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

Proactivity in Coordination Mechanisms for Multi-Robot Exploration of Indoor Environments: A Graph-Based Approach

Thesis advisor:
Prof. Francesco Amigoni

Candidate:
Alex Perugini, 876359

Academic Year 2018-2019

Alla mia famiglia

Abstract

This thesis deals with the problem of exploring unknown environments by means of a team of robots. Specifically, attention is given to the coordination among them to complete the exploration as fast as possible. Starting from two already existing coordination mechanisms, we propose some variants, based on a different approach to the proactive use of the idle robots before they are actually needed in the exploration. The aim of the proposed variants is to exploit what is known about the environment during the exploration in order to improve the base coordination mechanisms. The idea is to determine a convenient waiting location for idle robots, from which they can move when they are needed. To do this, a representation of the environments using graphs has been introduced. They are built on the partial representation of the map and updated as the exploration goes on. The robots not currently needed are moved to a location computed through the values of centrality measures for these graphs. We analyze two different types of graphs, as well as two centrality measures. Their combination produces four variants for each coordination mechanism considered. The proposed variants have been compared to the base coordination mechanisms according to the time required to complete the exploration of a number of environments. Results obtained show that the proposed variants do not enhance performance in general, but they perform better than the corresponding base mechanisms in some specific environments.

Sommario

Questa tesi riguarda il problema di esplorare ambienti sconosciuti mediante un gruppo di robot, ponendo l'attenzione su come i robot si coordinano per eseguire tale compito nel minor tempo possibile. Partendo da due meccanismi di coordinamento già esistenti, proponiamo delle varianti degli stessi. In particolare, l'aspetto su cui questo lavoro si fonda è un utilizzo proattivo dei robot inizialmente non necessari ai fini dell'esplorazione. L'obiettivo è quello di sfruttare esplicitamente, rispetto ai meccanismi di base, le conoscenze relative all'ambiente che si sta esplorando al fine di far attendere tali robot in una posizione conveniente da cui partire, una volta che saranno necessari. Per far sì che essi siano posizionati in maniera efficace, abbiamo rappresentato l'ambiente con dei grafi, costruiti a partire dalla mappa parziale dell'ambiente ed aggiornati con il proseguire dell'esplorazione. Il miglior punto in cui posizionare i robot non necessari è poi individuato mediante il calcolo di metriche di centralità sui grafi ottenuti. Abbiamo analizzato due differenti tipi di grafi, così come due metriche di centralità. Le loro combinazioni hanno permesso di ottenere quattro varianti per ognuno dei meccanismi di coordinamento considerati. Tali varianti sono state confrontate con i meccanismi di coordinamento di partenza sulla base del tempo richiesto per completare l'esplorazione di diversi ambienti. I dati ottenuti dal loro confronto mostrano che le varianti proposte non migliorano l'esplorazione in generale, ma almeno una tra le varianti proposte fornisce risultati migliori rispetto al corrispondente meccanismo di coordinamento di partenza in specifici ambienti.

Contents

Chapter 1

Introduction

The use of mobile robots is increasing consistently in the last years. They are used in a wide variety of scenarios, like search and rescue, logistics, surveillance, exploration, just to mention few of the applications. Some of the most interesting examples of such systems in recent times are represented by PMORPH and Colossus (Figure 1.1). PMORPH is a robot employed in the disaster site of Fukushima to investigate the situation in sectors of the nuclear power plant unaccessible by humans, due to the high radioactivity values [41]. Colossus is a firefighter robot employed as a first responder during the Notre-Dame de Paris fire. Given the high temperatures reached in the interiors of the cathedral, it was impossible for human firefighters to enter [32]. Both these robots are teleoperated, thus, a person is needed to control them, and it is a hard task. Moreover, teleoperation imposes limits in such scenarios. This happened different times on the Fukushima disaster site, as the cables used to control the robots got stuck, and the use of wireless communication was not possible. For this reason, a certain degree of autonomy is desirable.

Both the examples presented above exploit a single robot. When dealing with scenarios like the exploration of an unknown environment or search and rescue, the use of a team of robots is usually beneficial to the completion of the task [10, 24, 34]. In both these contexts, the aim is to build a map of the environment, and the presence of more robots allows to speed up the coverage of a bigger area in the same amount of time. Moreover, a team of robots is able to produce a map more accurate because of the increased amount of redundancy in the measurements [7]. The advantages of employing a team, rather than a single robot, point out, even more, the need for autonomy. Considering that teleoperating a team of robots is an extremely complex operation [12], robots should be able to autonomously coordinate both to

(a) PMORPH. Image taken from [41]

(b) Colossus. Image taken from [26]

Figure 1.1: Examples of teleoperated robots

reduce the time required to complete the task, and to avoid collisions.

The setting of this thesis is the one just described, the exploration of an unknown environment through a team of robots. The focus is on the coordination mechanisms regulating the behavior of the team of robots. The coordination mechanisms considered split the team of robots into two sets, an active set and an idle set, following the idea originally introduced in [33]. The first one is composed of the robots which are exploring the environment. While robots which are not yet needed for the exploration form the idle set. These robots wait for the moment when they are assigned to explore a location. A good location where moving the idle robots is the main aspect considered in this thesis. Moving them proactively towards a suitable location is beneficial to the performance of the exploration, in the case in which that location represents a good spot from where to start once needed.

The aim of this thesis is to include information about the topology of the explored environment in the proactive allocation of idle robots. This is done by the definition of two graphs and the computation of two centrality measures on them. The combination of these elements produces four possible ways of computing the location where to move the idle set. Their application has been tested on two coordination mechanisms in which the proactive movement of the idle robots is done according to a naive approach. Further analysis is carried on to relate some features of the explored environment to metrics that can be obtained during the exploration itself.

The coordination mechanisms on which this thesis is based are the proactive versions proposed by [13] of the *reserve* and *buddy system* mechanisms, introduced originally in [33]. In the case of [13], the proactivity is implemented by moving the idle robots to the barycenter of the polygon whose vertices are the locations of the active agents. In their original formulations, both reserve mechanism and buddy system mechanism keep the idle robots waiting at the initial location. The results reported by [13] shown how a proactive movement of the idle set of robots provides an enhancement to performance, particularly in the case of reserve. Given this result and the consideration that proactivity through the barycenter computation does not directly take into account any aspect of the environment, being only based on the locations of the active robots, through this thesis we inspect whether the inclusion of topological aspects may enhance the exploration even more.

Coordination mechanisms represent just one of the main components in the solution of the exploration problem, and they are present only in the case of multi-robot exploration, as they address the problem of deciding which robot goes where. In the exploration of an unknown environment, two further components needed are the SLAM algorithm and the exploration strategy.

The SLAM algorithm solves the Simultaneous Localization And Mapping problem. It comes out because, as a robot moves in the environment, its location is hard to determine even if it knows its initial pose, due to uncertainty in the odometry. Moreover, this is complicated by the fact that the complete map is not known, thus only a partial knowledge of the environment is available to the robot. The SLAM algorithm provides a solution to this problem, allowing to integrate the data coming from sensors to simultaneously localize the robot on the partial map built up to that moment and update that same map. A wide range of different algorithms for solving SLAM problems have been proposed in the literature [7, 39], and they also differ in the kind of map produced. Models for the map can be topological or metric. In the former case, the output of the SLAM algorithm is a graph [23], which provides information about the structure of the environment. In the latter case, the output is a map providing the exact locations of the obstacles [28]. Once a partial map is available to the robots, they have to decide the next locations to explore. This is done by following an exploration strategy [22, 43, 47, 48]. It takes as input the output map of a SLAM algorithm and detects a set of candidate locations to explore. It also provides the policy to select the next one to explore. When dealing with a team of robots, once the candidate locations are identified by the exploration strategy, the coordination mechanism decides how the robots of the team should be allocated to them. They basically answer to two different questions: *Where to go next?* for the exploration strategy and *Who goes where?*

for the coordination mechanism. In [3], it is analyzed the relative impact of the exploration strategy and the coordination mechanism on the exploration, assessing that the first one affects more the performance on highly structured indoor environments. On the other hand, the role of coordination mechanisms is dominant in less structured indoor environments.

Graphs are widely employed in the literature about exploration [5, 6, 14, 19], as they are particularly useful to get rid of the geometrical aspects of the environment and to focus only on its structural properties. By doing so, it is also possible to reduce the spatial complexity of the map with respect to a metric representation of the same environment [14]. Once a topological representation of the environment is provided, it is possible to define exploration strategies that identify nodes of the graph as candidate locations [5, 22]. They are a strong modeling tool that we consider able to include information about the environment to enhance the proactivity in the coordination mechanisms. Two graphs are tested, the topological graph and the visibility graph. The first one is a graph aimed at capturing the connectivity between regions, generated from the occupancy grid produced by the SLAM algorithm. The second graph is a historical graph whose nodes are both located in the positions where the robots moved and represent the candidate locations identified by the exploration strategy. These graphs are included in the coordination mechanisms through centrality measures, which are measures aimed at identifying the most central nodes in a graph. Depending on the centrality measure employed, the concept of central node varies. Among all the existing centrality measures, the two considered here are closeness and betweenness. The first one considers as central a node whose average distance from all the other nodes is low. While the second one considers as central a node along the largest number of shortest paths between two other nodes. These two measures are considered as worth to be tested for their capacity of providing a good location where to proactively locate the idle robots. They are applied largely in social network analysis, being able to track the most influential nodes in a network or the ones most crucial to grant the connectivity of the network [21, 29, 30].

To evaluate the impact of the proposed graph-based approaches with respect to the barycenter-based mechanisms of [13], we perform a series of experiments, testing each possible combination of graphs and centrality measures on both the coordination mechanisms considered. In this way, it is possible to analyze the relative impact of a different graph or a different centrality measure on performance. All the experiments have been performed in MRESim, a simulator for multi-robot exploration of 2D environments [35], which allows us to test the proposed mechanisms over a number of environments for varying team sizes. The metric we consider in the evaluation of the goodness of a certain mechanism is the number of discrete time steps required to complete the exploration. Moreover, also availability and interference are evaluated, as defined in [13]. Even though they result not to be a discriminating factor among the various mechanisms, unless when distinguishing between reserve and buddy system.

When considering the whole set of environments tested, the proposed variants to the proactivity provide no benefits in general, making the approaches based on the barycenter to be preferable, given also their intrinsic lower complexity. If the focus is restricted to single environments, our experiments show that the exploration may benefit from the adoption of the proposed coordination mechanisms, in some cases. Among the proposed graph-based mechanisms, the best one is the one using the topological graph and the betweenness as centrality measure, as it grants results even better than the corresponding barycenter-based mechanism on some environments, and tend not to worsen on the others. On the contrary, the visibility graph as defined in this thesis results not to be a good approach for two main reasons. In general, it does not provide any enhancement to exploration with respect to the topological graph in terms of completion time. Moreover, its higher number of nodes and edges than the topological graph makes the computation of the centrality measures slower.

This thesis is structured as follows. In Chapter 2, the state of the art for the two main aspects regarding multi-robot exploration is reported. The chapter starts with some of the exploration strategies presented in the literature, giving particular attention to the one employed in our experiments. Then, the attention shifts to coordination mechanisms, showing the differences between online and offline mechanisms. In Chapter 3, a formal definition of the exploration problem considered in this thesis is provided. It is also highlighted the setting of the experiments, starting from the representation of the environments given as output by the SLAM algorithm and a presentation of the features used to discriminate among them. Then, agents are described, as well as the exploration strategy employed. The last section of the chapter illustrates the coordination mechanisms proposed by [33] and their extensions by [13]. Chapter 4 contains the description of the graphs employed and the centrality measures employed. It is also reported the definition of interference and availability. In Chapter 5, it is provided an in-depth analysis of the coordination mechanisms, pointing out the common structure which characterizes them all and the modifications needed to implement each mechanism considered in this thesis. In the end, some needed optimizations for the proposed mechanisms are reported. Chapter 6 contains the results of the whole set of experiments performed. In the last section, it is also reported a possible interpretation of some of the metrics considered, and their relation with the features of the environments. Chapter 7 reports the conclusions about the results obtained, together with some possible future work.

Chapter 2

State of the art

The exploration of an unknown environment pursued by a team of robots is a complex problem, tackled in different ways across the literature. It can be informally defined as the process of producing a representation of the environment, in the following referred to as map, which can be used for future navigation. The map can fall into two categories; it is said to be topological if it is a graph modeling connectivity between regions, while it is metric if it provides the exact locations of obstacles. Two of the most used metric models for the map are occupancy grids and coverage maps. *Occupancy grids* model the environment as a grid where each cell can be marked as *free* or *obstacle* if already scanned through sensors, or *unknown* if it has not been scanned yet. Also, *coverage maps* are a grid-based representation of the environment, where each cell contains the posterior probability of being covered by an obstacle. This provides different advantages compared to occupancy grids, like for example the possibility to finely model a wall not parallel to x- or y-axis of the grid, without the need of enlarging it to match the discretization.

The mapping process is carried on by one or more robots able to perceive the environment utilizing sensors of different types. The most common are lasers [11, 22] and sonars [24], even if other types of sensors are sometimes used, like a laser-limited sonar [47, 48] which is a combination of the two, or a Microsoft Kinect sensor [33], which provides 3D measurements based on an RGB camera and a depth sensor. The map is progressively updated by including the information obtained from sensors on the partial map known at that moment.

Looking at exploration from a conceptual point of view, two main phases can be identified. The first one concerns the detection of the best locations to explore next in the partial map built so far. The other one deals with the allocation of robots to these candidate locations. Thus, the whole exploration can be seen as an iterative two-step procedure where, once the map is updated, a set of possible points of interest are chosen, based on some criteria, and then to each robot is assigned a goal location. These two steps are repeated until the exploration can be considered finished. The *exploration strategy* is the algorithm that selects the candidate locations, while the *coordination mechanism* is the one allocating the robots to them.

2.1 Exploration strategy

As previously described, the exploration strategy is an algorithm providing the candidate locations robots should visit to maximize their knowledge about the environment. There are two crucial aspects in this, that are the definition of a candidate location and the criterion used to choose the best one.

How a candidate location is defined depends strongly on the representation of the environment. As presented in the previous section, this can be divided into two categories, topological or metric, which turns into graph-based or grid-based representations, even if other data structures are possible, like in [2] where the map is stored as two lists of line segments.

The first step of the exploration strategy consists in generating a set of candidate locations. Its generation has a high impact on the definition of the algorithm and it is the core in frontier-based strategies [22, 43, 47, 48], where the focus is mostly in this generating step, rather than in the choice of the next location among the possible candidates. Other strategies, on the contrary, skip the generation of this set by considering the whole set of known cells or a portion of them. This is possible because of the particular implementation of the criterion used to choose the next best location [36].

The choice of the next location to explore among the set of candidate ones is done in different ways across the literature. An example of this in a metric representation is in [2], where a comparative review of four strategies for single robot exploration is performed, distinguishing among a random approach (used as benchmark), a greedy one and two complex *ad hoc* procedures, testing their performance over different environments. Therefore, the choice can be done according to different criteria and the two main factors affecting it are the amount of expected new information obtainable from the location, usually referred to as utility, and the cost of reaching it. In a topological representation, strategies like a Depth-First Search [18] and Breadth-First Search [44] are naturally possible algorithms on undirected graphs, while [1] proposes an interesting approach to solve the exploration problem on directed graphs.

In the following, some exploration strategies are presented, distincted into three categories: *information gain-based*, *frontier-based* and *topological strategies*. We give particular attention to the second one, being the one used in this work.

2.1.1 Information gain-based strategies

Information gain-based strategies as presented in [36] are probabilistic strategies usually employed on coverage maps, and consequently extensible to occupancy grids. Candidate locations are chosen among the known cells of the grid, according to the expected change in the entropy obtainable by moving a robot there. Given a posterior probability distribution $p(c)$ of a cell c, its entropy $H(p(c))$ is defined as

$$H\left(p\left(c\right)\right) = -\int_{c} p\left(c\right) \log p\left(c\right) dc$$

While, the information gain for a given cell c and measurement z taken from the pose x is

$$I\left(c, x, z\right) = H\left(p\left(c\right)\right) - H\left(p\left(c|x, z\right)\right)$$

Then, each known cell in the grid is considered as a possible candidate location and the one providing the highest expected entropy reduction, i.e., information gain, is selected. This method provides suitable locations because the information gain for a completely known cell is near zero, thus the approach tends to assign candidate locations in the proximity of uncertain cells, increasing the knowledge of the environment. This partially justifies the absence of care in generating the set of candidate locations, rather it is preferred to take the whole set of known cells and to check the information gain each one can provide.

The impact on the performance of this brute-force search is high, for this reason, in [36], besides the basic strategy presented so far, two modifications are also introduced. The first one reduces the number of candidate locations from the whole set of known cells in the grid to the ones in a local window, which has to be completely explored before moving on. Rather than reducing the set of candidate locations, the second one modifies the way in which the next one to explore is chosen by introducing the cost to reach that location from the robot pose.

In [4], it is presented the *A-C-G strategy* which defines a conceptually similar approach but takes explicitly into account the contribution to entropy of the points sensed from the candidate location, discriminating between already sensed points and the ones sensed for the first time. It also includes a factor proportional to the distance from the robot location and the candidate one.

In [37], an information gain strategy is integrated into the localization and mapping phase. This allows deciding which action to perform at each step of the exploration, by taking into account the trajectory and map uncertainty.

2.1.2 Frontier-based strategies

As presented above, information gain-based strategies mainly focus on the process of deciding the next location to explore, giving less attention to the definition of the set of candidate locations. This is clear by considering the base strategy of [36], where all the known cells in the grid are possible candidate locations and the method implicitly cuts out the ones not providing new knowledge. In frontier-based strategies, the focus is shifted to the creation of a good set of candidate locations.

As defined in [47], the paper originally introducing this strategy for single robot exploration, a *frontier* is the boundary region between explored and unexplored space. The idea is that, by assigning a robot to its closest frontier as location to explore, the line between explored and unexplored space is pushed continuously, until the whole environment is mapped. In that case, occupancy grids are used to model the environment and with that representation, it is pretty straightforward to find out a frontier, identifiable as a cluster of adjacent free cells whose neighbors are unknown.

This simple idea works extremely well in practice and for this reason, it has been used widely in the literature, producing a lot of extensions and adaptations to the various cases, like [48] where the strategy is extended to multi-robot scenarios.

In [43], the Leader-Follower exploration algorithm is presented. It focuses on the roles assumed by the robots during the exploration, which can be dynamically changed, according to the distance from the assigned location. Candidate locations detection is done by identifying frontiers and, differently from the strategy of [47] and [48], the next to explore is not chosen as the nearest one. Indeed, it looks for the pair of frontiers maximizing the sum of the rewards for the leader and the follower, where the reward function is composed of a utility term minus the cost to reach the frontier.

[22] provides an extension of this exploration strategy to topological maps, rather than occupancy grids. An interesting aspect of this is in the strict relation between the frontiers and the nodes of the graph. As the environment is progressively mapped, frontiers are detected and classified according to geometric information about the environment into free area or transit area, defined as the area where the robot transits between two spaces (rooms, corridors, and so on). Once classified in one of these two categories, the next frontier to explore is selected through a cost-utility function, composed of three terms: the geometric and the semantic utility, and the topological cost. The geometric utility corresponds to the size of the frontier; a bigger frontier offers a bigger range to acquire new information. The semantic utility is related to the classification of the frontier, being a transit area preferable over a free area, despite its smaller size. The topological cost is a cost term associated with the connectivity between frontiers. It assigns a fixed small cost to consecutive frontiers, while if to reach a frontier, a robot has to pass by other frontiers, the cost of that one is proportional to the number of crossed frontiers. Once a frontier has been explored, it is added as a node in the graph. The proposed cost-utility function is then linearly related to the utilities and the relation with the cost factor is a reverse exponential. The algorithm guided by this function is shown to have good performance both in terms of exploration time and traveled distance against some benchmark algorithms.

2.1.3 Topological strategies

Topological strategies rely on a graph-based representation of the world. This is useful to neglect the geometrical features of the environment and to focus on its structure. As shown in [14], the complexity of using geometric maps grows exponentially as the environment becomes larger and this justifies the use of topological maps. Moreover, the use of a directed graph can also simulate the case of one-way streets, where the robot is allowed to go in one direction and not in the opposite one [6], which would be impossible to describe just relying on geometric maps.

In this kind of strategies, candidate locations are nodes of the graph and the next one to explore is decided in various ways, strongly depending on the type of graph used. In fact, in the literature, both undirected and directed graphs are used, with a further distinction whether or not the vertices are identifiable. A vertex is identifiable if it can be recognized by a robot when revisited. This is not always guaranteed because the robot may have limited sensor capabilities or the appearance of vertices may change.

Undirected graphs with distinguishable vertices are the most straightforward case. Each vertex is labeled uniquely and the robot is allowed to traverse the edges in both ways [18, 20]. In [9] it is presented an extension of the Depth-First Search algorithm to the multi-robot scenario both for graphs and trees. A particular aspect of the model used is that edges are considered as opaque, this means that from either end, it is not clear where the edge goes. In [5] it is analyzed the problem of piecemeal exploration, this states that the robot can traverse a limited number of edges before going back to the source vertex. It is a realistic context in which the robot has a limited amount of fuel or battery and needs to refill it after a fixed number of steps or traveled distance. The algorithm proposed for this problem is based on Breadth-First Search and another important aspect is the use of the concept of *frontier vertex*, defined as a vertex incident to unexplored edges.

Undirected graphs with anonymous vertices introduce some difficulties and to get rid of them, markers are needed to distinguish between explored and unexplored area [19, 25, 44]. In [19] it is shown that one marker is sufficient to allow the robot to build a graph isomorphic to the environment in low-order polynomial time and the use of multiple markers may improve the performance. In [44] two enhancements are presented both to single-robot and multi-robot exploration in such environments, provided by the use of a Breadth-First Search and the exploitation of local neighbors information.

In the case of directed graphs, the robot movement is strongly limited with respect to undirected graphs. Clearly, Depth-First Search is not always possible because backtracking is not guaranteed to be applicable. Different algorithms have been proposed to deal with these models [1, 6, 17]. In [1] it is proposed an algorithm to visit all nodes and edges with a subexponential upper bound on the number of edge traversals. In [6] it is defined an algorithm able to explore a directed graph with anonymous vertices by using two robots through the simultaneous learning of the graph and a homing sequence. This is done by keeping multiple possible maps, updating them through a sequence of movements, then checking their correctness. It also states that it is not possible to efficiently learn the same kind of graph utilizing a single robot with a constant number of pebbles without prior knowledge on the number of vertices. In this case, pebbles are used similarly to the markers stated above. They can be dropped by a robot at a certain vertex to make it recognizable when revisited and eventually, they can be also picked up by the robot to place them at another node. Previously it has been presented [19], which shows that same problem solvable with one marker in low-order polynomial time in the case of an undirected graph.

2.2 Coordination mechanisms

In a multi-robot scenario, once candidate locations are detected on the map, it comes out the problem of how the robots in the team have to be assigned to them in order to maximize the knowledge about the environment. Moreover, even if a random allocation is possible, it is clearly preferable an assignment of agents to candidate locations which minimizes some metrics like the time taken to explore the environment or the distance traveled by the robots.

The answer to this is provided by the coordination mechanism, which is the algorithm that assigns robots to candidate locations according to some criteria. Coordination mechanisms are distinguished into online and offline. Online mechanisms assign robots to candidate locations by taking into account the actions currently done by the other members of the team [10, 11, 34]. In offline mechanisms, in contrast, roles are assigned to robots before the exploration starts and offline coordination can be divided into two further categories, fixed and variable [40]. In fixed offline coordination, robots act according to the roles defined before the beginning of the exploration and they stick to these roles, without altering them [24,31,33,42]. In variable offline coordination, robots can exchange their roles dynamically as the exploration goes on [43].

Coordination based on an online mechanism is weighed down by the need for more communication among the agents. Before an allocation is made, an agent has to know other agents poses and targets locations. On one hand, this implies a lot of communication to make proper assignments; on the other hand, this allows to perform choices aimed at maximizing the performance of the system. To clarify this, it is interesting to anticipate the algorithm proposed in [34] and analyzed more in-depth in the following section. This algorithm provides that every time the map is updated and the set of candidate frontiers is detected, each robot communicates its expected gain obtainable from the exploration of each frontier in the set. After having received them all, the central executive computes the next location for each robot, in a way to provide the highest possible gains for the whole system.

Through the use of roles, offline mechanisms require little to no communication once the exploration is started, making the robots and the whole system easier to implement. The other side of the coin is that robots move almost freely, with the possibility of interference among them and the redundancy of assignments to the same target location.

The relation between exploration strategy and coordination mechanism is quite tight and the relative impact each one has on the performance is hard to establish. A work in this sense is [3], where different exploration strategies and coordination mechanisms are compared in two different environments. What comes out is that in structured environments, like an office one with a lot of rooms and corridors, the detection of good candidate locations is preferable over a good assignment of robots to them. In an open environment, the contrary holds, being the coordination mechanism able to increase the amount of area explored in the same amount of time, making the impact of the exploration strategy less relevant.

2.2.1 Online mechanisms

Online coordination mechanisms allocate robots to target locations exploiting current information about other robots actions. To achieve this, robots need to communicate with each other. This has been done in different ways across literature, using different techniques [10, 11, 34].

In [34] the communication is performed through the use of *bids*. Every time a robot receives a map update from the central mapper, it sends a bid with a list of costs and information gains for each frontier to the central executive. As the central executive gets all the bids, computes the assignment maximizing the difference between information gain and cost for any robot and assigns the frontier to that robot. Once an assignment is fixed, the other bids are discounted by a certain value to take it into account. This procedure is iterated as long as there are no remaining robots or tasks. The discount factor is fundamental, being the main factor introducing the online coordination aspect of this algorithm, in fact, if bids were not discounted, each robot would go towards the frontier with the highest estimated utility, not taking into account other robots assignments. Also [10] and [11] perform coordination by considering the utility of each frontier computed as the difference between the information gain it can provide and the cost to reach it.

In [10] every time a robot is assigned to a certain frontier, the utility of the other frontiers is discounted by a value proportional to the probability of being in the visibility area from the assigned one. In [11] this approach is extended to limited communication scenarios.

[46] differs from the previous works because the algorithm proposed uses a topological map, rather than a metric one. The topological map is built as the Voronoi graph of the partial map, known up to that moment. A Voronoi graph is a graph in which nodes consist of points of the free space equidistant

from the closest obstacles. An edge connects a pair of nodes if they are adjacent in the map. Once the Voronoi graph is computed, it is segmented in a way to create frontiers at *critical points*, like doorways. At this point, for each robot is computed the cost for reaching each map segment and the optimal allocation is then found by applying the Hungarian method, which is an algorithm able to provide the optimal solution with minimal cost.

2.2.2 Offline mechanisms

Offline coordination mechanisms provide a definition of roles prior to the beginning of the exploration. By sticking to these roles, coordination among robots needs little to no communication, which is one of the main advantages of this approach. Roles definition may also be modified at run-time, like how is done in the Leader-Follower algorithm [43] presented above, where the role depends on the distance from the assigned frontier.

Two major works following this approach for this thesis are represented by [33] and the further extension provided by [13]. In [33] three coordination mechanisms, namely *reserve*, *buddy system*, and *divide and conquer* are presented. The *reserve* mechanism splits the team into two smaller teams where one is left idle at the initial position, while the second one is sent to explore frontiers. As new frontiers are found, idle robots are progressively turned into active agents and assigned to them. Once all the initially idle robots are active agents, the exploration is carried out without further coordination. *Buddy system* works in a similar way, with the difference that rather than considering single robots, pairs of robots are considered. At the start of the exploration, some pairs are sent to explore frontiers, while the others remain idle at the starting position. Once a branching point is found, that is a zone of the environment where different spaces meet, like a T-shaped junction, for example, the pair is split and each robot explores a different branch. If another branching point is found, one branch is explored by the single robot which discovered it, and the other one is assigned to a pair from the idle set, which then turns into active. In *divide and conquer*, at the beginning of the exploration, all the robots move together following a leader, then as a branching point is found, the team splits into two halves. A new leader for the second team is decided and each team is assigned to a branch. This splitting approach goes on while there are teams composed of more than one robot and, after that, they proceed in an uncoordinated way.

[13] modified these mechanisms proposing respectively *proactive reserve*, *proactive buddy system*, and *side follower*. The idea behind the first two mechanisms is to move the idle set from the starting position, towards a

better position, nearer to the possible branching points. This would allow the robots turned into active to reach the assigned frontiers in less time, once they are called. The waiting position for the idle team is computed as the barycenter of the locations of the active agents. This thesis expands this approach modifying the way in which this waiting position is computed by taking the topology of the environment into account. Differently from *divide and conquer*, the *side follower* mechanism organizes the agents into groups of three, rather than a single group. The idea is that each robot in the group has a preferred direction for the frontier to explore: the left robot tends to explore frontiers on the left, the right robot prefers the ones on the right and the robot in the center explores the ones in front of it. These modifications are shown to have very good performance when compared with the benchmark ones, particularly *proactive reserve* which is usually better than all the other considered strategies. *Proactive buddy system* outperforms the simple *buddy system* mostly on open environments, resulting in similar or worst performance on the others. *Side follower* also performs generally better than *divide and conquer*, particularly on the environments reflecting the structure for which it is designed, that is a central corridor with spaces on the sides.

Chapter 3

Problem setting

3.1 Exploration problem

The exploration problem can be defined as the problem of mapping an unknown environment by means of a team of robots. Thus, the objective is the construction of a map of the environment but this introduces further problems, concerning for example the localization of each robot dealing with the various sources of uncertainty, how to distribute the team, and how to coordinate them. These are three of the main aspects considered in the development of a solution to this problem, providing the major impact on performance. To compare different approaches the main metric used is the time taken by the exploration [11, 13, 33], but also the distance traveled by the robots is sometimes considered [5].

A formal definition of the *exploration problem*, in the following abbreviated as *EP*, can be provided through a 4-ple $\langle A, E, P_0, T \rangle$ where:

- A is the set of robots used for the exploration;
- E is the environment to explore;
- P_0 is a vector of the initial poses of the robots of the team;
- T is the termination criterion.

These four elements characterize the instance of EP addressed, while the solution of the same consists in producing a map M of the environment E. To do this, each robot is able to perceive the environment through one or more sensors, characterized by the size of the spanned area. The map M consists in a 2D occupancy grid representing the areas of E. The center of each cell c is characterized by its coordinates in a global coordinate system, provided as a vector (x_c, y_c). Each cell contains a value which tells if that cell is free, occupied by an obstacle, or still unknown. Clearly, the first two values refer only to already explored cells, while the third value refers to those cells which are still not scanned by any robot of the team.

The experiments presented in this work are run on MRESim, a 2D simulator aimed at testing multi-robot exploration scenarios [35]. In particular, it allows to create an instance of *EP* through a configuration file which provides the robots in the team, their initial poses and a PNG image representing the environment to explore. The vectors P_0 employed have in common the disposition of the robots along a straight line suitably spaced to make them start from the same initial location. The termination criterion used is the percentage of explored area and it can be modified by varying a variable in the code.

The following sections describe the system used for the experiments, focusing at first on the environment and the agents, then moving the attention to the exploration strategy and the coordination mechanisms considered in this thesis.

3.2 Localization and mapping

The problem of mapping an unknown environment and simultaneously localize a single robot on the partial map built is one of the fundamental problems of robotics, called *SLAM problem*, from *Simultaneous Localization And Mapping*. The robot knows its initial pose and, as it moves, the uncertainty about its pose increases, due to uncertainty in the odometry. For this reason it becomes necessary to localize the robot on the map, even a partial one. A formal description of it, as provided in [39], is suitably done in a probabilistic framework and distinguishes two versions, the *online* and the *full SLAM problem*.

Let X, U, and Z be three statistical variables representing respectively the sequence of poses assumed by the robot, i.e., the *path*, the sequence of odometry measurements, and the sequence of measurements provided by the sensors. Let also μ be the true map of the environment. The *full SLAM problem* is then defined as the problem of estimating the posterior probability of the map together with the whole path traveled by the robot, that is $p\left(X, \mu | Z, U\right)$. The *online SLAM problem* aims at estimating the posterior of the actual location x of the robot, rather than the whole path, together with the map. Thus, $p\left(x, \mu | Z, U\right)$.

What happens in practice is that as the robot moves in an unknown environment, it perceives the surroundings through its sensors, has an estimate of its pose from its odometry measures and these are used to reduce the uncertainty both about the map built and the location of, or the path followed by, the robot, depending on the version of the SLAM algorithm implemented. As the exploration goes on, the partial map approaches to a complete map of the environment and it is also possible to get rid of the noise in the measurements.

The model used for the map can be both topological [23] or metric [28]. In particular in this thesis, the representation provided as output by the SLAM algorithm is metric, consisting in an occupancy grid.

3.2.1 Environments

This work considers only indoor environments. Reasons behind this restriction are related to the work of [13], of which this thesis is an extension, in order to provide a comparison in the same type of environments. Moreover, this focus is justified by the application contexts, being the exploration of indoor environments more common with respect to outdoor ones.

The representation of the map provided by the SLAM algorithm used in this work is a two-dimensional occupancy grid. Moreover, being applied to a multi-robot scenario, it comes out the difficulty of merging the maps computed by solving the SLAM problem for each agent. The solution to this is a centralized approach by means of a base station, which collects all the individual local maps and combines them into a single global map.

Similarly to [13], the environments are classified according to some features characterizing them. In particular, the aspects considered are the size, the openness, and the parallelizability. The first one is easy to formally define. For the second and the third one, a formal definition can hardly be provided being influenced by many factors. To make the definitions more readable, recalling that M denotes the map of the environment modeled as an occupancy grid, let $s : M \to \mathbb{R}$ be an auxiliary function which takes as input a cell of the map and provides its area. The features considered can be defined in the following way.

- Size: the amount of free area in the map. Thus, $S(M) = \sum_i s(f_i)$ with f_i being a free cell. The size of the environment clearly affects the average time taken by robots to explore it. The smaller the environment, the lower the amount of resources needed to map it. From this, the distinction into small (S) and large (L) environments.

- Openness: the property of an environment to be composed of large open spaces (O) rather than cluttered ones (C). It is also related to the possibility for the agents to follow different paths in the case of an open environment, opposed to a channeled exploration towards specific directions for a cluttered one.

- Parallelizability: the property of an environment to enforce the spreading of the robots during the exploration. A formal definition to this is hard to provide, but it can be informally presented considering that if the environment allows the team to spread, it is highly parallelizable (HP). Otherwise, if the robots are forced to stick together, it is lightly parallelizable (LP).

To make these distinctions clearer, it is worth looking at some of the environments used in this work and at how they are classified according to these features. In Figure 3.1, two very different environments are presented.

The environment in Figure 3.1a is a simple maze, composed of a series of few corridors. This is classified as S-C-LP, because its dimension is small and the corridors make the exploration to be channeled into specific directions. Thus, robots are not allowed to spread into it, forced by the long corridors, justifying the classification as cluttered and lightly parallelizable.

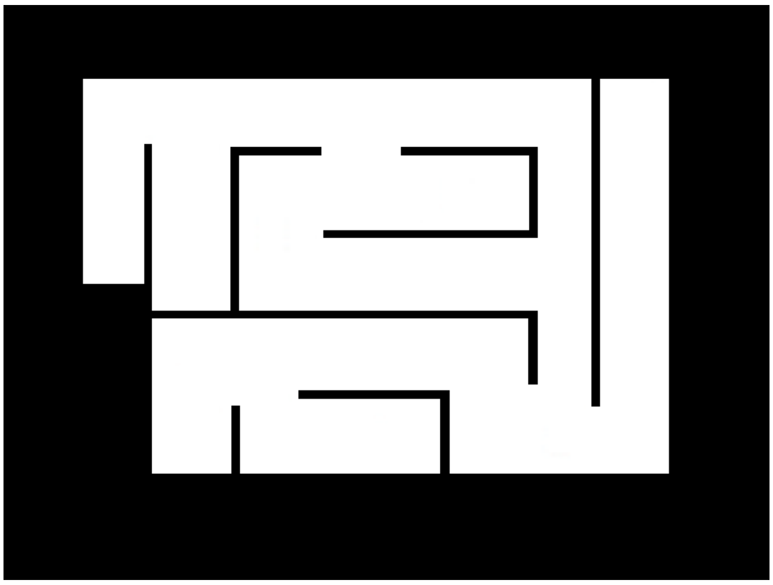
The second one (Figure 3.1b) is likely to be the map of an office. It is composed of a lot of rooms, corridors and spaces with different sizes. It is classified as L-C-HP because the amount of free area is really large and is mainly composed of rooms and limited spaces. For the configuration of the corridors, the robots tend to spread in different directions, without sticking close to each other. For this reason, the environment is classified as highly parallelizable.

Given the difficulty in providing a formal definition for the openness and the parallelizability, the possibility to correlate them with some metrics measurable during the exploration is analyzed in-depth in Chapter 6.

3.2.2 Agents

In MRESim, an agent is characterized through:

- a number and an ID to uniquely identify it;
- its pose, expressed by a vector $P = [x, y, \phi]$, where (x, y) are the coordinates of the grid cells occupied by the robot and ϕ is its orientation;
- the sensing range, set to the default value;
- the communication range, assumed to be infinite;


(a) S-C-LP environment

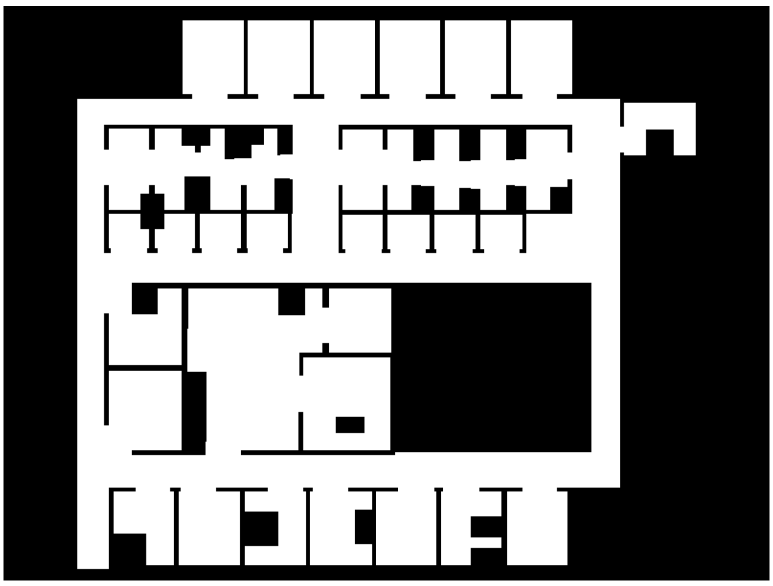

(b) L-C-HP environment

Figure 3.1: Example of two different environments

- the battery life, assumed to be infinite as well;
- its type, it can be the base station, a relay, or an explorer.

The communication range and the battery life are assumed to be infinite because the focus of this work is on the results provided by the use of different coordination mechanisms. In this way, it is possible to simplify the mechanisms not to consider scenarios in which the robots run out of fuel or they are unable to communicate, even if these situations are of particular interest in a more realistic context and object of study as in [5] and [11], respectively.

As stated above, the type of the robot, not to be confused with its role, can be: base station, relay, or explorer. The *relay* type is never used in the simulations performed for this work, being useful in cases in which communication between a robot and the base station is not possible using a direct link. Indeed, each team used is composed of one base station and from four to eleven explorers.

The *base station* is the central coordinator of the team and stores the global map computed by merging the partial maps provided by the agents. It is configured similarly to other agents but its pose is fixed to the initial one. It is important for the purpose of exploration because allows agents to coordinate by means of a centralized unit, rather than a decentralized approach, which would make the merging of the map way more difficult. An example of a decentralized approach is in [7], where only as soon as two robots meet, they are able to merge their maps and then proceed with the exploration based on this updated map.

The *explorer* is the main kind of agents employed, being the one which moves in the environment to map it. Apart from the configuration parameters presented above, it is also characterized by a finite speed. Explorer are equipped with a laser range sensor allowing them to scan a semicircle of radius equal to the sensing range in front of them. The value for the sensing range is specified in the configuration file. Sensor readings are obtained after every step and, as an explorer perceives previously unknown cells with its laser sensor, it communicates the measurements to the base station which merges them with the current map and updates it. The updated map is then sent to each agent.

3.3 Exploration strategy

The frontier-based exploration presented in [48] is applied to identify the candidate locations to explore. A frontier is the boundary region between

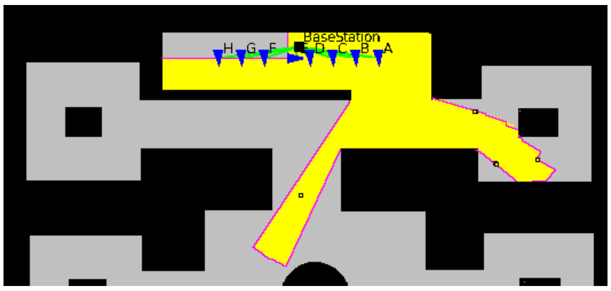


Figure 3.2: Frontiers detected on the map

known and unknown space and by moving towards frontiers, this boundary is pushed forward accordingly.

As highlighted in the previous section, the model for the map is an occupancy grid. Each robot has its own local map, updated through the sensor measurements, and merged at the base station to provide a global map, which is the one used in the frontiers identification phase.

Frontiers are identified by at first drawing the contour of the known space. The contour, represented by the pink line in Figure 3.2, depicts a clear distinction between the known space, the yellow one, and unknown space, the gray one. Then, the contour is split into smaller segments such that each segment is included within two obstacles. Due to the representation based on occupancy grids, segments are composed of a series of cells, whose centers are used as vertices of a polygon. If the area of this polygon is higher than a certain threshold, then the cell located at its barycenter is a frontier. In the figure, the identified frontiers are indicated with the little squares. The presence of three frontiers in the room on the right is due to the obstacle in the middle of that room, which splits the contour into three segments. Another aspect to notice is that the lower frontier on the left is located in the middle of the known space, not on the border, and this is due to the shape of the polygon, being it convex.

3.4 Coordination mechanisms

A coordination mechanism is the algorithm providing the allocation of robots to the possible frontiers computed by the exploration strategy. Together with the SLAM algorithm and the exploration strategy, this completes the view on the system needed for the exploration. In fact, the process of exploring an unknown environment starts with the robots scanning an area through their sensors and these data, together with the initial locations, are used as input to the SLAM algorithm, which provides a partial map as output. This is given to the exploration strategy that, as stated above, performs a discretization of the polygon enclosing the known area and computes the frontiers, excluding the ones too small. At this point, the coordination mechanism is applied to find out an allocation of robots to frontiers.

The coordination mechanisms are the main focus of this work. In the following, the coordination mechanisms we consider are presented both as the original mechanisms from [33] and their extensions provided by [13].

3.4.1 Original mechanisms

In [33], three coordination mechanisms are presented, namely *reserve*, *buddy system*, and *divide and conquer*. They are all focused on how *proactive* the team members not strictly needed for the exploration are. The team can be separated in two sub-teams, the *active* and the *idle set*, with the first one composed of the robots whose goal is to explore one of the frontiers detected, while the second one is made up by the remaining robots. An example is useful to clarify this distinction and it is provided by Figure 3.3.

This is a snapshot of the exploration right after the one of Figure 3.2. Five frontiers are detected: one located between the robots G and H, one in the central corridor and three on the right. To each frontier is assigned a robot, except for the three frontiers on the right, which are assigned to one robot, rather than three. Being the distance among them within a certain clustering threshold, only a robot is turned into active and assigned to the nearest frontier of the three. Thus, a total of three robots is turned into active, depicted in purple in the figure, and assigned to a frontier, while the remaining five robots compose the idle set, depicted in blue.

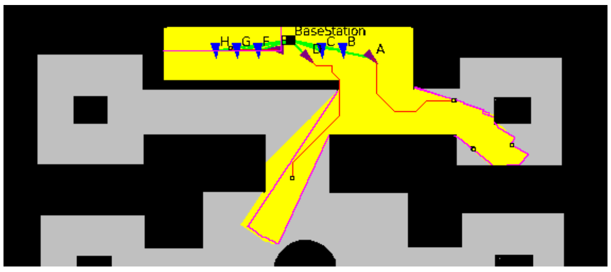


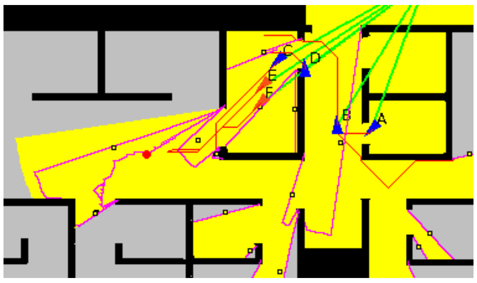
Figure 3.3: Separation into active set and idle set. Purple triangles are the robots of the active set. Each one is assigned to a different frontier and the red line is the path from the robot position to it. Blue triangles are the robots composing the idle set

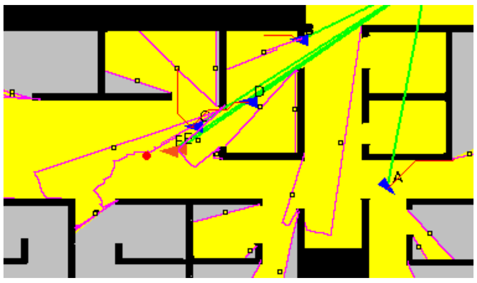
The policy for the active set is the same for all the mechanisms, to explore the closest frontier. The differences come out dealing with the idle set, in fact the way in which robots are proactively moved allows to distinguish among each mechanism. Moreover, this theme of a proactive use of the idle set is the leitmotiv linking the work started in [33] and extended both by [13] and this thesis.

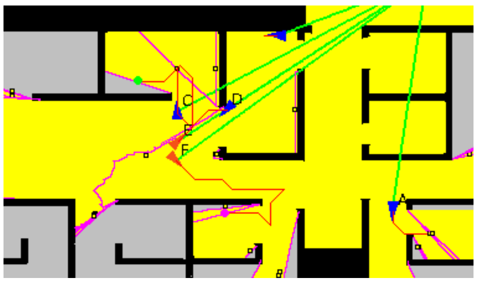
Reserve is the less proactive mechanism because as the name implies, each robot composing the idle set is left as a reserve at its initial location. As the exploration starts and the first frontiers are detected, the robots are split into the active set and the idle set, in a similar way to the example provided before. Then, the robots in the active set move towards their assigned frontiers and the ones in the idle set wait at their initial positions. As the exploration goes on and new frontiers are found, their number may be higher than the size of the active set, making some or all the robots from the idle set needed, which are then turned into active and assigned to a frontier. Once the idle set is empty and all the robots are active, the exploration proceeds in an uncoordinated way. This means that each robot is assigned to the closest frontier, without taking into account whether another robot has been already assigned to it.

Divide and conquer is the most proactive mechanism presented in that work because the idle set moves together with the active set. At the beginning of the exploration, active agents are assigned to the frontiers and the idle set is split in several subsets, one for each active agent. For example, assume that at the beginning, only a frontier is found. Then, an agent is marked as *leader* and assigned to explore it. The other robots follow it as it moves towards its frontier, until at least another frontier is detected. For the sake of the example, assume that at this point there are two frontiers. In this case, a robot from the idle set is turned into active and marked as leader. The idle set is split in two: one half follows the first leader, the other half follows the other one. This goes on until the idle set is empty, after which the exploration proceeds in an uncoordinated way. Differently from the reserve mechanism, this approach allows to have robots from the idle set nearer to the frontier to which they are going to be assigned. In this sense, the idle team members are considered to be more proactive with respect to reserve, where waiting at the initial location makes the distance between the robot turned into active and the assigned frontier higher.

Buddy system is considered to be halfway between the two previous mechanisms for what concerns proactivity of the agents. This comes clearer by looking at how the mechanism handles the idle set. As soon as robots are deployed on the environment, pairs are formed, composed of a robot marked as *leader* and another marked as *follower*, each one is the *buddy*


(a) The orange pair is moving towards the frontier, depicted as a red dot


(b) The pair reaches the assigned frontier


(c) The pair is split assigning the follower to the green dot and the leader to the pink one

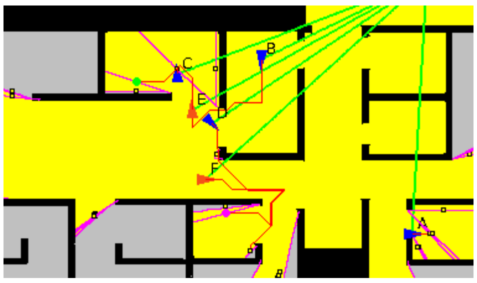

(d) The two robots move each one towards its own assigned frontier

Figure 3.4: Buddy split

of the other. Once frontiers are identified, the minimum number of leader agents is turned into active and assigned to them. Each follower follows its own leader towards the assigned frontier. The other pairs remain still at the initial location, similarly to the idle set in the reserve mechanism. When a branching point is met, that is a point where there are two or more frontiers, the pair is split and the leader is assigned to a frontier, while the follower to the other one. Here, the analogy with the divide and conquer mechanism can be seen.

An example of this situation is shown in Figure 3.4. The orange pair is going towards the assigned frontier highlighted by the red dot in Figure 3.4a. As this is reached (Figure 3.4b), being the number of frontiers high, the orange pair is split, assigning to each robot the closest frontier. These are represented as a green dot for the robot which previously was the follower, and as a pink one for the leader (Figure 3.4c). At this point, each robot goes towards its own frontier (Figure 3.4d).

If a robot split from its buddy finds another branching point, a pair from the idle set is called and assigned to one of the two frontiers, while the single robot goes towards the other. This clarifies why the buddy system can be seen as a mix of the reserve mechanism and the divide and conquer mechanism. It both keeps the idle set waiting at the initial location until the moment at which it is needed, as reserve, and each leader goes with a follower from which splits when a branching point is met, similarly to divide and conquer. In this way, when a follower is turned into active, it is already near to the assigned frontier, providing a similar advantage of divide and conquer, without the disadvantage of having the whole team moving close.

3.4.2 Proactive mechanisms

In [13], three modifications for the mechanisms presented in [33] are introduced. As stated previously, they focus on enhancing the proactivity of the idle set. The way in which this is done is pretty straightforward for what concerns reserve and buddy system, while it is a little more tricky for divide and conquer. The mechanisms proposed are named *proactive reserve*, *proactive buddy system*, and *side follower*.

The main problem with reserve mechanism is that, once a robot from the idle set is turned into active, it has to move from the initial location to its assigned frontier. The distance it has to travel may be lower if the robot is moved to a nearer position while it is still idle. This is exactly what this modified mechanism tries to do by moving the idle robots to the barycenter of the polygon whose vertices are active robots positions. In this way, they are likely to be nearer to the newly detected frontiers which have to be explored.

The buddy system faces a similar problem as the reserve mechanism: the robots in the idle set wait to be turned into active at the initial location. Thus, the proactive version of the buddy system moves the idle pairs at the barycenter of the polygon formed by the active agents locations, for the same reason explained above concerning the proactive reserve.

The problem of divide and conquer is different, being related to the interference caused by moving the whole team of robots together. Therefore, the team of robots is split into groups of three at the beginning of the exploration and roles are assigned to them. The central robot is the leader, the right one is the right follower and the left one is the left follower. These roles are statically determined and never modified. Moreover, they affect the assignment of frontiers to the group members, being the frontiers along the direction of the movement assigned to the leader, the ones on its right assigned to the right follower and the ones on the left to the left follower symmetrically. In this way, there is a trade-off between the interference caused by the number of robots moving together and the distance from the frontiers, reducing the first one without affecting too much the second one.

Chapter 4

Modeling and evaluation

In the previous chapter, the coordination mechanisms developed in [13] have been presented, giving particular attention to the contribution provided by proactivity. Concerning buddy system and reserve mechanisms, their proactive versions move the idle set at the barycenter of the polygon formed by the positions of the active agents. This ensures good results when compared with the original mechanisms of [33], in particular for proactive reserve. However, the location for the idle set is computed naively and might be exploiting only a part of the available information on the structure of the environment. For example, the barycenter of the positions of the active robots may fall into a small room, from which a robot in the idle set would have to get out once turned into active, making the move into the room almost useless. On the contrary, we are more interested in moving the robots in the idle set towards vantage positions providing good starting points for them.

To include the structure of the environment discovered so far directly into the coordination mechanism, graphs are considered. They provide a representation of the free space and, by computing some suitable metrics, called centrality measures, it is possible to find out a subset of nodes more influential on connectivity between spaces. In this way, the location where the idle set is proactively moved is a central point of the environment. The concept of centrality varies accordingly to the centrality measure used to compute such subset of nodes.

In the following, the graphs and the centrality measures used in this work will be presented and described in-depth. The last section is dedicated to providing an overview of the criteria used to evaluate the different coordination mechanisms tested.

4.1 Graphs

The use of graphs allows including a topological component in the coordination, which is done employing an embedded graph. A graph G embedded to a surface Σ is a representation of G on Σ such that points of Σ and arcs in it are associated with vertices and edges of G, respectively. We exploit this idea by creating a graph G embedded on the map M and then through the measures of the centrality of a node, nodes are ranked based on their values for the two centrality measures used in this work.

The definition of an embedded graph is independent of the correspondence between points and nodes, and between arcs and edges, which leaves the freedom to adapt it on the needs. Based on this, two different types of graphs are defined in the next sections.

These graphs differ strongly on how they are defined and how they model the environment. The topological graph is more focused on the topological properties of the environment, nodes and edges represent the structure of the known free space. The model it provides is not strictly affected by team-dependent factors like agents distribution or team size, being built only considering the connectivity properties of the map built so far. On the contrary, the visibility graph defined below is built according to the positions of the agents moving in the environment. In this way, it includes information related to the disposition of the agents and the paths they followed. It is indirectly affected by the structure of the environment, being the agents only capable of moving in the free space.

4.1.1 Topological graph

From now on, we call *topological graph* a graph isomorphic to the graph used by the simulator to compute the paths followed by the agents during the exploration. As presented in [35], the construction of this graph is based on the structure of the occupancy grid known up to that moment, and its building will be discussed in detail in Chapter 5.

On one side, the use of this graph is backed up by the implementation of the navigation system, because it is computed just once for both navigation and proactivity. In this way, it does not introduce further costs in the building and the update. On the other side, it is pretty limited in its representation. It provides a topological view of the environment, but it is hard to extend with elements like frontier nodes or information about the location of the robots. These two aspects characterize the second kind of graph tested, the visibility graph.

In Figure 4.2a, the topological graph of the environment is shown. This one has been produced by applying the building procedure on the whole map of the environment, when it has been completely mapped. This has been considered to show more clearly the properties of this graph. The disposition of the nodes is almost uniform and this can be particularly noticed by looking at their distribution in the large open area, as opposed to the room in the upper right corner. In the large open area, nodes tend to form a grid around the columns, and the disposition of the nodes is more dense in proximity of the lateral walls. On the contrary, in the small room in the upper right corner, as well as in the three openings in the lower part, the nodes are more coarse, and also the number of edges is reduced.

This almost uniform distribution highlights strongly the difference in structure with the visibility graph, shown in Figure 4.2b, where nodes are provided with varying concentrations among the various spaces.

4.1.2 Visibility graph

The *visibility graph* as defined in this work consists of a graph built on the notion of visibility, rather than on that of the navigability from node to node. It is composed of two different types of nodes, the *pose nodes* and the *frontier nodes*.

The first kind is the one composing the vast majority of the graph and each pose node has an historical meaning, being a pose assumed by an active robot during the exploration. As active agents proceed in their mapping task, their locations are stored as pose nodes every time a re-plan for one of them or a new location for the robots in the idle set is needed. These two events happen quite frequently in the initial part of the exploration, and thus they trigger the graph building function several times. However, the frequency with which this is done is variable, for this reason, the distribution of the pose nodes might be coarse in some spaces and more dense in others. This allows keeping the number of pose nodes in the graph reduced with respect to the case in which every time an active agent moves, its pose is used to set up a pose node. This helps in dealing with the computational complexity of the centrality measures, even if some more precautions need to be taken, as will explained in Chapter 5. To enforce this aspect, the location of an active agent is added as a pose node as long as there are no other pose nodes within a certain radius. Once a pose node is added to the graph, it is fixed and never modified as the exploration goes on. The motion of agents in the idle set is excluded to avoid an excessive concentration of pose nodes in some crucial areas, which would negatively affect the computation

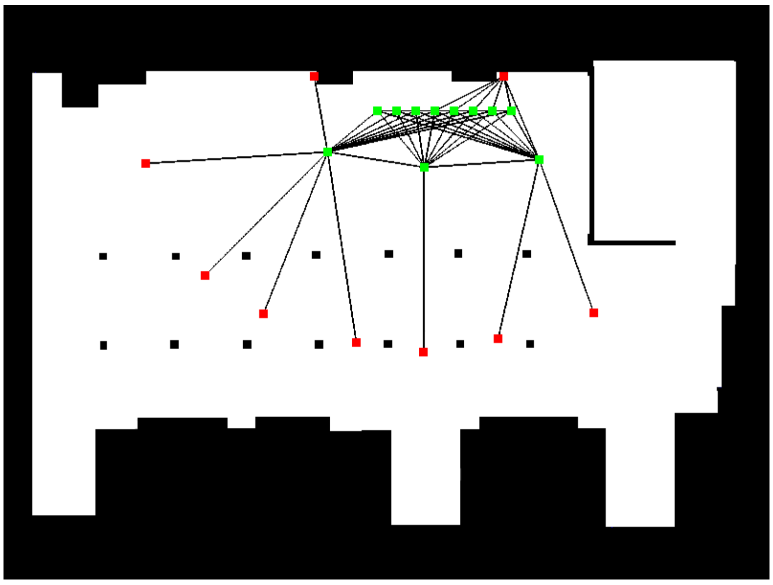


Figure 4.1: Visibility graph at the beginning of the exploration. Green dots represent pose nodes, connected by the straight black lines representing edges. Red dots are the frontier nodes

of centrality measures.

The second kind of nodes are the frontier nodes. As the name suggests, they allow to include the frontiers computed by the exploration strategy into the graph. This is a fundamental characterization of this graph that distinguishes it from the topological one. At each step in which the list of frontiers is updated, the same is done for the list of frontier nodes: the old ones which have been explored are removed and the new ones are added, making this type of nodes variable in time, differently from pose nodes which are fixed once they are put in the graph.

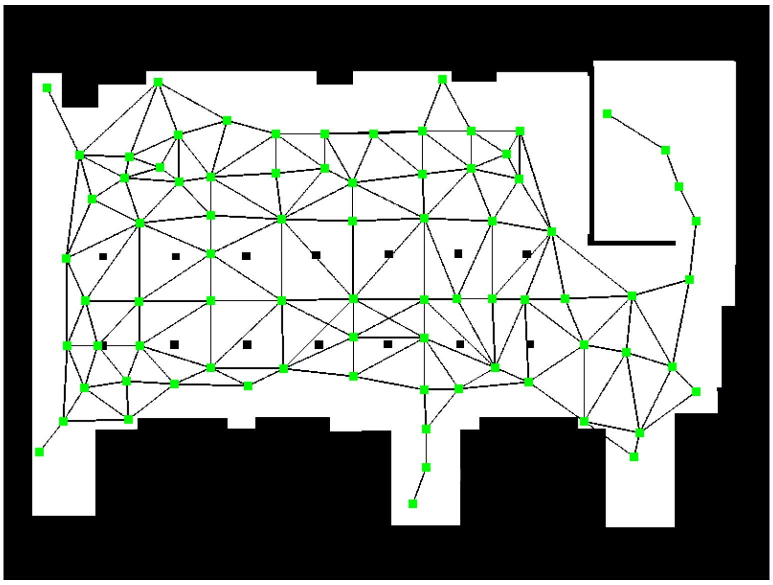
An edge models the notion of visibility: two nodes n_1 and n_2 are linked by an edge if and only if n_1 is within the sensing range of a robot placed in n_2 and there are no obstacles along the straight line segment connecting them. Moreover, it is excluded the possibility for an edge to go through unknown cells. This would be equivalent to assume free space in correspondance of the unknown space, and it would produce edges connecting nodes that are separated by obstacles not discovered yet. The relation of visibility is symmetric, thus the visibility graph is an undirected graph by construction, as the topological one. Edges are also characterized by a weight equal to the Euclidean distance between the nodes.

An example of visibility graph built in the initial steps of the exploration is reported in Figure 4.1. Due to the definition of the graph, both the number of nodes and edges is high with respect to the topological graph. This makes the representation of the graph hard to distinguish. For this reason, it is handful to see a small visibility graph, composed of few nodes, on which it is also possible to depict the edges connecting them. On the contrary, Figure 4.2b shows the same visibility graph at the end of the exploration. In this case, the high number of edges makes their representation not possible, therefore, they have been omitted. Accordingly to what stated previously, the distribution of nodes depends strongly on the paths followed by the robots and on the frequency with which the graph is updated. The structural differences with the topological graph in Figure 4.2a are clear just by looking at the two figures and all relate to the distribution of nodes, being widely less uniform. Moreover, there is also a remarkable difference in the number of nodes and edges between the two graphs, both being more in the visibility graph. The presence of frontier nodes, here painted in red, are also a discriminating factor between the two.

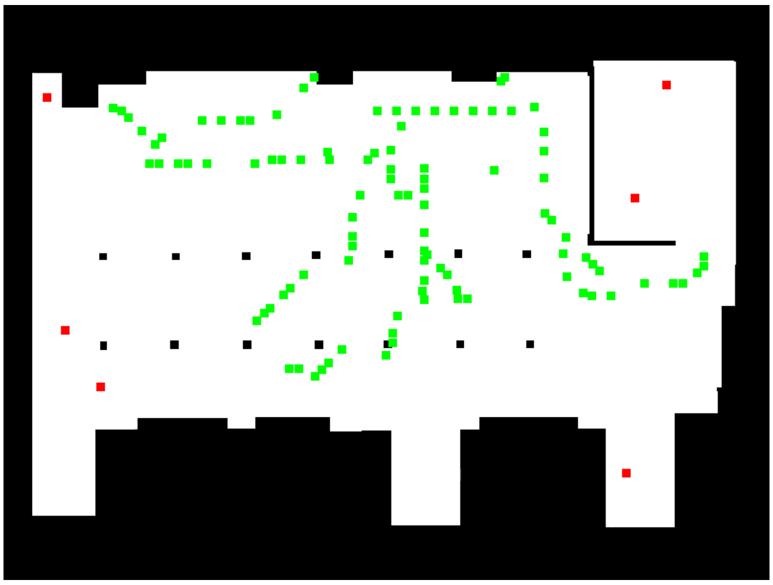
4.2 Centrality measures

Centrality measures have been exploited widely in the literature, especially in fields related to social networks [21], power grids [29], diseases [16], and computer virus spreading [45]. This is possible since centrality measures are applicable as long as the system is modeled by means of a graph and they provide a ranking of the nodes according to the metric applied. In fact, different measures may provide different rankings, depending on the network topology, because of the elusive concept of central node. An example of this is provided by the kite graph [27]. It is a simple graph composed of 10 nodes and 18 edges and it is depicted in Figure 4.3. The particularity of it is to be the smallest possible graph for which the nodes having the highest values of the three most basic centrality measures, namely degree, closeness, and betweenness, are all different.

To the author's knowledge, there are no previous works trying to apply the use of centrality measures to enhance the coordination of a team of robots performing exploration and due to the variability depending on the centrality measure used, in this work both closeness and betweenness have been tested. In the following sections, they are formally introduced with the reasons for their use.



(a) Topological graph. Green dots represent the nodes. Adjacent nodes are connected by an edge



(b) Visibility graph. Green dots represent the nodes, red ones are the frontier nodes. Edges are omitted for clarity reasons

Figure 4.2: Examples of graphs at the end of the exploration

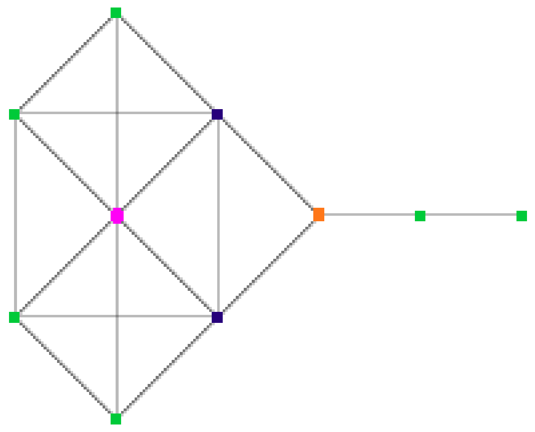


Figure 4.3: Kite graph. The pink node has the highest degree, the orange node has the highest betweenness and the blue nodes have the highest closeness. Green nodes are the remaining nodes

4.2.1 Closeness

The closeness centrality of a node is defined as the reciprocal of the sum of the lengths of the shortest paths between the node and all the other nodes in the graph [21]. This definition is strongly dependent on the number of nodes N in the graph, thus closeness is usually normalized by multiplying for $N-1$. In this way, closeness of a node can be defined as the reciprocal of the average distance between the node and all the other nodes and allows to compare its value for graphs of different sizes.

Formally, let m and n be nodes of the graph, and let d be a real-valued function which provides the length of the shortest path connecting two nodes, then the normalized closeness value $C(n)$ is defined as

$$C(n) = \frac{N-1}{\sum_{m\neq n} d(n,m)}$$

with N the total number of nodes in the graph, as defined above.

The original formulation of closeness centrality considers only unweighted graphs, but it has been extended to be applied also to weighted ones. The formal definition is the same assuming an appropriate modification in the implementation of the distance function d. Indeed, in an unweighted graph, it simply has to count the number of edges along the shortest path linking the two nodes in input, while on a weighted graph the cost of traversing an edge is equal to its weight and thus it has to be accounted accordingly [30]. In the latter case, the distance function d is $d(m, n) = \sum_{e \in E} w(e)$, where m and n are nodes of the graph, E is the set of edges of the graph composing the shortest path from m to n, and w is a real-valued function returning the weight of an edge.

Closeness tends to consider central nodes the ones whose distance from all the other nodes is lower on average. Therefore, going back to an exploration context, placing an agent at the location corresponding to the highest closeness node makes it possible to reach an assigned position, not known before, in an expected time lower than any other starting location with a lower closeness. This reasoning has been applied to the idle set of robots, which, placed in the node with the highest closeness, are likely to already be in a good spot when turned into active.

In Figure 4.4 an example of the high closeness nodes is shown for each kind of graph. For the sake of the example, as high closeness nodes are considered nodes with a value of closeness higher than the 75% of the max value. It is interesting to point out how these nodes are distributed mostly on the whole central area of the large open room for the topological graph (Figure 4.4a), while the pose nodes with high closeness are located in a narrower region of the central area in the visibility case (Figure 4.4b). Moreover, the distance between the nodes with highest closeness is significantly different for the two graphs.

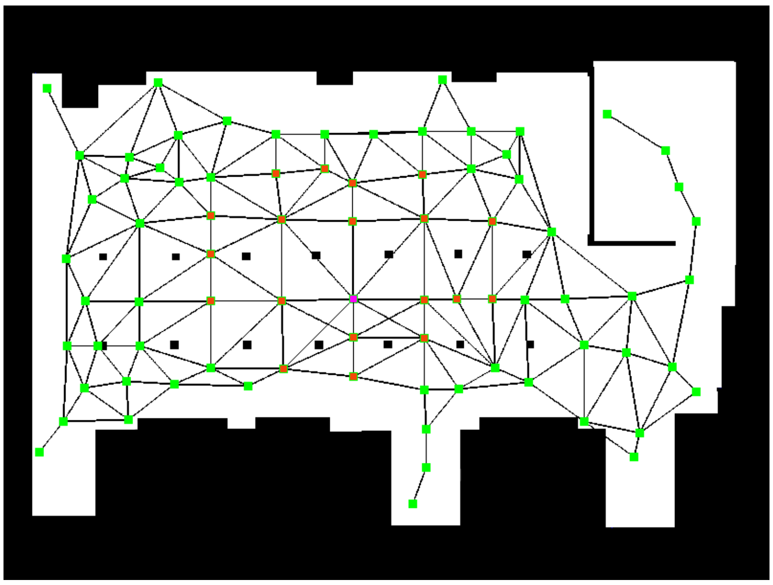
4.2.2 Betweenness

Betweenness centrality for a node of the graph measures how many of the total number of shortest paths between other nodes passes through that node [21]. Let m, n, and v be three nodes in a connected graph, σ_{mn} be the total number of shortest paths connecting m and n, and $\sigma_{mn}(v)$ be the number of those paths which go through v, then the betweenness $B(v)$ for the node v is defined as

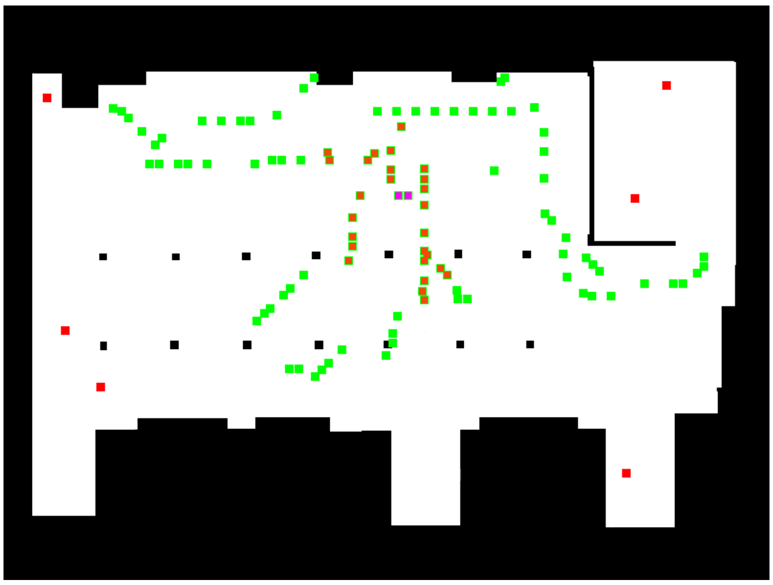
$$B(v) = \sum_{m \neq v \neq n} \frac{\sigma_{mn}(v)}{\sigma_{mn}}$$

where the sum is performed over each pair of nodes in the graph. The graph needs to be connected, otherwise, one σ_{mn} where m or n is a disconnected node would result in a division by zero. However, this is always granted in the context of this thesis because of the way in which graphs are built.

Betweenness followed an evolution similar to the one of closeness, being at first defined on unweighted graphs [21] and then extended to the case of



(a) Topological graph closeness



(b) Visibility graph closeness

Figure 4.4: Distribution of nodes with a closeness higher than the 75% of the max value for the topological and the visibility graphs. Purple dots are the highest closeness nodes, orange dots are the ones with a high value of closeness but not the maximum. Green dots are the nodes with a low value of closeness, and red dots are the frontier nodes. For the visibility graph, edges are omitted for clarity

weighted ones [30]. In the case of weighted graphs, weights impact on how shortest paths are computed, making it necessary the use of algorithms like Dijkstra's or Breadth-First Search to deal with them. Once the shortest paths are provided, the algorithm to compute the betweenness is the same. This is similar to closeness, where the introduction of weights on the edges only affects the computation of the distances between nodes.

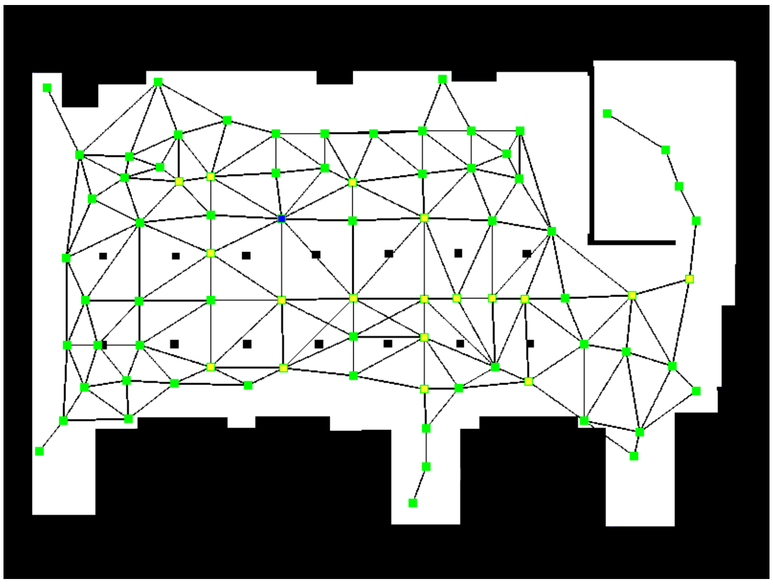
The idea behind this metric is to assign higher importance to the nodes which are along more shortest paths linking pairs of other nodes. In different works about social networks analysis [21], betweenness is used to find out which are the nodes having more control over the information flow. Nodes with a high value of betweenness are along more shortest paths, thus more information goes through them and their possible disconnection may cause loss of information or a separation of the graph into two sub-graphs. In other words, a node of this kind is likely to be fundamental for what concerns the connectivity of the graph, differently from a node with low betweenness. According to this, the approach based on betweenness has been conceived. Being interested in an effective positioning of the idle set of robots, a location with a high value of betweenness is an ideal candidate because it represents a crucial point for the connectivity of the environment and when the robots of the idle set are turned into active, it is likely that they are in a good position to navigate towards the assigned frontier.

In Figure 4.5, the nodes with high betweenness are plotted over the representation of the graphs provided before. The first thing that catches the eye is the difference in the number of this kind of nodes between the topological and the visibility graphs, being a lot more in the first one. This can be justified by considering the reduced size of the graph in that case, which makes every node likely to be on more shortest paths, thus with a higher value of betweenness.

4.3 Comparative metrics

The different coordination mechanisms proposed are compared based on a practical measure, as the time taken to fulfill the termination criterion, and on theoretical-based measures, namely interference among robots and their availability.

The use of time as comparison metric is intuitive when looking at some of the application contexts. Teams of robots are often used in search and rescue scenarios, where the time needed to complete the exploration is a fundamental aspect to take into account [10,33,38]. Thus, a faster approach is overall preferable to a slower one in such a scenario.


(a) Topological graph betweenness

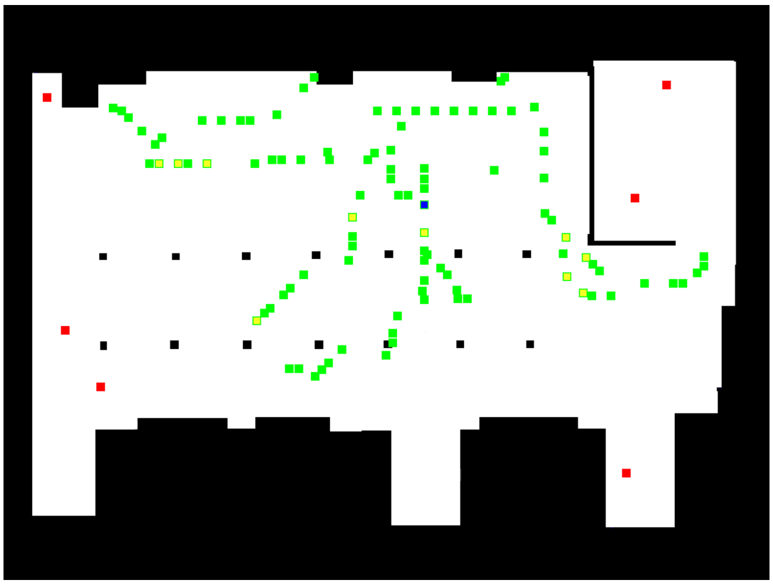

(b) Visibility graph betweenness

Figure 4.5: Distribution of nodes with a betweenness higher than the 50% of the max value for the topological and the visibility graphs. Blue dots are the nodes with highest betweenness, yellow dots are the ones with a high value of betweenness but not the maximum. Green dots are the nodes with a low value of betweenness, and red dots are the frontier nodes. For the visibility graph, edges are omitted for clarity

In the simulations run in this work, the termination criterion used is the exploration of the 95% of the environment. At the end of each run, the number of discrete time steps taken is stored and then compared during the analysis of the results. The decision of considering the number of time steps rather than the absolute time taken by the exploration has been carried on to exclude machine-dependent aspects from the results. The mechanisms implemented, in particular those computing betweenness, are way more computationally intensive than the ones based on the barycenter computation, and comparing their results on the effective time taken would have been faked by the computing capability of the machine running simulations.

Time taken to complete the exploration has been considered sufficient to characterize from a practical point of view each mechanism analyzed. Nevertheless, two further measures are used to evaluate the mechanisms, which are named interference and availability. They are at first introduced in [33] and formalized in [13].

4.3.1 Interference

Interference quantifies the average distance held by agents during the exploration and the higher the distance, the higher the value. A high value of interference is desirable because as the average distance among robots increases, it reduces the possibility of crashes and the complexity in the management of the system. Moreover, it is also an indirect measure of how parallel the exploration is being carried on because it increases as the robots are spread on the environment. According to this, the value of interference for a particular coordination mechanism can provide useful information about the amount of parallelizability exploited as compared to other mechanisms. The value of interference can also be correlated to the features of the environment presented in Chapter 3, and may provide insights useful to classifiy an environment according to those features, as shown in Chapter 6.

The value of interference λ for an agent a is computed at each step t of the exploration by calculating the average distance between a and all the other agents a' in the team A, thus it can be formally written as

$$\lambda_t(a) = \frac{1}{|A|-1} \sum_{a' \in A | a' \neq a} d\left(p_t(a), p_t(a')\right)$$

where $|A|$ is the size of the team, p_t is a function providing the position of an agent at time t, while d is the function that computes the distance between two positions of the environment.

This definition only relates to one agent at a particular instant of the exploration. To completely characterize the value of interference for the whole exploration, it needs to be at first generalized over the set of agents composing the team and then over the entire time needed to complete the exploration. In this way, its value computed for a specific coordination mechanism is comparable with the ones provided by other mechanisms over the same exploration problem.

The first generalization consists in extending the definition of interference to all the robots in the team, rather than considering only a single robot, and this is simply done by averaging the values of interference of each agent:

$$\lambda_t = \frac{1}{|A|} \sum_{a \in A} \lambda_t(a)$$

At this point, it is possible to integrate this expression over the entire time interval taken by the exploration, which is considered to take values in $[0, \ldots, t_T]$ with t_T being the time step of the fulfillment of the termination criterion T. The definition of the interference for the coordination mechanism applied to a specific instance of the exploration problem considered is

$$\lambda = \frac{1}{t_T + 1} \sum_{t=0}^{t_T} \lambda_t$$

This states that the interference for the whole exploration can be computed by averaging over the total time taken the values of interference for the team. In this way, the value of interference obtained can be exploited to compare different coordination mechanisms, avoiding that differences in the duration of the exploration impact on this measure.

4.3.2 Availability

Availability is a measure of the distance between an agent and its assigned location. It can be formally defined at first for a single agent a at a certain time step t as

$$\alpha_t(a) = d(p_t(a), g_t(a))$$

where α is the symbol used for the availability and g_t is a function returning the location of the frontier assigned to the agent in input, if it is an active agent, or the location where the agent is proactively moved, if it is an idle agent. The two auxiliary functions d and p_t are the same presented in the previous section to define the interference.

Similarly to what done for the interference, this definition can be generalized to the whole team and the whole exploration. Following similar reasoning and with the same meaning of the symbols, the availability for the whole team at a certain instant t of the exploration is

$$\alpha_t = \frac{1}{|A|} \sum_{a \in A} \alpha_t\left(a\right)$$

recalling that A is the set of agents composing the team and $|A|$ is its cardinality. Averaging this result over the whole exploration time t_T defines availability for a coordination mechanism applied to a specific instance of the exploration problem, thus it turns out to be

$$\alpha = \frac{1}{t_T + 1} \sum_{t=0}^{t_T} \alpha_t$$

This value of the availability is independent of the time taken to meet the termination criterion and therefore allows a comparison among different coordination mechanisms, without being affected by their respective performance in terms of time. It is important to highlight how availability has the opposite trend of interference. Indeed, a mechanism with low availability assigns robots to locations near their current positions. In this way, agents can scan unknown portions of the environment sooner with respect to a mechanism with an higher value of availability because of the shorter distance to travel to reach the assigned location.

This metric has a two-fold interpretation. From one side, it shows whether a coordination mechanism assigns robots to far or close targets. On the other side, it can provide insights on the effectiveness of the proactivity when comparing two mechanisms which differ only in this aspect. This, in particular, is the setting of this thesis. The differences of the mechanisms analyzed concerns only the proactive allocation of the idle set, and thus a mechanism that has a lower value of availability is likely to place this set of robots in a position nearer to the frontiers. Strong evidence of this relation between proactivity and availability is provided by [13] in the comparison between reserve and proactive reserve, where by moving the idle set towards the barycenter of locations of the active agents, the latter method ensures agents to travel a shorter distance once turned into active with respect to the former mechanism.

It is worth to point out also that the absolute value of both the availability and interference is highly affected by the particular configuration of the environment explored and by the team size, for this reason, comparisons among mechanisms based on them do make sense only if done on the same instance of exploration problem.

Chapter 5

Coordination mechanisms

This thesis is aimed at exploiting measures related to the environment into a proactive allocation of idle agents. This tries to be a further improvement of the mechanisms proposed in [33], with respect to [13]. The proposed mechanisms are analyzed in detail in this chapter, giving particular attention to how the various elements presented before, namely graphs and centrality measures, are employed.

At first, a common structure for the coordination mechanisms previously introduced in Section 3.4 is defined, and it is shown how a different implementation of the various functions allows the definition of the single mechanisms. This is done starting from reserve and buddy system, as presented in [33], being the building blocks of the other mechanisms proposed. Then, the focus shifts to the proactive versions developed by [13]. At this point, it is presented how the proposed proactive mechanisms modify some components of this common structure to include the use of topological aspects.

5.1 Original mechanisms

Reserve and buddy system have been introduced intuitively in Section 3.4.1, and here a more detailed account of how they are implemented is given. Recalling that the buddy system is conceived as a mix from reserve and divide and conquer mechanisms, it is clear why they share some algorithms.

The general structure for the coordination mechanisms analyzed can be roughly summarized of four elements:

- planning function;
- activation function;

- goal function;
- proactivity function.

For some aspects, the planning function works as a common interface provided by the mechanism. As the simulation requires an agent to move, the step to take is returned by this function. It calls the appropriate function coherently with the state of the agent, whether it is active or idle. It also initializes the exploration by setting the starting agent and forming the active and the idle sets.

The remaining three functions allow characterizing a specific coordination mechanism. They enclose the logic about the assignment of frontiers to active agents, what triggers idle agents to be turned into active, and where they are waiting until that moment.

In particular, the proactivity function is the focus of this work. In the previous chapter, the importance of graphs and centrality measures has been introduced. The last section of this chapter shows how these elements are included in the proposed coordination mechanisms, giving also attention to some optimizations performed to avoid an excessive impact of their computation.

5.1.1 Reserve

Reserve mechanism divides the team of robots into two sub-teams, one composed of active agents and the other composed of idle ones. This separation is done by the planning function as soon as the exploration begins, once the first set of frontiers is computed.

At first, the distance of each agent from each frontier is computed, then the agent of the pair with the lowest distance is set as the starting agent and assigned to that frontier. This assignment is iterated until every frontier is assigned to an agent or vice versa. The set of assigned agents composes the active set, while the remaining agents if any, form the idle set. After this initialization step, the planning function is invoked either every time an agent has reached its goal or a certain amount of time has passed since the last planning. In both cases, it is in charge of calling the right function among the activation function, the goal function, or the proactivity one.

Once an agent is active, which frontier is assigned to it is computed by the goal function. To do this, the planning function provides it with the updated list of frontiers. The assignment is simply done considering the closest frontier to the agent location.

The activation function can be invoked only for agents of the idle set, being the function aimed at turning them into active. This function checks whether there are any unassigned frontiers and, if so, the agent is turned into active and assigned to the closest one. If this is not the case and all the frontiers are assigned, the proactivity function is called.

The reserve policy is such that the idle set waits at the initial location and thus, the proactivity function always returns the current position of the agent. In this way, agents of the idle set stay still, until the activation function assigns them to a frontier.

5.1.2 Buddy system

Buddy system proceeds in a way similar to the reserve one. This method is characterized by the use of paired robots and each robot in the pair is the buddy of the other, from this the name of the mechanism. Pairs are created before the exploration begins, by assigning a role to each robot of the team. The roles are two, either an agent is a leader or a follower and they are assigned to split the team into two halves. In the case of odd number of robots, the extra agent is assigned leader role.

Apart from this division into leaders and followers, the team is also split into an active set and an idle set. The active set is composed of leaders assigned to frontiers and their buddies. The idle set is composed of the pairs waiting at their initial locations.

The first call to the planning function selects the starting pair. This is done similarly to the reserve mechanism, by looking for the robot of the team whose distance from the closest frontier is lower. If it is a leader, it is assigned to that frontier. If it is a follower, its buddy is retrieved and being it the leader, it is assigned to that frontier. As for reserve, this approach is iterated as long as there are no more leaders or frontiers to assign. All the leaders assigned to a frontier and their buddies compose the active set. The idle set is composed of the pairs whose leader is not assigned to a frontier, if any.

Further calls to the planning function distinguish whether the calling agent is a leader or a follower. In the former case, it simply invokes the goal function, while in the latter it checks whether the pair needs to be split. This happens when the two closest frontiers are enough distant from each other. This is a branching point and the pair is split. The leader is assigned to the closest frontier. The follower is now considered a leader itself and is assigned to the second closest frontier. In the case in which the two closest frontiers are not enough distant from each other, the point is evaluated not to be a branching point, then the goal function for the follower is called. In fact, the goal function is implemented differently whether the considered agent is a leader or a follower.

The goal function for leaders is the same as reserve. It assigns the agent to the closest frontier, while the goal function for the follower moves the agent towards the frontier assigned to its leader. This makes the pair go together in the direction of the assigned frontier.

The activation function for the buddy system follows the same principle of the reserve one. If there are unassigned frontiers, the closest idle leader to each one is computed and turned into active by assigning that frontier to it. This has the side effect of turning into active also its buddy because the goal function will now make the follower follow its leader.

Buddy system keeps the idle set waiting at the initial position. Therefore, when called by an agent, the proactivity function simply returns the current location of that agent.

5.1.3 Base proactive mechanisms

Both the proactive buddy and the proactive reserve employ the same activation, planning and goal functions described for the original mechanisms. The only difference is in the proactivity function.

The proactivity function in the previous mechanisms is trivially implemented and consists only of returning the position of the agent calling it. This makes the agents stay still at their initial locations until the activation function turns them into active. As discussed in the previous chapters, this makes the average distance between the agents and the assigned frontiers higher, penalizing performance. To overcome this, a new proactivity function has been proposed in [13]. Robots of the idle set are moved to a position likely to be nearer the future assigned frontiers. This position will be referred to as *proactivity goal* in the following. Experimental results show this modification to actually be an enhancement in the case of reserve mechanism. The proactive version outperforms the base version on almost every environment on which it has been tested. The proactive buddy system, on the contrary, is impacted less by the proactivity.

For both proactive reserve and proactive buddy system, the proactivity function assigns to idle agents a proactivity goal computed as the barycenter of the polygon whose vertices coincide with the locations of the active agents. It can be formalized as

$$g_t(a) = \frac{1}{\mid Act \mid} \sum_{a' \in Act} p_t(a')$$

recalling the notation from the previous chapter, where a and a' are agents, in particular, a is an idle agent and a' iterates over Act, the set of active agents. $p_t\left(a'\right)$ is a function providing the location of the agent a' at the time t and $g_t\left(a\right)$ is the proactivity goal assigned to the idle agent a at time t. The subscript t for the goal function is possible since the locations of the active agents at time t are used to compute the barycenter. This is coherent with the communication capabilities assumed for the agents, which are always able to communicate, and there is no delay in the propagation of the information.

This proactivity function is characterized only by the actual locations of the active agents. The environment is not explicitly included in this formula. To take its features into account directly, an approach based on graphs and centrality measures is proposed in the following section, characterized by a different implementation of the proactivity function.

5.2 Proposed proactive mechanisms

The proposed proactive mechanisms differ from the previously defined ones mostly for the proactivity function. There are no other conceptual differences in the other elements of the structure. However, a small variation to the implementation of the goal function is introduced, as it will be pointed out in the next section.

The proactivity function is modified in a way to include the computation of either the closeness or the betweenness and to use their values to obtain the proactivity goal for an agent. Independently of the measure needed, the proactivity function follows the same steps. At first, it is checked if the graph representing the environment has been modified since the last computation. If not, the proactivity goal returned is the same as the previous step, avoiding useless computations. Otherwise, a new proactivity goal has to be evaluated. This is done by determining the value of the considered measure for the nodes of the graph and then retrieving the most central one, i.e., the one with the highest value of the measure. If more than one node is found, the returned point is their barycenter. However, this case has a very low frequency and the number of nodes in a tie never exceeds two during the experiments.

As presented in the previous chapter, the proposed proactive mechanisms exploit two different types of graphs, combined with two centrality measures. All their combinations have been tested both applied to the reserve and the buddy system, for a total of four different proposed variants for each mechanism.

In the following sections, it is presented how the mechanisms have been adapted to include the generation of the graphs and some optimizations aimed at allowing an efficient computation of the centrality measures. These were needed because of the high complexity of the algorithms used, which made the average simulation cycle to greatly increase its duration.

5.2.1 Graph building process

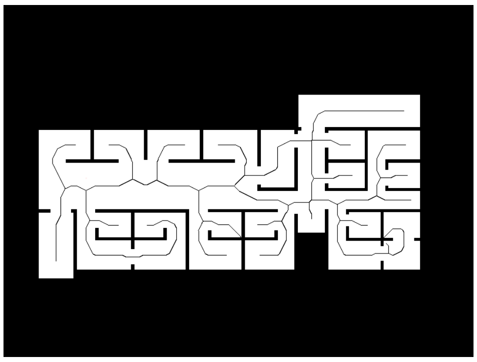
In the proposed mechanisms, two types of graphs are tested. As already stated in Chapter 4, they are referred to as topological graph and visibility graph. Both have been implemented and tested to check how their use would affect performance. Their structures are built to enforce different aspects of the environment. The topological graph is built on the notion of *navigability*, that is, it aims at capturing the connections among spaces. The visibility graph is based on the concept of *visibility*, which is more related to the possibility for the robots to perceive a certain location from another one through their sensors.

Recalling that the map is modeled as an occupancy grid, each node is characterized by its coordinates in the global coordinate system. Both graphs have been implemented employing adjacency lists [15]. Each node holds a list containing all the nodes linked to it and the Euclidean distance separating them. This has been preferred over a representation through adjacency matrices because both the graphs tend to be sparse, in particular the topological one. To confirm this, the average degree of the nodes of the topological graph is two, whereas it has a high variability for the visibility graph. However, even in open environments, where the number of nodes visible from a node is high, the average degree is always lower than half of the total number of nodes.

In the following, how these graphs are built is shown in detail, starting from the topological graph, then focusing on the visibility graph.

Topological graph

The topological graph is a graph isomorphic to the one used by the navigation system to compute the paths followed by robots [35]. In the following, this last one is referred to as navigation graph to avoid confusion. Whether it is toward a frontier for an active agent or toward a proactivity goal for an idle one, every time the path from an agent position to its goal is needed, the navigation graph is checked. In case the time since the last update is higher than a threshold or the occupancy grid of the environment has changed, it is recomputed. This is done in the following way. The obstacle cells of the


(a) Skeleton of the free space, represented by the black line

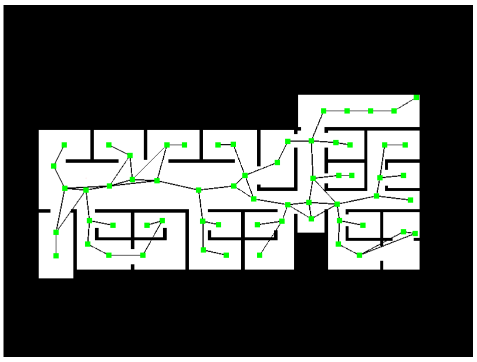

(b) Navigation graph. Green dots represent the nodes

Figure 5.1: The skeleton of the free space, and the navigation graph obtained from it through discretization

occupancy grid are progressively enlarged until the skeleton of the free space is found. An example of skeleton is shown in Figure 5.1a. It is a line connecting the various sections of the environment, through the free space. The skeleton is then discretized into a set of nodes that are going to be the nodes of the navigation graph. Discretization starts by selecting the first kind of nodes corresponding to branching points, then it looks for points filling the gaps among them. These points are added as nodes if the distance from the closest node is higher than a certain fixed value. At this point, the set of nodes for the navigation graph is complete but a further pruning is applied to remove the nodes too close to an obstacle. Each cell of the occupancy grid is then mapped to the nearest node and this provides a partition of the map into different polygons, each one associated with a node, which is useful in speeding up the computation of the shortest paths. Two nodes are adjacent if the polygons associated with them share one side and this allows finding the edges of the graph, completing its construction. The navigation graph computed on the same skeleton of Figure 5.1a is reported in Figure 5.1b. In this way, it is possible to compare how the skeleton is discretized, and the way in which nodes are connected through edges.

Once the navigation graph is computed, the topological graph is built with the same set of nodes and with the same adjacency relations. The only difference is that the edges of the navigation graph do not keep track of the distance between nodes, while the edges of the topological graph do. Indeed, the topological graph is a weighted undirected graph, where the weight function is a distance function. Given the structure of this graph, the difference between Euclidean distance and the effective length of the path between the two nodes is negligible, thus the first one has been used for efficiency reasons.

The update of the topological graph is equivalent to the computation of a new navigation graph and it is done every time a path for an agent is needed. Thus, referring to the general structure of a coordination mechanism presented before, it may be done by each of the four functions because each one implies the computation of a path for the agent. This ensures the topological graph be updated frequently.

Visibility graph

As presented in Chapter 4, the visibility graph is composed of pose nodes and frontier nodes. Pose nodes represent the positions assumed by the robots at various time instants. Frontier nodes are used to include the location of frontiers known at a certain moment into the graph.

It is initialized at the beginning of the exploration, considering as pose nodes the set of initial positions of the robots. The frontier nodes are included as soon as the first scan provides the first set of frontiers. After this, it is never rebuilt from scratch, like it happens for the topological graph, rather the visibility graph is progressively updated to match the movements of the robots, the discovery of new frontiers, and the exploration of old ones. This holds because its construction is carried on entirely during the calls to the goal function. In both the reserve and the buddy system, this function comprises the update of the list of frontiers to properly assign robots. This provides the possibility to remove old frontier nodes and include the new ones. During this step, also the actual locations of active robots are used to generate pose nodes of the graph and to add them. Once all these nodes have been included, edges are generated. They model the notion of visibility, thus an edge links two nodes if and only if a robot located in one node is able to perceive the other node. This is a restrictive condition, in fact if two nodes are linked by an edge, then it is granted the existence of a straight path connecting them. On the contrary, knowing that exists a path connecting two nodes provides no information about the possibility for a robot located in one node to perceive the other node. A simple example of this is the case of two nodes located on opposite sides of a wardrobe. It is clearly possible to define a path connecting them, but the sensors of a robot placed on one side cannot provide measurements about what is on the other side. Moreover, edges built in this way are straight lines and as for the topological graph, the distance function providing their weight is the Euclidean distance.

5.2.2 Centrality measures

The main problem with closeness and betweenness is related to their computational complexity. They require the knowledge of the shortest paths connecting each pair of nodes and of their length. This is particularly true for betweenness, while closeness only needs the last piece of information. As the size of the graph grows, it may be difficult to compute the proactivity goal in a reasonable amount of time. For this reason, two major optimizations have been performed. The first one is to compute the matrix of the distances between each pair of nodes in an efficient way and store it, allowing to avoid the repeated computations of the length of the shortest paths. The second one relates to how betweenness is computed.

Distance matrix

The distance matrix is a square matrix containing the distances between each pair of nodes of a graph. Both the topological and the visibility graphs are weighted undirected graphs, thus it is useless to have a square matrix, being it symmetric by construction. Moreover, the weights considered are always non-negative, for this reason, the distance of a node from itself is always zero. This allows reducing the size of the matrix from $N \times N$ to an $(N-1) \times (N-1)$ triangular matrix, where N is the number of nodes.

This data structure is essential in speeding up the computation of closeness. Recalling that the closeness of a node is defined as the inverse of the average distance of it from all the other nodes, it can be computed as

$$C(n) = \frac{1}{avg\left[row(n) \cup col(n)\right]}$$

where $row(n)$ and $col(n)$ provide respectively the elements of the row and the column of the distance matrix associated to node n. While avg is simply the function computing the average. The union is to ensure considering each element, being the distance matrix triangular. Thus, in a single sweep of the whole matrix, the value of the closeness of each node can be computed.

The main issue about the distance matrix is its construction. It is an instance of the All Pairs Shortest Path problem, that is the problem of finding the shortest paths linking each pair of nodes. An algorithm able to solve this is the Floyd-Warshall algorithm [15].

Consider a graph G which nodes are $V = \{1, 2, \ldots, N\}$ and a subset $K = \{1, 2, \ldots, k\}$ of V for a generic k. Let m and n be two nodes of G, and p the shortest path connecting them composed only of vertices in K. The path p is an ordered sequence of nodes $p = \{m, i_1, \ldots, i_j, n\}$, starting with m, the source node, and ending with n, the arrival node. The subset of nodes $\{i_1, \dots, i_j\}$ is the set of *intermediate nodes*. It is composed of the set of nodes composing a path without the starting and the arrival nodes. At this point, k can be an intermediate node of p, or not.

If k is an intermediate node of p, then the length of the path from m to k, plus the length of the path from k to n is lower than the one from m to n having nodes in $K \setminus \{k\}$ as intermediate nodes. This holds because the path p going through k is the shortest path between m and n. On the contrary, if k is not an intermediate node, then this last path is shorter. By progressively increasing the value of k to consider the whole set of nodes as possible intermediate nodes, it is possible to find all the shortest paths for each pair of nodes of the graph.

The complexity of the algorithm is $\Theta\left(N^3\right)$, where N is the number of nodes in the graph. This is manageable in the case of the topological graph because N is limited by the graph building algorithm. On the other hand, the number of nodes in the visibility graph is much higher because no pruning is applied. A possible solution consists of forcing a minimum distance within pose nodes. However, as the exploration goes on, N might be sufficiently high to make the application of the Floyd-Warshall algorithm heavy. To overcome this issue, two factors allow to come up with an efficient solution.

First of all the visibility graph is updated at every step and is never built from scratch, after the initialization. The second main aspect is that the Floyd-Warshall algorithm is an example of dynamic programming, thus the solution provided for the graph at time t_1 can be used as a starting point to compute the solution for the graph at time $t_2 > t_1$. In fact, paths computed with a reduced set of intermediate nodes, suppose $\{1, 2, \dots, k-1\}$, are the shortest paths possible if the graph had nodes in $\{1, 2, \dots, k-1\}$. Then, the extension of the intermediate nodes to $\{1, 2, \dots, k\}$ provides the optimal solution for the graph with nodes in $\{1, 2, \dots, k\}$. This idea can be applied by looking at the graph at time t_1 like the one with intermediate nodes in $\{1, 2, \dots, k-1\}$, while the one at time $t_2 > t_1$ as the one with intermediate nodes in $\{1, 2, \dots, k\}$.

In this way, the distance matrix of the visibility graph is computed at the beginning of the exploration and consequently updated as new nodes are added to the graph. The complexity of each update is linear in the number of nodes, including the new ones.

Apart from computing all the distances between each pair of nodes, the Floyd-Warshall algorithm also allows to keep track of the shortest paths linking them. This is fundamental to optimize the computation of betweenness, as discussed in the following section.

Betweenness

Efficient computation of betweenness requires the preemptive computation of the shortest paths between each pair of agents. Assume to have these paths stored in a matrix, that, similarly to the distance matrix, is an $(N-1)\times(N-1)$ triangular matrix. Even if this is available, finding the betweenness $B(v)$ of a node v requires to go through each pair of nodes m and n, such that $m \neq v \neq n$, and retrieve the list of shortest paths linking them. The number of paths composing this list is equal to the total number of paths connecting the two nodes. This value has been referred to as σ_{mn} in Section 4.2.2. To compute betweenness for the node v, it is also needed $\sigma_{mn}(v)$. Thus, the list of shortest paths linking m and n needs to be iterated to count the number of shortest paths going through v. At this point, the ratio of $\sigma_{mn}(v)$ and σ_{mn} can be computed, and the result has to be summed with the same value obtained from each other pair of nodes, m and n different from v, in the graph. Similarly to the application of the Floyd-Warshall algorithm, this algorithm may be a problem for the case of a visibility graph, where the number of nodes N might increase enough to make the computation unfeasible in a reasonable amount of time.

The main aspect of improvement is in the computation of $\sigma_{mn}(v)$, the number of shortest paths from m to n going through v. This is made possible by the *Bellman criterion* [8], which states that a node v of a graph lies on a shortest path between two nodes m and n, if and only if $d(m,n) = d(m,v) + d(v,n)$. Even if apparently obvious, this criterion allows to compute $\sigma_{mn}(v)$ as

$$\begin{cases} 0 & \text{if } d(m,n) < d(m,v) + d(v,n) \\ \sigma_{mv} \cdot \sigma_{vn} & \text{otherwise} \end{cases}$$

where σ_{mv} is the value of the count of the shortest paths from node m to node v, and σ_{vn} is the analogous from v to n. The idea is that if v is along a shortest path from m to n, then $\sigma_{mn}(v)$ can be computed as the number of shortest paths from node m to node v multiplied by the number of the ones from v to n, to include all the possible combinations.

Exploiting this information, the betweenness can simply be computed according to the definition in Section 4.2.2. This also allows avoiding the storage of the whole list of shortest paths for each pair of nodes because it is only needed their count and the distance matrix. The Floyd-Warshall algorithm can be suitably modified to fill the matrix of counts by updating it as new shortest paths are found. This can be done while checking if the length of the path going through the considered intermediate node is lower than the previously known path. If the new path is lower, then the counter is reset to one. If the length is equal, then the counter is increased by one. If the new path is longer, then the counter remains equal because the new path is not a shortest path.

Once both these matrices are completed, computation of the betweenness for the whole set of nodes requires for each node to iterate over all the possible pairs of other nodes. Thus, the computation has cubic complexity in the number of nodes but it is independent of the length of the paths considered. On the contrary, this would affect the computation in the case the entire set of paths had to be scanned to check whether node v was present or not.

Reduced node-set

The optimizations provided before to speed up the computation of the measures, in particular for betweenness, work well in practice. This happens both for the topological graph and for the visibility graph, where the number of nodes is higher, even of two to three times. In some cases, this ratio can also go up to four times, producing an impact on the computation. After all, the complexity holds a cubic relation with the number of nodes.

Moreover, another aspect taken into consideration is related exclusively to the visibility graph. The value of both the centrality measures for frontier nodes is almost uninformative. This can be explained easily considering that frontier nodes are marginal nodes on the edges of the graph, because of their definition. Furthermore, being the sensing range longer than the distance a robot covers in a time step, the distance between pose nodes is lower than the distance between pose nodes and frontier nodes. Thus, the impact of frontier nodes on the value of the centrality measures is negligible.

To deal with both these theoretical and empirical considerations, it has been decided to restrict the set of nodes for which centrality measures are computed. The reduced node-set R is then composed of pose nodes n such that n either is adjacent to a frontier node n_f or is adjacent to a pose node n' adjacent to n_f. To have a clear definition of how this set is composed, it is worth defining a function $adj(n)$ for a node n of a graph returning the set of nodes adjacent to n and indicating as F the set of frontier nodes. An auxiliary set A_F can be defined as the set of nodes adjacent to a frontier node, namely

$$A_F = \bigcup_{n_f \in F} adj\left(n_f\right)$$

Then, the definition of R can be rewritten as

$$R = \left(A_F \cup \bigcup_{a_f \in A_F} adj\left(a_f\right) \right) \setminus F$$

R composed in this way allows a trade-off between considering the whole set of nodes and just the ones adjacent to a frontier node.

Thanks to all the tricks exposed so far, it has been possible to reduce the average simulation cycle time of the proposed mechanisms to values comparable to the ones of the mechanisms exploiting the barycenter and used in [13].

Chapter 6

Experimental evaluation

In this chapter, the experimental evaluations are presented. As already pointed out, the considered coordination mechanisms enhanced through proactivity are buddy system and reserve. The proposed proactive mechanisms exploiting centrality measures of two different kinds of graphs are compared to the barycenter-based coordination mechanisms presented in [13].

The configuration of a simulation is characterized by three aspects:

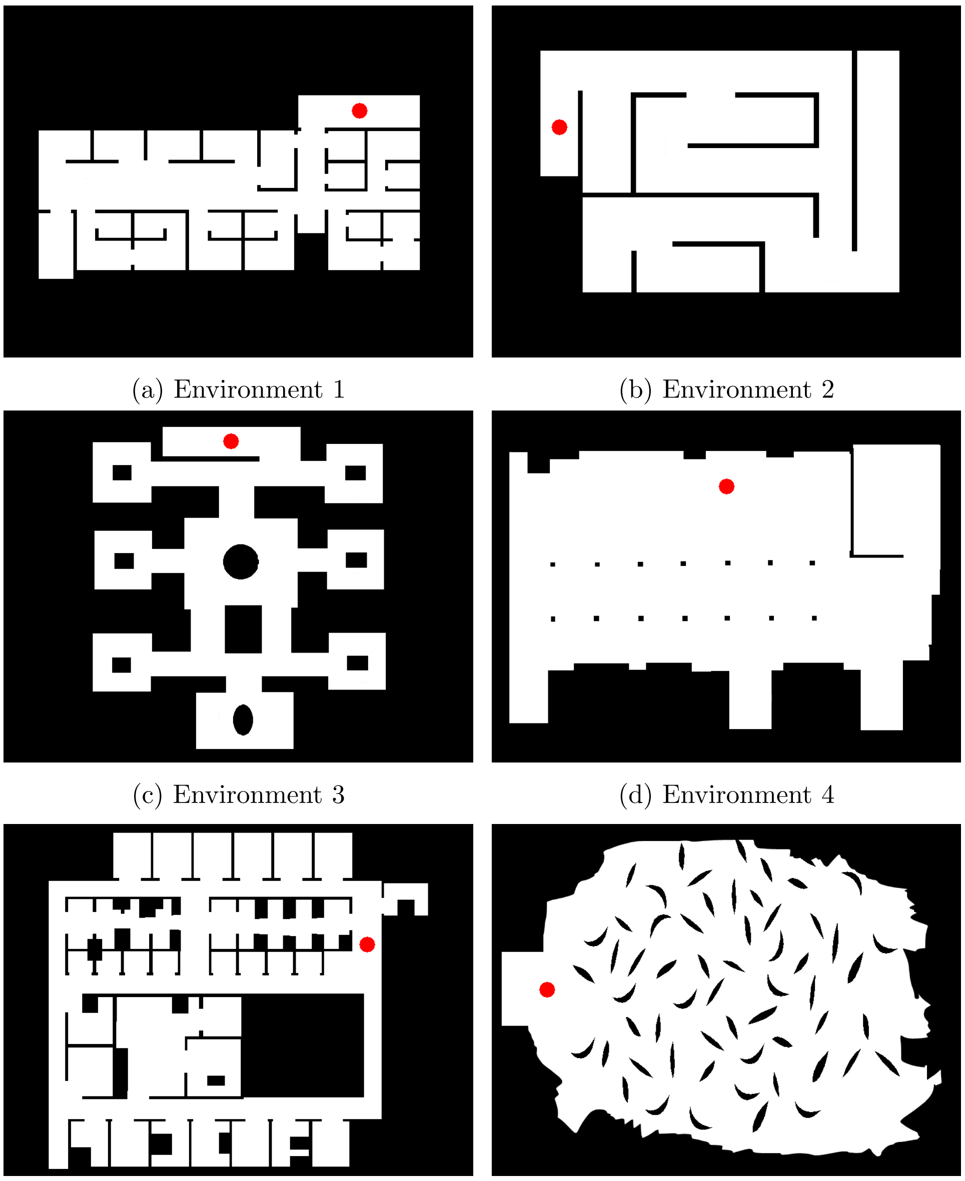
- size of the team of robots,
- environment,
- coordination mechanism

and each configuration has been tested ten times. All the simulations are run in MRESim, a Java-based simulator.

To analyze the impact on the exploration of the team size, teams from four to eleven robots have been used. Teams of two and three robots have been excluded because there would have been no effect from proactivity, being all the robots actively exploring, and none in the idle set. The environments used are presented in the next section, with a focus on their features. Then, the results of the simulations are reported and compared. In the end, it is studied the possible relation between the features of the environment and some measurable quantities, like the number of frontiers during the exploration and the value of interference.

6.1 Environments

To provide a more comprehensive analysis of the performance of each mechanism, they have been tested over the six indoor environments reported in


(a) Environment 1 (b) Environment 2

(c) Environment 3 (d) Environment 4

(e) Environment 5 (f) Environment 6

Figure 6.1: Environments tested

Environment	1	2	3	4	5	6
Size	150066	370879	520255	805367	1040954	1330885

Table 6.1: Amount of free cells for each environment

Figure 6.1, where the red dot represents the initial location of the team. They have been classified according to the features presented in Chapter 3, namely the size, the openness, and the parallelizability. In this way, it is possible to highlight any possible relation between coordination mechanisms and these features of the environment, as we do in the last section of this chapter.

The first environment is a small office (Figure 6.1a). Apart from the central corridor which traverses the environment horizontally, the spaces are rooms linked by small apertures. Thus, it can be considered cluttered. The starting point for the team is a room located in the upper right corner of the environment. This makes the team split into two subteams. One focuses on the cluster of rooms below the starting point and the other one proceeds in exploring the left section of the environment. This behavior forced by the structure of the environment makes the exploration to be lightly parallelizable. Moreover, this environment is the smallest one among those we considered, as shown in Table 6.1, where the amount of free area for each environment is reported.

The second environment is a simple maze, reported in Figure 6.1b. It provides a few points where the team can split, making the parallelizability to be very low. The team starts from the leftmost vertical corridor and moves on, following the structure of the maze. There are mainly two branching points and a different split of the team in such points is the main distinguishing factor between the various tested mechanisms. The size of the environment is the double of the previous one, nevertheless it can be considered small as well.

The third environment in Figure 6.1c can be a bunker characterized by various rooms, each one built around a central structure, and connected by corridors. The exploration starts in the upper room and the team mainly moves towards the lower part. Every corridor represents a branching point that directs a subteam to one of the lateral rooms. The exploration is always channeled into specific directions, which justifies a classification as a cluttered environment. The amount of free area is higher than the previous environments, even though it can still be considered as a small environment. The regular structure of this environment allows the exploration to benefit from an increasing number of robots. Thus, it is highly parallelizable.

Figure 6.1d shows the fourth environment. It is a large room, with some regular columns, and a smaller room on the right. This is the first of the three large environments considered. Robots are deployed in the large area, in particular in the upper central part marked by the red dot. Due to the size and the openness of the space, at the beginning of the exploration, the number of frontiers is low. This is because a frontier is produced on the contour separating known and unknown cells only for those segments of the contour between two obstacle cells. Thus, this environment can intuitively be classified as large and open. On the contrary, parallelizability is a little trickier. At first sight, given its structure, the environment might appear as highly parallelizable. This is because it allows the team to spread into the large room. On the other hand, the effect of the increase in team size is reduced with respect to other environments with high parallelizability.

The fifth environment is a large office and is represented in Figure 6.1e. It is composed of mainly two sections, both composed of different contiguous rooms, but in the upper section these are more and smaller than the ones in the lower section. Thus, the environment is large and cluttered. Moreover, it is also highly parallelizable because it allows robots to spread in different directions. This is also reflected in the boost of performance provided by the increase of the team size. Robots start from the upper right section, in the proximity of a long corridor connecting the two sections of the environment. This location is marked as the red dot in the figure. The priority given to the exploration of this corridor has a high impact on the performance, as will be shown more in detail later on.

The sixth environment is a large environment with a lot of randomly placed obstacles in it, and it is reported in Figure 6.1f. It is the largest of the tested environments. The high presence of obstacles allows us to classify it not as an open environment, rather as a cluttered one. This is reflected in the high number of frontiers discovered at the same time. For this reason, the environment can also be classified as highly parallelizable. In fact, robots tend to spread. Similarly to the fourth environment, this spreading is not coupled with a decrease in the time needed to complete the exploration as the size of the team increases.

These environments represent a wide variety of different scenarios in which the exploration might be pursued, even if not exhaustive. A major focus is given to the relation between the features used to classify the environments and the variations in the performance of the various mechanisms. The analysis will be carried on also considering other aspects like the number of frontiers detected and the performance of teams of a given size.

6.2 Mechanisms comparison

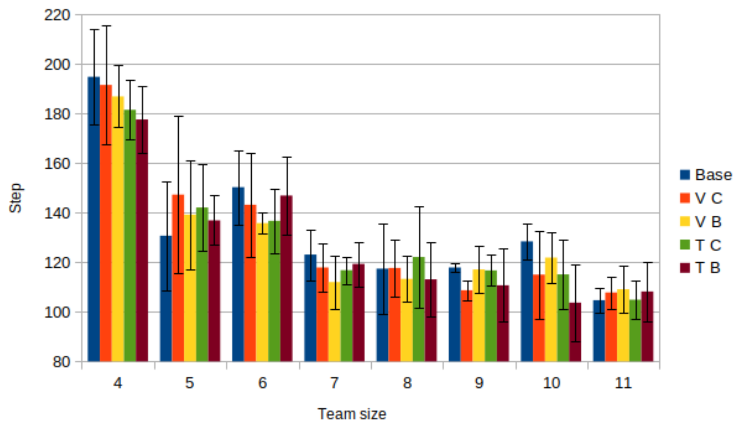
The mechanisms compared are the proactive versions of the *buddy system* and *reserve*. The proposed variants are based on how the proactivity goal is computed. The mechanisms presented in [13] are used as a benchmark for the mechanisms developed in this thesis and are referred to as base mechanisms. The proposed mechanisms are characterized by the kind of graph built during the exploration and the centrality measure calculated for it. Two graphs are used, the *topological graph* and the *visibility graph*. Also the centrality measures are two, namely *closeness* and *betweenness*. How these elements are included in the coordination mechanisms is widely described in Chapter 4.

The main comparison metric considered is the number of steps required to complete the exploration. In our case, exploration is completed when the 95% of the free area is explored. Even other metrics are used, like the *interference* and the *availability*, as presented in Chapter 4. Moreover, some attention is also given to the relation between the features of the environment, the team size, and the number of frontiers. It is shown how this information, together with the interference, might provide insights on the structure of the environment, particularly on the openness and on how much parallelizable its exploration is.

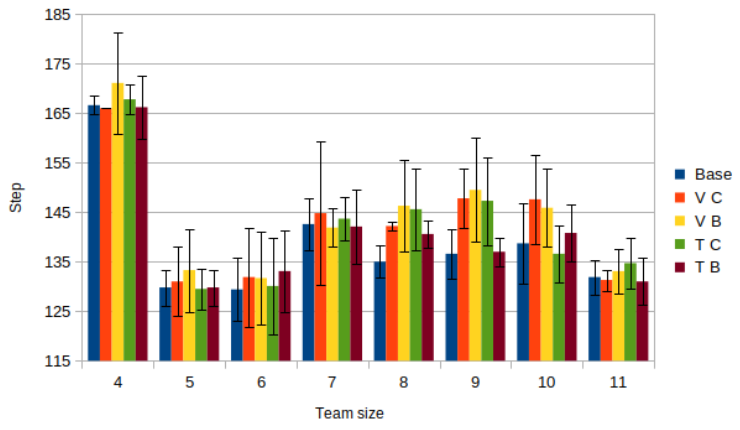
The analysis starts at first with the proactive buddy system, then moves on to the proactive reserve. Common notation is used to simplify the legends in charts and the analysis of the results. T and V represent the type of graph employed, the first one for the topological graph and the second for the visibility one. Similarly, B and C are used to refer to the centrality measure computed, betweenness in the first case and closeness in the second one.

6.2.1 Proactive buddy system

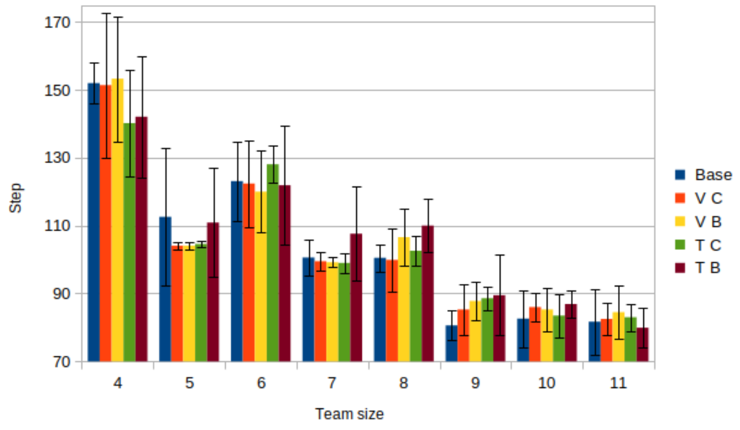
It is important to state from the beginning of the analysis that, in general, both the base proactive buddy system and the graph-based variants tested perform similarly if looking at the average completion time. This holds as long as we consider the whole sets of experiments, without focusing on a single environment. Thus, in general, the steps required by a team exploring an unknown environment are independent of the specific computation employed for the proactivity goal, among the ones considered. On the contrary, if restricting the analysis to a particular environment, it is shown that the exploration may benefit from the adoption of the proposed graph-based



(a) Environment 1

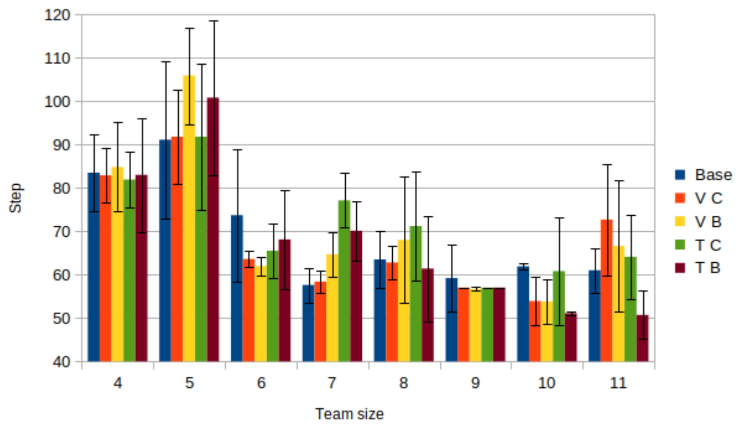


(b) Environment 2

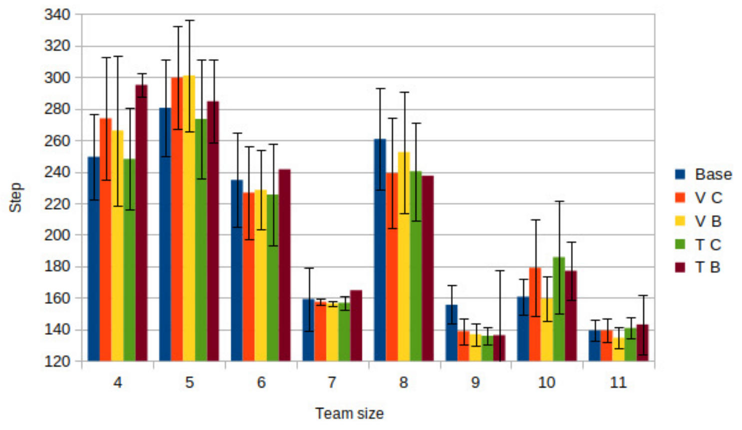


(c) Environment 3

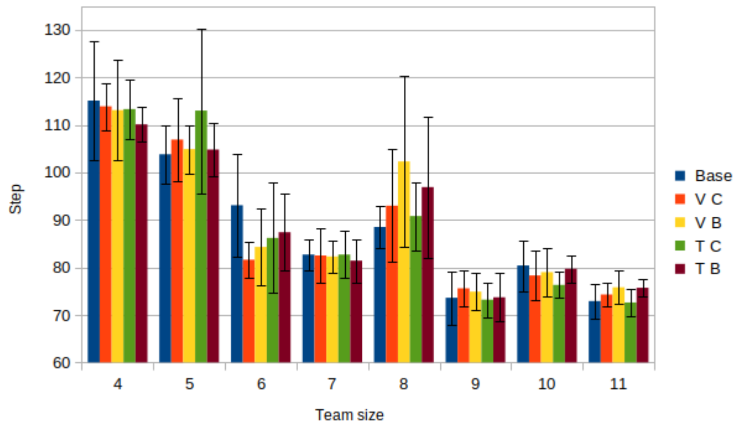
Figure 6.2: Results for the proactive buddy system on the small environments



(a) Environment 4



(b) Environment 5



(c) Environment 6

Figure 6.3: Results for the proactive buddy system on the large environments

approaches. The number of steps required to completion by the different approaches to proactivity is reported in Figure 6.2 for the small environments, and in Figure 6.3 for the large ones.

The first environment, the small office-like one, is the one where the use of graph-based approaches provides more benefit with respect to the base mechanism, as shown in Figure 6.2a. The reason behind this is that the graph-based mechanisms tend to compute the proactivity goal in correspondance of the branching point right after the exit of the starting room. This is important because when the robots in the idle set are turned into active, they are assigned to frontiers on the left side of the environment, and from this position they are able to reach their destination in a small amount of time. On the other hand, the base mechanism either places the idle set in the middle of the vertical corridor or moves it to the right section of the environment. This happens because the number of active robots allocated to the right section of the environment is usually higher than the ones assigned to the left section, shifting the barycenter more towards this section. Moreover, T B is the mechanism requiring less time and provides an average improvement of 5%, computed over the whole team sizes tested. Another aspect worth to be highlighted is that mechanisms exploiting betweenness perform generally better than the ones using closeness on the same kind of graph. In particular, for teams smaller than nine robots, V B performs better than V C. This trend swaps as the team used becomes bigger. A possible reason behind this behavior is related to the fact that, when using teams of ten and eleven robots, V B locates the proactivity goal in the lower part of the vertical corridor, in particular towards the right section. On the contrary, V C moves the idle set in a position nearer to the left section of the environment, and, as pointed out before, the frontiers to which the idle robots are assigned once turned into active are located mostly on this side of the environment. Recalling that the number of runs for each configuration is fixed, it is also interesting to notice that the standard deviation for the base mechanism is usually higher than the one of the graph-based mechanisms, particularly for teams of up to eight robots. A higher standard deviation is also shown by V C.

In the maze environment, the base mechanism completes the exploration in a smaller amount of steps with respect to all the others (Figure 6.2b). This holds for almost every team size. The base mechanism requires less time to explore than graph-based approaches because whether using closeness or betweenness, the idle set is located farther from the frontiers. While, due to the structure of the environment, the active agents tend to be nearer one another and the idle set is then moved closer to them with the base mechanism. The effect of this small difference in the location of the idle set is the variation in performance measurement. T B is the one requiring shorter time concerning other graph-based approaches. The average difference between this mechanism and the base one is around 1%, thus they perform similarly. Even though, the base mechanism appears to be preferable due to the lower differences as the team size varies.

For the third environment, all the mechanisms perform in a similar way. The chart of Figure 6.2c shows little differences among the mechanisms. Even though, it is possible to notice that closeness-based mechanisms perform slightly better than the betweenness one, under the same graph. The similarity in performance is easy to explain in this context and it is related to the fact that, due to the regularity in the structure of the environment, the distance between the barycenter and the proactivity goal computed through centrality measures is small. Another reason is that, once the active robots get out of the starting room, the number of frontiers increases, making the robots of the idle set to be turned into active. This reduces the impact provided by the proactivity and, consequently, the possible differences between the mechanisms.

Similarly to the previous environment, in the forth environment the mechanisms show similar performance on average. From the chart of Figure 6.3a it emerges that for teams whose size is greater than eight robots, T B tends to perform better than the others. This holds in particular when comparing to the base mechanism, from which the improvement is of the 10%. A possible reason behind this behavior is related to the fact that starting from the initial position, the first set of active agents tends to spread in every direction. The barycenter of the polygon whose vertices coincide with the positions of the active agents is located at a distance which can be covered in a few steps below the initial position. On the other hand, computing the proactivity goal as the highest betweenness node makes the idle set to be placed between the two rows of obstacles. The closeness of the same graph is such that the idle set is located almost halfway between the proactivity goals computed by the base method and the betweenness-based one. Another important factor is that, using the topological graph, it is preferable to use the betweenness. The opposite holds for the visibility graph. Due to the openness of the environment, the differences between these two graphs are more marked with respect to what happens in a cluttered environment, which might explain this difference. However, as already stated, all the mechanisms perform almost the same in general.

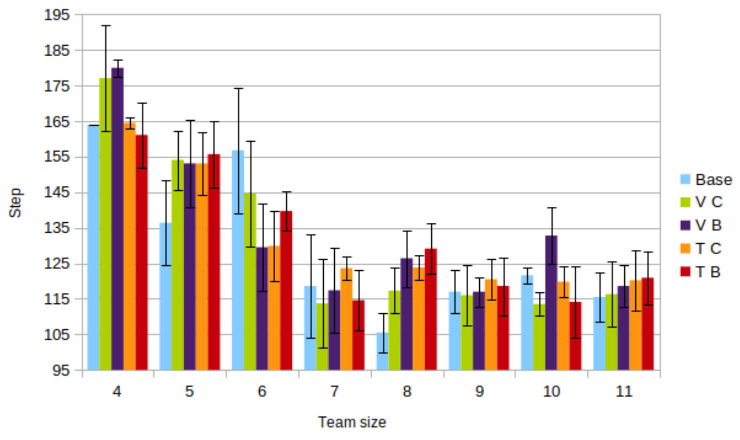
The fifth environment is the large office-like environment and the results on it are reported in Figure 6.3b. In this case, T C performs better than all the other methods. The benefit of this approach can be seen in the comparison with the base mechanism, which is outperformed for every team size except for the team of ten robots. The average improvement is 2%, but excluding the performance on the team of ten robots, the overall improvement would have been 4%. For what concerns the topological graph, T C requires 8% less time to complete exploration with respect to T B for teams of up to eight robots. From that size on, they behave similarly. The mechanisms using the visibility graph have almost similar performance. The chart presents a particular aspect that is even team sizes tend to perform worse than the odd ones. This is clear looking at what happens when the team size is increased from seven to eight robots. All the methods get worse, taking even more time in general than the case of a team of six robots. The explanation is related to the long vertical corridor linking the upper and the lower section of the environment, highlighted in the previous section. What happens is that if the team is even, no robot is sent along that corridor, thus the lower section of the environment remains unexplored until the very late stages of the exploration, where the robots can reach it through the vertical corridor on the left side. On the contrary, when the team size is odd, since the beginning of the exploration at least one robot follows that corridor and explores the lower section of the environment. This makes the exploration to require less time. When the number of robots is equal to ten, this effect is mitigated by the idle set which waits in a position near the initial location and is able to reach the lower section faster. Moreover, due to the highly fragmented structure of the environment, characterized by a lot of rooms, the number of frontiers is generally high, turning the idle set into active in the initial steps of the exploration.

The sixth environment, similarly to the third and the fourth environments, requires the same amount of time on average for each mechanism. All the mechanisms tend to show a monotonic decreasing time to complete the exploration in Figure 6.3c as the team size increases. Even though there is a slight worsening for the team of eight robots, common to all the mechanisms. This is due to the fact that, for that particular team size, the team is more channeled to explore the upper part of the environment, and then, explores the lower section of the environment. While, for the other team sizes, a subset of robots explores the lower section, making the exploration faster. For the visibility graph, the performance is almost independent of the centrality measure used. On the contrary, for what concerns the topological graph, for small teams of four and five robots, the betweenness seems to provide better results, with an improvement of 5%. As the team size increases, after a plateau measured for the teams of six, seven, and nine robots, closeness performs slightly better. Even though the differences are small, being the improvement of 3%. The absence of significant differences in the time required to complete the exploration by both the base mechanism and the graph-based ones might be related to the fact that a lot of frontiers are discovered in the early stages of exploration. This makes idle robots to be turned into active soon. If the team size is sufficiently large to still have an idle set, the various mechanisms compute proactivity goals near to each other, or at least the distance between them is small as compared to the size of the environment. Similar to what happens in the fourth environment, with the active agents moving in almost every direction, the barycenter of their positions and the node with the highest closeness tend to be very close. The node with the highest betweenness is usually located slightly lower in the environment, however along the same vertical axis. This holds both for the topological and the visibility graph, in general, and is a possible explanation of why performance is independent of how proactivity goal is computed.

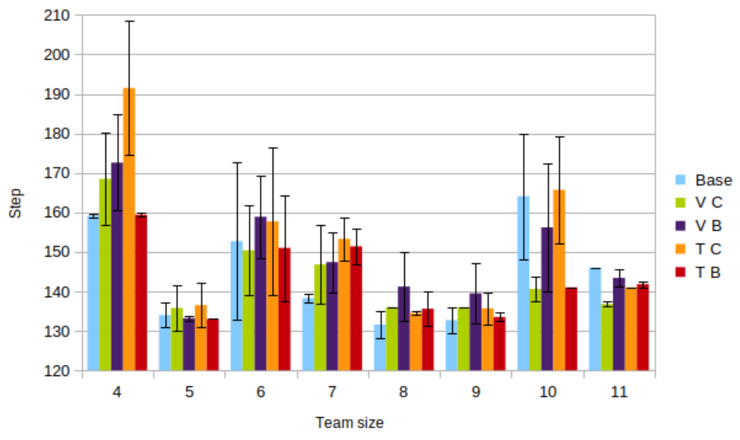
6.2.2 Proactive reserve

The usage of graph-based approaches in the proactivity function of the proactive reserve mechanism affects performance negatively, in general. Similarly to the proactive buddy system, this is true as long as we consider the whole set of environments. When focusing on a single environment, employing one of the proposed mechanisms may enhance exploration performance. The results of all the variants of the proactive reserve are shown in Figure 6.4 and Figure 6.5.

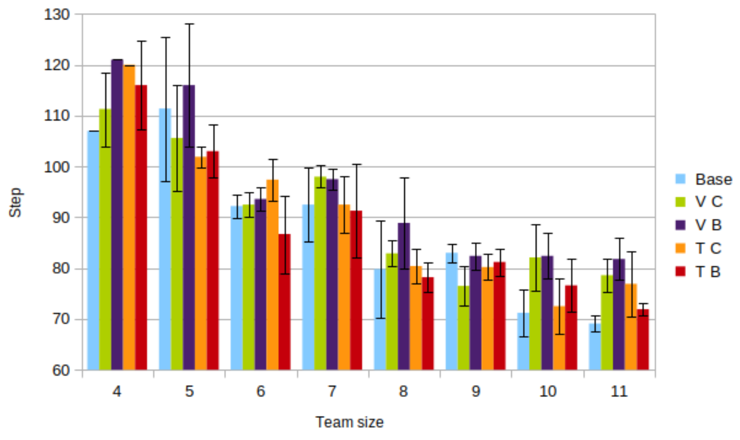
The first environment is characterized by the prevalence of the base mechanism with respect to the proposed ones (Figure 6.4a). The time required to complete the exploration is almost the same for both the mechanisms working on the topological graph and V C. The usage of V B causes a worsening of performance, particularly for the configuration of ten robots, because the idle set is moved to the right section of the environment. As discussed for the proactive buddy system case, new frontiers are discovered on the left section of the environment, thus moving the idle set on the right makes idle robots when turned into active to be far from these frontiers. On the contrary, the other mechanisms tend to place the idle set more on the left side of the intersection at the lower part of the vertical corridor. Moreover, except when the team is composed of six robots, the closeness is always preferable over the betweenness for the visibility graph. This is a substantial difference from the proactive buddy system case, where for



(a) Environment 1

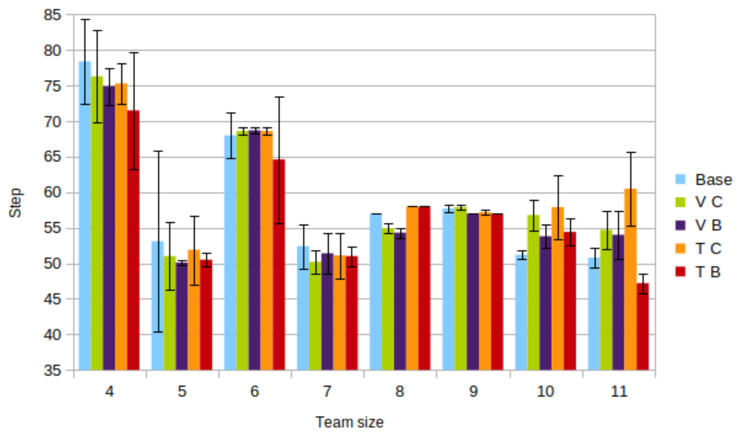


(b) Environment 2

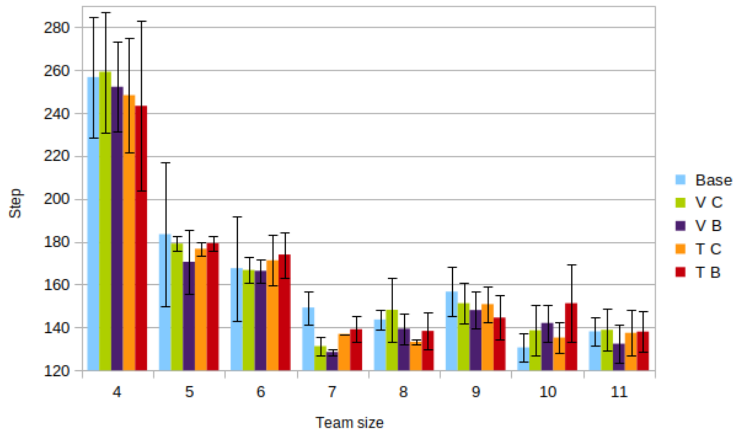


(c) Environment 3

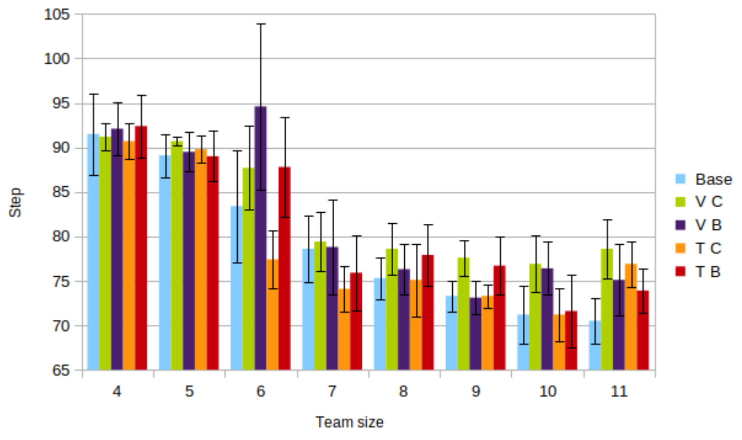
Figure 6.4: Results for the proactive reserve on the small environments



(a) Environment 4



(b) Environment 5



(c) Environment 6

Figure 6.5: Results for the proactive reserve on the large environments

teams of less than nine robots, betweenness provided better results.

In the second environment, the simple maze, three mechanisms perform better than the others in general. They are the base one, V C and T B. Their results are plot in Figure 6.4b. It is interesting to notice that the two graph-based mechanisms that perform better use a different centrality measure. Similarly to the case of the proactive buddy, a possible explanation to the worse results of the other mechanisms is related to the fact that in these situations the idle set tends to remain too distant from the active set, and consequently from the frontiers to which the robots of the idle set are assigned once turned into active. A plus point for T B is that the standard deviation is lower than the half for all the other mechanisms.

The results for the third environment in Figure 6.4c show that both the base mechanism and T B have the best performance. Closeness-based mechanisms are just slightly worse. T B is preferable over the base mechanism, except for teams of ten robots. For this environment, the same considerations done for the proactive buddy system still hold, in particular the fact that the proactivity goals computed by the various mechanisms tend to be near one another, making the differences between their performance reduced with respect to other environments. However, the usage of V B shows to provide worse performance for almost every team size with an average time 8% longer than the two best mechanisms. The explanation to this is related to the fact that the idle set is usually placed in a position in the middle of the two squared lateral rooms, while all the other mechanisms place it more in the proximity of the central room.

In the fourth environment (Figure 6.5a), all the mechanisms seem to have very similar performance. Due to the reduced amount of time required for large teams, T B provides a slight improvement with respect to the others, while T C is usually the worse. This one has the peculiarity of worsening performance as the team size increases. Even if this happens also for the V C, the worsening for T C is higher. As a consequence of this, under the same kind of graph, using betweenness is preferable over closeness.

The fifth environment is the only one where V B shows the best performance, which is the 4% better than the base mechanism. The chart with the results is the one in Figure 6.5b. Differently from the proactive buddy system case, for the proactive reserve, every graph-based approach provides an enhancement in the performance with respect to the base mechanism. Moreover, T C, which is the best for the proactive buddy system, is the second-best one in this case and provides benefits to the exploration that result in a performance slightly worse than that of the visibility graph-based.

For the sixth environment, the base mechanism and T C perform better than the others, with an improvement from 3% to 5%. In Figure 6.5c, it is possible to see that they both show a monotonic decreasing of the effort required for exploration as the team size increases, and reach a plateau for teams of more than six robots. Their performance is similar for every team size, except for the teams of eleven robots, where the base mechanism outperforms the other. For what concerns the visibility graph, its usage makes the exploration slower in general. Even though for teams from six to ten robots, V B provides performance similar to the one of the base mechanism. On the other hand, V C seems to make the exploration completely unaffected by the variations in the team size. In fact, for teams with more than seven robots, the number of steps is almost the same. Also for the other mechanisms, there is a plateau, as already pointed out, but this one is flat, while the others are slightly decreasing.

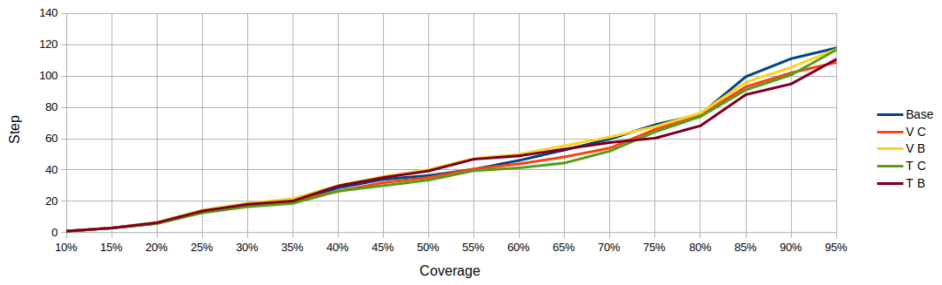
6.2.3 Coverage

Even if the main comparison metric is the time used to complete the exploration, it is also of interest to look at a few examples of how the coverage evolves over time. This allows to check whether a mechanism maps the environment faster than another one, then it requires more time to cover the last spots. Also in this case, the results reported are obtained averaging over ten runs for each configuration.

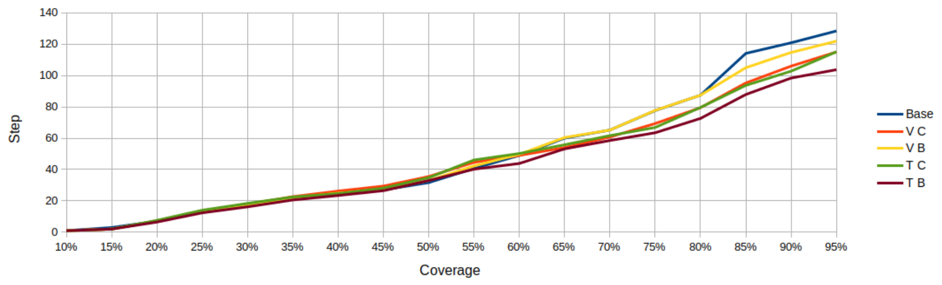
Figure 6.6 reports the trend of the coverage for the proactive buddy system in the first environment. The team sizes shown are respectively of nine (Figure 6.6a) and ten robots (Figure 6.6b).

In the first figure, it is interesting to notice that the two best performing mechanisms are T B and V C, whether the remaining three mechanisms require more time. The trend for T B and V C is slightly different as the first one is slower in covering the same amount of area up to almost the 70%, then it requires less time to map the last areas. This behavior might be due to the covering of the rooms in the left section, as T B usually moves there one robot more than V C. Another interesting behavior is the one shown by T C, as it is similar to the one of V C and even slightly faster for most of the exploration, then to fill the remaining 5% it takes more time than V C. The time to complete the exploration is similar to the base mechanism and V B in the end. A possible explanation to this is that the subteam allocated to explore the left section focuses more on the rooms on the top side, which provide a small amount of explored area, rather than prioritizing the ones on the bottom side.

Figure 6.6b depicts a different scenario, as the trends of the mechanisms

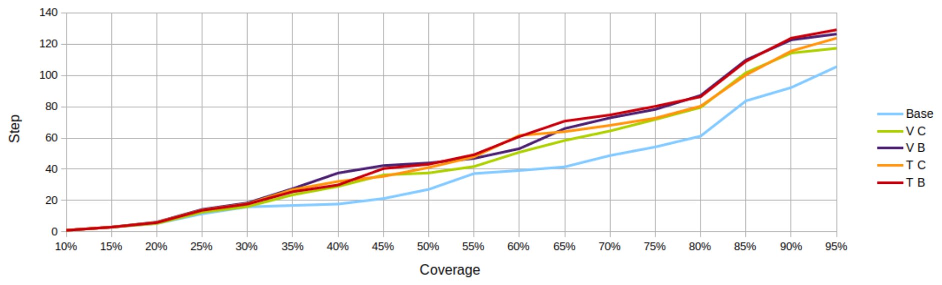


(a) Team of nine robots

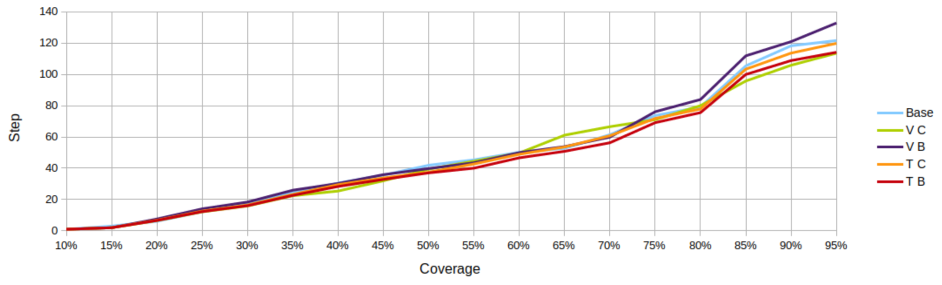


(b) Team of ten robots

Figure 6.6: Trend of the coverage for the proactive buddy system in the first environment



(a) Team of eight robots



(b) Team of ten robots

Figure 6.7: Trend of the coverage for the proactive reserve in the first environment

after the 70% of the area has been covered behave quite differently. T B is able to cover the remaining 30% in a small amout of time, which suggests a good positioning of the robots. This is opposed to the behavior of the base mechanism and V B. Similarly to the case of nine robots, this happens because robots are moved more to explore the top side of the left section, rather than the bottom one.

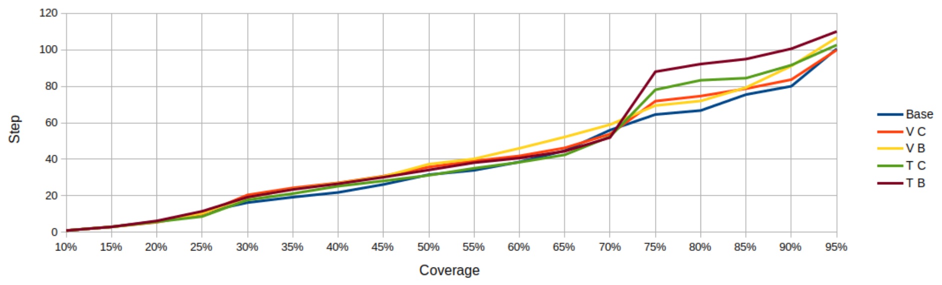
In Figure 6.7, the trend for the proactive reserve on the same environment is shown. In this case, the team sizes reported are eight and ten robots.

Figure 6.7a allows to see that the base mechanism covers an area higher than all the other mechanisms in a reduced amount of time since the beginning of the exploration. This is because of the spreading of the robots, which is more efficient with respect to the other mechanisms. Robots are evenly distributed among the left and the right sections, while all the proposed graph-based mechanisms tend to move a smaller subteam to the left section. Moreover, given that the differences are present since the beginning, it suggests that proactivity may play an important role in this case. The proactivity goal computed with the base mechanism is located in a more suitable position than the ones computed through centrality measures, making all the graph-based mechanisms to behave poorly. This also confirms what noticed in Section 6.2.2.

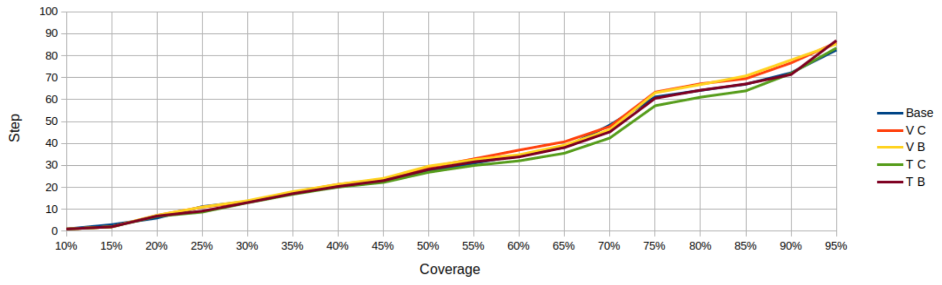
A different behavior is shown in Figure 6.7b, in which all the mechanisms have a similar trend. However, to complete the exploration, once the 80% of the environment is known, V C and T B are able to do this faster than the others. V B takes a lot of time to cover the remaining unknown area, particularly the last 5%, which is usually represented by one of the two rooms on the left side of the environment. It is also worth to notice the case of V C. In the number of steps it covers the 65% of the area, the other mechanisms cover the 70%. From that point on, the slope of the curve is smoother than the other mechanisms, and ends up being one the fastest. A possible explanation to this is related to a different prioritization given to rooms either on the top side of the left section or to the right section.

Figure 6.8 reports the trend of the coverage for the proactive buddy system in the third environment. The team sizes shown are respectively of eight (Figure 6.8a) and of ten robots (Figure 6.8b). In Figure 6.9, it is reported the same charts for the proactive reserve, considering the same team sizes.

Figure 6.8a has an interesting trend, as all the mechanisms differ strongly once explored the 70% of the environment. Even though, they end up almost in the same time. T B is the fastest in reaching that value of percentage,

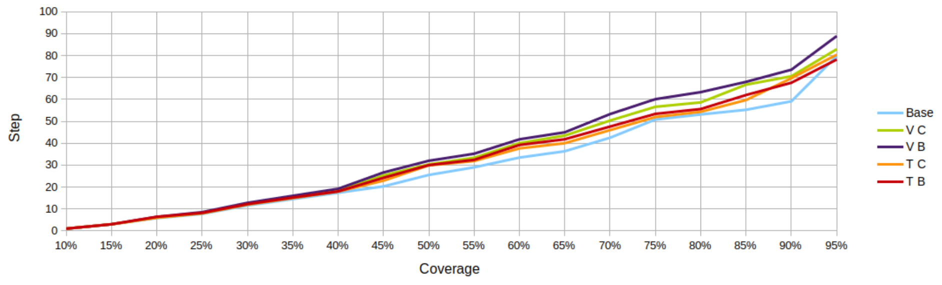


(a) Team of eight robots

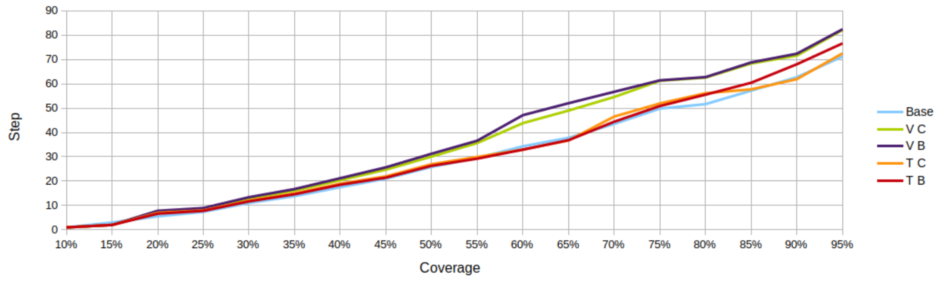


(b) Team of ten robots

Figure 6.8: Trend of the coverage for the proactive buddy system in the third environment



(a) Team of eight robots



(b) Team of ten robots

Figure 6.9: Trend of the coverage for the proactive reserve in the third environment

then the amount of time required to cover a 5% of the area is higher than all the other mechanisms. This is because the robots are allocated to frontiers in a way to push them more towards the lower section of the environment, allowing them to cover large areas in a reduced amount of time. After this, they have to move back to fill some remaining frontiers in the lateral rooms. On the other hand, the base mechanism tends to spread the robots more uniformly, avoiding a backtrack to complete the exploration. The remaining three mechanisms have a behavior which is a mixture of these two, and the amount of backtracking required is what makes the differences in the completion time.

The chart in Figure 6.8b depicts a case in which the trend of the exploration is independent of the proactive mechanism used. Even though there are some variations in the slope of the curves, the time required to complete the exploration is the same in the end. This confirms what stated in Section 6.2.1. The proactivity goals computed by all the mechanisms are near to each other. Thus, for large teams the impact of proactivity is reduced, while, in the case of Figure 6.8a, it is possible to see how a different positioning of the idle robots with a smaller team may affect how the exploration is carried on, even though it does not affect the total time required to complete it.

Figure 6.9a shows a behavior similar to the one of Figure 6.8b. All the mechanisms have similar trends, even if with some variations. Nevertheless, the time required to complete the exploration is the same, except for V B. The time it requires to cover the same amount of area is higher than all the other mechanisms almost at any step of the exploration. Also the slope of the curve for V B seems similar to the other ones, thus this indicates that the behavior shown by the robots is similar to the other mechanisms, but with some delay. It might be caused by a worse positiong of the robots computed by the proactivity function because the delay is present since the initial stages of the exploration.

In Figure 6.9b two groups can be identified. One is composed of the mechanisms employing the visibility graph, and the other composed of the remaining ones. Similarly to the case of V B with the team of eight robots described previously, these mechanisms have a certain delay since the beginning. This is probably due to a slight variation in the proactivity goal, which makes the exploration slower. On the other hand, the other mechanisms almost have the same behavior and completion time.

Even if the analysis of this section is restricted to few cases, it highlights some interesting behaviors, which are mostly two. Either the differences in area covered are there since the beginning, or happen in the last stages of the exploration. In particular, the first case is highly marked in Figure 6.7a and Figure 6.9b. A similar behavior happens also in Figure 6.6a for T B, which is slower in the beginning, but then, the initial allocation is beneficial on the long run as this mechanism is one of the best performing in this case. The second case is shown in Figure 6.6b, Figure 6.7b, and Figure 6.8a. The last one is particularly interesting because after the 70% of the coverage up to the 90%, mechanisms behave quite differently, but they reach the end of the exploration almost in the same number of steps.

This analysis allows to infer that a different computation of the proactivity goal affects the exploration in various ways, and the differences might be mitigated as the exploration goes on, or affect directly the time to complete it. Moreover, it also suggests that the time required to complete the exploration is not a metric able to capture all the differences produced by the variations in the proactivity function. In fact, some of them can be found only by looking at the trend of the coverage over time.

6.3 Environment features

The analysis carried on up to this moment focused on the general performance provided by the various mechanisms with respect to the base versions. The results give us a hint about some possible relations between the features of the environment and some aspects strictly related to the exploration.

6.3.1 Parallelizability

As presented in Chapter 3, the parallelizability of an environment is related to the spreading of the agents during the exploration. At the same time, it might be intuitive to think that if the exploration of an environment is highly parallelizable, then it may benefit from larger teams, rather than smaller teams. On the other hand, if it is lightly parallelizable, then the effect of increasing the team size should be reduced or it might even affect negatively the performance. An example of this negative behavior is the maze environment. Both Figure 6.2b and Figure 6.4b show that for the team of five robots, the exploration time is at a minimum point. As the team size increases, also the time required increases. This is an important example because it happens both for the proactive buddy system and the proactive reserve, which allows in a certain sense to claim that it could be a property of the environment, rather than of the particular mechanism used.

Another example in this sense is represented by the forth environment. It is an open environment, thus it might be intuitive to say that it is highly

Environment	1	2	3	4	5	6
Maximum number	17	6	18	13	23	31
Average number	9	2	8	7	16	22

Table 6.2: Maximum and average number of frontiers for each environment

parallelizable. Looking at the performance of the different mechanisms in Figure 6.3a and Figure 6.5a, it is possible to notice two major aspects. First of all, T B reaches a minimum for the team of eleven robots, both for the proactive reserve and the proactive buddy system. The second aspect is that, for the proactive buddy system, an increase in the team size causes a slight improvement in performance, in general, while for the proactive reserve the best performing team is the one composed of five robots. This might be counterintuitive when considering the structure of the environment. On the contrary, the explanation to this is that, being the environment almost free from obstacles, each robot can perceive a large amount of space with each scan. This makes the use of a lot of robots redundant, in general. However, as already pointed out, a mechanism like T B is able to overcome this redundancy and get better performance with teams of big sizes.

Rather than trusting the intuition, a possible way to relate the parallelizability of the environment to a measurable metric is the number of frontiers detected during the exploration. In particular, the maximum and the average number of frontiers for each environment are reported in Table 6.2. These have been measured during the whole set of experiments, for each configuration.

The second environment is obviously the one with the lowest amount of frontiers. The fourth environment is the second last. The fact that in these two environments an increase in the team size corresponds to a worsening of the performance and the fact that they are the ones with the lower amount of frontiers are likely to be related in some way. Also because of the extreme differences in the structure. The first and the third environment have almost the same number of frontiers, both the maximum and the average. Nevertheless, the third environment benefits more from increasing team sizes than the first one. The reduction in the number of steps required from the team of eleven robots to the one of four robots is 30% for proactive reserve on the first environment, while it is of 34% for the third one. For the proactive buddy system, the variation is equal for the two environments and is 43%. However, the number of frontiers alone provides only one point of view, it does not capture the whole picture. In fact, the sixth environment is the one with the highest number of frontiers, both for the maximum and the average values. Despite this, the difference in the performance of the proactive reserve for the teams of eleven robots with respect to the team composed of four robots one is only of the 18%. A value that is comparable to the one of the second environment, the maze. This is explicative of the intrinsic difficulty of providing a quantitative measure for such an elusive concept as the parallelizability. For this reason, also the values of interference and availability are taken into account.

6.3.2 Interference and availability

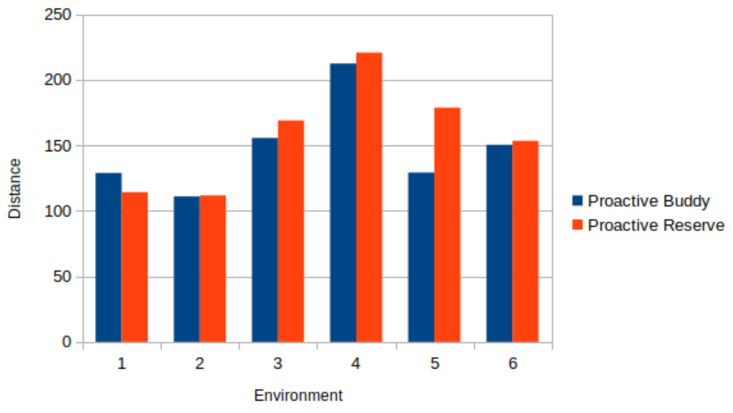
The interference and availability as defined in Chapter 3 can also provide some help in discriminating among the various kinds of environments. There are no major differences in the values of interference and availability for the single variants of the proactive buddy or the proactive reserve. Thus, they have been averaged and the interference for each environment is reported in Figure 6.10a, while the availability is plotted in Figure 6.10b.

The interference plot shows that the environments can be clustered into three groups. The first one is composed of the first and the second environment, providing the least amount of interference. The second group is composed of the third, the fifth, and the sixth environment, which have values of interference higher than the first group, but significantly lower than the forth environment. This one alone composes the third group.

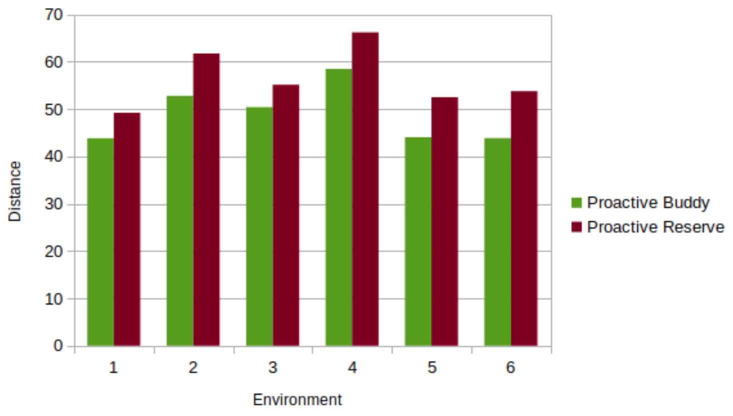
The low spreading of the second environment is a direct consequence of it being a simple maze. For the first one, the interference measure is so low because the team tends to divide into two subteams, one exploring the section on the right and the other one the one on the left. However, the distance among robots in each subteam is low and this explains the value of interference. Such value is very similar to the one obtained using the proactive buddy system on the fifth environment.

As already explained in Section 6.2.1, in the fifth environment, the proactive buddy shows a very different behavior depending on the parity of the team. In the case of even teams, the whole team of robots explores the upper part of the environment, making them be near to each other, reducing the value of interference, which turns out to be comparable to the one of the first environment on average, despite the difference in the size of the two. For this reason, the value of interference for the proactive reserve turns out to be more reliable to try to characterize the environment with respect to the proactive buddy one, being less affected by characteristics of the particular mechanism.

That being said, the second group of environments is composed of the



(a) Interference



(b) Availability

Figure 6.10: Values of interference and availability for each environment

third, the sixth, and, for what explained above, the fifth environment considering the proactive reserve. The most counterintuitive aspect is that the average distance between agents in the third and the sixth environment is almost the same, despite the size. However, this is explicable by considering that in the third environment robots are allowed to spread into the lateral rooms, while some robots keep exploring towards the lower part of the environment. In the sixth one, robots proceed from the left part of the environment to the right and the team of active robots is almost distributed along a vertical line. This makes the distance among them relatively small.

For the fourth environment, the higher value of interference is due to the openness which characterizes it. Robots are able to go in different directions, making the average distance among them high with respect to the other environments.

The availability seems to provide no possible insights on the classification of the environments. The chart of Figure 6.10b allows retrieving two aspects. First of all the value of availability for the proactive reserve is higher than that of the proactive buddy system one on each environment. This is due to the presence of the buddy in the buddy system, whose aim is exactly to reduce the availability, being already near the frontier to which it is assigned. Apart from this consideration, similarly to what has been done with the interference, it is possible to notice that the environments can be divided into two groups. The first one composed of the first, the fifth, and the sixth environment, while the second one composed of the remaining ones. Even though, the lower magnitude of the differences in the values of availability for each environment with respect to the interference case makes it hard to consider availability as a discriminatory element.

In conclusion, the value of interference, together with the number of frontiers, both the maximum and the average value, can help in retrieving some features of an environment. In fact:

- a low value of interference paired with a low amount of frontiers suggest that the environment is characterized by few branching points and that robots are likely to be forced to follow a certain path, without spreading;

- a low value of interference with a high number of frontiers may characterize an environment composed of contiguous rooms or with a lot of near obstacles;

- a high value of interference coupled with a low number of frontiers indicate that robots spread in different directions and the environment is not fragmented, but it is composed of large spaces, otherwise, the number of frontiers would be high;

- a high value of interference and a high number of frontiers may represent the case of an environment with many obstacles, where frontiers are discovered along different directions and the robots spread.

Relating this to the classification of the features provided in Chapter 3, only the third case of the list, that is a high value of interference coupled with a low number of frontiers, suggests that the environment can be classified as open. All the other cases refer to a cluttered environment. Dealing with parallelizability, only the forth case of the list suggests that the environment considered is highly parallelizable, even though this relation is more elusive and harder to get than the one for openness.

This is a possible interpretation of the results and it is not intended to be exhaustive. The idea is that the underlined aspects might provide an insight to classify the environments, but they are likely not to capture all the aspects.

Chapter 7

Conclusions

This thesis aims to extend the concept of proactivity developed in [13] through the use of graphs. The coordination mechanisms considered are the proactive buddy system and the proactive reserve. The proposed coordination mechanisms are compared to the benchmark ones according to the number of time steps required to complete the exploration. This is considered as the main metric in discriminating between the different mechanisms. The decision of considering the number of time steps rather than the absolute time taken by the exploration has been carried on to exclude machine-dependent aspects from the results. Two more metrics have been taken into account to completely characterize the various mechanisms, interference and availability.

7.1 Mechanisms comparison

For both the proactive buddy system and the proactive reserve, none of the proposed modifications to the proactivity functions seem to significantly enhance performance in general. Averaging the time taken by each mechanism over the whole set of environments tested, it comes out that all the mechanisms perform similarly in the case of the proactive buddy system. On the other hand, graph-based approaches tend to affect negatively proactive reserve, making the exploration slower. For this reason, the proposed coordination mechanisms are not definitely preferable to the base proactive mechanisms. This is also backed up by the increased computational effort required by the graph-based approaches. Even though all the optimizations explained in Chapter 5, the computation of centrality measure is more demanding than the one of the barycenter of a few points. Thus, even if the number of steps required to complete the exploration is similar, the time required by the graph-based approaches would be higher than that of the base mechanisms. This holds in particular for the visibility graph, where both the number of nodes and edges of the graph are higher than those of the topological one. Therefore, without prior information about the environment, the use of the base mechanisms is preferable. As already pointed out in Chapter 6, if restricting the analysis to some specific environments, the graph-based approaches can actually enhance performance.

For the proactive buddy system, there are some differences in the behavior of the proposed mechanisms with respect to the base mechanism in the first, the second, and the fifth environment. They are all cluttered environments, and both the first and the second are lightly parallelizable, differently from the fifth, which is highly parallelizable. Nevertheless, both the first and fifth are office-like environments. In the remaining three environments, all the mechanisms perform similarly. The base mechanism is outperformed on the first environment mostly by every graph-based approach, except for V C. In particular, the best performer is T B. The enhancement provided on this environment is of particular interest because the base proactive buddy system is slower than the base proactive reserve, while T B is even better than this last one. In the fifth environment, T C performs better than the base mechanism but, given the high variability of the runs in this environment as opposed to the small enhancement provided, this is not as significant as in the case of the first environment. For the second environment, all the graph-based approaches take more time than the base mechanism, except T B. Thus, for what concerns the proactive buddy system, T B can be considered the best candidate among the proposed mechanisms because the enhancement in performance on the first environment is significant, being around 5%, and the performance on the other environments is comparable to the base mechanism. At the same time, V C is the worst-performing as it affects negatively the performance on the second environment and, on all the other environments, it requires an amount of time similar to the base mechanism.

For the proactive reserve, the base mechanism is the best performing one, as already pointed out. The second-best mechanism is T B, which is also the second-best for the proactive buddy system. This happens because, in all the environments, the performance of T B is similar to that of the base mechanism. There are some consistent differences in the fourth and sixth environments. In the fourth one, T B provides an increase of 3%, while in the sixth the opposite holds, as the base mechanism requires the 2% less time than T B. In the case of proactive reserve, the worst mechanism is V B. Even if it requires less time than the base mechanism on the fifth environment, it makes the exploration slower in all the other environments.

Apart from the base mechanisms that is preferable over the proposed mechanisms, an interesting aspect that emerges is that T B provides the best performance among the graph-based mechanisms, both for the proactive buddy system and the proactive reserve. Moreover, the performance of T B is comparable to the one of the base mechanism, in the context of both the proactive buddy system and the proactive reserve. This makes this mechanism worth to be inspected more, and a good starting point for future works in this sense.

The use of the visibility graph makes the exploration require more time to be completed. Thus, the visibility graph, as defined in this thesis, appears to be a model not able to provide useful information in computing a good allocation for the idle set. This is shown in particular by the proactive reserve, where this kind of graph is outperformed by all other mechanisms. A possible improvement in this model might be in the criteria with which the pose nodes to add to the graph are selected. In fact, knowing that the topological graph provides such an improvement with respect to the visibility graph, it might be worth trying to apply a discretization to the pose nodes similar to the one applied in the building of the topological graph. In this way, the distribution of pose nodes over the map would be more regular than the one of the visibility graph considered in this work, and the centrality measure might provide a better allocation of the idle set.

Results obtained also point out some interesting aspects in comparing proactive buddy system and proactive reserve. The first one appears to be preferable in highly structured environments. On the contrary, the proactive reserve tends to behave better on open environments, and on all those environments which allow robots to spread. This is a direct consequence of the idea on which these mechanisms are based. In fact, the proactive buddy system is conceived to optimize the exploration of T-shaped junctions, which are likely to be present in highly structured environments. The proactive reserve allows robots to spread more since the beginning of the exploration, and from this comes out the more efficiency in exploring open environments.

Even if the main comparison metric used is the time required to complete the exploration, we analyzed also the trends of the coverage for all the coordination mechanisms on two environments. Even if in some cases the exploration is carried on in a similar way, independently of the usage of a graph-based mechanism or the base mechanism, interesting cases in which there are differences in these trends are present. In fact, this analysis allowed us to notice two different behaviors related to the variations in the proactivity function considered: either the differences are mitigated as the exploration goes on or they affect directly the time to complete it. These conclusions are interesting because the first one indicates that differences in the proactivity goal may impact only how the exploration is carried on, rather than affecting the time taken to complete it. On the other hand, the second conclusion comes from the fact that, in some cases, the trend of the coverage is similar among the mechanisms, but some of them are characterized by a certain delay. This is likely to be a direct consequence of the different computation of the proactivity goal. Therefore, what emerges from this analysis is that in comparing the performance of the coordination mechanisms, including the evolution of the coverage, may provide a deeper understanding of the consequences of using different proactivity functions, more than focusing only on the total time required to complete the exploration.

7.2 Interference and availability

For each mechanism considered in the experiments, both the base mechanisms and the proposed ones, the values of interference and availability have been evaluated and compared. These values have been taken into account to analyze the whole set of mechanisms under different perspectives. The aim of including graphs and centrality measures was to reduce the value of availability for the robots in the idle set. In this way, when they are turned into active, the distance to the assigned frontiers is expected to be lower than in the case of the base mechanisms. For what concerns interference, no major differences were expected. In fact, even if the proactivity goal is computed differently, the distance between robots in the team is expected to be similar on average.

Concerning the values of interference and availability, the only significant differences come out when comparing the proactive buddy system and proactive reserve in the same environment. This is a result already known from [13], and confirmed by our experiments. What emerges from our experimental phase is that the values of interference and availability are not affected by the variations of the proactivity function. Dealing with the proactive buddy system, this is somehow coherent with the results of the time required by the exploration we obtained because, as pointed out previously, both the base mechanism and all the graph-based approaches perform similarly. On the other hand, the base proactive reserve performs generally better than the proposed variants, except T B. However, this is not captured by the interference and availability measures, whose values differ of few units from the base mechanism to the graph-based ones. The magnitude of the difference is so small that it can hardly be used to justify the differences in performance. Moreover, in some cases, the evaluation of these metrics favorites slightly one of the graph-based approaches with respect to the base mechanism, but the latter takes less time to explore than the former.

This confirms the idea that the measures of interference and availability are not able to fully capture the dynamics of the exploration. They do not provide enough information when comparing coordination mechanisms differing only for some aspects, like proactivity in our case. A possible explanation of this is related to the reduced impact of proactivity on these measures. In fact, in our experiments, proactivity is usually needed in the first quarter of the exploration (at most). Thus, the remaining 75% of the exploration affects more both interference and availability than the first 25% does. To confirm this, it is worth to point out the case of the first environment, as an example, for the proactive buddy system. For all the teams with less than ten robots, all the pairs in the idle set are turned into active in the first 20% of the time required to complete the exploration. When the team is of ten and eleven robots, there is at least one idle pair up to the 30% of the total time required for exploration. For this reason, these two metrics are not capable of providing insights about the differences in the base mechanisms when compared to the graph-based variations. Nevertheless, they are a useful metric of comparison to analyze coordination mechanisms that present strong differences in their structure, as shown in Chapter 6.

7.3 Environment features

The extended testing performed with the whole set of considered mechanisms allowed us to focus also on the possibility of correlating some features of an environment to the metrics taken during the exploration. The features on which we focused are the ones described in Chapter 3, namely the size, the openness, and the parallelizability.

For the size, a practical measure has been provided together with its formal definition, it is simply the amount of free area. A rough estimate of it can be known prior to the beginning of the exploration. If this is not available, its value can be evaluated as the exploration goes on, even though the exact value would be known only at the end of the exploration, in such a case.

The openness has been defined as the characteristic of an environment to be composed of open spaces, rather than cluttered ones. Knowing this before the exploration begins might be exploitable as some coordination mechanisms perform better than others in this kind of environment. Relating this consideration to the mechanisms considered, it is true for the proactive reserve as opposed to the proactive buddy system, being this last one conceived to speed up the exploration of branching points. On the other hand, in Chapter 6, it has been proposed a possible relationship among the number of frontiers, the interference, and openness. The first two quantities can be computed during the exploration, and their values can be used to provide a rough estimate of the openness of the environment, whether it is open or rich in obstacles. Then, such an estimate might be exploited to adjust the coordination mechanism employed as the exploration goes on.

The parallelizability is the third feature considered. It is intuitively presented as the property of an environment to enforce the spreading of the robots during the exploration. This definition would be strictly related to the value of interference measured, being it the average distance among the robots during the exploration. However, as explained in Chapter 6, this intuitive definition could be enriched by considering that, if an environment is highly parallelizable, its exploration should benefit more from larger teams than a lightly parallelizable environment. Obtaining such knowledge requires to run the exploration with varying team sizes and then analyze whether the exploration is faster with larger teams, or not. Such analysis is only possible a posteriori and it cannot provide information exploitable. For this reason, similarly to what done with the openness, the suspect is that there is a possible relation between the interference and the number of frontiers, even though this relation is more elusive and harder to get than the one for openness.

As already pointed out, knowing before the exploration begins that the environment which is going to be explored is open makes the proactive reserve to be preferable to the proactive buddy system, as it provides better performance on this kind of environments. Something similar can be told if it is known that the environment is highly parallelizable. Also in such a case the proactive reserve seems to be a better choice. On the other hand, if we know that the environment is highly structured and lightly parallelizable, the proactive buddy system might provide better performance.

7.4 Future work

This thesis is an early work in the introduction of centrality measures into the proactivity functions. There are a lot of possible modifications that can be tested, both dealing with the definition of the graphs and the centrality measures used. An example might be the inclusion of frontier nodes into the topological graph. Having this kind of graph shown in general to have better performance than the visibility one, it might be the case to test whether the inclusion of frontiers allows enhancing the performance even more. Moreover, among all the proposed variations, T B seems to be a good starting point for future works.

The centrality measures used are just a small example of all the possible ones which can be tested. Other examples in this sense are degree centrality or the eigenvector centrality, which both relate the concept of the centrality of a node to the number of connections it has with the other nodes. They are less related to the distances between nodes, differently from closeness and betweenness, thus might exploit differently the structure of the graph, and, consequently, of the environment. Other aspects which might be further investigated are related to the extensive use of the graphs during the exploration, and not limiting their use only to the proactivity function.

From the analysis on the coverage emerged that the completion time does not capture all the consequences of different implementations of the proactivity function. Thus, including the trend of the coverage can help to completely characterize a comparison between mechanisms differing for this aspect. Moreover, a consistent analysis of this kind may also point out the possibility to define a coordination mechanism which dynamically changes during exploration, behaving like one of the base mechanisms at the beginning, then switching to a graph-based approach in later stages, for example.

The features of the environment considered in this thesis are size, openness, and parallelizability. They are just some of the features which can be possibly considered. In fact, the openness just discriminates whether the environment is composed of large spaces or is rich of obstacles. A further distinction might be in the nature of the obstacles, like if they are walls which split the space into rooms, or just random objects. Moreover, given the intrinsic difficulty both in the definition and in the exploitation of the parallelizability, it may be worth to consider different aspects affecting it, like the spreading of the agents coupled with the amount of area explored by each one. These two aspects, together with the modifications proposed for the openness might provide a more comprehensive characterization of the environment. In the case in which all these aspects can suitably be evaluated during the exploration, they might be used as building blocks to the definition of a coordination mechanism able to dynamically adapt to the features of the environments on which it is employed.

Bibliography

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

[2] F. Amigoni. Experimental evaluation of some exploration strategies for mobile robots. In *Proceedings of the 2008 IEEE International Conference on Robotics and Automation*, pages 2818–2823, 2008.

[3] F. Amigoni, N. Basilico, and A. Quattrini Li. How much worth is coordination of mobile robots for exploration in search and rescue? In *RoboCup 2012: Robot Soccer World Cup XVI*, pages 106–117. Springer Berlin Heidelberg, 2013.

[4] F. Amigoni, V. Caglioti, and U. Galtarossa. A mobile robot mapping system with an information-based exploration strategy. In *Proceedings of the First International Conference on Informatics in Control, Automation and Robotics*, pages 71–78. SciTePress - Science and Technology Publications, jan 2004.

[5] B. Awerbuch, M. Betke, R. L. Rivest, and M. Singh. Piecemeal graph exploration by a mobile robot. *Information and Computation*, 152(2):155–172, aug 1999.

[6] M. Bender and D. Slonim. The power of team exploration: two robots can learn unlabeled directed graphs. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 75–85. IEEE Comput. Soc. Press, 1994.

[7] D. Benedettelli, A. Garulli, and A. Giannitrapani. Cooperative SLAM using m-space representation of linear features. *Robotics and Autonomous Systems*, 60(10):1267–1278, oct 2012.

[8] U. Brandes. A faster algorithm for betweenness centrality. *The Journal of Mathematical Sociology*, 25(2):163–177, jun 2001.

[9] P. Brass, F. Cabrera-Mora, A. Gasparri, and J. Xiao. Multirobot tree and graph exploration. *IEEE Transactions on Robotics*, 27(4):707–717, aug 2011.

[10] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun. Collaborative multi-robot exploration. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 1, pages 476–481 vol.1. IEEE, 2000.

[11] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider. Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3):376–386, 2005.

[12] A. Caltieri and F. Amigoni. High-level commands in human-robot interaction for search and rescue. In S. Behnke, M. Veloso, A. Visser, and R. Xiong, editors, *RoboCup 2013: Robot World Cup XVII*, pages 480–491. Springer Berlin Heidelberg, 01 2014.

[13] M. Cattaneo. An analysis of coordination mechanisms for multi-robot exploration of indoor environments. MSc thesis, Politecnico di Milano, 2017.

[14] R. Chatila and J. Laumond. Position referencing and consistent world modeling for mobile robots. In *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 138 – 145. Institute of Electrical and Electronics Engineers, 04 1985.

[15] T. Cormen. *Introduction to Algorithms*. McGraw-Hill, 2010.

[16] A. Dekker. Network centrality and super-spreaders in infectious disease epidemiology. In *Proceedings of the 20th International Congress on Modelling and Simulation*, pages 331–337. Modelling and Simulation Society of Australia and New Zealand, dec 2013.

[17] X. Deng and C. H. Papadimitriou. Exploring an unknown graph. *Journal of Graph Theory*, 32(3):355–361, nov 1999.

[18] A. Dessmark and A. Pelc. Optimal graph exploration without good maps. In *Algorithms — ESA 2002*, pages 374–386. Springer Berlin Heidelberg, 2002.

[19] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes. Robotic exploration as graph construction. *IEEE Transactions on Robotics and Automation*, 7(6):859–865, 1991.

[20] P. Fraigniaud, L. Gasieniec, D. R. Kowalski, and A. Pelc. Collective tree exploration. *Networks*, 48(3):166–177, 2006.

[21] L. C. Freeman. Centrality in social networks conceptual clarification. *Social Networks*, 1(3):215–239, jan 1978.

[22] C. Gomez, A. C. Hernandez, and R. Barber. Topological frontier-based exploration and map-building using semantic information. *Sensors*, 19(20):4595, oct 2019.

[23] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard. A tutorial on graph-based SLAM. *IEEE Transactions on Intelligent Transportation Systems Magazine*, 2:31–43, 12 2010.

[24] D. Guzzoni, A. Cheyer, L. E. Julia, and K. Konolige. Many robots make short work: Report of the SRI international mobile robot team. *AI Magazine*, 18(1):55–64, Mar. 1997.

[25] F. Hoffmann. One pebble does not suffice to search plane labyrinths. In *Fundamentals of Computation Theory*, pages 433–444. Springer Berlin Heidelberg, 1981.

[26] Colossus - ROBOTS: Your guide to the world of robotics. `https://robots.ieee.org/robots/colossus/`.

[27] D. Krackhardt. Assessing the political landscape: Structure, cognition, and power in organizations. *Administrative Science Quarterly*, 35(2):342–369, jun 1990.

[28] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the National Conference on Artificial Intelligence*, 11 2002.

[29] A. B. M. Nasiruzzaman, H. R. Pota, and M. A. Mahmud. Application of centrality measures of complex network framework in power grid. In *Proceedings of the IECON 2011 - 37th Annual Conference of the IEEE Industrial Electronics Society*, pages 4660–4665. IEEE, nov 2011.

[30] M. E. J. Newman. Scientific collaboration networks. II. shortest paths, weighted networks, and centrality. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 64(1):016132, jun 2001.

[31] E. Olson, J. Strom, R. Morton, A. Richardson, P. Ranganathan, R. Goeddel, M. Bulic, J. Crossman, and B. Marinier. Progress toward multi-robot reconnaissance and the MAGIC 2010 competition. *Journal of Field Robotics*, 29(5):762–792, jun 2012.

[32] L. Peskoe-Yang. Paris firefighters used this remote-controlled robot to extinguish the notre dame blaze. `https://spectrum.ieee.org/automaton/robotics/industrial-robots/colossus-the-firefighting-robot-that-helped-save-notre-dame`.

[33] J. G. Rogers, C. Nieto-Granda, and H. I. Christensen. Coordination strategies for multi-robot exploration and mapping. In J. P. Desai, G. Dudek, O. Khatib, and V. Kumar, editors, *Experimental Robotics*, pages 231–243. Springer International Publishing, 2013.

[34] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. Younes. Coordination for multi-robot exploration and mapping. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, volume 4, pages 852–858. AAAI Press, 01 2000.

[35] V. Spirin, J. de Hoog, A. Visser, and S. Cameron. MRESim, a multi-robot exploration simulator for the rescue simulation league. In R. A. C. Bianchi, H. L. Akin, S. Ramamoorthy, and K. Sugiura, editors, *RoboCup 2014: Robot World Cup XVIII*, volume 8992, pages 106–117. Springer International Publishing, 01 2015.

[36] C. Stachniss. *Robotic Mapping and Exploration.* Springer Berlin Heidelberg, 2009.

[37] C. Stachniss, G. Grisetti, and W. Burgard. Information gain-based exploration using Rao-Blackwellized particle filters. In *Robotics: Science and Systems I*, pages 65–72. Robotics: Science and Systems Foundation, jun 2005.

[38] C. Stachniss, O. Mozos, and W. Burgard. Speeding-up multi-robot exploration by considering semantic place information. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 2006, pages 1692–1697, 05 2006.

[39] S. Thrun and J. J. Leonard. Simultaneous localization and mapping. In *Springer Handbook of Robotics*, pages 871–889. Springer Berlin Heidelberg, 2008.

[40] E. Todt, G. Rausch, and R. Suarez. Analysis and classification of multiple robot coordination methods. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 4, pages 3158–3163 vol.4. IEEE, 2000.

[41] I. Tsitsimpelis, C. J. Taylor, B. Lennox, and M. Joyce. A review of ground-based robotic systems for the characterization of nuclear environments. *Progress in Nuclear Energy*, 111:109–124, 03 2019.

[42] R. Vincent, D. Fox, J. Ko, K. Konolige, B. Limketkai, B. Morisset, C. Ortiz, D. Schulz, and B. Stewart. Distributed multirobot exploration, mapping, and task allocation. *Annals of Mathematics and Artificial Intelligence*, 52(2-4):229–255, apr 2008.

[43] B. Wang and S. Qin. Multi-robot environment exploration based on label maps building via recognition of frontiers. In *Proceedings of the 2014 International Conference on Multisensor Fusion and Information Integration for Intelligent Systems (MFI)*, pages 1–9. IEEE, sep 2014.

[44] H. Wang, M. Jenkin, and P. Dymond. Enhancing exploration in graph-like worlds. In *Proceedings of the 2008 Canadian Conference on Computer and Robot Vision*, pages 53–60. IEEE, may 2008.

[45] Y. Wang, H. Liu, B. Ren, and J. Chen. Node centrality analysis of multiplex networks under computer virus spreading. In *Proceedings of the 3rd International Conference on Robotics, Control and Automation - ICRCA '18*, pages 255–260. ACM Press, 2018.

[46] K. Wurm, C. Stachniss, and W. Burgard. Coordinated multi-robot exploration using a segmentation of the environment. In *Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1160–1165. IEEE, sep 2008.

[47] B. Yamauchi. A frontier-based approach for autonomous exploration. In *Proceedings of the 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, pages 146–151. IEEE Comput. Soc. Press, 1997.

[48] B. Yamauchi. Frontier-based exploration using multiple robots. In *Proceedings of the Second International Conference on Autonomous Agents - AGENTS '98*, pages 47–53. ACM Press, 1998.