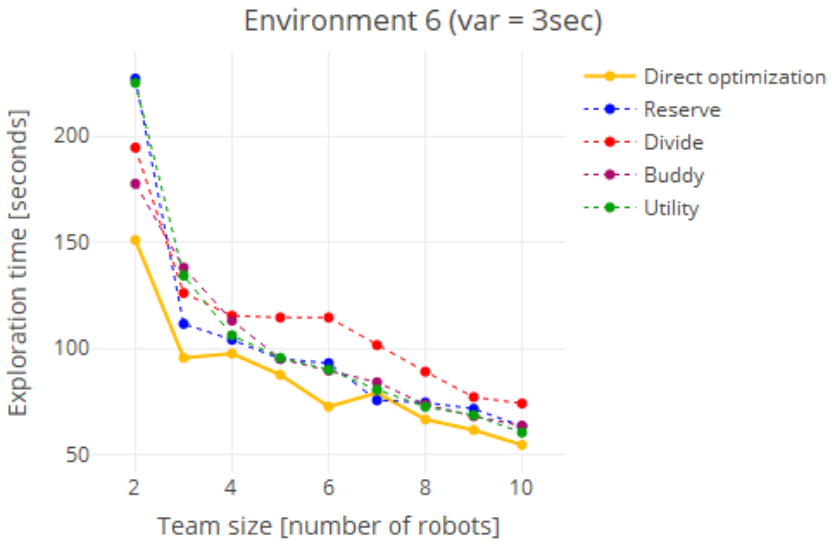


Figure 6.34: Direct optimization exploration time on environment 6.

affects the offline mechanisms mostly over cluttered environments, which force the agents to abandon their roles early in the exploration. The last diagrams

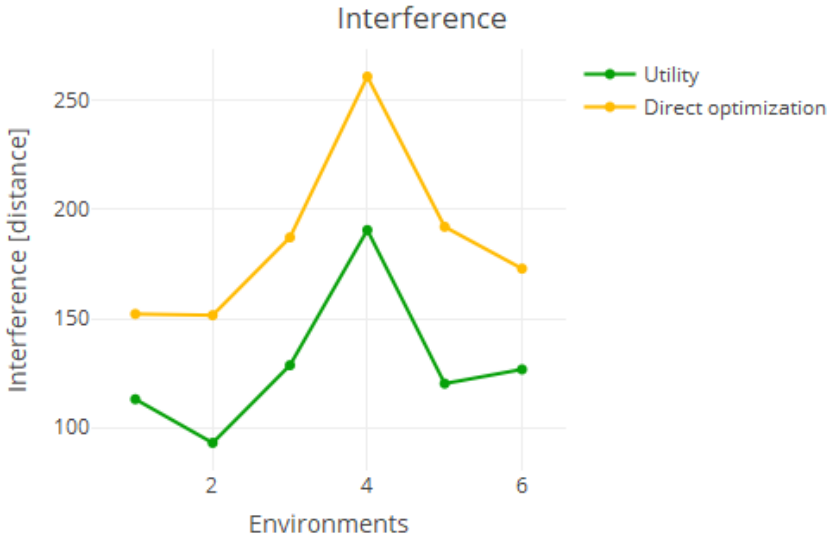


Figure 6.35: Direct optimization vs. utility based interference.

(in Figure 6.35,Figure 6.37, and in Figure 6.36), finally, show the interference, availability and decision time measures of the direct approach, comparing them with the ones obtained by the utility mechanism. The results show, increasing

Figure 6.36: Direct optimization vs. utility based decision time.

of a little amount the decision time, the direct optimization improves the interference measure for all the environments. However, the availability distance is higher for the direct approach, meaning that agents are allocated to far frontiers, on average. This is probably due to its initial offline component. In any case, also here emerges the greater impact of the interference over the availability in determining the overall exploration time.

6.2.5 Mechanisms comparison

Summarizing the results obtained by the original mechanisms proposed in this thesis, the combination of online and offline approaches deployed in the direct optimization mechanism seem to outperform the other mechanisms on almost all the environments. An exception is represented by the third and fourth environment that, being open, allow offline mechanisms to behave similarly to the direct optimization approach. The side follower, offline, mechanism gives a significant contribution only in environments that present a particular structure: it, indeed, has a particularly good performance over the third environment that is well organized around its central corridor. However, the side follower mechanism appear to specific to be generalized over the others environment types. The proactive reserve gets very close to the direct optimization in open environments, more than how the proactive buddy does. This can be due to the

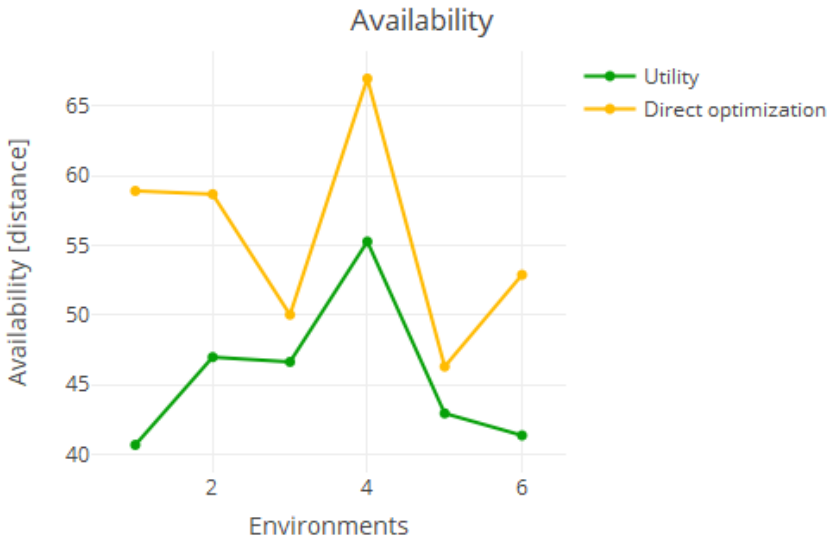


Figure 6.37: Direct optimization vs. utility based availability.

initial difference of the performance of their plain versions, the reserve and the buddy system. The buddy system approach tends to have a lower interference distance, due to the closeness of the pairs in the initial stages of the exploration while improving the availability. The reason for the reserve to outperform the buddy, then, can be that the interference measure has a greater impact on the availability metric in terms of exploration time.

Chapter 7

Conclusions

The aim of this thesis is twofold: propose a general framework to describe the coordination of multiple robots and test different coordination mechanisms under the framework proposed. In particular, the framework is used to evaluate some aspects of the mechanisms a priori, before running experiments. The coordination mechanisms are both online and offline.

In this chapter, we present the conclusions of the work, giving a final analysis of the different mechanisms and evaluating the effectiveness of the framework in describing the coordination. The first section summarizes the criteria adopted to analyze the performance of the mechanisms with the experiments run in MRESim and their results. The second section evaluates the a priori assumptions made in Chapter 5 on the possible behaviour of the mechanisms and derives the final considerations about the goodness of the framework proposed.

7.1 Mechanisms comparison

Our original mechanisms have been compared with the benchmark ones in the experimental activity described in Chapter 6. The comparison has been driven by four main elements: the exploration time, the interference, the availability (as defined in Chapter 4), and the decision time. The exploration time is considered as the main indicator to analyze the results derived from the experiments. Interference and availability, instead, are intended as a way to explain the results generated and to corroborate the qualitative discussion in Chapter 5. Indeed, the analysis of the possible interference and availability during the exploration can be made, with relative ease, before testing the mechanisms, only basing on their algorithmic structure. Finally, the decision time indicates the time taken by the mechanisms to allocate all the agents (at each step of the exploration) and provides a practical estimate of the complexity of the coordination algorithms implemented.

The experiments have been performed over six 2D environments, chosen to represent the different types of spaces introduced in Chapter 3. We evaluated

the results according to the specific mechanism-environment-team configuration triple to which they relate. Since runs in the environments are non-deterministic, MRESim experiments are not guaranteed to provide the same results for different runs of the same configuration triple. For this reason, the experiments are run multiple times with the same mechanism-environment- team configuration and the results are averaged over the instances.

From the results obtained, the decision time revealed to not be a factor since all the mechanisms performed similarly in that sense. The average decision time is established around one second (taken to allocate all the robots), with a minimum variance. The exploration time, instead, showed how the best mechanisms have been the proactive reserve and the direct optimization. While the proactive reserve is fully offline, the direct optimization combines offline and online mechanisms. They have been both capable of outperforming their benchmark versions in the different environments. The proactive reserve increased the availability (decreasing the average allocation distance) of the agents without negatively affecting the interference measure. This has made possible to overcome the main issue related to the reserve mechanism. The direct optimization, instead, tackled the problem that the utility based mechanism had in evaluating the initial allocations. The utility based mechanism, and in general the online coordination mechanisms, suffer the initial conditions of the exploration, in which the utility matrix of the agent-location pairs is set to its initial values. The agents have not enough knowledge of the environment to properly update the matrix until some allocations have been made. This is the main difference between online and offline mechanisms, which, instead, perform well in the initial steps of the exploration and tend to deteriorate their performance as the exploration proceeds towards the final stages.

In general, the offline mechanisms appeared better on open environments, where the spread of the robots is favored by the many directions available during the exploration. The boundary region on which the frontiers are allocated, indeed, expand concentrically over the environment, centered in the agents poses. This allows the offline mechanisms to spread the agents without having the online control of the coordination. On the other hand, the online approaches, after the early stages of the exploration, performed better on fragmented environments which seemed to require an online control to react to the frequent changes in the available frontiers. The frontiers discovered over fragmented environments, indeed, are usually discovered either in constrained directions (in the case of corridors) or scattered over the environment (if the obstacles are randomly positioned or there are many rooms).

7.2 Model evaluation

The coordination framework proposed is mainly characterized by three elements: the state of the exploration, the coordination mechanism, and the evaluation criteria. The state of the exploration represents the information that the agents collected of the environment. It is composed of: the currently known grid, the

agents poses, and their goals. The coordination mechanism is seen as a function that allocates the robots to the available frontiers, computed on the basis of the known grid. Finally, the evaluation criteria is intended as a combination of interference and availability, two measures formalized to characterize and evaluate the mechanisms a priori, without the need for experiments.

The aim of the framework is to describe, through a common model, different types of coordination mechanisms, assuming the communication among the team members to be always possible. A specific coordination mechanism is intended as an instance of a generic algorithm. An instance is generated through the definition of the *computeUtility()* and *handleFreeLocations()* functions, which characterized the way in which the mechanism allocates the agents to the available frontiers.

Once defined a general model for a coordination mechanism, the different instances of the model are easier to compare. To compare two mechanisms, the exploration time is normally considered as the main factor to analyze. However, the overall exploration time is shown to be difficult to estimate without running a large amount of experiments. For this reason, the measures of availability and interference are introduced. The availability measures estimates the average distance that the agents have to travel to reach their assigned locations, while the interference measure is defined as the average distance among the agents and indicates how much the team members are spread on the environment. These measures are strictly related to the structure of the coordination mechanism, in the framework proposed. This makes it possible to evaluate the behaviour of a mechanism a priori, knowing only the type of environment to be explored and the structure of the coordination mechanism.

The results of the mechanisms in terms of interference and availability confirmed that the two measures, together, can provide a rough estimate of the real exploration time. However, mechanisms with the lower availability distance and the higher interference measure not always resulted the ones with the lower exploration time. This means that the two measures are not sufficient to provide a clear view of the expected exploration time because they cannot capture all the dynamics of the mechanisms.

The model proposed for the mechanisms seem to generalize several types of coordination mechanisms, from online to offline settings. However, the representation of the allocation function, based on a utility matrix internally maintained by the mechanism, appear more suitable for online mechanisms. In many cases these mechanisms rely on the estimate of the utilities of the available frontiers and lend themselves to be represented by means of the *computeUtility()* and *handleFreeLocations()* functions. The offline mechanisms, on the other hand, are based on the definition of roles and do not refer directly to the utility of the frontiers discovered. Despite this, different offline mechanisms have been proposed under the model introduced, which appeared versatile enough to be adapted also at these cases.

7.3 Future work

Future work can be developed in different directions from what presented in this thesis. Several mechanisms can be implemented instantiating the model proposed. Of particular interest could be to investigate the mechanisms that try to allocate the robots taking directly into account the interference and availability measures. An example of such a mechanism would allocate each robot by minimizing the overall interference and availability of the entire team, at each step of the exploration. In this sense, the relative effects of interference and availability should be studied in depth and assessed to understand if dominating classes of mechanisms could be individuated.

Moreover, the framework proposed could be enhanced to provide a more clear definition of offline mechanisms or to consider the communication requirements of the team members. This thesis, indeed, assumes the communication to be always available and in many cases this is not true in real world scenarios.

Bibliography

[1] S. Albers and M. R. Henzinger. Exploring unknown environments. *SIAM Journal on Computing*, 29(4):1164–1188, 2000.

[2] F. Amigoni. Experimental Evaluation of Some Exploration Strategies for Mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2818–2823, 2008.

[3] F. Amigoni and V. Caglioti. An information-based exploration strategy for environment mapping with mobile robots. *Robotics and Autonomous Systems*, 58(5):684–699, 2010.

[4] F. Amigoni and A. Gallo. A multi-objective exploration strategy for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3861–3866, 2005.

[5] Basilico N. Amigoni F. and Quattrini Li A. Moving from how to go there? to where to go?: Towards increased autonomy of mobile robots. *New Trends in Medical and Service Robots. Mechanisms and Machine Science*, 20:345–356, 2014.

[6] M. Betke B. Awerbuch and R. Rivest. Piecemeal graph exploration by a mobile robot. In *Proceedings of the 8th Annual ACM Conference on Computational Learning Theory (COLT)*, pages 374–386, 1995.

[7] T. Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (SLAM): Part i. *IEEE Robotics and Automation Magazine*, 13(2):99–108, 2006.

[8] T. Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (SLAM): Part ii. *IEEE Robotics and Automation Magazine*, 13(3):108–117, 2006.

[9] M. Bender and D. Ron. The power of a pebble: exploring and mapping directed graphs. In *Proceedings of the 30th annual ACM symposium on Theory of Computing (STOC)*, pages 269–278, 1998.

[10] M. A. Bender and D. K. Slonim. The power of team exploration: Two robots can learn unlabeled directed graphs. pages 75–85, 1994.

[11] A. Giannitrapani D. Benedettelli, A. Garulli. Cooperative SLAM using M-Space Representation of Linear Features. *Robotics and Autonomous Systems*, 60:1267–1278, 2012.

[12] J. Ko D. Fox and K. Konolige. Distributed multirobot exploration and mapping. *IEEE special issue on multi-robot systems*, 94(7):1325–1339, 2006.

[13] A. Cheyer D. Guzzoni and L. Julia. Many robots make short work. *Artificial Intelligence Magazine*, 18(1):55–64, 1997.

[14] R.P. Bonasso D. Kortenkamp and R. Murphy. AI-based mobile robots: Case studies of successful robot systems. *MIT Press*, 1998.

[15] I. Bogoslavskyib D. P. Strom and C.Stachniss. Robust exploration and homing for autonomous robots. *Journal of Robotics and Autonomous Systems*, 20, 2015.

[16] F. Dellaert and S. Thrun. Square Root SAM. *The Internation Journal of Robotics Research*, 25(12):1181–1203, 2006.

[17] A. Dessmark and A. Pelc. Optimal graph exploration without good maps. In *Proceedings of the European Symposium on Algorithms (ESA)*, pages 374–386, 2002.

[18] J. Stromand R. Morton E. Olson. Progress towards multi-robot reconnaissance and the magic 2010 competition. *Journal of Field Robotics*, 29(5):762–792, 2012.

[19] Y. Cao et al. Cooperative mobile robotics: antecedents and directions. *Autonomous Robots*, 4(1):7–27, 1997.

[20] N. Basilico F. Amigoni and A.Q. Li. How much worth is coordination of mobile robots for exploration in search and rescue? *RoboCup 2012: Robot Soccer World Cup XVI*, 2013.

[21] J. Folkesson and H. Christensen. Graphical SLAM - A selfcorrecting map. In *Proceedings of the International conference on robotics and automation (2004)*, volume 1, pages 383–390, 2004.

[22] L. Freda and G. Oriolo. Frontier-based probabilistic strategies for sensor-based exploration. In *Proceedings of the IEEE International Conference on Robotics and Automation (2005)*, pages 680–687, 2005.

[23] P. Newman G. Dissanayake and S. Clark. A solution to the simultaneous localisation and map building (SLAM) problem. *IEEE Transactions of Robotics and Automation*, 17(3):229–241, 2001.

[24] E. Milos G. Dudek, M. Jenkin. Robotic Exploration as Graph Construction. *IEEE Transactions on Robotics*, vol. 7(6):859–865, 1991.

[25] C. Stachniss G. Grisetti and W. Burgard. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2432–2437, 2005.

[26] B. P. Gerkey and M. J. Mataric. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23(9):939–954, 2014.

[27] M. Golfarelli and S.Rizzi. Solving open polygonals in elastic correction of dead-reckoning errors. In *Proceedings of the IEEE/RJS Internation Conference on Intelligent Robots and Systems (IROS)*, pages 905–911, 1998.

[28] H. H. Gonzalez-Banos and J. C. Latombe. Navigation strategies for exploring indoor environments. *The International Journal of Robotics Research*, 21:829–848, 2002.

[29] C. Nieto-Granda H. Christensesn and J.G. Rogers III. Coordination strategies for multi-robot exploration and mapping. *The International Journal of Robotics Research*, 33(4):519–533, 2014.

[30] L. Brunet H. L. Choi and J. P. How. Consensus-based decentralized auctions for robust task allocation. *IEEE Transactions on Robotics*, 25(4):912–926, 2009.

[31] F. Hoffmann. One pebble does not suffice to search plane labyrinths. *Fundamentals of Computation Theory*, 117:433–444, 1981.

[32] Singh S. Hollinger G. and Kehagias A. Improving the efficiency of clearing with multi-agent teams. *International Journal of Robotics Research*, 29(8):1088–1105, 2010.

[33] H.Wang and P. Dymond. Enhancing exploration in graphlike worlds. In *Proceedings of the Canadian Conference on Computer and Robot Vision*, pages 53–60, 2008.

[34] H. Christensen J. Folkesson, P. Jensfelt. The M-Space Feature Representation for SLAM. *IEEE Transactions on Robotics*, 23:1024–1035, 2007.

[35] S. Russell K. Murphy. Rao-Blackwellized Particle Filtering for Dynamic Bayesian Networks. *Sequential Montecarlo Methods in Practice*, pages 499–516, 2001.

[36] M. Keidar and Gal A. Kaminka. Robot exploration with fast frontier detection: Theory and experiments. In *Proceedings of the International conference on autonomous agents and multiagent systems (2012)*, 2012.

[37] H. Lau and A. NSW. Behavioural approach for multi-robot exploration. In *Proceedings of the Australasian Conference on Robotics and Automation, December, 2003*, 2003.

[38] D. Lee and M. Recce. Quantitative evaluation of the exploration strategies of a mobile robot. *International Journal of Robotics Research*, 16(4):413–447, 1997.

[39] F. Lu and E. Milios. Globally Consistent Range Scan Allignment for Environment Mapping. *Autonomous Robot*, 4:333–349, 1997.

[40] D. Koller M. Montemerlo, S. Thrun. A Factored Solution to the Simultaneous Localisation and Mapping Problem. In *Proceedings of the National Conference on Artificial Intelligence (Edmonton 2002)*, volume 37, pages 81–91, 2002.

[41] P. Newman. On the structure and solution of the simultaneous localisation and map building problem. *PhD thesis, Univ. of Sydney*, 2000.

[42] J. Xiao P. Brass, A. Gasparri. Multirobot Tree and Graph Exploration. *IEEE Transactions on Robotics*, 27(4):707–716, 2011.

[43] L. Gasieniec P. Fraigniaud and D. Kowalski. Collective tree exploration. *Networks*, vol. 48:166–177, 2006.

[44] P. Panaite and A. Pelc. Impact of topographic information on graph exploration efficiency. *Networks*, vol. 36:96–103, 2000.

[45] L. E. Parker. Alliance: An architecture for fault-tolerant multi-robot cooperation. *IEEE Transactions on Robotics*, 14(2):220–240, 1998.

[46] K. M. Krishna R. Sawhney. On fast exploration in 2d and 3d terrains with multiple robots. In *Proceedings of the International conference on autonomous agents and multiagent systems (2012)*, pages 73–80, 2009.

[47] M. Self R. Smith and P. Cheeseman. Estimating uncertain spatial relationships in robotics. *Autonomous Robot Vehnicles*, 1990.

[48] D. Fox R. Vincent and J. Kol. Distributed multirobot exploration, mapping, and task allocation. *Annals of Mathematics and Artificial Intelligence*, 52(2):229–255, 2008.

[49] Wolfram Burgard Reid Simmons. Coordination for multi-robot exploration and mapping. In *Proceedings of the National Conference on Artificial Intelligence (2000)*, volume 4, pages 7–27, 2000.

[50] W. Burgard S. Thrun and D. Fox. Probabilistic Robotics. *The MIT Press*, 2005.

[51] R. Sim and N. Roy. Global a-optimal robot exploration in slam. In *Proceedings of the IEEE International Conference on Robotics and Automation(2005)*, pages 661–666, 2005.

[52] A. Visserand B.A. Slamet. Including communication success in the estimation of information gain for multi-robot exploration. In *Proceedings of the 6th International Symposium on Modelling and Optimization*, pages 680–687, 2008.

[53] C. Stachniss and W. Burgard. Exploring unknown environments with mobile robots using coverage maps. In *Proceedings of the 18th Joint Conference on Artificial Intelligence*, pages 1127–1134, 2005.

[54] C. Thorpe and H. Durrant-Whyte. Field robots. *International Symposium on Robotics Research*, 2001.

[55] S. Thrun and J.J. Leonard. Simultaneous Localisation and Mapping. *Robotics and Cognitive Approaches to Spatial Mapping*, 38:13–41, 2008.

[56] Lovekesh Vig and Julie A. Adams. Multi-robot coalition formation. *IEEE Transactions on Robotics*, 22(4):637–649, 2006.

[57] M. Moors W. Burgard and D. Fox. Collaborative multi-robot exploration. In *Proceedings of the IEEE International Conference on Robotics and Automation (2000)*, pages 476–481, 2000.

[58] M. Moors W. Burgard and M. Schneider. Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3):376–378, 2005.

[59] C. Papadimitriou X. Deng. Exploring an Unknown Graph. In *Proceedings of the IEEE 33th Annual Symposium on Foundations of Computer Science*, pages 355–361, 1990.

[60] B. Yamauchi. A frontier-based approach for autonomous exploration. In *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 1997.

[61] B. Yamauchi. Frontier-based exploration using multiple robots. In *Proceedings of the Second International conference on autonomous agents (Minneapolis 1998)*, pages 47–53, 1998.