Multi-Agent Exploration of Indoor Environments Under Limited Communication Constraints

Victor Spirin

Exeter College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2016

Abstract

This thesis considers cooperation strategies for teams of agents autonomously exploring an unknown indoor environment under limited communication constraints. The primary application considered is Urban Search-and-Rescue, although other applications are possible, such as surveying hazardous areas. We focus on developing cooperation strategies that enable periodic communication between the exploring agents and the base station (human operators). Such strategies involve an inherent trade-off between allocating team resources towards facilitating communication and increasing the speed of exploration.

We propose two classes of approaches to address this problem: using opportunistic rendezvous to guide the team behaviour, and explicitly arranging rendezvous between agents. In the opportunistic approach, the allocation of team resources between exploration and communication can be indicated with a single numerical parameter between 0 and 1 – the return ratio – which leads to complex emergent cooperative behaviour. We show that in some operating environments agents can benefit from explicitly arranging rendezvous. We propose a novel definition of a rendezvous location as a tuple of points and show how such locations can be generated so that the topology of the environment and the communication ranges of agents can be exploited. We show how such rendezvous locations can be used to both improve the speed of exploration and to improve team connectivity by allowing relays to contribute to the overall exploration.

We evaluate these approaches extensively in simulation and discuss their applicability in search-and-rescue scenarios.

Acknowledgements

First of all, I would like to express my gratitude to my supervisor Stephen Cameron. He got me interested in robotics when I was an undergraduate student, and without his encouragement, I may not have ended up doing a DPhil. His guidance and support made my time as a research student enjoyable and rewarding, and I could not have asked for a better supervisor. Stephen first interviewed me during undergraduate admissions more than a decade ago; that interview included a question on the Fibonacci sequence. Interestingly, the Fibonacci sequence also plays a role in this thesis. In a way, my time as a student in Oxford comes to an end the way it started!

I was fortunate to have been supported in this work by the Clarendon and the Mary Frances Cairncross Scholarships, without which I could not have pursued this DPhil.

Many thanks to Arnoud Visser who hosted me at University of Amsterdam several times and with whom we have had many fruitful discussions, and to Julian de Hoog for sharing his codebase with me and for his valuable advice.

Special thanks go to my colleague Helen Flynn for the hours and hours of fun we have had in the Skylab and at various RoboCup travels abroad, and who showed me how to see the world from an Irish perspective. It is no exaggeration that without her, this thesis might not have been completed! I am also grateful to Helen for proofreading this thesis, even though she had no interest in the subject matter.

Finally, I would like to thank my family for their support and understanding over these few years: my Mum, as I would never have even applied to do my undergraduate degree at Oxford, let alone pursue a DPhil, if not for her support and confidence in me; my Dad, who had to drive on the wrong side of the highway to get out of a traffic jam on the way to the airport to make sure I got to my admissions interviews on time.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Background and motivation

Recent advances in the field of robotics have allowed mobile robots to be used in a wide range of applications, from search-and-rescue operations and environmental surveying to exploration of other planets and bomb disposal. Teams of robots can also be used in such areas as manufacturing, medical, military, domestic and others. Using robots presents a number of advantages:

- Robots can have greater physical capabilities than humans. They may be stronger and faster; have sensors that humans don't have, such as infra-red or sonar, allowing them to collect more information about the environment than humans could; and they may be able to reach areas that are not accessible to humans, due to their physical size and method of locomotion (for example, flying robots may be able to reach high areas unreachable to humans).

- Robots can be more durable than humans, and able to operate in environments too dangerous for humans. They are able to withstand extreme high and low temperatures, radiation, toxic chemicals, and lack of oxygen.

- Robots may be able to make better decisions, and more quickly. For example, robots may be able to exchange information, plan paths, and evaluate and decide on cooperation strategies much faster than humans.

- Robots may be able to operate for longer periods of time without the risks of making mistakes due to tiredness.
- Robots are expendable.

Using teams of multiple robots instead of single robots offers a number of advantages. Distributing the task among several robots can speed up the completion of the task [107]. Such teams are more robust to individual agent failure. However, effectively operating a multi-robot team by either a single operator or by a team of operators is a difficult task. In order to increase the effectiveness of the robot team and to reduce the stress for the human operators, a degree of autonomous exploration and cooperation among the robots in the team is important.

In [9], unmanned aerial vehicles (UAVs) were used to survey a severely damaged power plant. The team consisted of more than 3 people to operate a single UAV – one person was ensuring the safety of the operation, one was piloting the UAV and one was controlling the camera on the UAV and monitoring the live video feed. Structural engineers were giving instructions to the team in order to acquire the information that they needed to assess the situation. Furthermore, an additional person was needed to aid the navigation of the UAV. Indoor exploration with the UAV was impossible due to clutter inside the area to be surveyed and lack of GPS, creating problems for indoor navigation. Time on site was limited and so full exploration of the environment with a single UAV was impossible. Having a team of highly autonomous robots to survey the environment would be highly beneficial in similar scenarios.

In 2012, after a series of earthquakes in Northern Italy, a team of robots consisting of two unmanned ground vehicles (UGVs) and two UAVs was deployed to survey the damage to the historical buildings in the area and to the artefacts that they contain [70].

In Fukushima in 2011 a Quince robot was deployed. WiFi communication would only have allowed the robot to be used in a very limited area, so cabled

communication was used. In one of the missions the communication cable got damaged and the robot ended up stuck [139]. This highlights the need for better robot autonomy and better strategies for dealing with the limited communication in operating environments.

1.2 Approach

This thesis aims to address the problem of exploring an unknown indoor environment by a team of autonomous robots where high-bandwidth communication is limited. The goal is to sense the environment as efficiently as possible, balancing the speed of exploration with keeping the human operators updated on progress as frequently as possible. To achieve this goal, some of the robots from the team are diverted from the task of exploration to the task of providing an ad-hoc infrastructure that provides periodic communication with the base station. As a result, in many scenarios the speed of exploration is in conflict with the frequency of periodic communication – one can be optimized at the expense of the other. In this thesis, we explore this trade-off relationship and propose a novel method for team coordination that is:

- Flexible to the desired trade-off between communication frequency and exploration speed.
- Allows for an easy specification of the relative importance of exploration speed vs communication frequency. This specification can be updated during the mission and propagated throughout the team.
- Highly robust to individual agent failure and dynamic changes in the environment.

Unlike other approaches based on periodic communication [30, 31, 32, 51, 76, 131] which use a fixed team hierarchy with a number of agents designated as relays throughout the mission or arrange meetings at specific locations and

times, our proposed approach uses implicit coordination which improves the robustness of team behaviour and still yields efficient performance. We show that our exploration strategy leads to complex emergent behaviour where some agents start acting as designated relays and form relay chains. The number of agents in the relay role is automatically adjusted throughout the exploration to satisfy user preference for team behaviour. In the proposed approach, the human operators can specify the relative importance of exploration speed vs communication frequency as a ratio, namely the ratio of information (or in a more general case, utility) collected by the team but not yet delivered to the operators, to the information already known at the base station. This lets the operator specify team behaviour as a fraction between 0 and 1, where 0 means that a greedy exploration strategy is used without any explicit attempts at communication, and 1 means that the team attempts to communicate with the operators after any new information is discovered.

The implicit coordination approach has a number of weaknesses. If an agent gets lost, it may be difficult to trace the location of the agent as the rendezvous are not arranged in advance and therefore do not have a pre-arranged time associated with them, which may help to narrow down the time when the agent got lost. In large open environments with relatively very small communication ranges agents are unlikely to communicate with each other without explicitly arranging a meeting, which may lead to inefficient exploration with lots of repeated coverage of the environment. Additionally, in some situations explicit planning for rendezvous location and timing can allow robots to exploit their communication ranges to greatly reduce the distance they have to travel, saving their energy, reducing risks of getting stuck when traversing difficult terrain, and improving team performance. For example, when exploring a multi-floor building, navigating the staircase connecting the floors can be difficult for many robots and should be avoided. It could then be highly beneficial to arrange rendezvous at either end of the staircase than to risk having one agent attempt to traverse the staircase in order to reach the base station. In the second part

of the thesis we present a novel approach for incorporating the communication ranges of the agents into their planning. Our approach builds on [30, 31, 32] to enable robots to plan for communication through thin walls and hard-to-navigate terrain. We show that using this approach can lead to improvements in both communication frequency and exploration speed.

The contributions in this thesis include:

- A highly scalable, robust approach for exploration with implicit coordination for communication.
- A coordination approach that explicitly plans for communication through obstacles and rough terrain.
- Development of a discrete-time simulator MRESim[1] which was originally introduced in [30]. The simulator is proposed as a middle ground of abstraction between the Agent and Virtual Robot competitions of RoboCup Rescue. Our work on MRESim won the 2014 Infrastructure competition of the RoboCup Rescue Simulation League [114].

[1] `https://github.com/v-spirin/MRESim`

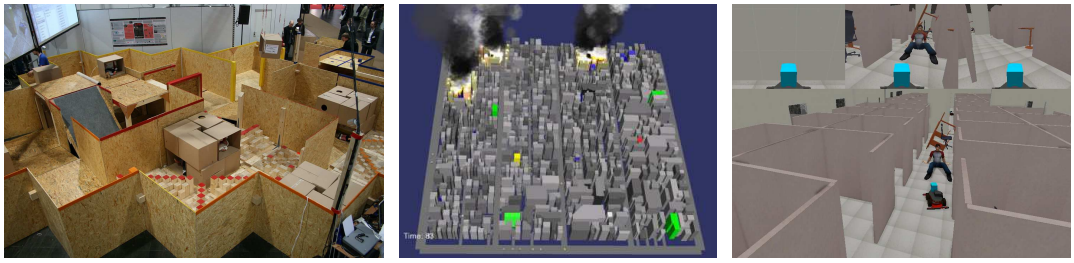
Chapter 2

Literature Review

We start the literature review with a brief overview of the recent developments in the field of rescue robotics. We then provide description of the state-of-the-art in low-level autonomous mobile robot tasks on which the approaches discussed in this thesis depend, such as Simultaneous Localization and Mapping (SLAM), environment representation, map merging and path planning. A review of literature on multi-agent exploration is given next, followed by a discussion of state-of-the-art approaches for dealing with limited communication in multi-agent exploration with a focus on exploration of indoor environments.

2.1 History of rescue robotics

The Great Hanshin Earthquake of 1995 in Kobe, Japan, claimed thousands of lives and prompted the continuing development of rescue robotic systems to help mitigate the consequences of such disasters. Rescue robots can assist human responders during search operations in environments that are difficult for humans to access or that might be dangerous. They can also increase the speed of rescue operations, which can save lives [117]. One of the results of this push for the development of rescue robotic systems was the proposal of the RoboCup Rescue League in 1999 [61] and the creation of the league as part of the RoboCup competitions in 2001. Robotic competitions provide a common testing environment for robotic systems, make it easy to benchmark approaches against each other, allow for comprehensive testing of the proposed systems and steer the development towards applying the systems in the real world [54]. The RoboCup Rescue League consists of several competitions, shown in Fig. 2.1. The Rescue Robot competition focuses on low-level behaviour of individual robots and features a testing arena built from standardized components that must be navigated by teleoperated and autonomous robots to locate simulated victims [110]. This competition includes Best-in-Class awards for teams that demonstrate reliable and best-in-class performance in such areas as mobility, autonomy and mapping, or manipulation. The Rescue Simulation competition features a simulated large-scale city immediately after a disaster; hundreds of simulated responder units must cooperate with each other in order to stop fires, clear the roads and rescue potential victims [4, 118]. The Virtual Agents competition provides a middle-ground between the two [26]. It uses a high-fidelity 3D physical simulator (formerly USARSim [67], now Gazebo [123]) with validated robot and sensor models, where a team of robots (usually up to 8) must navigate an unknown indoor or outdoor environment and locate the victims. The team is controlled by a single operator but in recent years the competition includes a "hands-off" period where the operator has no control of the



(a) (b) (c)

Figure 2.1: RoboCup Rescue League competitions. (a) Rescue Robot competition test arena [1]; (b) 3D visualization of a typical scenario in the Rescue Simulation competition [125]; (c) A screenshot of 3 robots searching for victims in a USARSim simulation as part of the Virtual Agents competition.

robots and the team must perform autonomous exploration, shifting the focus from designing human-robot interfaces towards full autonomy. The DARPA Robotics Challenge [96] was started in 2013 with the goal of creating robots that can interact with human environments and use human tools. The focus here was on low-level behaviour and control of a humanoid robot in a rescue scenario. The virtual part of the competition, using the Gazebo simulator [3], attracted nearly 100 participants.

2.2 Mapping and navigation

A good on-line mapping and navigation system is essential for a team of robots to be able to cooperate successfully in exploring an unknown environment.

Simultaneous localization and mapping (SLAM) systems can be grouped into two categories: online and full SLAM. The online SLAM algorithms are incremental, estimating the current pose of the robot and discarding past measurements after they have been processed. The full SLAM approaches estimate the entire trajectory of the robot based on all the observations, instead of estimating just the current pose [120]. When referring to graph SLAM algorithms, there are similar concepts of front-end and back-end systems. Front-end systems can be thought of as dealing with local changes in the map, estimating the current pose of the agent and building a graph representing the spatial

constraints between the local maps (usually using a form of scan matching). Back-end systems deal with global changes, estimating the poses along the full path of the agent that best satisfy the constraints (for example, for loop-closing) [46].

A survey of the history of the SLAM problem and the proposed solutions is given in [36]. Here we present a brief review of the recent advancements in this area.

2.2.1 HectorSLAM

RoboCup Rescue Robot League uses environments built from standardized elements from the DHS-NIST-ASTM International Standard Test Methods for Response Robots [93]. These aim to resemble real operating settings for rescue robots and include terrain of various complexity. More detailed information about the test environments used is available in [54]. One of the challenges in the league involves having a robot autonomously navigate and map an arena which includes ramps and other obstacles. Recent years have seen steady increases in the quality of the maps produced by the robots (see Fig. 2.2) [93]. Here we will describe the approach HectorSLAM used by the Best-in-Class Autonomy 2012, 2013, 2014 and 2015 award winning team Hector.

In their approach to SLAM, team Hector take advantage of the high frequency of updates and the precision of modern laser scanner (LIDAR) systems. In addition, they take into account the unreliability of odometry in USAR settings due to high risks of wheel skid and robot drift when navigating uneven terrain. A schematic view of the mapping and navigation system used is shown in Fig. 2.3. The system consists of two modules running in parallel – the SLAM module and the navigation module. The navigation module uses an Extended Kalman Filter (EKF) to take the measurements from an Inertial Measurement Unit (IMU), the 2D pose estimation from the SLAM module (which is sent to the navigation module as soon as the pose estimation becomes available) as well as measurements from other sensors if available (while GPS is generally

Figure 2.2: On the left is the final map produced in 2007 by the Best in Class Autonomy team Resko from University of Koblenz-Landau (Germany). On the right is the final map by the 2012 Best-in-Class Autonomy team Hector from TU Darmstadt (Germany). This team also won Best-in-Class Autonomy in 2013, 2014 and 2015 [2]. Image taken from [93].

useless in indoor USAR scenarios, other sensors such as a compass, an altimeter of odometry might be available and their measurements could be incorporated into the EKF). The navigation module then outputs a full 6D pose estimate, which is projected on the 2D mapping plane and fed into the SLAM module. The SLAM module uses this pose estimate as the start estimate in the scan matching process. The scan matching process uses a fast variant of gradient descent (the Gauss-Newton method) to perform fast near-real-time matching of the point cloud obtained from the laser scanner to the existing occupancy grid map. Scan matching is performed on several separate occupancy grid maps stored at different resolutions in an attempt to avoid getting stuck in local minima during gradient descent. A more detailed description of the approach is available in [66].

The SLAM system developed by team Hector is available as an open-source ROS (Robot Operating System) package *hector_slam* [64]. ROS [97] is an open-source middle layer between robot hardware and the software modules and is widely used by the robotics community.

This is a purely front-end system which does not include pose graph opti-

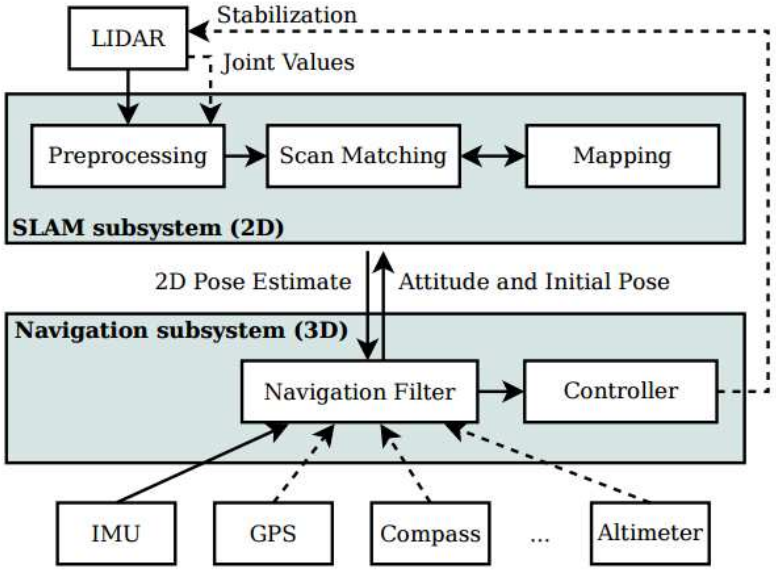


Figure 2.3: A schematic view of the mapping and navigation system developed by team Hector. Image taken from [66].

mization, including loop closing. RoboCup Rescue Robot League has relatively small but technically challenging environments, and the approach is able to produce accurate maps without explicit loop-closing. In [99], HectorSLAM was successfully incorporated into a graph-based SLAM approach. HectorSLAM was used to create local maps. New local maps were created and added to the graph when the robot has either travelled a fix distance, or when the covariance of the local map indicates that the robot is unable to localize itself within the local map (for example, when navigating featureless corridors, where detecting robot motion from the scanmatcher alone is difficult). When a potential loop in the graph is detected, the graph is optimized – the positions of the local maps are adjusted to satisfy the loop closing constraints, and the local maps themselves are adjusted to reduce the error between consecutive scans belonging to connected local maps. [99] indicates that the resulting maps are improved over HectorSLAM for large environments containing long featureless corridors, while still being able to run on-line (with an overhead in each scan-matching iteration of only 0.5ms, and an 8ms overhead when loop-closing is performed).

2.2.2 Particle filter approaches

One of the most popular ROS SLAM packages is GMapping[1], an approach based on Rao-Blackwellized particle filters [48]. Using particle filters is a popular and promising approach to solving SLAM ([23, 47, 48, 60, 82, 83]). In approaches based on particle filters, usually each particle represents a possible trajectory of the robot (that is, a set of robot poses), and the associated map (for example, an occupancy grid map). The samples are weighted with their likelihood, given the sensor observations. When new observations become available, each particle samples a new pose given the measurements and updates its map given the new pose. Then, the weights are recalculated for each particle, and the particles are re-sampled. There are two main problems with using particle filters. Firstly, due to the space complexity, it is important to keep the number of particles small. Secondly, when the particles are re-sampled, it is possible that important particles might get lost. In GMapping [48], the number of particles required is reduced by using a scan-matcher on the observed laser scans, which increases the quality of the particles. The frequency of resampling is also reduced by only resampling when the set of particles is believed to reflect the target distribution poorly, which usually happens when a loop in the environment is closed.

2.2.3 Visual SLAM

Visual SLAM has been an active area of research in recent years. The goal of visual SLAM is to use image data from a visual light, infrared or an RGB-D camera in order to produce a map of the environment and localize the robot in the map. A comprehensive survey of literature on this topic is available in [43].

[1] http://wiki.ros.org/gmapping

2.2.4 Representing the environment

One of the simplest and most common ways of representing the environment is as a two dimensional grid. Each cell in the grid represents an area in the environment, storing the probability that the area contains an obstacle. Maps produced in this way are called occupancy grid maps [38], and are most useful for representing environments that can be considered flat for navigation purposes. The occupancy grid can be augmented with additional information, such as elevation data, to create a "2.5D" representation of the world [65]. Quadtrees [106] can be used to store occupancy grids more efficiently, particularly when the grid is mostly empty [69, 128]. A similar approach based on octrees has been used to represent maps in three dimensions [135]. However, when a single surface is used for robot navigation, a 2.5D occupancy grid is usually sufficient.

Topological maps, generally, are higher-level abstractions of occupancy grid maps. They can represent whole locations (rooms or areas) as one node in a connected graph. Nodes on the graph are connected with edges if there is a direct path between the nodes, and the weights of the edges correspond to path costs. This greatly reduces the state space, making path planning more efficient. Building compact topological maps is more complicated than building occupancy grid maps and they do not allow a robot to navigate effectively within each location. Combining topological maps with occupancy grid maps was originally proposed in [119]. In this approach, a topological map is built from an occupancy grid by creating a Voronoi diagram, which consists of a set of points equidistant from 2 or more obstacles. The points on the Voronoi diagram where clearance is locally minimized are called critical points; the occupancy grid is partitioned around these points.

Semantic maps are a concept related to topological maps. In a semantic map, areas of the environment are labelled with human-level knowledge, such as 'corridor', 'room', 'kitchen', 'office' etc. Such labelling can aid both human

responders using the map produced by the robots and robot autonomous behaviour (for example, robots searching for human survivors in a building may have prior information that victims are likely to be in particular types of rooms [75]).

[42] uses conditional random fields on the Voronoi graph (called Voronoi random fields) built from the occupancy grid in order to come up with semantic labellings of areas of the map (room, corridor, doorway). Then, a semantic map is built according to this labelling. [115] uses features extracted directly from the laser scans in order to classify areas according to semantic labels. [77] uses prior information about the type of building being mapped (e.g. school, office, apartment building) to infer the likely semantic labelling of areas.

2.2.5 Merging maps from multiple agents

Merging maps produced by multiple exploring agents is a challenging problem. Occupancy grid maps can be merged using computer vision inspired techniques such as the Hough transform [24]. This approach has been shown to perform well, merging maps from public data repositories in less than 200ms, even if robots that produced the maps do not know their relative localizations in the maps being merged [24]. In a USAR scenario, it can be assumed that robots are localized at the start of the mission with regards to the base station and to each other; frequent rendezvous among agents allow agents to re-localize with respect to each other, making the map merging problem easier to solve. In [53], accidental rendezvous among agents is used to facilitate map merging. When two robots have line-of-sight communication, they are able to localize themselves relative to each other. If they are also localized within their local maps, the map merging process becomes straightforward. In [41], it is assumed that robots start exploring the environment from unknown locations. When two robots enter communication range, they attempt to localize themselves relative to each other and arrange a line-of-sight rendezvous to verify their mutual localization. If the rendezvous succeeds, they assume that the localization was

correct and merge their maps.

2.2.6 Path planning

One of the most crucial elements of efficient autonomous exploration which is common to almost any strategy or approach used is the ability to quickly and efficiently plan paths in the environment. The ability to generate accurate paths through the environment enables agents to more accurately predict the time it would take them to reach unexplored frontiers and other locations of interest. This in turn allows them to generate plans that are closer to optimal. If it takes too long for agents to (re)compute paths, it raises a number of problems. They may need to use cruder estimates such as straight-line distances to estimate costs, which can result in very sub-optimal plans. They may have to re-plan less frequently which can reduce their ability to quickly react and adjust to changes such as changes in the environment itself in the case of operating in a dynamic environment or changes to the state of the agent or the team. And finally, they may not be able to navigate to a certain area of the environment at all if the path cannot be calculated in a reasonable time. In that case the agent may have to assume that part of the environment to be not reachable from its current location. Here we will give an overview of some of the techniques for planning paths efficiently on occupancy grid maps.

2.2.6.1 A* search

Perhaps the most popular algorithms for path planning is A* [50]. It is essentially an extension of Dijkstra's algorithm [35], utilising heuristics. It performs best-first search on the occupancy-grid map, using the lowest known heuristic cost at each node. It always finds a path, if one exists. If the heuristic never overestimates the real path to the goal and is consistent, then the path will also be optimal. The implementation of this algorithm is very easy and straightforward on occupancy grid maps. However, it can be very slow with a sufficiently high resolution of the occupancy grid. A recent optimization of

A*, Jump Point Search [49], significantly reduces the search-space by eliminating the search-space symmetry. Unlike many other popular approaches, it preserves optimality of A*, while being an order of magnitude faster on many occupancy grid maps, particularly ones that contain large open spaces.

2.2.6.2 Rapidly exploring random trees

Another popular approach for robot path planning is using rapidly exploring random trees [73]. This is an approach based on random sampling that does not guarantee optimal paths, but is fast and allows for path planning for non-holonomic vehicles, as well as in higher dimensions [74]. The algorithm starts with the start location being in the path tree. Until a path to the goal location has been found, a location is randomly sampled from the environment, is checked for validity, and if there are no obstacles on the straight line between the sampled location and the nearest node already in the tree, an edge is created between the two nodes.

2.3 Multi-agent exploration

When multiple agents are exploring the same environment, an important problem is to assign areas of the environment to different agents in such a way that you don't have several agents exploring the same area, resulting in wasted effort. Also, in some cases it may be beneficial to send agents to completely different sectors of the environment in order to avoid collisions. In addition, it may be important to identify areas of the environment that, once explored, would give other agents more new frontiers to explore and thus enable more efficient exploration / frontier allocation (for example, in indoor environments such areas could be corridors). In this section we will talk about some established as well as emerging techniques for dealing with this problem.

2.3.1 Frontier exploration

Using frontiers to distribute the exploration task among multiple agents is a common approach to multi-agent exploration. A frontier is a boundary between the explored and unexplored parts of the occupancy grid map [137]. Agents can choose a frontier to explore by estimating their path costs of reaching the frontiers, as well as those of other agents in their vicinity, with the goal of maximizing overall exploration utility. Estimating the expected information gain from navigating to a frontier is important for efficient exploration. In [45], the information gain from a frontier is estimated by sampling locations near a frontier and simulating casting rays from that location that pass through the frontier cells (see Fig. 2.4). The concept of frontier polygons was introduced in [129]. Laser measurements at distances close to the maximum range of laser range-scanners become sparse and therefore cannot guarantee absence of obstacles. Areas sensed at a short range (2 metres given the full scanner range of 20 metres) are considered *safe space*. A frontier polygon is then defined as the polygon bounded by frontier cells, obstacle cells and safe space boundary cells. The frontier polygons are likely to be free of obstacles but need to be explored further, and their area can be used to estimate the information gain from exploring the frontier polygon.

In order to allocate the frontiers to robots well, some metric is required that measures how good each assignment is. Usually, an utility function is defined which incorporates multiple criteria. The utility function in [137] simply favours the nearest frontier; [45] incorporates both the distance to the frontier and the expected information gain into the utility with the following utility function:

$$U(f) = A(f)e^{-\lambda d(f)}$$

where $U(f)$ is the utility of navigating to frontier f, $A(f)$ is the measure of information gain from frontier f, λ is a tuning parameter and $d(f)$ is the cost of navigating to the frontier (often this is just the distance). A large λ means that the utility function favours nearby frontiers, meaning that the robot will

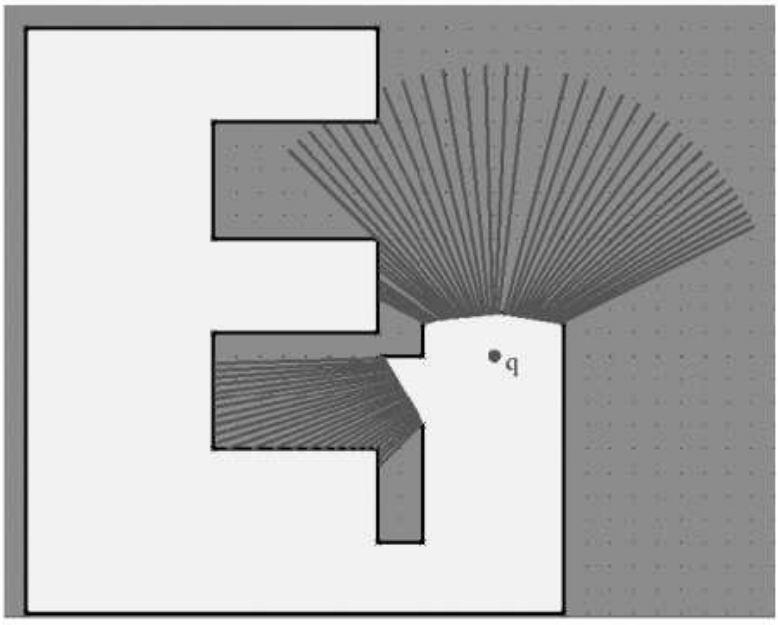


Figure 2.4: The potential information gain of sensing through a frontier from a candidate location q is estimated as the area that is 'visible' by casting rays through the frontier cells from q. Image taken from [45].

try to fully explore the local area before moving on. A small λ will get the robot to explore the largest frontiers first, meaning that it will often push deep into the environment and 'fill in the details' later. [126] has a similar utility function $U(f) = A(f)/d^n(f)$, where the parameter n plays a similar role to λ above. [127] additionally includes the probability of communication with the base station at the target location into its utility function. The utility functions are usually defined ad hoc, designed to combine specific criteria and parameters. [14] proposed an approach to defining utility functions that is grounded in decision-theoretic Multi-Criteria Decision Making (MCDM). This allows the definition in a systematic manner of utility functions for any number of criteria that can be mutually dependent.

2.3.2 Assigning agents to frontiers

In order for a robot team to explore an environment effectively, the exploration task needs to be divided among the agents in the team. In the context of

frontier exploration, that means assigning an exploration frontier to each agent in such a way as to maximize overall team utility. [138] uses a naive approach where each agent decides on the best frontier to visit independently of other agents. This can often result in several agents picking the same frontier to explore, leading to inefficient exploration. In [20], the utility of a frontier is estimated based on the expected area that an agent may sense when it arrives at a frontier. Once an assignment of an agent to a frontier that maximizes utility is made, the utility of sensing the unknown occupancy grid cells in the vicinity of the frontier is reduced. This makes it less likely that other agents will be assigned to nearby frontier cells. [136] use the Hungarian method [71] to get an assignment of agents to frontiers that is guaranteed to maximize overall utility. In [127] a faster greedy assignment method is described. First, approximate utilities are computed for every agent/frontier assignment using Euclidean distance instead of the true path length which is costly to calculate. For the assignment with the maximal approximate utility, the true utility is computed. If the assignment still has the maximal utility, then the frontier is assigned to the agent, and all pairings that include this frontier and this agent are removed; otherwise, a new maximal approximate utility is considered. This method of assignment can be used centrally (where one agent or the base station computes the assignments for all the agents, and then communicates the assignments to them) or in a distributed way, where each agent performs the assignments for themselves based on their beliefs about the locations of other agents – the latter being more robust to communication drop-out.

Often several frontiers can be located in the same area, and multiple agents may be assigned to explore these frontiers. This may be inefficient as a single agent may be able to quickly sense all frontiers in the area, while other agents may be better utilized sensing other areas of the environment. Additionally, several agents navigating the same area may interfere with each other, blocking the paths and affecting the sensor measurements. To address this issue, [136] partitions the available frontiers into groups by area and performs the assign-

ment hierarchically. First, agents are assigned to areas; then, in each area, agents are assigned to individual frontiers. [140] has agents planning *tours* of frontiers, instead of only considering the utilities of visiting one frontier at a time. A number of goals, up to a maximum, are added to the tour of each agent in order of shortest distances between the goals. A market approach is then used to exchange goals between agents. Each agent auctions off each of the goals to all the agents who take into account their costs of reaching the goal taking into account their own tours. The agents then place bids, and the agent who can reach the goal while minimizing the costs acquires the goal.

Another class of approaches for allocating agents to frontiers are the approaches utilising potential fields. Using imaginary forces acting on a robot for obstacle avoidance was originally proposed in [59]. The general idea behind these approaches is that obstacles exert repulsive forces on the robots and goals exert attractive forces. These forces diminish in effect with distance from the source of the force. The sum of these forces determines the direction of travel of the robot, steering them towards the target areas while avoiding the obstacles on the way. In [72], an additional repulsive force between robots is added to ensure that robots are spread throughout the environment. These approaches are generally favoured for their simplicity and elegance and the ability of robots to make good decision with only local planning. However, there are several common problems associated with such approaches. Agents can get stuck in local minima, unable to reach the goal. It may be impossible to pass between obstacles that are close to each other, or robots may oscillate when attempting to navigate narrow passages [68]. [101] attempts to overcome the problem of getting stuck in local minima by assigning one of the robots to act as a leader, acting independently of what the other robots do, using path planning to reach the nearest frontier. In [100], a solution to the oscillation problem is proposed. The main advantage of these approaches is that they can lead to complex emergent behaviour by only making local decisions, which increases the robustness of the system. However, they may not be able to fully

utilize the additional information that may be obtained through higher-level planning and coordination. A trade-off may involve using higher-level planning to allocate navigation targets among agents and using potential fields for local navigation and obstacle avoidance.

2.3.3 Particle Swarm Optimization techniques

In some multi-robot exploration scenarios, agents may be able to evaluate their proximity to the goal of the exploration using on-board sensors. One example of such a scenario would be attempting to locate the source of a gas leak in a large indoor environment using a large team of homogeneous robots [78]. In these types of scenarios, Particle Swarm Optimization (PSO) [37] techniques can be useful. PSO is an optimization algorithm inspired by social biological systems. A swarm of particles is searching a multi-dimensional space; initially, the particles are moving in random directions with random speeds. At each location, each particle is able to evaluate its utility function (indicating its proximity to the goal). Each particle stores the location where it was able to sense its maximal value of the utility function (locally best location), as well as the location where the maximal utility was observed by all its neighbours (globally best location). At every step, the velocity vector of the particle is updated using a linear combination of its current velocity and the directions towards the locally and globally best locations observed so far. By assigning a particle to each robot, and using the concentration of gas in the air as the utility function, teams of robots are able to efficiently locate the sources of gas leaks [78].

When using the above approach, the search may get stuck in a local solution. In addition, the approach does not take into account the potential presence of obstacles in the environment. In [28] these weaknesses were addressed. The utility function was augmented to take into account the presence of obstacles nearby, with the effect of guiding robots away from sensed obstacles. Furthermore, the team of robots was separated into several swarms. If a swarm does

not find an improved global solution after several timesteps, it is punished by having the worst performing robot excluded from that swarm. The excluded robots perform a random walk in the environment. If a swarm performs well, it has a small probability of getting rewarded and including the best performing unassigned robot. Experimental results in simulation show that this approach is effective at avoiding local optima when performing the search task.

2.3.4 Other approaches

Reinforcement learning has been successfully applied in a variety of applications in the field of robotics [63]. Using reinforcement learning for multi-robot exploration tasks remains challenging, with problems including managing the large state and action space, as well as assigning credit to the actions of individual agents. In [44], hierarchical reinforcement learning is used for agents to learn a policy to optimize the communication needed for proper coordination, given the communication cost. In hierarchical reinforcement learning, the overall task is decomposed into subtasks that are important for solving the problem. A policy for each of the subtasks is then combined into the overall hierarchical policy. The policies are learned using a temporal-difference learning method [116]. In [80], a team of robots uses Q-Learning [132] in order to learn the allocation of roles among the team in order to cooperatively perform a foraging task while repairing broken or stuck robots. A survey of recent work in multi-agent reinforcement learning is available in [21]. While it remains a young field of study, the use of reinforcement learning for multi-agent cooperative exploration shows much promise for the future.

In [8], it is assumed that during the exploration mission access to some areas of the environment may be blocked, requiring the cooperation of multiple robots to clear the passages. Heuristics are proposed that can be used to decide when a robot should stop its own exploration in order to assist other robots; these heuristics are shown to decrease the time required to explore unknown environments.

2.4 Dealing with limited communication

In search-and-rescue scenarios there is often no functioning pre-existing communication infrastructure available. Wireless communication often has limited range and is unreliable due to attenuation caused by obstacles and debris that might be present in the environment. Fiber-optic tethers may break or tangle and can limit robot mobility [84]. In this section we outline the methods designed to attempt to overcome the constraints of limited communication. The approaches can be roughly separated into those that attempt to keep agents in communication range of each other and/or with the base station (for example, to enable teleoperation of the robots), and those that only aim to establish periodic communication for data exchange.

2.4.1 Keeping the team connected

The probability of having communication with the base station near a frontier is taken into consideration when deciding frontier utility in [127]. The result is that agents avoid leaving the communication range of the base station. In [10], agents attempt to maintain communication with the base station via a multi-hop ad-hoc network [94]. If during the exploration, an agent loses line-of-sight connectivity with the base station, it backtracks until connectivity is re-established and then acts as a stationary relay node for other agents which continue the exploration mission. While effective at extending the overall communication range of the base station towards the goal, exploration may proceed slowly due to most of the agents in the team acting as static relays. [86, 87, 88, 89] attempted to make the provision of the communication infrastructure transparent to the operator of a single robot by using a convoy of simpler relay robots which ensure the primary robot maintains communication with the operator. In [102], communication with the base station is maintained by, at every timestep, sampling the possible configurations of robot positions in the following timestep. Then, the configuration with the best utility is selected;

configurations which lead to a loss in team connectivity are penalized, while the ones which lead to exploring a frontier are rewarded. Unlike the approach in [94] described above which uses decentralized planning, in [102] centralized coordination is used. In [92], a heterogeneous team of robots is used. The team consists of exploring robots (called *frontier nodes*, or FN) and a relay-deploying robot (referred to as a relay-deployment node, or RDN). While the exploring robots use an efficient exploration strategy without regard for communication constraints, the relay-deploying robot attempts to deploy relay nodes to provide them with the communication infrastructure while minimizing its travel cost and the number of relays deployed. An illustration of the approach is given in Fig. 2.5. Another approach for maintaining communication in a robot team, particularly when operating in an outdoors environment, is to use a multi-tier approach [131]. A team consisting of 3 tiers is proposed: ground robots that perform sensing, exploration, and carry out tasks on the ground; UAVSearchers, providing communication links and sensing the environment from above; and UAVCommBridges, which are high-altitude autonomous aerial vehicles, and are there primarily to provide communication links for the other agents. An illustration is provided in Fig. 2.6. In [18, 39, 55, 62], agents communicate implicitly by dropping tags in the environment that can be read and written to by other agents, reducing the reliance on the availability of long range wireless connectivity.

Some approaches try to keep *sub-teams* of robots connected to each other. This leads to robot packs exploring areas of the environment together, maintaining communication. [103] uses a centralized approach, where one agent in the pack assumes the role of the central controller. Possible robot configurations for the next step are sampled and the utility of each configuration is evaluated, with configurations where any agent gets disconnected from the pack getting penalized. In some cases, the pack can end up in a *deadlock* – the communication constraints may be preventing agents from reaching their target frontiers or navigating around obstacles [104]. An example of a deadlock

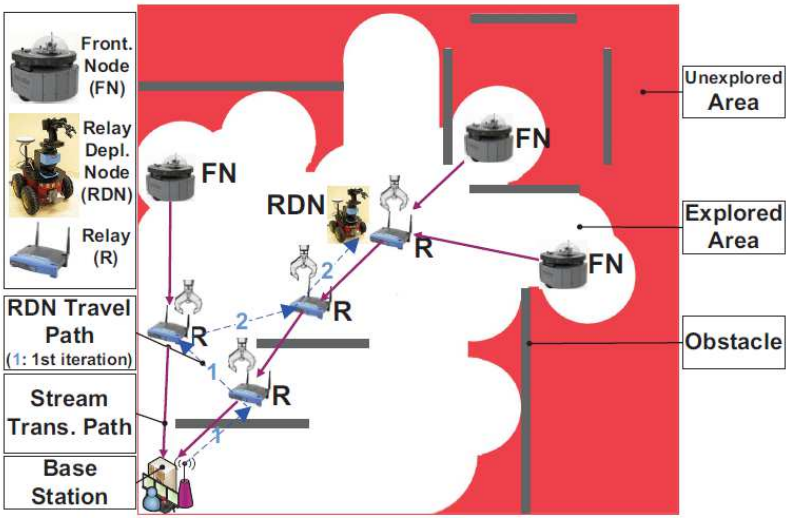


Figure 2.5: Using a heterogeneous team to deploy relay nodes in the environment. Image taken from [92].

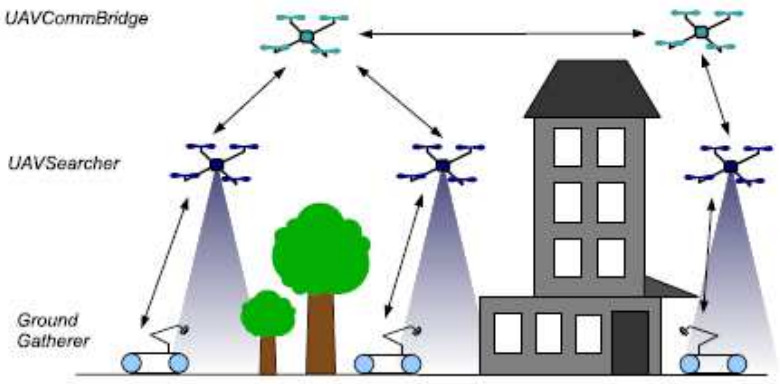


Figure 2.6: A multi-tier approach to providing communication infrastructure. Image taken from [131].

is shown in Fig. 2.7. Once a deadlock is detected, one of the stuck robots is selected as a meeting point, the connectivity constraint is temporarily dropped as other agents in the team head towards that meeting point. Once they are near, the connectivity constraint is re-enabled and exploration resumes as normal, with the deadlock situation resolved. This approach was shown to allow packs of robots to fully explore the environment. In [111], a distributed bidding mechanism is used to keep robots that are already within communication range close to each other when exploring new frontiers. Agent may drop out from the packs; then, if they establish communication with an agent in another

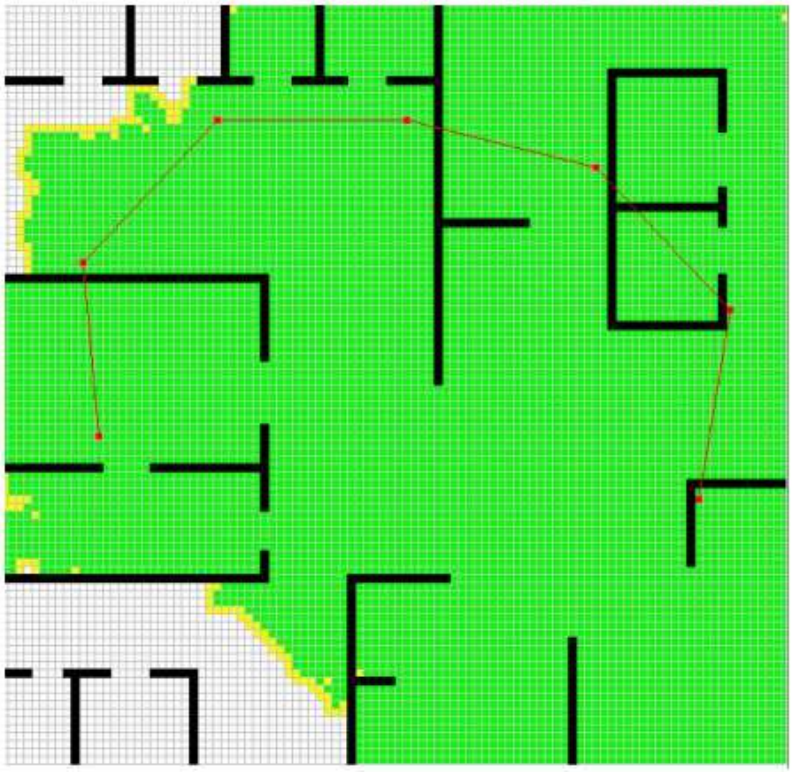


Figure 2.7: An example of a robot pack in a deadlock. Robots are shown in red, lines between robots indicate communication links. Explored area shown in green, frontiers are in yellow, unexplored space in white, obstacles are in black. Image taken from [104].

sub-team, they join it. If agents belonging to two sub-teams establish communication, the sub-teams merge together. Keeping agents within communication range of each other leads to reduced total exploration time; simulation results reveal that robots tend to form clusters during the exploration.

2.4.2 Periodic communication

While maintaining wireless communication within a robotic team is highly beneficial, it can also hinder the progress of the team towards the goal of exploring and mapping the unknown environment as quickly as possible. Relaxing the connectivity constraints may allow the robots to explore the environment more efficiently while periodically communicating with each other and with the base station to exchange information. A naive approach would be to sim-

ply have agents perform frontier exploration without regard for communication constraints, and return to the base station when they run out of frontiers to explore. This method has several downsides – if the team spends a long time without communicating with the base station, the human operators may not know what the state of the team is (the robots might have got stuck or got disabled), and any information gathered by the team is not available to the operators until the end of the exploration. Furthermore, if members of the team spend a long time without the opportunity to communicate with each other, team coordination may suffer. Consider, for example, the case where two robots end up following the same path, with the second robot just outside the communication range of the first robot at all times, re-sensing the same areas as the first robot. In this case, the work of the second robot is wasted – had they been cooperating effectively, they might have explored the environment a lot more efficiently. An natural extension of this strategy is a *periodic return* approach, where agents return to the base station after progressively increasing amounts of time spent exploring. While this improves the team coordination by periodically sharing knowledge, this approach can still be very inefficient. [51] examine a related scenario where a team of agents attempt to search an environment for which a map already exists where the team must regain full connectivity at fixed intervals and show that the problem is NP-hard.

2.4.2.1 Role-based exploration

Role-Based Exploration [30] allows a team to explore the environment with periodic communication while reducing redundant behaviour among the agents. This is achieved by dividing the team into two groups of agents: explorers and relays. The task of the explorers is to explore as much of the environment as possible and return it back to pre-agreed rendezvous points at pre-arranged times. The task of relays is to communicate information between rendezvous points (where they communicate with explorers or other relays), or between a rendezvous point and the base station. Teams of agents have a rigid hierarchy

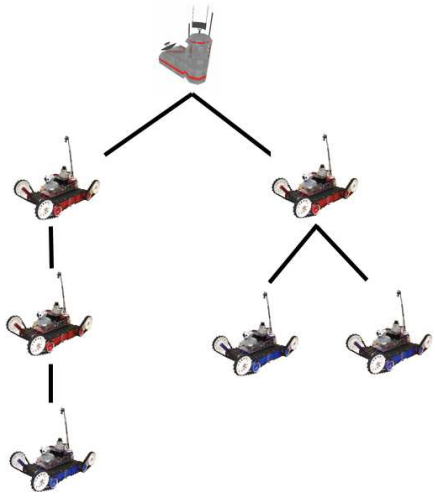


Figure 2.8: An example of an agent hierarchy for Role-Based Exploration. At the top of the tree is the base station; red agents are relays; blue agents are explorers. Image taken from [32].

tree which is manually selected before the agents enter the environment, but agents may switch positions in the tree throughout their mission [31]. However, the shape of the tree itself does not change. Fig. 2.8 shows an example of such a tree.

When an explorer meets its parent relay, they exchange information about the environment, ensuring that they both have the same knowledge about it. Then, the explorer suggests a rendezvous point, normally near a frontier that it plans to explore next, and a fall-back rendezvous point in case the primary one cannot be reached, which is especially useful in dynamic environments (if the explorer and the relay happen to meet before they both reach their meeting point, they act as if they have reached it and proceed to exchange information and to re-plan their next meeting). Because the explorer and the relay share the same information, the explorer can predict how long it will take for the relay to get back to its own parent, and then travel to the new meeting point, and can therefore decide when it should stop exploring the new frontiers and get back to the agreed rendezvous point [30].

The selection of the rendezvous points is therefore crucial for the performance of the algorithm, as it affects how long the agents get to spend exploring

the environment, and how often they deliver their information back to the base station. The further away from the base station the meeting points are, the more biased the exploration effort is to exploring deeper into the environment instead of relaying the information back. Also, selecting meeting points at junctions and in corridors where the communication range is wider leads to increased performance of the exploration approach [32].

In [130], this approach is extended to teams of UAVs performing search-and-rescue in outdoor wilderness scenarios. Here, all of the team-mates periodically meet at the pre-arranged rendezvous locations where they decide how many and which members of the team will be acting as a relay chain for the team in order to closely meet the communication latency requirement specified by human responders.

Chapter 3

Assumptions and Methodology

In this chapter we describe the assumptions about the problem domain and define key performance metrics that we will use throughout the thesis to evaluate proposed approaches. This is followed by the description and discussion of the simulation environment used.

3.1 Assumptions

In this thesis we make the following assumptions:

1. Robots are able to build reasonable planar maps of their environments, and are able to localize themselves within those maps well. Given the performance of recent scan-matching algorithms, we believe it is a reasonable assumption to make even today. In discussing the approaches presented in this thesis, we do consider the possibility of some localization errors.

2. Robots have reasonable low-level control, obstacle avoidance, and low-level wireless communication protocols. Robots are able to navigate obstacle-free paths in their operating environment. While we do not assume any particular type of robots, the assumption is that the robots are equipped with a laser range scanner sensor and have on-board computing and storage capacity. Additionally, the robots are equipped with high-bandwidth communication hardware and are able to form ad-hoc multi-hop communication networks when in range.

3. The environments are assumed to be indoor and static, and can be represented adequately as 2D maps of obstacles (that robots cannot navigate through) and free space (that robots can navigate through). The free space may be further subdivided into various degrees of traversability (that is, robots may be able to move with different speeds through different types of free space). In the latter case we assume that robots are able to sense the type of free space in their near vicinity (for example, as in [121]).

4. All robots start their exploration next to each other and next to a base station. There is only one base station, which does not move during the mission. The goal of the robots is to sense and map the environment and deliver the information back to the base station. We do not make any

assumptions about how much of the environment is known prior to the start of the exploration.

3.2 Performance metrics

In order to analyse and compare exploration strategies, we define the following performance metrics. We make a distinction between the metrics that measure the overall performance of the team with regards to the mission goals, and the metrics that measure the internal state of the team and affect the mission goals indirectly.

3.2.1 Metrics directly capturing exploration goals

The following metrics reflect the exploration goals directly:

1. Total time to complete the mission $t_{complete}$. In the scenarios we are interested in, the mission is defined as mapping the entire environment (or, the target percentage of the environment), and delivering that information back to the base station. This metric is to be minimized.

2. Knowledge about the environment at the base station at time t, $K_{BS}(t)$. The mission itself can have a time limit imposed on it (in RoboCup Rescue Virtual Robots league, this is typically 20 minutes), and in that case, information known at the command centre at the end of that time limit is important. Also, as soon as information is available at the base station, it can be used by human responders to aid them in their decision-making. Therefore, this metric is to be maximized for every time t.

3. The responsiveness of the team to base station commands. As new information becomes available to human responders, it may be useful to prioritize the exploration of a particular region on the map or to change the exploration strategy of the team. If the environment becomes too

dangerous, it may be necessary to terminate the mission early. In addition, this metric also gives an indication of how often information from the agents becomes available at the base station, allowing the human responders to have visibility of the team progress. We define this metric as the average and the maximum time it takes for messages from the base station to reach a particular agent, as well as all the agents in the team. We assume that the base station sends out a message to all agents once in every timestep, and measure the times it takes for each of these messages to reach every agent in the team. An agent does not have to directly communicate with the base station in order to receive messages - such messages can be ferried by other agents. Let A be the set of agents in the team. Let L_k^i be the timestep at which the agent $i \in A$ received the message that the base station sent out at time k. Then we define the maximum latency for agent i at time t_{max} as $MaxLat^i = \max_t(L_t^i - t)$ and average latency as $AvgLat^i = \frac{\sum_{t=0}^{t_{max}}(L_t^i - t)}{t_{max}}$. Correspondingly, the maximum and average latencies for the whole team are defined as $MaxLat = \max_i MaxLat^i$ and $AvgLat = \frac{\sum_{i \in A} AvgLat^i}{|A|}$. These metrics are to be minimized.

A somewhat related metric, used in [91], is the maximum amount of time that an agent has spent without a wireless connection to the base station. This latter metric is more useful when there is a requirement for a robot to be teleoperated, and is not very useful for evaluating strategies with periodic communication like the ones discussed in this thesis, as agents are expected to only maintain connectivity with the base station long enough to periodically exchange information.

Note that the above metrics are conflicting to some extent: while it is possible for one exploration strategy to be better than another in all of these metrics, in most cases a trade-off has to be made. For example, minimizing $t_{complete}$ is likely to require all agents in the team to be exploring areas outside of the communication range of the base station, while minimizing $MaxLat$ or

AvgLat is likely to require most of the agents in the team to be providing a multi-hop network to increase the communication range of the base station. Specifying the desired trade-off is non-trivial.

3.2.2 Metrics with an indirect effect on the exploration goals

The metrics below have an indirect effect on the overall exploration goals. Nevertheless, they give us valuable insight into the behaviour and performance of the strategies, and allow us to better analyse the relative strengths and weaknesses of the approaches.

1. The combined knowledge of the whole team at time t, $K_{team}(t)$. While generally, this information is of little use to human responders until it is delivered to the base station, it nevertheless gives us a valuable insight into the overall speed of the exploration and gives us an indication of where a strategy could be improved by allocating more resources to deliver the information to the base station faster.

2. The amount of time each agent spent sensing new areas of the environment, t_{sense}

3. The amount of time each agent spent providing communication infrastructure. This can be measured as $t_{complete} - t_{sense}$. Another related metric that could be useful is the amount of time each agent spent navigating towards its parent in order to deliver the information.

4. Number of times base station received new information from the agents throughout the mission.

5. The amount of time spent on replanning. An important factor when comparing various exploration strategies is the amount of time required to decide where to go next.

3.3 Simulation environment

Evaluating new multi-agent approaches in simulation offers a number of advantages over evaluations with physical robots in a real environment. Conducting simulated experiments in simulation is cheap, as it does not require access to expensive hardware. It does not require the time or the space for building large-scale test environments. An appropriate choice of simulation abstraction allows for rapid prototyping of multi-agent exploration strategies, where the researchers do not have to worry about the implementations of low-level components that they are not interested in evaluating. The simulated experiments are highly repeatable – any configuration can be reproduced exactly. A greater amount of data from the simulation can be acquired, stored and analysed. Finally, the simulated experiments can run in faster-than-real-time; many different simulations can be running at the same time, meaning that many more parameters, configurations and scenarios can be explored. However, simulation results may not always translate well to physical experiments – any model or abstraction can only approximate the behaviour of the real multi-agent system. Furthermore, even experiments with real robots using a test arena may not translate well to highly unpredictable real robot deployments.

In this work we used a discrete-time multi-agent exploration simulator MRESim [114], originally introduced in [30] and further developed as part of this DPhil. Here we will give a brief description of this simulator; a more thorough description is available in Appendix A. Environments are modelled as two-dimensional occupancy grids, where each cell represents a fixed area of the environment (typically a square 5cm by 5cm). Each cell can either be free or occupied by an obstacle. Each agent occupies a single cell at any time; collisions between agents are not considered, so multiple agents can occupy the same cell. In each timestep, each agent can re-plan, move a fixed distance determined by their speed, sense the area with a simulated laser range scanner and communicate with other agents within communication range. Perfect

sensing and localization are assumed. While this assumption is unrealistic, it allows us to abstract away from solving the SLAM problem and focus on the higher-level behaviour. The effect that imperfect sensing and localization may have on the approaches proposed in this thesis is discussed in later chapters. Given the assumptions made in section 3.1, we believe MRESim presents the right level of abstraction and the results obtained in simulation should translate reasonably well to the real world.

Chapter 4

Team Coordination Without Explicit Communication Planning

In this chapter we explore the trade-off between using team resources for exploration and providing the communication infrastructure. A novel approach for periodic communication is presented where rendezvous between agents is not explicitly planned. This approach is extensively evaluated in simulation, and is followed by a discussion of the results.

4.1 Motivation

An agent in a team deployed on an exploration mission in an unknown environment can be performing the following roles:

1. It can be sensing new areas of the environment.

2. It can be providing communication infrastructure for other agents in the team, which are sensing areas of the environment beyond the communication range of the base station. For example, it can be acting as a relay in a multi-hop ad-hoc network. In that case, it will need to be positioned in a location where the communication ranges of at least two other agents (or static communication points, such as the base station or a retransmitter deployed in the environment) overlap. It can also be ferrying information from one agent to another, by moving from the communication range of one agent to the communication range of another.

While an agent can be performing both of these roles at the same time, usually it will have to fulfil one role at the expense of another. For example, when ferrying information, the path of the agent will usually be planned through previously mapped territory, and any new sensing along the way will be done opportunistically. Because of that, spending more total team time on providing communication infrastructure will usually mean slower overall exploration time, although it can have a number of benefits. Such expected benefits are as follows:

1. Help prevent repeated coverage by sharing information on explored areas across the team.

2. Facilitating more efficient task allocation among the members of the team.

3. Providing more frequent updates on the progress of the team to the base station.

4. Giving greater control over the overall team and individual agents to human operators over the course of the mission

In section 4.3 we describe a novel approach to providing periodic communication between agents in the team and the base station. The approach does not rely on pre-arranged meetings between two or more agents (specifying the meeting place and time in advance), and is therefore highly robust to individual robot failure, errors in localization of robots or changes in robot plans. In addition, the approach is highly flexible in specifying the team resource allocation between communication and exploration.

4.2 Defining desired team behaviour

The desired allocation of resources between exploration and communication in a robotic team can be specified by a human operator in a number of ways. The most obvious approach would be to specify communication goals directly; an example might be to stipulate a maximum time that agents are allowed to remain out of range of the base station. A similar approach used to allocate resources in a team of unmanned aerial vehicles performing a search mission in a wilderness rescue scenario is described in [130]. However, the human operator may not know in advance what that parameter should be, or how it should increase as the exploration mission progresses further away from the base station. Values too low may lead to inefficient exploration, where agents do not have enough time to explore new frontiers before having to return to the base station again.

Another return condition for the agents could be the amount of new information obtained – for example, every 10 square metres covered. However, the cost of delivering information to base increases as the exploration effort progresses and the team explores new frontiers further away from the base station; it may seem to be beneficial to correspondingly increase the amount of new information (or, more generally, its utility) that should be obtained before

returning with it to the base station.

4.3 Method description

In the proposed approach, an agent can be in one of two states: the *Explore* state, in which case the agent is heading towards a frontier, or the *Return* state, when the agent is heading back to the base station.

At the start of the exploration, all agents start in the *Explore* state. At each timestep, when an agent is in the *Explore* state they will first check if it is time to replan; replan if necessary; and continue on the planned path to their goal. The replan step consists of two stages. First, the agent checks if it needs to change into the *Return* state. The process for that is outlined below. If the conditions for changing the state are not met, the agent remains in the *Explore* state, selects a new frontier cell to navigate towards, and plans a path to it. In this thesis, the frontier cell selection is performed using the "Beyond Frontier Exploration" strategy [129], however our approach does not depend on using any particular planning strategy. When an agent is in the *Return* state, it is heading back into the communication range of the base station in order to deliver the new information that it has obtained. The flowchart of the state transitions is presented in Fig. 4.1.

In our approach, state transitions can happen in two circumstances. They can either happen during the planning phase of the timestep, or during communication with other agents (as well as with the base station).

4.3.1 Representing the environment

In the sections below we assume that occupancy grid maps [38] are used to represent the environment. However, the proposed approach can be adjusted to be used with other map representations. Usually, when occupancy grid maps are used for frontier exploration, it can be assumed that each cell contains two bits of information – whether the cell represents free or unknown space, and

Bit	Information
0	Free space
1	Safe space
2	Obstacle
3	Known at base
4	Relayed by another agent

Table 4.1: Information stored for each cell of an occupancy grid for each agent.

whether the cell has an obstacle in it or not (the actual occupancy grid representation includes probabilistic measures of whether each cell represents an obstacle or free space, however for navigation and planning a binary representation can be obtained by thresholding each cell with an appropriate value). In [129], this representation was expanded to include an additional bit of information – whether the cell represents safe space or not. In our work, we add two additional bits – (i) whether the cell is known by the base station or not, and (ii) if we are responsible for delivering the information about this cell to the base station or it is being relayed to the base station by another agent – a total of 5 bits, meaning that each cell can be represented by a single byte. The role of these two additional bits is explained in subsection 4.3.3.

4.3.2 State transitions during planning

During the replan stage, agents that are in the *Explore* state may change into the *Return* state (see Fig. 4.1). The condition for changing the state of the agent that we propose in this work is the ratio between the amount of information about the environment known by the base station, and the total amount of information collected by the team of agents. For example, if we are simply interested in mapping out the environment, then we could use the ratio of mapped area known at the base station to the area that has been sensed by the exploration team. This target ratio is a parameter that is set for all agents in the team at the start of the exploration, but may be changed by the base station during the mission. When an agent believes this target ratio is exceeded, they change into the *Return* state. As each individual agent does

not always have full knowledge about the location of other agents, how much information they have collected or delivered to the base station, each agent has to estimate these variables. In this thesis, we calculate the current information ratio as follows:

$$Ratio = \frac{BaseKnowledgeBelief}{BaseKnowledgeBelief + NewInfo}$$

where

$$BaseKnowledgeBelief = freeCellsKnownAtBase + freeRelayedCells$$

$$NewInfo = freeCells - freeCellsKnownAtBase - freeRelayedCells$$

The agent then switches into the *Return* state if the following inequality holds:

$$Ratio < TargetRatio$$

In the above equations, $freeCells$ is the total count of known free space cells in the occupancy grid of the agent, $freeCellsKnownAtBase$ is the number of those free cells that are marked as being known by the base station, $freeRelayedCells$ is the number of free cells that are marked as being relayed by another agent and $TargetRatio$ is the target ratio parameter set for the explorations strategy.

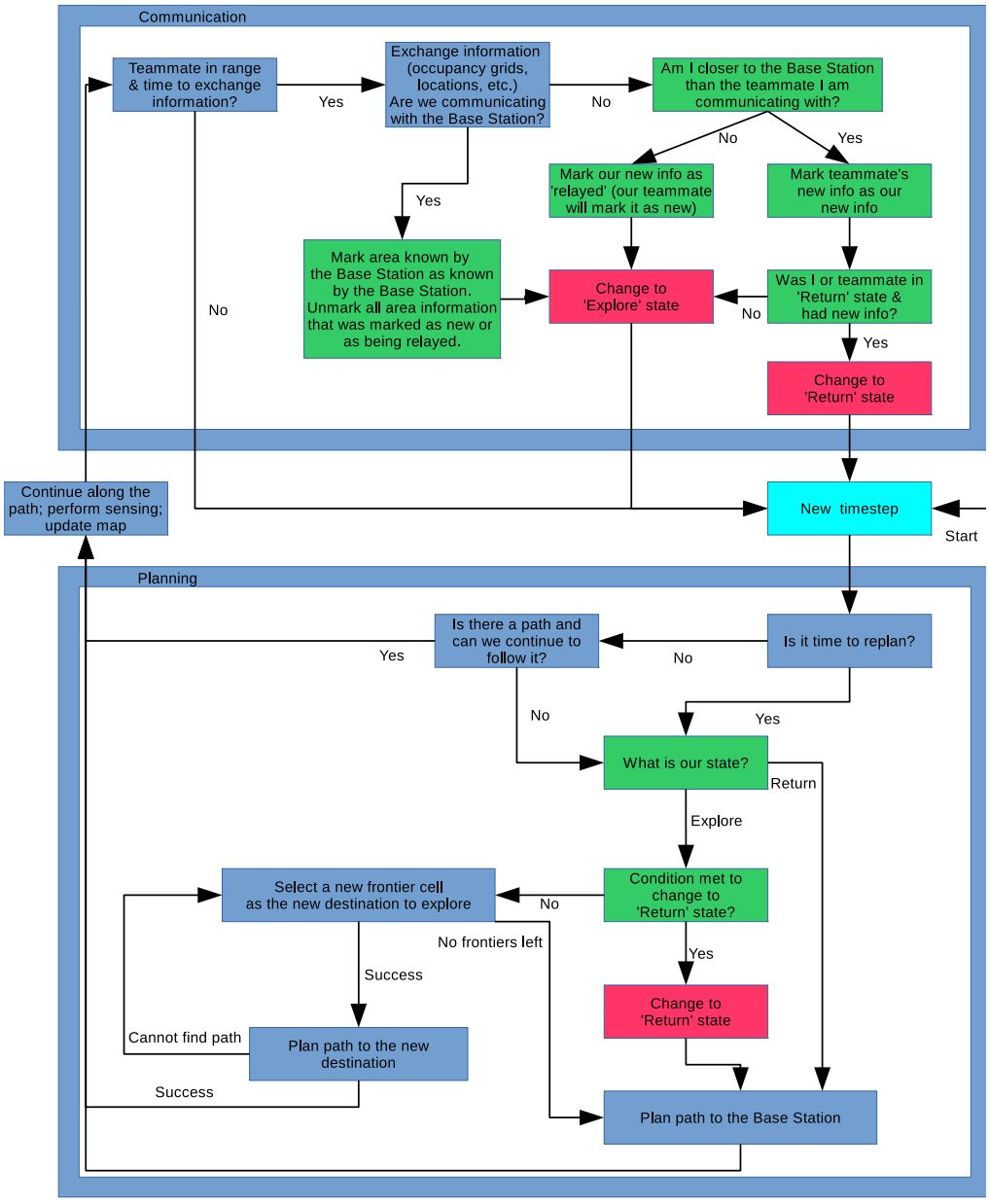


Figure 4.1: Blue squares represent general frontier exploration framework. Green and red squares represent our additions to the framework. Red squares are steps where an agent changes its state.

4.3.3 State transitions during communication

When two agents enter communication range of one another, they exchange their occupancy grid maps, merging the information they receive on the location of free space, obstacles, safe space and cells known at the base station into their own map. Both agents then go into the *Explore* state, so that they can reevaluate the state they need to be in given the new information available to them in the subsequent replanning step. After this exchange of information takes place, both agents will have the same maps, which means that during the next replanning step, both agents may end up switching to the *Return* state. Since they have the same occupancy grid maps, they will both be trying to relay the same information back to the base station, resulting in duplication of effort. We overcome this problem by having only the agent that is closest to the base station at the time of data exchange assume responsibility for new information; the other agent marks those cells as being relayed by someone else, and will not count them as new information when considering changing to the *Return* state in the subsequent replanning stages. For the description of the process for merging the occupancy grid maps and marking cells as being relayed, see subsection 4.3.6.

4.3.4 Preventing oscillations

The state transition above depends on which of the two communicating agents is located nearest to the base station. Suppose that agent A is in communication range with agent B, and agent A is located nearest to the base station. Agent A is in the *Explore* state, while agent B is in the *Return* state. Then after communicating with each other, both agents will change states: agent A, being the one nearest to the base station, will proceed to base, and agent B will head out to the next frontier. It is possible, however, that in the following timestep, agent B will end up nearest to the base station, while A and B are still in communication range, which will lead to the two agents changing state again

Input: Agent deciding on state transition A
Data communicated from the teammate B
Distance threshold $threshold$

```
1 if (|A.DistToBase() − B.DistToBase()| > threshold) or
  (A.HasNewInfo() and B.HasNewInfo()) then
2 |   A.MergeCells(B.OccupancyGrid());
3 |   if A.DistToBase() < B.DistToBase() then
4 |   |   A.SetState(Return);
5 |   end
6 |   else
7 |   |   A.SetState(Explore);
8 |   end
9 end
```

Algorithm 4.1: Algorithm for preventing oscillations when deciding whether to change state when communicating with a teammate.

and may cause them to pass each other, returning to the original situation and leading to oscillating behaviour. This could happen due to errors in path planning, localisation or mapping, or if the agents happen to move past each other while the communication is actually taking place.

In order to prevent such oscillations from happening, we impose the following additional condition on performing state changes and relaying of data during communication between two agents A and B: the difference in distances to base between agents A and B needs to be greater than a specified threshold, where that threshold is set to at least twice the distance that each agent can travel in one timestep. Note, however, that this rule should only apply if only one agent is already responsible for relaying all the new data between agents A and B – otherwise, it may lead to a situation where A and B, if the distance between them is less than the threshold, never allocate a relay among themselves, and both head back to the base station to try to deliver the new information. To avoid that, we only invoke the above rule if at least one of the agents has all their new data already marked as being relayed by another agent. The process is described more precisely in Algorithm 4.1.

4.3.5 Reducing redundant space coverage

The state transition functions defined above, together with the fact that agents have limited information about the true state of the world at the time of their state transitions, may mean that an agent that had previously changed into the *Return* state may need to reevaluate that decision after receiving additional information from communicating with another agent. As a result, the agent will switch back into the *Explore* state, having to backtrack to the frontier it had been exploring before (or to a new, nearby frontier), resulting in wasted effort in previously attempting to navigate back to the base station. There are several ways of attempting to reduce such wasted effort. Each agent can act conservatively: that is, be optimistic about the amount of information known by the base station and pessimistic about the new area sensed by their teammates. This, however, may result in the agent deciding to switch to the *Return* state too rarely. Instead, the approach we took in this thesis is to ensure that once an agent changes into the *Return* state, information is to be delivered to the base station even if the agent later finds out that it was not yet necessary to switch to the *Return* state. More specifically, when two agents communicate with each other, and one agent was in *Return* state and had non-zero amount of information that it was responsible for delivering to the base station, the agent that assumes responsibility for that information after the communication (typically, the agent nearest to the base station) will change into the *Return* state, with the other agent switching to *Explore* state (as opposed to both agent switching to the *Explore* state and then deciding whether they need to change to the *Return* state during the replanning stage). This is reflected in the state transition to the *Return* state in the communication stage shown in Fig. 4.1.

4.3.6 Merging occupancy grids

When two agents enter each other's communication range, they exchange their occupancy grid maps. The maps exchanged are in a state *before* communication and map merging has taken place. Once the maps have been exchanged, each agent attempts to merge the map received into their own. After the communication and map merging by both agents is complete, both agents should end up with mutually consistent merged maps (e.g. the first 4 bits of each cell should be the same in both maps, and only at most one agent must be responsible for relaying the new information back to base). The merging of the first 3 bits for each cell is a straightforward process if one agent has already explored that cell but for another that cell is still unexplored. In that case, the first 3 bits of the unexplored cell in the agent's map are replaced with the corresponding bits of the explored cell from the received map.

However, it is possible that both agents have already explored a cell and hold conflicting information about it in the first 3 bits. This can occur due to sensor error, localization error or dynamic obstacles being sensed, such as another robot. In this work, we resolve the conflicts when merging maps as follows:

1. **Safe space always has priority over unsafe space**. For example, if a cell is marked as an obstacle in safe space for one agent, but as free unsafe space for another agent, the merged map will have the cell marked as an obstacle in safe space. The reason for this is that measurements in safe space (which is a shorter range of the laser scanner than its full range, typically 10% of the scanner's full range [129]) are likely to be more accurate than at longer ranges.

2. **Free space has priority over obstacles.** That is, if both (or neither) agents have the cell marked as safe, but one has it marked as an obstacle, the resulting map will have the cell marked as free space. This increases the likelihood that robots do not get trapped in narrow spaces, unable

to plan a path out, due to erroneous "obstacles" on the map blocking their way out. A robot planning a path through an actual obstacle that it thinks is free space is less of a problem, as when the robot gets closer to the obstacle it will usually be able to sense it and stop before a collision takes place, and replan the path accordingly. In addition, local planners may be used to manoeuvre around some obstacles if their planned path ends up lying through them.

In order to merge information stored in the 4th bit (which denotes if the cell is known by the base station), a simple disjunction of the corresponding bits in the cells of each map is performed. If the agent is communicating with the base station, the agent sets that bit to 1.

Finally, we need to merge the 5th bit, which denotes if we are responsible for relaying the information about this cell to the base station, or if this cell is being relayed by someone else. Information in this bit is only necessary if this cell is not yet known at the base station, that is if the 4th bit is 0. This bit of information needs to be merged in such a way that, until the information about the cell is delivered to the base station, exactly one agent is responsible for relaying the information. We accomplish this as follows:

1. If the agent did not have any information about the cell before the merging took place, it sets the 'relayed' flag to be the same as it is set in the teammate's occupancy grid.

2. If at least one agent has the cell marked as 'new' (not marked as 'relayed'), then the agent marks the cell as 'new' if the agent is closer to the base station; otherwise, it marks the cell as 'relayed'. This ensures that only one agent is responsible for delivering the information about the cell to the base station after communication has taken place.

Note that it is still possible for several agents to be trying to deliver the same information back to the base station. An example of that would be two agents

that have independently explored the same area of the environment. Then, as long as two agents relaying this data – originating from different explorers – do not communicate with each other, there will be more than one agent relaying the same information to the base station.

4.4 Simulation experiments

The goal of this section is to evaluate the behaviour of the proposed strategy in different types of scenarios. Experiments are conducted in simulation, with the goal of exploring 95% of the free space of the environment.

4.4.1 Approaches under consideration

In this section we compare the performance of the following exploration strategies:

- Greedy Exploration – with this strategy, agents allocate frontiers among themselves whenever possible, explore the environment and only attempt to return to base after the whole environment has been explored. Agents communicate with each other and with the base station if they happen to enter the communication range, but otherwise they do not deliberately make an effort to communicate with other agents.

- Role-Based Exploration (RB) [30] – agents are divided into two equal groups, explorers and relays. Each explorer gets assigned one designated relay. It is true that in certain types of environments, it would make sense to have different hierarchy of agents (for example, in a corridor environment, it would make sense to have a long chain of relays serving one explorer; but in an environment with 4 separate areas to be explored, it would make sense to have at least 4 explorers, with the remaining agents equally distributed as relays to those explorers). However, this thesis deals primarily with the situations where there is little prior information

about the environment when the robotic team is sent in; as such, the goal is to use the same configuration for all of the tested environments in order to evaluate the adaptability of each approach. Role-based exploration with a branch depth of 2 (where half of agents are assigned the role of explorer, and each explorer gets assigned one relay) has been shown to adapt to various types of environments [31] – in fact, it was suggested that more complex configuration may actually lead to worse performance due to the increased amount of planning required. Therefore, we chose that configuration in order to evaluate the adaptability of our proposed opportunistic approach against an approach with pre-configured roles and pre-planned rendezvous locations. In the version of role-based exploration used in these simulated experiments, explorers allocate frontiers among themselves and return to pre-agreed rendezvous locations with their relays at pre-agreed times. Relays move between the base station and the pre-agreed rendezvous locations. The meeting times are determined by the time it takes for the relay to navigate to the base station and then get to the rendezvous location. The subsequent rendezvous location is selected as the point at which the explorer had to turn back to go to the previous rendezvous location to meet its relay.

- Our approach (opportunistic relaying) with return ratios between 0.2 and 0.9 (R0.2, R0.3, ..., R0.9) – the approach described in this chapter. When an agent believes that the amount of information known by the base station represents less than the defined ratio of the total information known by the agent, the agent attempts to return to the base station. If they meet another agent, the agent closest to base proceeds to the base station, while the other agent continues the exploration.

- Our approach (opportunistic relaying) returning after each 10% of the environment has been explored (R10%) – similar to above, except this time we explicitly tell agents the total amount of information they are to

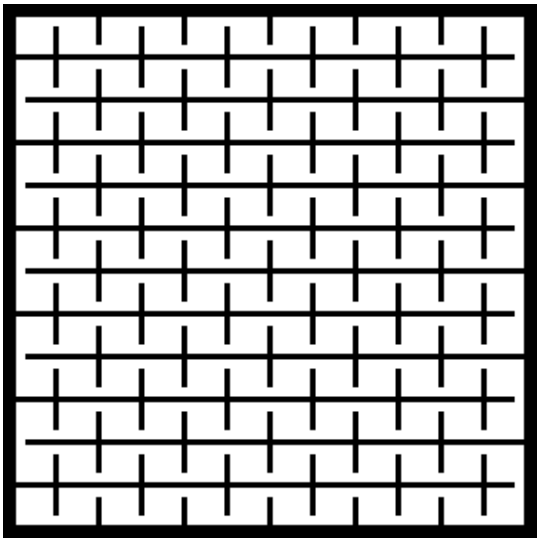


Figure 4.2: Corridor environment, 12 square cells wide and 12 square cells tall for a total of 144 cells. Base station is in the top-left cell. Communication is limited to line-of-sight.

discover in the environment, and ask them to attempt to relay the information after they have sensed each subsequent 10% of the environment. This is likely to be unrealistic in practice, as we would need to have a priori knowledge of the size of the accessible environment to be mapped, but is included here for comparison of returning after a constant size of area is sensed.

4.4.2 Corridor environment

The first environment we use in our evaluation is a simple corridor environment, shown in Fig. 4.2. It consists of 144 equally sized square "rooms", and is 12 "rooms" wide and 12 "rooms" tall. Each square "room" is a 20 by 20 occupancy grid. The obstacles are positioned in such a way, that:

1. There are no junctions or dead-ends, the environment can be navigated from start to finish without backtracking and represents one long corridor.

2. Agents can only sense one square "room" at a time and they need to be inside the "room" to fully sense it.

3. Agents can only maintain line-of-sight communication if they share the same square "room", or sometimes when they are in adjacent "rooms".

We only allow line-of-sight communication in these simulations. Our proposed strategy relies on chance encounters between agents to establish relay chains; reducing the communication range of each agent to their immediate vicinity allows us to reduce the effect of chance and to study the performance of the strategies under pessimistic assumptions. In all of the simulations in this subsection, the base station is located in the top-left square, which is also where all of the agents start the exploration mission.

First, we consider a mission with only two agents. A simple setup like this should allow us to see the basic properties of each examined strategy.

4.4.2.1 Greedy Exploration

Greedy exploration has a very simple behaviour in this environment, which is independent of the number of agents in the team. At any time, there is only one available frontier, and agents do not return back to base until they have explored all of the environment. Therefore, agents simply navigate along the corridor to the end, and then turn back. The consequence of this is that increasing the number of agents does not lead to a significant increase in performance, agents simply end up traversing the corridor side-by-side. An interesting observation is that when running the simulation with one agent, it takes the robot 608 time steps to sense every cell in the corridor, but 398 time steps to return back – agents move slightly slower while sensing unknown territory due to both less efficient path planning, and the need to fully sense each part of the corridor before moving forward. The speed of agents while sensing is therefore $398/608 \approx 0.65461$ times the speed of agents through known territory. While these exact figures are specific to the simulation used, this effect is consistent with what we would expect to happen in the physical world, assuming the world is static. The total time to explore the environment with one robot and return to the base station is 1023 timesteps (the agent spends a few time steps

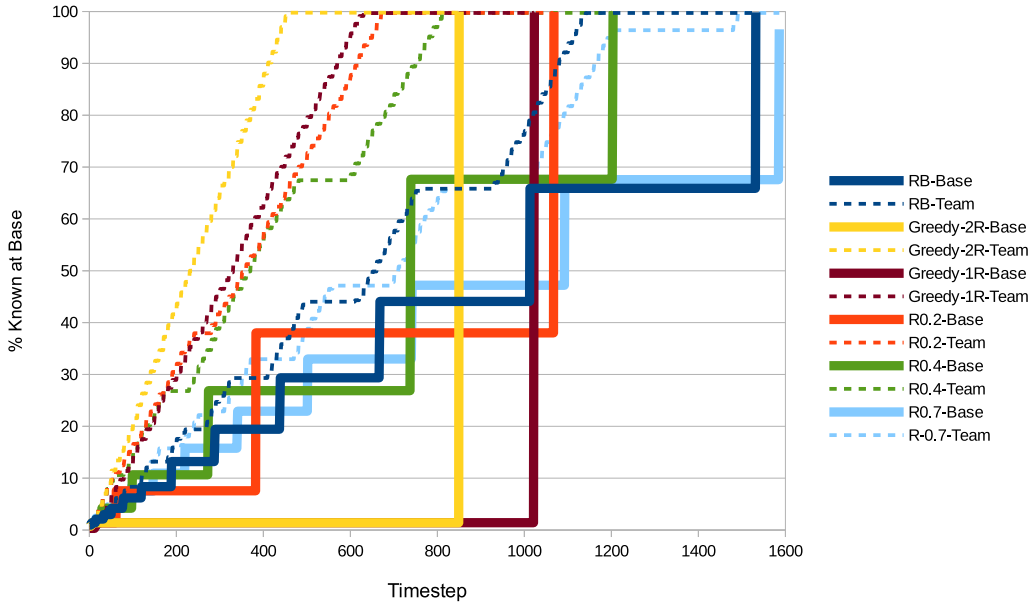


Figure 4.3: Percentage of the environment known by the overall team (dashed lines) and the base station (solid lines) for missions with 2 agents exploring the corridor environment.

at the start of the exploration moving randomly to accumulate some initial data). There is no relaying in this configuration, so the base station only has information about less than 5% of the environment until after 1023 timesteps have passed from the start of the mission (see dark red line in Fig. 4.3). After the agents leave the communication range of the base station at the start of the mission, they are not able to receive commands from the base station until they return back at the end of the mission, so the base station has no information about the progress of the mission until the very end.

4.4.2.2 Planned rendezvous

Role-based exploration with two robots on this map gives us an interesting insight into the fundamental behaviour of role-based exploration. Let us consider a discrete corridor environment, similar to the one we are considering in this subsection, but where:

1. Agents can communicate only when they are sharing the same discrete

cell in the environment.

2. Agents start in cell 1, where the base station is located.

3. Any number of agents can be in the same cell at the same time.

4. It takes 1 time step for any agent to move from their current cell to an adjacent cell.

5. Agents agree to rendezvous in the farthest explored cell, that is the cell where the explorer had to turn back to meet its relay. This is the farthest cell from the base station at which rendezvous can be arranged, if it is to be arranged in explored space.

Then the behaviour of the robots looks like what is shown in Fig. 4.4. The rendezvous locations are selected in cells 1, 2, 3, 5, 8, 13, 21, etc., following the Fibonacci sequence. Because the following rendezvous location is selected to be the furthest explored cell, the ratio of the number of cells known by the base station to the number of cells known by the explorer at the moment when the explorer turns back to meet the relay is going to be the ratio of two consecutive Fibonacci sequence numbers (e.g. 1/2 in step 2, 2/3 in step 5, 3/5 in step 9, 5/8 in step 16 etc.). This ratio is calculated in the same way as we calculate the condition for agents to turn into the *Return* state in our approach. The limit of this ratio is the inverse of the golden ratio, which is $= \frac{\sqrt{5}-1}{2} \approx 0.618$. Note that in role-based exploration, the next rendezvous location has to be agreed upon during the previous rendezvous. As such, the new rendezvous has to be located in an area that has already been explored by that point. This means that if the following conditions are true:

- The relay has the same speed or faster than that of the explorer

- Rendezvous is timed such that the explorer and the relay arrive at the rendezvous at the same time

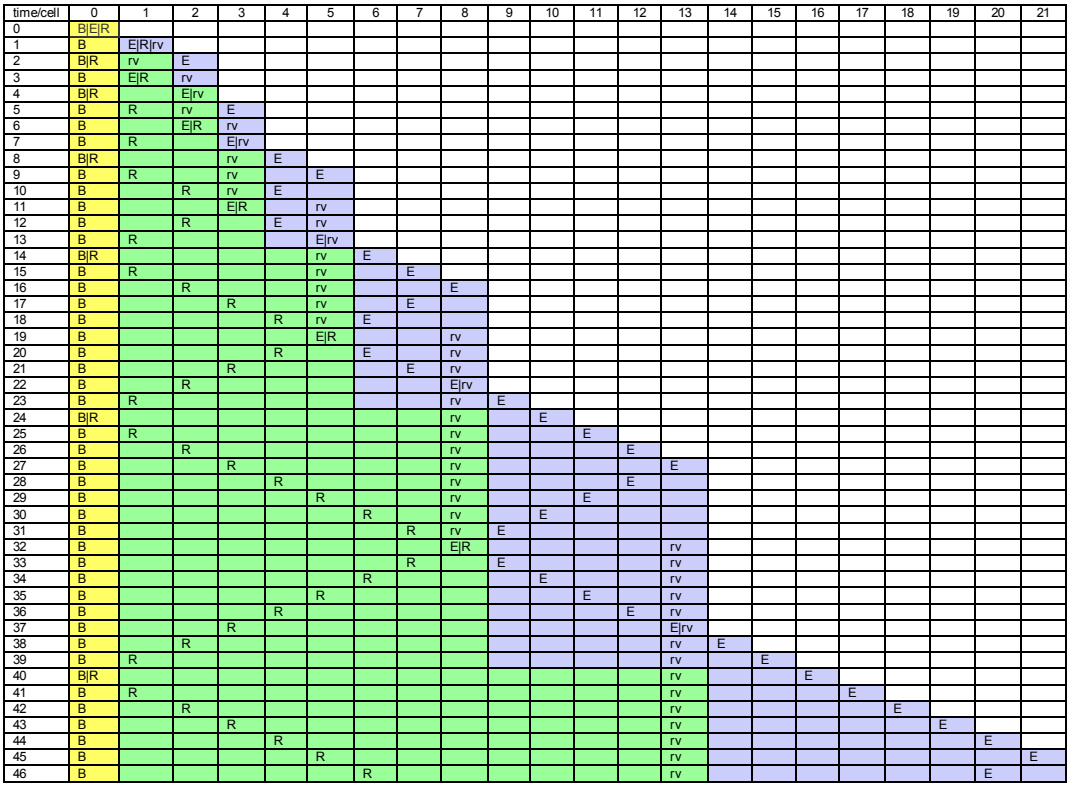


Figure 4.4: An explorer and a relay exploring a corridor of length 21 using role-based exploration over 46 timesteps. Cell containing the Base Station is indicated with the letter B, E represents the location of the Explorer and R represents the location of the relay. The current rendezvous location is indicated with letters *rv*. Agents can only communicate when they share the same cell. The starting cell is shown in yellow, green cells are known by the base station and blue cells have been mapped by the explorer.

- Two agents have to be near each other to communicate

then the return ratio is not going to be smaller than ≈ 0.618, as that would require for the rendezvous to happen in space that is yet unexplored at the time of the previous rendezvous. Larger return ratios are possible by selecting a rendezvous location deeper in explored territory (or if the relay has a short-cut to the rendezvous location, i.e. the environment is not corridor-like). Alternatively, if agents have a relatively large communication range and take that into account when planning, the effective return ratio can also be increased.

We previously established that in our simulation exploration speed is $\tau = 0.65461$ times that of the relaying speed. That is, $Vel_{exp} = \tau Vel_{rel}$, where

Vel_{exp} is the velocity of a robot moving through previously unexplored terrain, and Vel_{rel} is the velocity of the robot when moving through known terrain. What then can be said about when the explorer has to return to rendezvous with the relay in role-based exploration, given the difference in exploration speed and relaying speed? Let RV_1 and RV_2 be the cell numbers of where two subsequent rendezvous take place, and RV_3 be the point at which the explorer turns back to head back to RV_2 (and, correspondingly, the location of the following rendezvous). Both the explorer and the relay start at RV_1, at which point the relay heads back to the base station in cell 0 and then proceeds to RV_2. Therefore, the explorer has time $TimeAvail = \frac{RV_1+RV_2}{Vel_{rel}}$ to explore new cells and return to RV_2. That is, $\frac{(RV_2-RV_1)}{Vel_{rel}} + \frac{RV_3-RV_2}{Vel_{exp}} + \frac{(RV_3-RV_2)}{Vel_{rel}} = TimeAvail$. Hence, $RV_1+RV_2 = (RV_2-RV_1)+\frac{RV_3-RV_2}{\tau}+(RV_3-RV_2)$, and so $RV_3 = \frac{2\tau}{\tau+1}RV_1+RV_2$. Let $\beta = \frac{2\tau}{\tau+1}$. Then,

$$RV_{n+2} = \beta RV_n + RV_{n+1} \tag{4.1}$$

We are interested in finding the ratio $\alpha = \lim_{n\to\infty} \frac{RV_n}{RV_{n+1}}$, which is what the ratio of the area known by the base station to area known by the explorer, at the time when the explorer turns back to head to rendezvous, converges to. As $n \to \infty$, by definition of α:

$$RV_{n+1} = \alpha RV_{n+2} \tag{4.2}$$

and

$$RV_n = \alpha^2 RV_{n+2} \tag{4.3}$$

Then, substituting (4.2) and (4.3) into (4.1), we get:

$$RV_{n+2} = \beta\alpha^2 RV_{n+2} + \alpha RV_{n+2} \tag{4.4}$$

As $RV_{n+2} \neq 0$, we can simplify (4.4) to get the following quadratic equation:

Timestep at which explorer headed back to RV	**Ratio of base station known occupancy grid cells to explorer's**
11	67.279%
26	70.449%
39	71.200%
59	67.677%
90	73.969%
138	63.303%
216	67.921%
326	66.243%
498	66.579%
755	66.918%
1136	66.006%

Table 4.2: Ratios of the number of occupancy grid cells known by the base station to total number of cells explored by the explorer at the timesteps when the explorer turned back to head to the meeting point with its relay using Role-Based Exploration on the corridor environment with 2 agents: 1 explorer and 1 relay.

$$\beta\alpha^2 + \alpha - 1 = 0 \tag{4.5}$$

Solving (4.5) for $\alpha > 0$ we get:

$$\alpha = \frac{\sqrt{4\beta + 1} - 1}{2\beta} \tag{4.6}$$

Therefore, if exploration speed is $\tau = 0.65461$ times that of the relaying speed, α converges to ≈ 0.658. The relationship between α and τ is shown in Fig. 4.5.

In simulation, the behaviour of the agents ends up being similar to what we predicted above. The information ratios at the time when the explorer turns back to return to the relay are shown in Table 4.2. This result implies that in the simple corridor environment, we should be able to obtain the same effective behaviour as Role-Based Exploration using our opportunistic rendezvous approach with the return ratio of approximately 0.658.

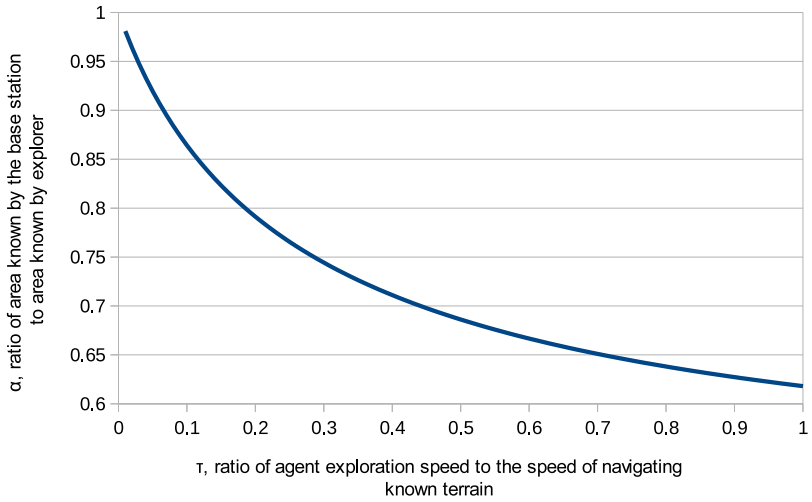


Figure 4.5: Graph showing the ratio α of area known to the base station to area known by the explorer, at which explorer should head to rendezvous in order to meet the relay at the boundary between area that is known and unknown to the base station. The relationship between α and the ratio of explorer speed while sensing new areas to its speed when navigating known territory is displayed. Exploration of a corridor environment with one explorer and one relay is assumed.

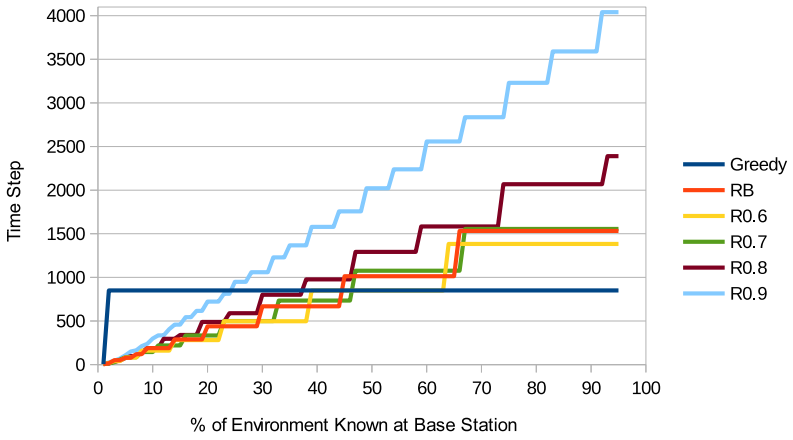
4.4.2.3 Opportunistic rendezvous

It may appear that due to the exploration speed being significantly slower than relaying speed (which is also a sensible assumption to make in a real-world rescue scenario - sensing a room should take longer than simply navigating through it along a pre-planned path), it would be more efficient to have an explorer rendezvous with the relay in territory *already known* to the relay - this way, the relay would always be traversing pre-explored space and therefore its navigation speed will be maximized. However, the simulation results contradict that assumption. The most consistently good results are achieved with our approach with return ratios between 0.2 and 0.7. Performance gets a lot worse at return ratios higher than 0.7. The reason for this is that relaying information too often leads to the explorer spending a lot of time in already explored territory, and little time sensing new areas, resulting in slow progress. Such behaviour could still be desirable if a mission is not time-critical and the base station

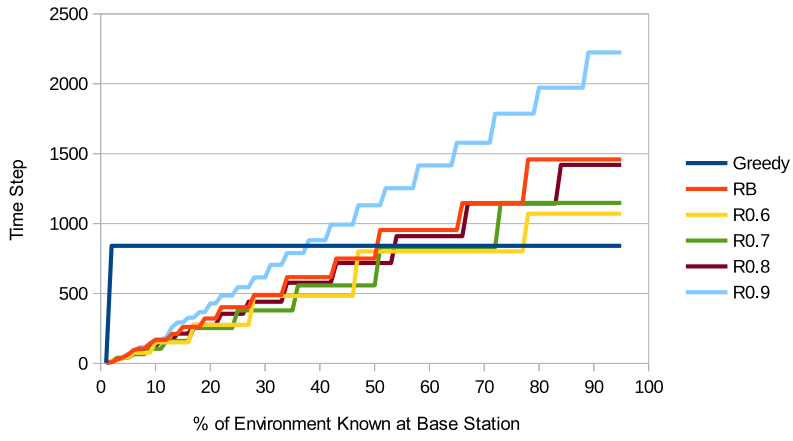
requires close control over the actions of the exploring robots. However, better alternatives could include setting up a communication infrastructure to enable teleoperation of the robot, or requiring the agent to communicate with the base station for new instructions after a specific area or amount of area has been sensed – the precise return conditions for each exploration leg could be given to the explorer at subsequent rendezvous. Note that unlike exploration strategies where the rendezvous time and location have to be explicitly agreed during the previous rendezvous, with our approach the return conditions for the explorer can be more flexible, such as "return after fully exploring this corridor".

Fig. 4.6 shows the number of timesteps after which the base station has information about at least $N\%$ of the environment for each N. With 2 agents, the greedy strategy performs reasonably well compared to other approaches – it takes only 850 timesteps to obtain and deliver to base the full map of the environment. Other approaches have less than half of the map known at base at that point. The downside is that in the first 850 timesteps human responders get no information from the exploration team whatsoever. Both R0.9 and R0.8 appear to perform slower than approaches with return ratio of 0.7 and smaller at nearly every stage of the exploration mission. Role-Based exploration behaves very similarly to R0.7 and R0.6, as expected. Approaches with return ratios between 0.2 and 0.5 behave similarly to each other, as can be seen in Fig. 4.7. As expected, as we increase the return ratio from 0.2 to 0.9, the overall mission time increases. With linear increases in the return ratio, the overall mission time appears to increase exponentially.

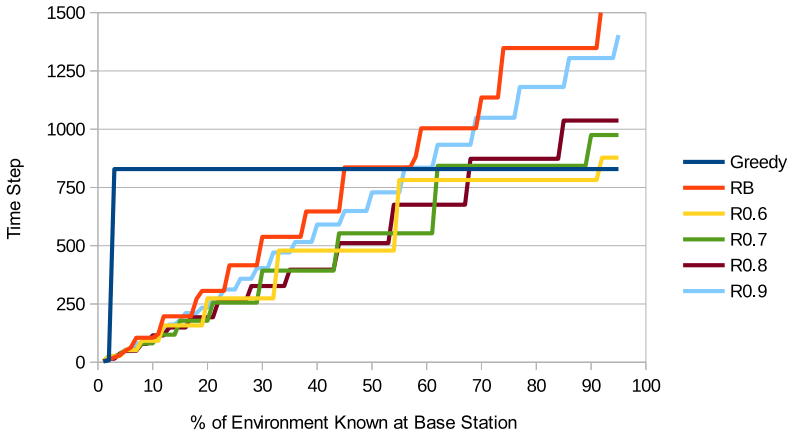
As expected, using a lower return ratio performs better in the later stages of exploring the environment, while using a higher return ratio yields better performance in the early stages (that is, the base station gets information about *some* of the environment faster, at the cost of delaying the completion of the overall mission). To help assess which method yields better overall performance over the whole time of the mission, Table 4.3 presents a comparison of exploration strategies evaluated on the corridor environment for teams of 2, 4



(a) 2 agents.

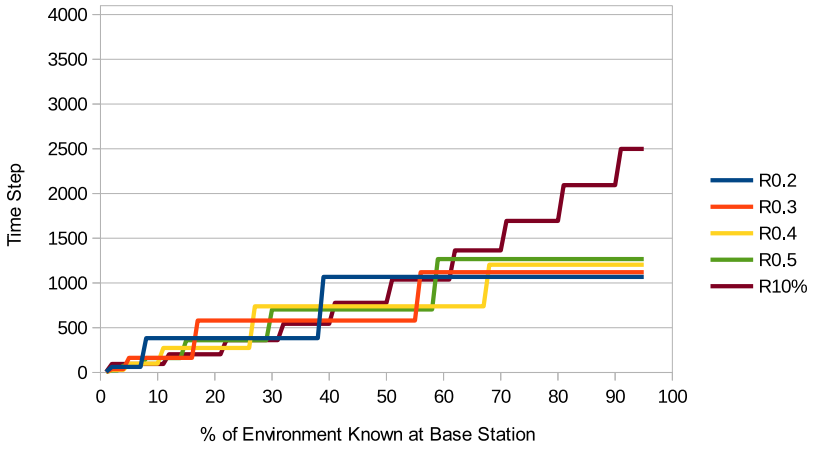


(b) 4 agents.

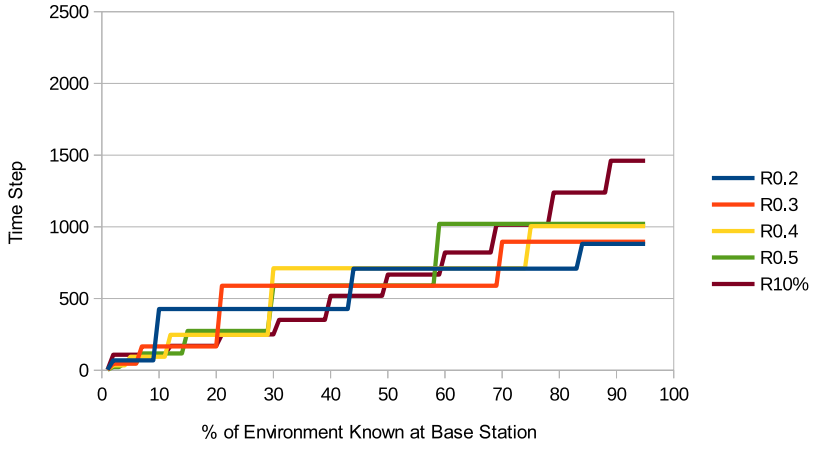


(c) 8 agents.

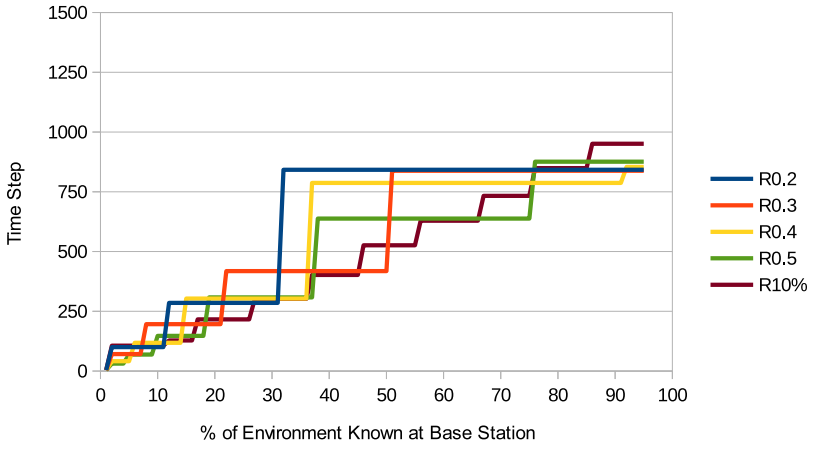
Figure 4.6: Graphs showing the number of timesteps after which information about each percentage of the environment becomes available at the base station. 2, 4 and 8 agents are exploring the corridor environment using Greedy, Role-Based, as well as our approach with return ratios of 0.6, 0.7, 0.8 and 0.9. The graphs for return ratios between 0.2 and 0.5 are available in Fig. 4.7.



(a) 2 agents.



(b) 4 agents.



(c) 8 agents.

Figure 4.7: Graphs showing the number of timesteps after which information about each percentage of the environment becomes available at the base station. 2, 4 and 8 agents are exploring the corridor environment using our utility return approach with return ratios of 0.2, 0.3, 0.4 and 0.5, as well as with a policy of returning after each 10% of the environment has been explored.

and 8 agents. For each increment of 1% of the total area of the map, we consider the timestep for each strategy at which at least that amount of information had become available at the base station. Let the best time of getting n% of the map known at base be t_n^{best}. Then for each exploration strategy s and each percentage of the total map n% we can calculate $\frac{t_n^s}{t_n^{best}}$, which represents how many times slower strategy s is at getting n% of the environment known at base compared to the strategy that is the fastest at getting n% of the map known at base. The maximum such value over n between 3% and 95% for each strategy s is presented in the column labelled "Max (ratio)". The average such value over the values n considered is listed in the "Avg (ratio)" column. Likewise, the "Max (timestep)" and "Avg (timestep)" present the maximum and average values for $t_n^s - t_n^{best}$, which takes into account the absolute number of timesteps by which each strategy is worse than the best one considered for each n. The best value in each column is highlighted in green. As more agents are available in the team, using a higher return ratio appears to yield better overall results. Using a return ratio between 0.6 and 0.7 seems to give good performance for team sizes between 2 and 8 agents.

As the number of agents in the team is increased, greedy exploration on the corridor environment is unable to make good use of the additional resources. At any given time, there is usually only one frontier to explore, meaning that all robots stay close together leading to little change in the overall team efficiency. An interesting result here is that as the number of agents is increased from 2 to 4 and 8, the speed of the role-based strategy to deliver each percentage of the environment to the base station does not seem to improve significantly either. We can see this in Fig. 4.6. Our approach adapts to the availability of additional resources well – with 8 agents, even R0.9 ends up performing better than Role-Based exploration; R0.6 outperforms greedy exploration for most of the mission, getting information about the first 91% of the environment to base faster than the greedy approach. Using return ratios between 0.2 and 0.4 leads to little improvement as the number of agents in the team is increased, with

	2 agents			
	Max (ratio)	**Avg (ratio)**	**Max (timesteps)**	**Avg (timesteps)**
Greedy	65.38	3.89	837	311
RB	2.21	1.65	793	364
R0.2	4.76	1.63	524	239
R0.3	2.85	1.51	416	200
R0.4	2.88	1.41	379	196
R0.5	2.62	1.48	528	250
R0.6	1.87	1.5	644	283
R0.7	2.31	1.65	816	377
R0.8	2.81	2.07	1540	633
R0.9	4.75	3.33	3190	1384
R10%	7.31	1.74	1649	454
	4 agents			
	Max (ratio)	**Avg (ratio)**	**Max (timestep)**	**Avg (timestep)**
Greedy	93.44	4.67	832	373
RB	2.06	1.68	751	328
R0.2	7.66	1.56	333	106
R0.3	5.11	1.43	342	114
R0.4	3.77	1.43	461	153
R0.5	2.44	1.40	432	172
R0.6	2.22	1.33	363	147
R0.7	2	1.36	440	178
R0.8	1.94	1.54	579	270
R0.9	2.78	2.25	1383	645
R10%	12	1.49	620	174
	8 agents			
	Max (ratio)	**Avg (ratio)**	**Max (timestep)**	**Avg (timestep)**
Greedy	55.26	4.05	814	375
RB	2.14	1.71	718	329
R0.2	12.5	1.80	568	199
R0.3	8.875	1.51	359	119
R0.4	5.125	1.47	479	136
R0.5	3.875	1.27	245	80
R0.6	3.25	1.21	256	70
R0.7	3.25	1.21	214	79
R0.8	1.87	1.20	255	93
R0.9	1.74	1.5	575	240
R10%	13.25	1.39	169	58

Table 4.3: Comparison of exploration strategies evaluated on the corridor environment for teams of 2, 4 and 8 agents. The best value in each column is highlighted in green. Detailed explanation of the values in this table is given in Section 4.4.2.3.

team behaviour being similar to the greedy approach.

Let us examine the behaviour of Role-Based exploration as the number of agents in the team is increased. The result that the performance of this approach does not appear to improve significantly when more resources are available seems counter-intuitive. Let us look at how the frequency of base station updates is affected by the increase in the number of agents for each strategy. That is, how many times does the base station get communicated new information throughout the exploration mission. The figures for a number of different strategies are presented in Table 4.4. We can see that for all strategies apart from role-based exploration, the number of times that information is returned to the base station stays relatively constant, or decreases, as the number of agents is increased. That means that the extra available resources help reduce the time it takes for the exploring agent to backtrack to meet the relays to allow the explorer to spend more time exploring. If agents had larger communication range, it would also allow for the information to be delivered faster to base, but since in our simulation agents need to be very close to each other to communicate, the actual speed of delivery is not improved by introducing more agents. The number of times that the base station gets updated increases quite a lot for role-based exploration as we increase the number of agents, unlike with other approaches. This suggests that the extra resources contribute more to increasing the frequency of base station updates, delivering information in smaller bits. The addition of the additional agents ensures that this does not negatively affect the amount of time it takes to explore and deliver each part of the environment to the base station. Fig. 4.8 shows what percentage of the environment is known to the team overall and to the base station over time for role-based exploration on the corridor environment with 2, 4 and 8 agents. Note that all of the graphs have the same slope, and as the number of agents is increased, the only difference is the increase in the frequency of base station updates. In contrast, Fig. 4.9 shows what happens when the number of agents is increased for our strategy with return ratio of

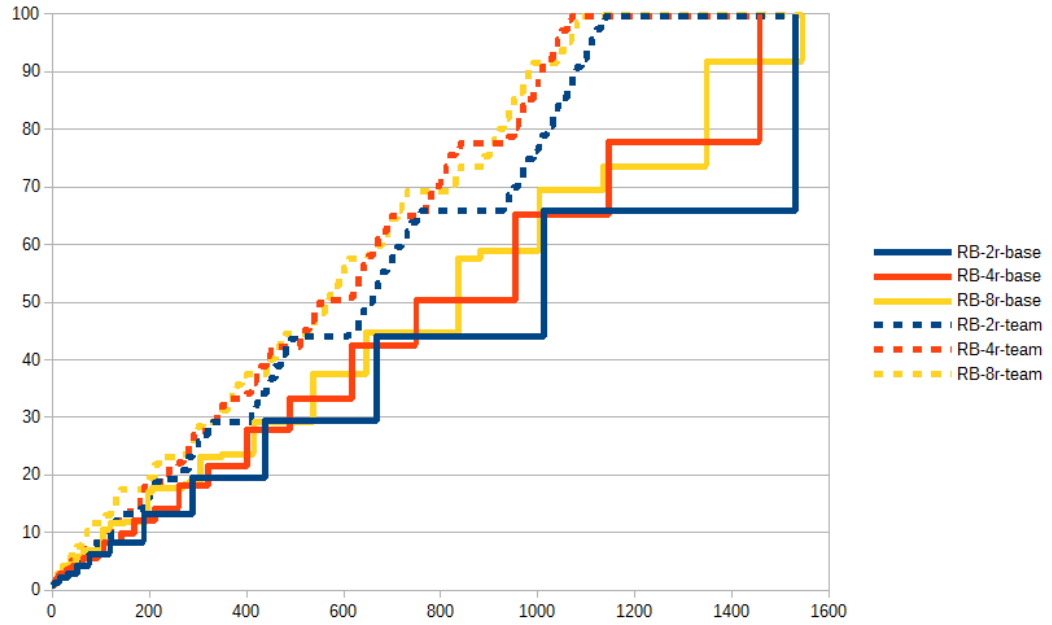


Figure 4.8: Graph showing percentage of the environment known to the team overall and to the base station over time for role-based exploration on the corridor environment with 2, 4 and 8 agents.

0.7 – the slopes of the graphs change, ensuring that the mission is completed in less time.

Fig. 4.10 shows how our strategy behaves in the corridor environment with 2 and 4 agents. On the left, at **timestep 372** we can see that agent B has just hit the return ratio of 0.7 and is returning to the base station in the top left corner. Agent A has recently communicated with the base station and is heading to explore a new frontier (and therefore, moving towards B). At **timestep 417**, A and B have just moved into communication range; A turns back to relay the new information (shown in green) to the base station. B marks the green area as being relayed by another agent (and assumes that the base station will soon have information about that area) and heads back to explore new frontiers. At **timestep 499**, A has just delivered the new information to the base station. Finally, at **timestep 551** B hits the return ratio again (while correctly assuming that the green area is known by the base station) and turns back. The situation is now similar to that at **timestep 372**, and

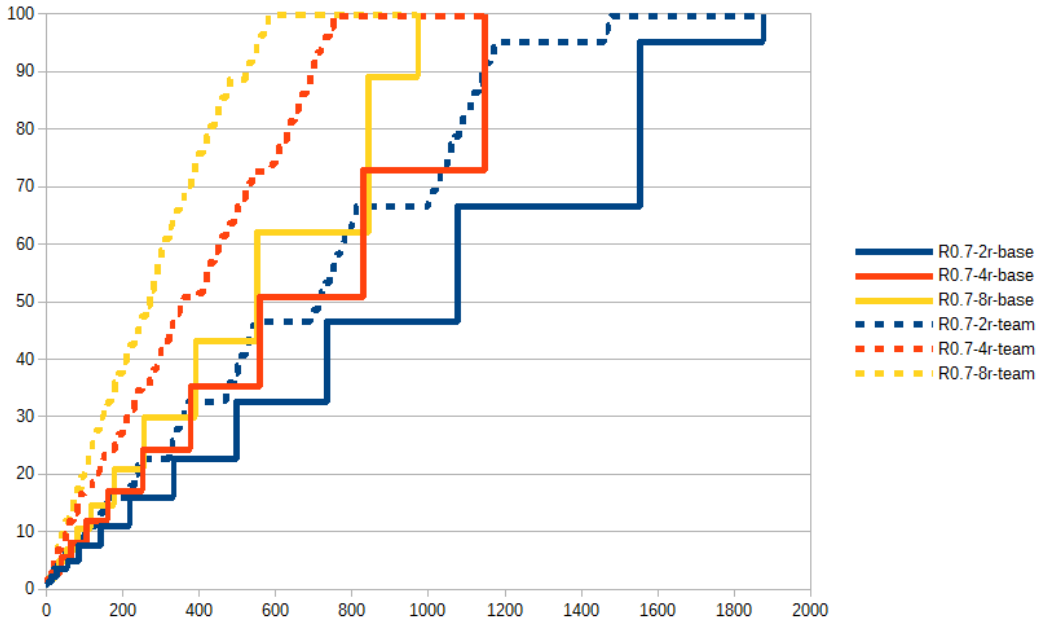
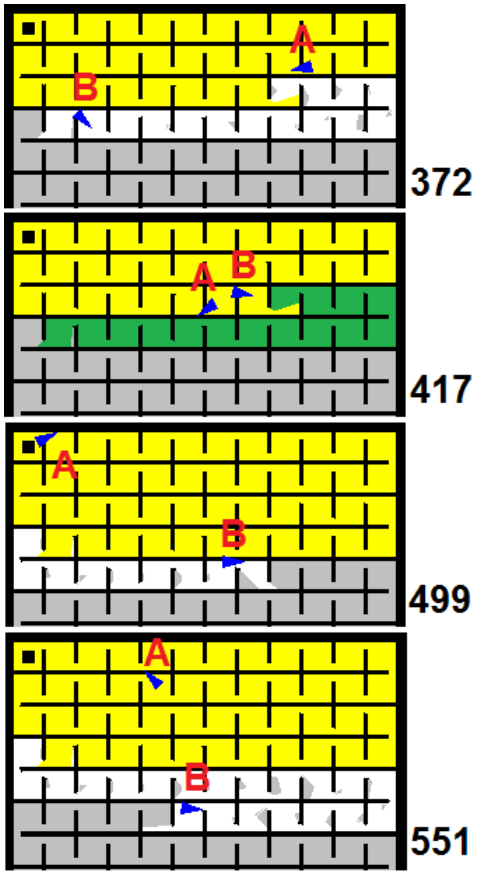


Figure 4.9: Graph showing percentage of the environment known to the team overall and to the base station over time for R0.7 strategy on the corridor environment with 2 (blue), 4 (red) and 8 (yellow) agents.

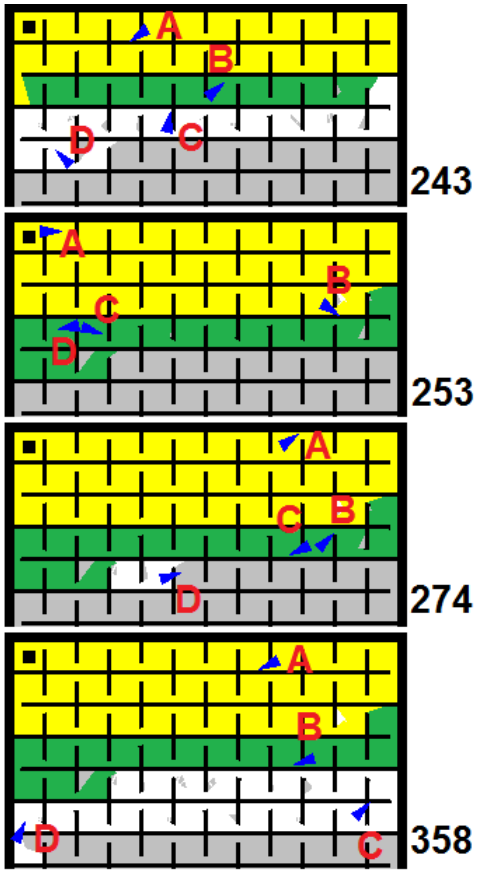
the exploration cycle continues until all of the environment is explored.

In the sequence on the right, the corridor is being explored with 4 agents using the same strategy as before. At **timestep 243**, D has just hit the return ratio and is attempting to deliver the information back to base. In the meantime, B and C are heading towards the frontiers near D, and A is relaying the information about the green area to base. At **timestep 253**, A has just exchanged information with the base station; D has encountered C, given the new area information to C and marked it as being relayed (shown in green), and turned back to go back to explore new frontiers. At **timestep 274**, D is exploring new areas, while B is now in charge of relaying the map of the area shown in green. At **timestep 358**, D hits the return ratio again, and the exploration cycle continues.

Fig. 4.11 illustrates the effect of increasing the number of agents with role-based exploration. On the left we can see the corridor being explored with 2 agents. At **timestep 326**, agent B and its relay A are heading towards



(a) Exploration of the corridor environment with 2 agents using return ratio of 0.7. Snapshots at timesteps 372, 417, 499 and 551 are displayed.



(b) Exploration of the corridor environment with 4 agents using return ratio of 0.7. Snapshots at timesteps 243, 253, 274 and 358 are displayed.

Figure 4.10: Exploration of the corridor environment using return ratio of 0.7. Obstacles are shown in black, explored space is in white, space known at the base station is yellow, green space is being relayed by an agent to the base station, represented with a black square in the top left corner.

the rendezvous point, shown as the purple circle. At **time 367** they make contact and arrange the next rendezvous. Agent A delivers the information to the base station at **time 440**, and at **time 498** agent B heads back to the rendezvous location. The figure on the right shows the effect of doubling the number of agents has on the performance of the team. Here we have two pairs of agents – explorer B with its relay A and explorer D with its relay C. At **timestep 267**, agent D has just had a rendezvous with its relay C, giving it information about the area shown in green. They also agree on their

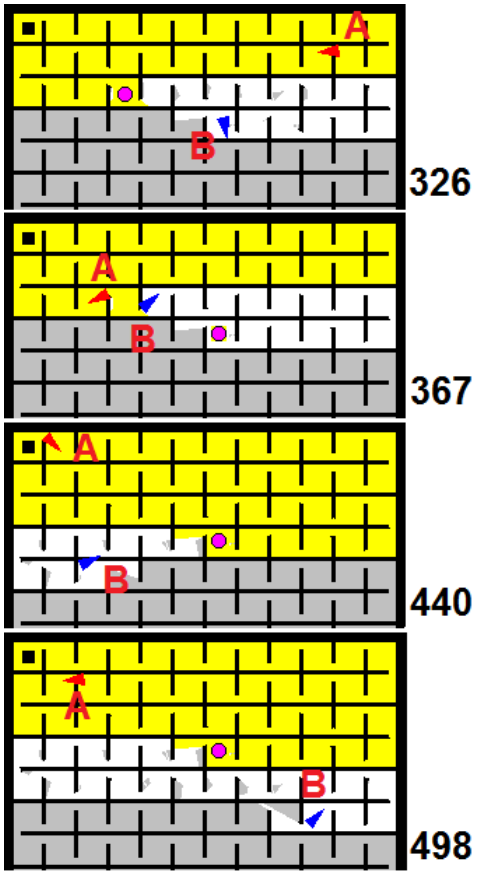
Exploration strategy	Number of times base station is updated (after 5 simulation steps)		
	2 agents	**4 agents**	**8 agents**
Role-Based	13	24	29
R0.2	5	5	4
R0.3	6	5	5
R0.4	7	6	6
R0.5	8	7	7
R0.6	10	9	9
R0.7	14	11	12
R0.8	20	18	16
R0.9	35	35	31

Table 4.4: Number of times base station knowledge is updated for each strategy on the corridor map for 2, 4 and 8 agents.

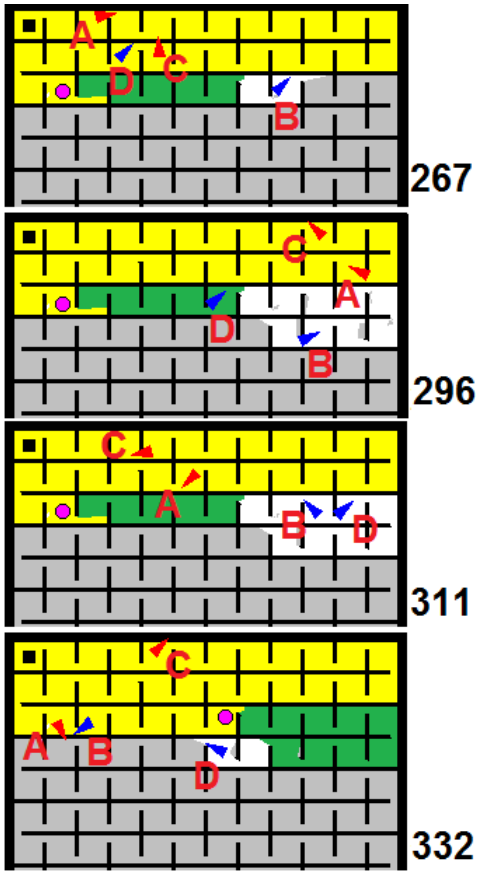
next rendezvous, which is on the edge of the green area, which is the furthest explored area that they know about (you can see the location of the rendezvous as the purple circle in the bottom picture). In **timestep 296**, B starts heading back to the rendezvous with C shown in purple (remember that the timing of the rendezvous is determined by the time it takes for the relay to navigate to the rendezvous location). Relays A and C have just communicated and swapped roles (for simplicity, here they will also swap names to keep the explorer-relay pairs consistent). At **time 311**, explorers B and D meet and swap roles. Finally, at **timestep 332** B meets with its relay A, and the exploration cycle continues. Note that the total area explored by agents B and D combined here is roughly the same size as the area explored by B alone in Fig. 4.11a, illustrating our earlier observation that doubling the number of agents in role-based exploration does little to increase the overall speed of exploration.

4.4.2.4 Distribution of team time

Fig. 4.12a shows the breakdown of team time on various activities for teams of 2 agents exploring the corridor environment. We can see that total team sensing time, shown in blue, stays relatively constant across all the strategies. That is to be expected, as for any exploration strategy it takes roughly the same



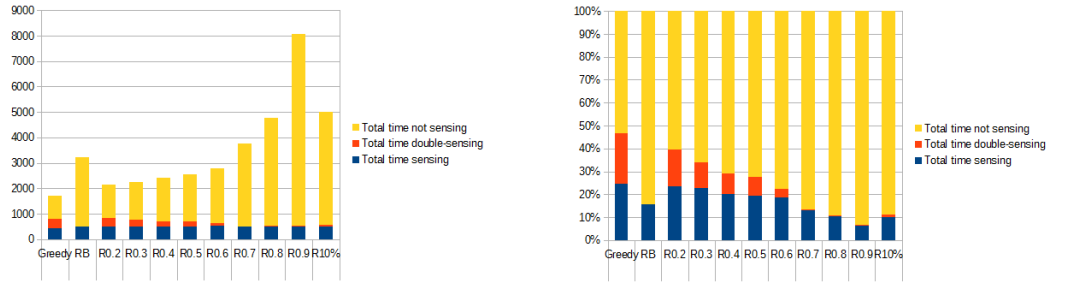
(a) Exploration of the corridor environment with 2 agents using role-based exploration. Snapshots at timesteps 326, 367, 440 and 498 are displayed.



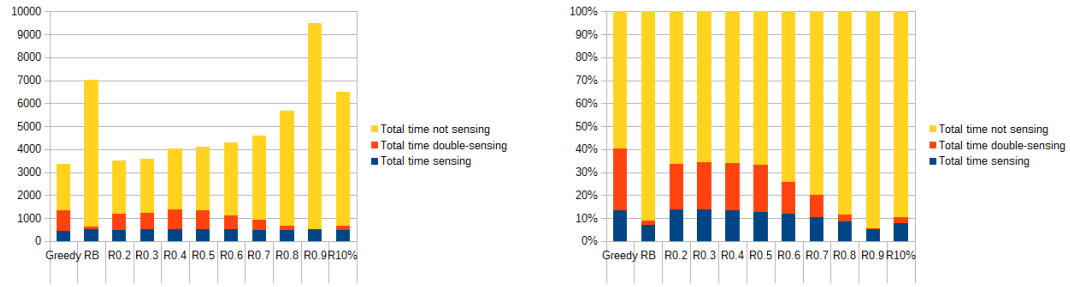
(b) Exploration of the corridor environment with 4 agents using role-based exploration. Snapshots at timesteps 267, 296, 311 and 332 are displayed.

Figure 4.11: Exploration of the corridor environment. Obstacles are shown in black, explored space is in white, space known at the base station is yellow, green space is being relayed by an agent to the base station, represented with a black square in the top left corner. Purple circles are rendezvous locations.

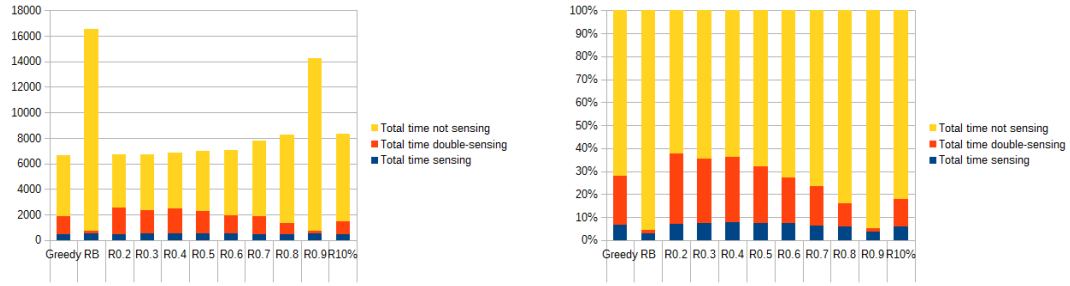
amount of time to sense each square room in the environment. However, we can see that as we require agents to relay information to base more frequently, they seem to spend exponentially more time in transit, not sensing new areas. We can also see that with return ratios less than 0.7, the relaying agent spends more and more time trying to explore the same parts of the environment that have already been explored by its teammate. The reason for this is that with the smaller return ratios, the explorer and the relay end up meeting in unexplored



(a) 2 agents.



(b) 4 agents.



(c) 8 agents.

Figure 4.12: Breakdown of team timesteps spent on several different activities for 2, 4 and 8 agents exploring the corridor environment. On the left, total number of timesteps agents in the team spent sensing new areas, total time spent sensing areas that have already been sensed by another agent (but not communicated to teammates or to base yet), and total time agents spent not sensing (i.e. navigating from one location to another – in this environment it corresponds to the amount of time spent ferrying information back to base by the team). On the right, the same stats are shown as a percentage of total time spent on the mission by the team.

territory for the relay; as a result, the relay has to explore that area before it can meet with the explorer. The tradeoff of such double-covering of unexplored area is that the exploring agent can spend more time exploring and backtrack less to meet the relay, which leads to improved team performance, particularly as the exploration effort moves deeper into the environment. This outcome

is impossible with pre-arranged planned rendezvous, as the explorer has to arrange the rendezvous location with its relay in an area that is known to both the agents at the time of their previous meeting. Double-covering is still possible with pre-arranged rendezvous by two explorers that are in different branches of the agent hierarchy tree.

With 4 agents using greedy exploration all agents still explore the environment together as a pack, with all agents within the communication range of each other. The behaviour is similar with return ratios under 0.3, with agents gradually dropping off from the pack to relay the information back to the base station until only one agent completes the exploration and returns back. With return ratios over 0.4 agents start off as before, but get to form a chain of relays before the environment is fully explored. Fig. 4.12b shows the breakdown of total team time spent on sensing, navigating and re-sensing areas that have already been explored. We can see that with 4 agents some re-sensing happens even with pre-agreed rendezvous (RB). The reason for this here is that we used two branches, one explorer and one relay each. While it ends up behaving as one long chain, some double coverage by the two explorers still occurs. We can see that this time no re-sensing occurs with return ratios above 0.9, where the relays never leave the already explored territory.

The breakdown of team time with 8 agents in the corridor environment is shown in Fig. 4.12c. Having 8 agents in the environment allows for efficient chains of relays to be established with return ratios even as high as 0.9. A relay ends up following the explorer quite closely at all times, resulting in the explorer not needing to backtrack much. Due to frequent encounters between agents, repeated sensing of the environment is also kept very low. Evaluating how much the agent that is actually sensing new frontiers has to backtrack with each strategy is difficult, as the agent that is sensing new frontiers may change throughout the mission.

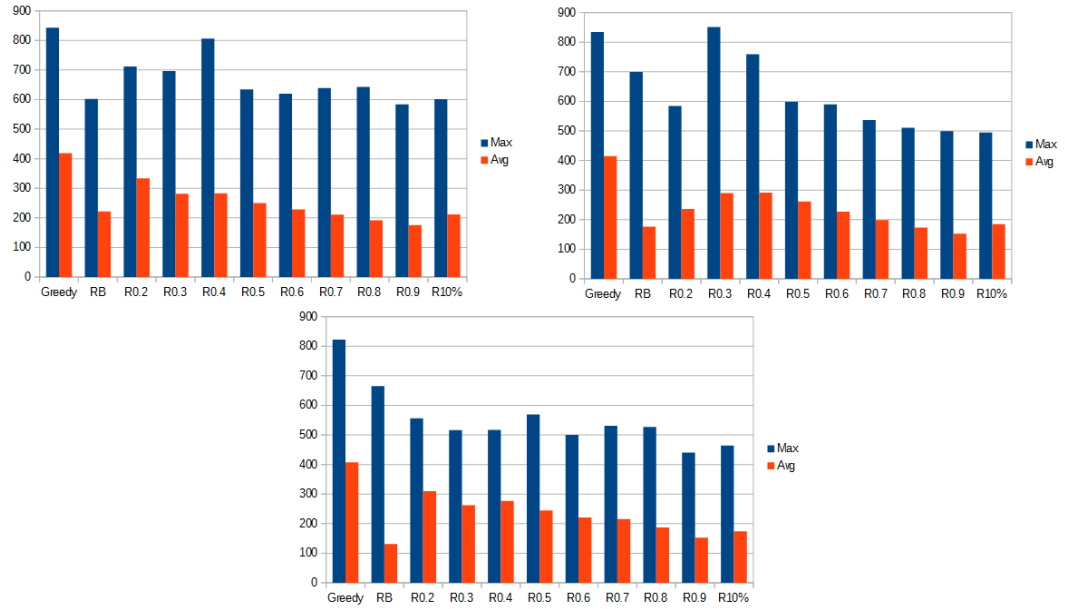


Figure 4.13: The maximum and average latencies of delivering a message sent each timestep to all the agents for the corridor environment. Top-left is for the corridor environment with 2 robots, top-right for 4 robots, bottom for 8 robots.

4.4.2.5 Base station message latency

The maximum and average latencies of delivering a message from the base station to each agent in the team over the course of the exploration mission are shown in Fig. 4.13. As expected, the values are highest for greedy exploration, and tend to get lower as more team resources are spent on communication as we increase the return ratio. The values for role-based exploration are close to the corresponding similar return ratios for a given number of agents in the simulation. An interesting result here is that the latencies seem to be getting smaller linearly and rather slowly as we increase the return ratios. Meanwhile, the total time to perform the mission seems to be increasing exponentially (see Fig. 4.6).

4.4.3 Four corridors environment

The next environment that we use for evaluation is an environment consisting of 4 separated corridors, shown in Fig. 4.14. The purpose of this evaluation is

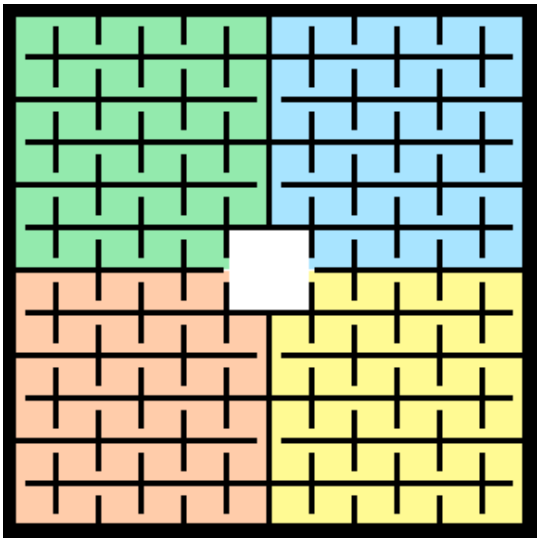


Figure 4.14: Four corridors environment. The base station is located in the centre of the map in the white area, which also serves as the starting location for the agents. The 4 separated corridors are highlighted with different colours.

to find out the effects of having multiple separate sectors in the environment that sub-teams of agents can explore. Is it beneficial to have sub-teams of multiple agents explore each sector, with some agents relaying for the explorers in their sub-team, or is it better to send smaller sub-teams off to explore each sector independently, relaying the information for themselves?

With four agents using role-based exploration, we have two explorers start by exploring one corridor each. They are followed by two relays. Once the first two corridors have been explored, the agents proceed to explore the remaining two corridors. In contrast to this, when either greedy exploration or our approach is used, all four agents get assigned a frontier in each of the corridors, resulting in all four corridors being explored at the same time. With our approach, the agents end up relaying the new information for themselves.

With eight agents, all strategies behave similarly – two agents go into each corridor, with one agent exploring and one agent relaying for them (apart from greedy exploration, where both agents explore the corridor side-by-side).

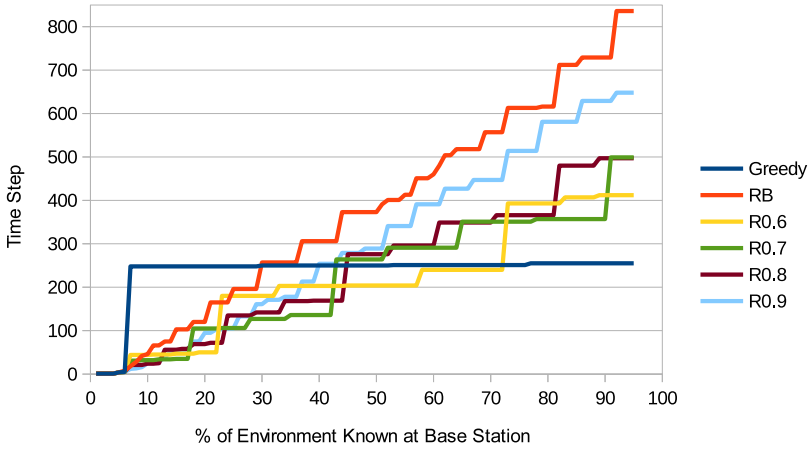
Once again, we examine the performance of the strategies by checking how many timesteps are required to get progressively more information about the

environment to the base station. The results are shown in Fig. 4.15 and Table 4.5. As we can see from the results, the return ratio of 0.2 behaves nearly exactly the same as greedy exploration, as the agents reach the ends of their respective corridors before they are triggered to relay the information. It takes both approaches almost 10 times as long as R0.9 and R0.8 to deliver information about the first 10% of the environment back to the base station for teams of both 4 and 8 agents. When role-based exploration is used with 4 agents in this environment, it performs worse than most other approaches considered, especially in the middle and later stages of the exploration. With 8 agents, using role-based exploration results in all four corridors being explored simultaneously. In this scenario its performance is closer to that of the other approaches. This result suggests that it is better to send out one agent into each corridor to explore sections of the environment simultaneously, instead of sending agents into corridors in pairs and exploring the sections of the environment sequentially (in this case, only exploring the last 2 corridors after the first 2 corridors have been explored). With both 4 and 8 agents, using return ratios between 0.3 and 0.8 appears to show good performance compared to the benchmarks, with values between 0.6 and 0.8 outperforming the benchmarks in most metrics for teams of both 4 and 8 agents.

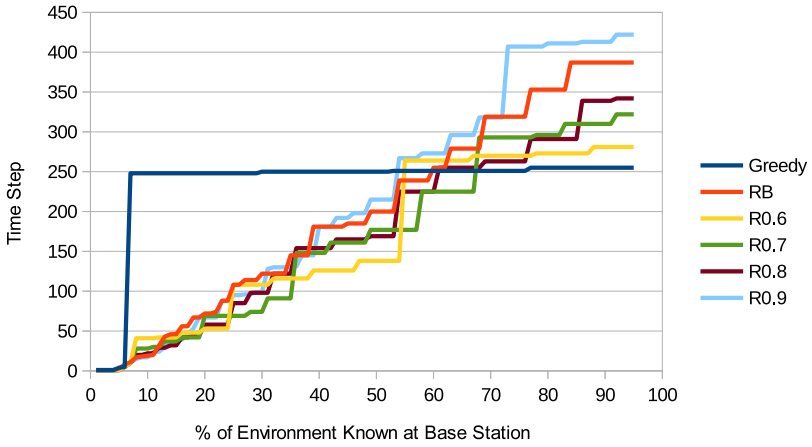
The main conclusion from running the simulations on this environment is that it appears to be better to explore all of the separate regions in the environment simultaneously with smaller sub-teams of agents, than to focus on a few of the regions first with bigger sub-teams.

	4 agents			
	Max (ratio)	**Avg (ratio)**	**Max (timestep)**	**Avg (timestep)**
Greedy	20.67	2.90	236	92
RB	3.43	2.53	581	236
R0.2	20.67	2.90	236	92
R0.3	12.42	2.08	137	71
R0.4	7.83	1.79	162	74
R0.5	5.41	1.75	193	80
R0.6	3.67	1.57	157	72
R0.7	2.58	1.53	244	80
R0.8	2.18	1.61	242	99
R0.9	2.67	1.90	393	161
	8 agents			
	Max (ratio)	**Avg (ratio)**	**Max (timestep)**	**Avg (timestep)**
Greedy	24.8	3.24	238	112
RB	3	1.56	170	77
R0.2	24.8	3.24	238	112
R0.3	8.27	1.71	109	28
R0.4	5.46	1.48	91	33
R0.5	3.8	1.39	92	34
R0.6	2.73	1.34	110	40
R0.7	1.87	1.31	133	45
R0.8	1.66	1.34	120	50
R0.9	1.99	1.58	194	91

Table 4.5: Comparison of exploration strategies evaluated on the "four corridors" environment for teams of 4 and 8 agents. The best value in each column is highlighted in green. Detailed explanation of the values in this table is given in Section 4.4.2.3.

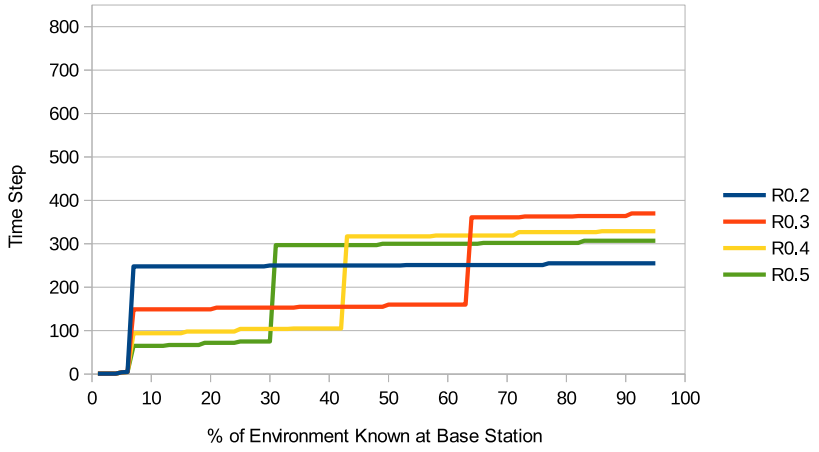


(a) 4 agents.

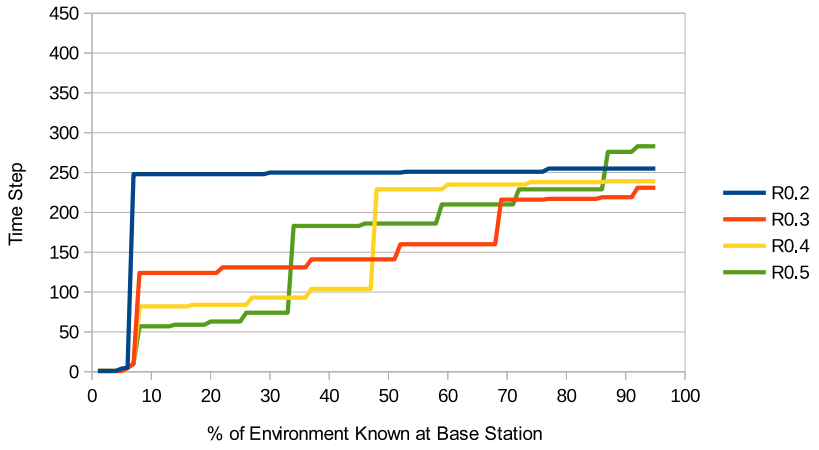


(b) 8 agents.

Figure 4.15: Graphs showing the number of timesteps after which information about each percentage of the environment becomes available at the base station. 4 and 8 agents are exploring the "four corridors" environment using Greedy, Role-Based, as well as our approach with return ratios of 0.6, 0.7, 0.8 and 0.9. The graphs for return ratios between 0.2 and 0.5 are available in Fig. 4.16.



(a) 4 agents.



(b) 8 agents.

Figure 4.16: Graphs showing the number of timesteps after which information about each percentage of the environment becomes available at the base station. 4 and 8 agents are exploring the "four corridors" environment using our utility return approach with return ratios of 0.2, 0.3, 0.4 and 0.5.

4.4.4 Grid environment

Next we evaluate the performance of the strategies on a grid environment, shown in Fig. 4.17. This map is very open; each square room is connected to its four neighbouring rooms. The purpose of this map is to assess how the strategies behave when accidentally encountering another agent is not very likely. This increases the chances that without explicitly negotiating rendezvous locations, agents may miss each other, not share information or form relay chains, which may result in a lot of repeated coverage of the environment and poor team performance.

Fig. 4.18 shows the breakdowns of overall team time spent on sensing new aras of the environment, re-sensing the areas that have already been sensed by other agents (but not known to the whole team), and time spent navigating from one location to another through previously explored territory. As we can see here, all approaches that do not explicitly negotiate rendezvous locations spend a large percentage of their time on repeated coverage of areas of the environment.

Fig. 4.19 shows the maximum and average latencies in delivering the messages from the base station to all the agents in the team throughout the exploration missions for 2, 4 and 8 agents. Here we can see that explicitly agreeing on rendezvous locations appears to lead to significantly faster propagation of messages from the base station in environments where agents are unlikely to accidentally enter the communication ranges of each other. As the number of agents in the team is increased, this advantage is diminished. While role-based exploration maintains similar communication latency with the increase in team size, the latency is decreased for the other approaches, as with an increased number of agents in the environment agents are more likely to communicate opportunistically, sharing information throughout the team.

Greedy exploration on this map with 2 agents does worse than most strategies considered even in terms of time taken to explore 90% of the environment,

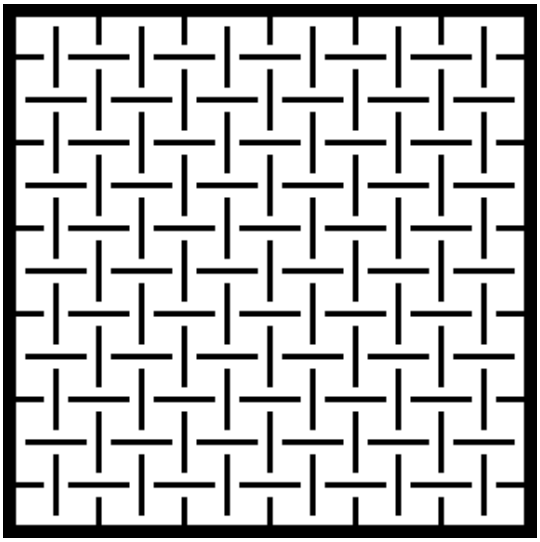


Figure 4.17: Grid environment. The base station and the starting positions for the agents are in the top left corner.

as shown in Fig. 4.20. Additionally, on this environment, decreasing the return ratio does not always lead to speeding up the completion of the mission. The reason for this is that agents are unlikely to enter each other's communication range and therefore may not know when the team has completed exploring the environment and it is time to return to the base station. This leads to significant repeated coverage of the environment and a delay in delivering the information to base. As the number of agents is increased to 4 and 8, greedy exploration performance improves, as agents are more likely to accidentally meet each other and share information. While the total team time spent on re-sensing the environment seems to be less affected by the number of agents in the team, the average time spent on re-sensing the environment per agent goes down, which leads to the increase in performance.

While using role-based exploration with explicitly agreed-upon rendezvous reduces the amount of repeated coverage, it also halves the number of agents actively involved in exploring the environment, which appears to result in worse overall performance than most of the other approaches considered. This result seems to hold for 2, 4 and 8 agent missions. Role-based exploration appears to behave similarly to our opportunistic rendezvous approach with the return

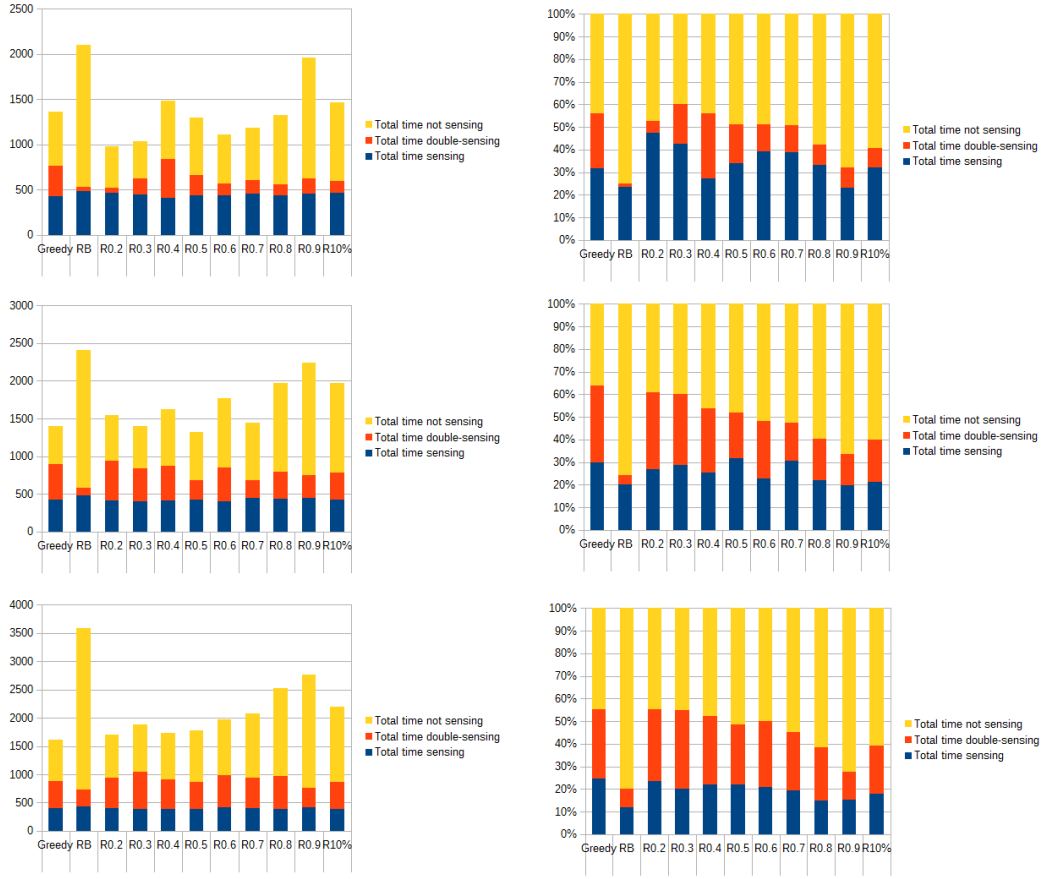


Figure 4.18: Breakdown of overall times spent by the team on sensing new areas, repeated coverage of areas already sensed by other agents, and time spent navigating between areas for the grid environment. Statistics from the run with 2 agents on top, 4 agents in the middle and 8 agents on the bottom.

ratio of 0.9, with similar amount of team time spent on facilitating communication and with similar communication latency. Interestingly, with 8 agents in the team, the opportunistic approach actually appears to outperform planned rendezvous in all of the metrics. The likely explanation for this is that the opportunistic approach is able to allocate resources better at different stages of the exploration – having more explorers in the early stages, when the exploration takes place closer to base and agents are able to self-relay effectively, and allocating more agents to facilitate communication as the exploration effort moves deeper into the environment. The density of agents in the environment is high enough such that agents are able to communicate opportunistically often

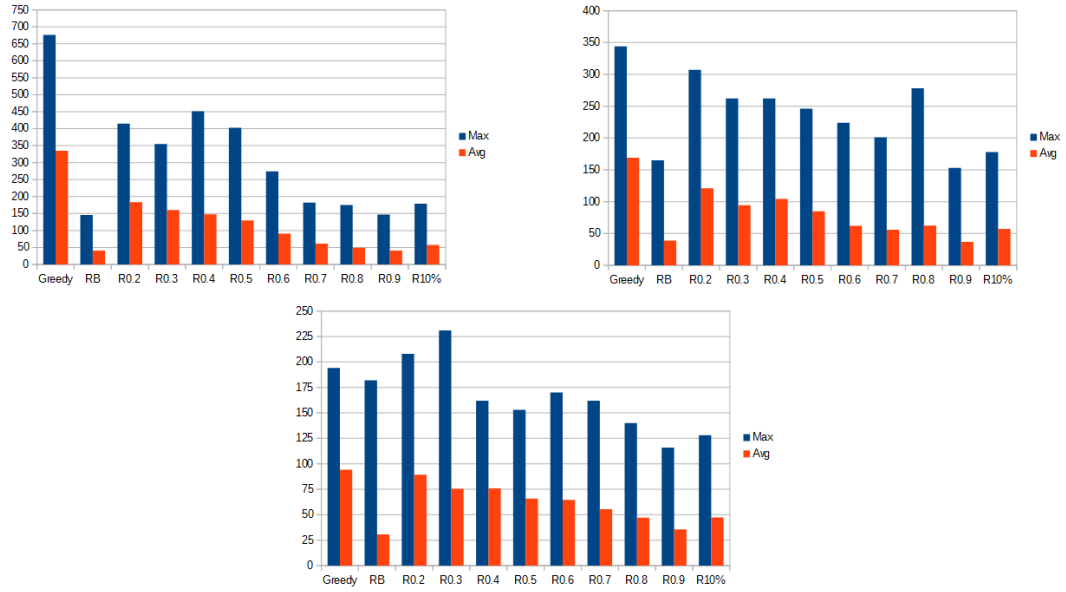


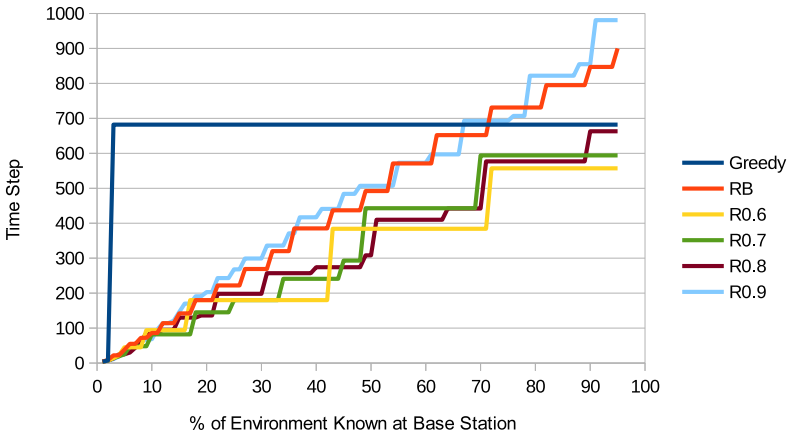
Figure 4.19: The maximum and average latencies of delivering a message sent each timestep to all the agents for the grid environment. Top-left is for the grid environment with 2 robots, top-right for 4 robots, bottom for 8 robots.

enough to maintain chains of relays.

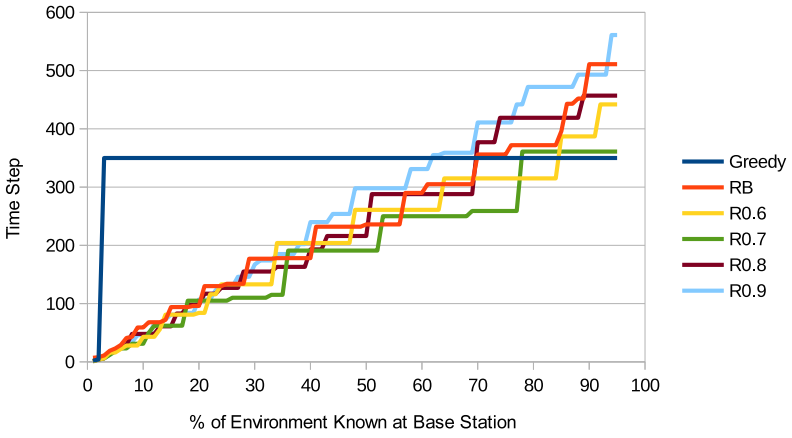
From Table 4.6 we can see that overall, using return ratios between 0.3 and 0.8 appears to produce good results on this map with team sizes between 2 and 8. Using a fixed return trigger of each 10% of the environment being explored appears to actually give the best overall performance, most likely because with this environment being open there is a short direct-line path from any location to the base station. Therefore, the cost involved in delivering information to base has little dependence on the current progress of the mission (knowledge at base station), and so delivering a constant amount of information to base all the time gives good performance.

4.4.5 Large grid environment

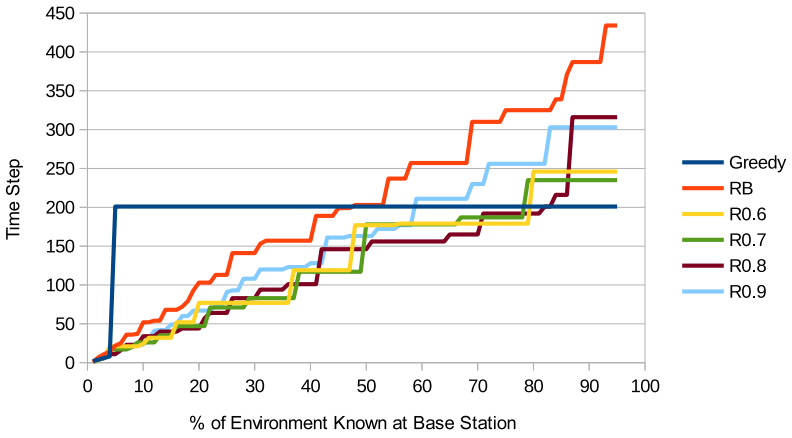
In the previous subsection we used a grid environment with a width and height of 12 20x20 rooms. To see how the size of the open environment affects the performance of the exploration strategies, here we use a similar environment, but with a height of 24 and width of 34 20x20 rooms, for a total of 816 rooms



(a) 2 agents.

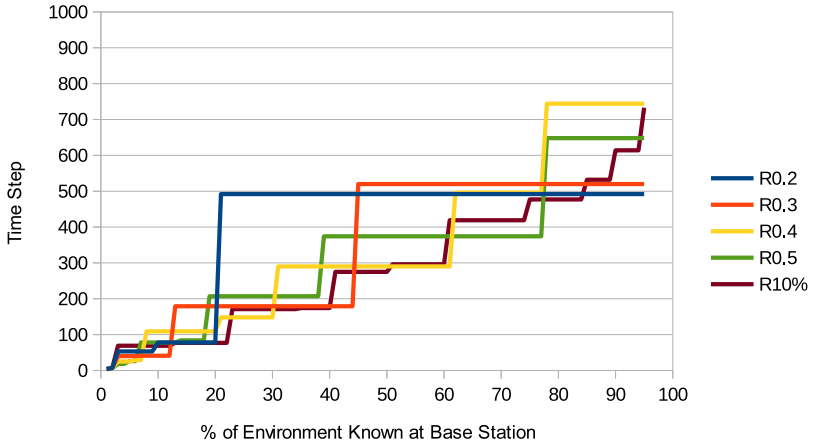


(b) 4 agents.

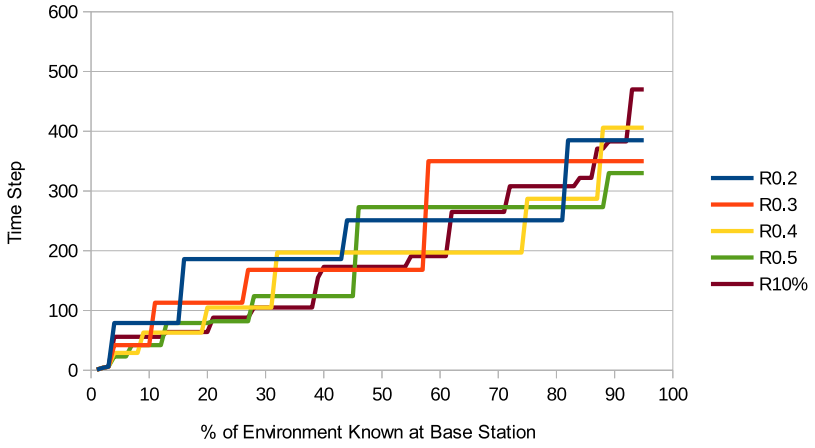


(c) 8 agents.

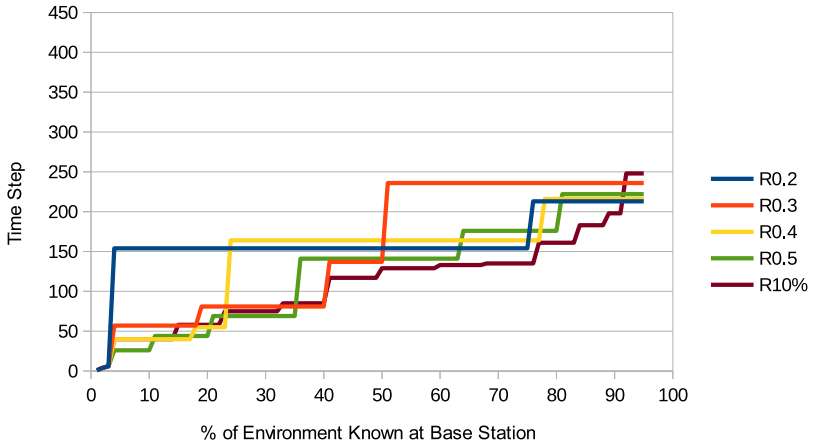
Figure 4.20: Graphs showing the number of timesteps after which information about each percentage of the environment becomes available at the base station. 2, 4 and 8 agents are exploring the grid environment using Greedy, Role-Based, as well as our approach with return ratios of 0.6, 0.7, 0.8 and 0.9. The graphs for return ratios between 0.2 and 0.5 are available in Fig. 4.21.



(a) 2 agents.



(b) 4 agents.



(c) 8 agents.

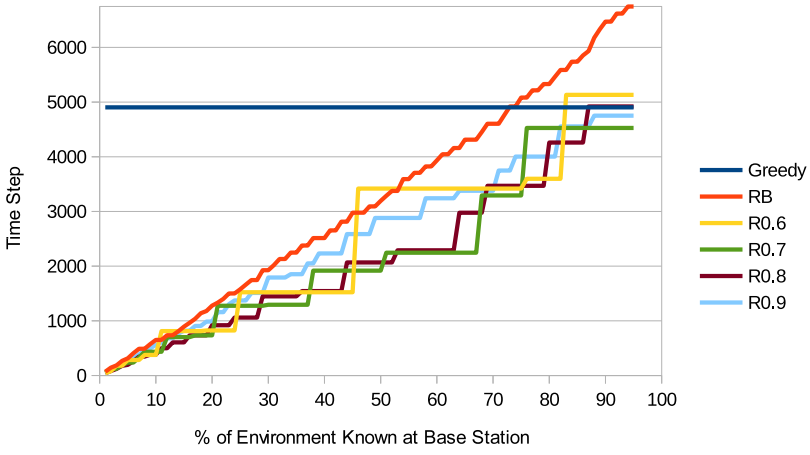
Figure 4.21: Graphs showing the number of timesteps after which information about each percentage of the environment becomes available at the base station. 2, 4 and 8 agents are exploring the grid environment using our utility return approach with return ratios of 0.2, 0.3, 0.4 and 0.5, as well as with a policy of returning after each 10% of the environment has been explored.

	2 agents			
	Max (ratio)	**Avg (ratio)**	**Max (timesteps)**	**Avg (timesteps)**
Greedy	62	5.71	671	416
RB	2.88	1.85	408	203
R0.2	6.39	1.90	415	147
R0.3	3.64	1.45	246	89
R0.4	2.66	1.42	267	97
R0.5	2.69	1.39	200	76
R0.6	2.34	1.35	205	63
R0.7	2	1.35	220	83
R0.8	2.57	1.40	203	84
R0.9	3.16	1.90	489	224
R10%	6.27	1.27	241	33
	4 agents			
	Max (ratio)	**Avg (ratio)**	**Max (timestep)**	**Avg (timestep)**
Greedy	58.33	4.34	344	190
RB	2.11	1.57	181	86
R0.2	6.58	1.73	127	68
R0.3	3.5	1.51	159	63
R0.4	2.42	1.26	133	31
R0.5	1.92	1.24	105	34
R0.6	1.94	1.40	118	64
R0.7	1.82	1.27	110	42
R0.8	2.13	1.50	222	84
R0.9	2.09	1.65	231	112
R10%	4.67	1.29	140	34
	8 agents			
	Max (ratio)	**Avg (ratio)**	**Max (timestep)**	**Avg (timestep)**
Greedy	18.27	2.87	190	89
RB	2.41	1.97	233	103
R0.2	19.25	2.47	146	58
R0.3	7.125	1.64	107	50
R0.4	5	1.54	100	40
R0.5	3.25	1.25	64	22
R0.6	2.625	1.29	85	31
R0.7	2.125	1.24	74	28
R0.8	1.73	1.25	133	31
R0.9	1.90	1.48	142	57
R10%	5	1.21	47	7

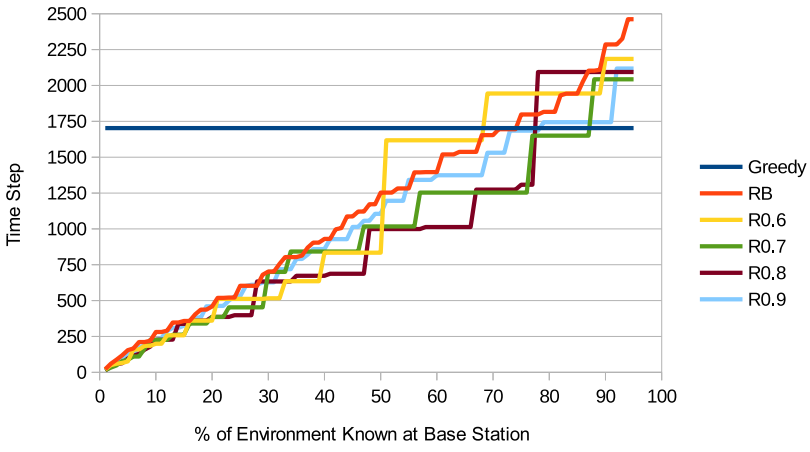
Table 4.6: Comparison of exploration strategies evaluated on the grid environment for teams of 2, 4 and 8 agents. The best value in each column is highlighted in green. Detailed explanation of the values in this table is given in Section 4.4.2.3.

compared to 144 in the smaller grid environment. The starting position for the agents and for the location of the base station are in the top-middle room of the environment. We ran the simulations with 2, 4 and 8 agents.

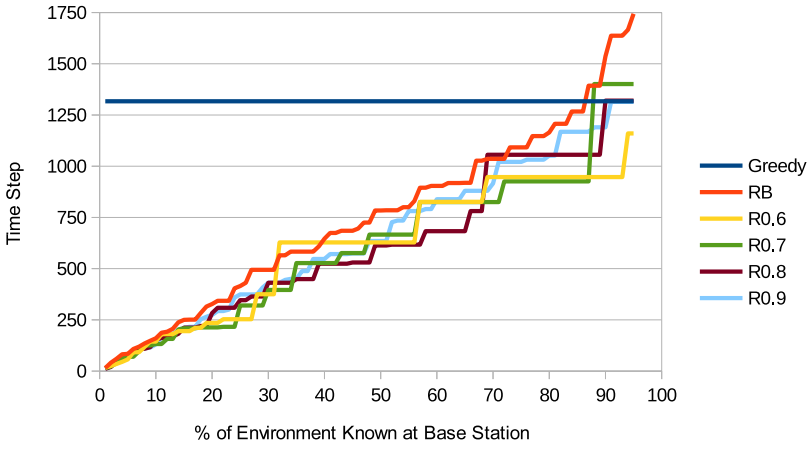
As can be seen from Figs. 4.22 and 4.23, pure greedy approach performs really badly here for 2, 4 and 8 agents. This highlights the importance of team communication in large open environments to prevent redundant covering of the environment by explorers. However, role-based exploration is surprisingly slower than our proposed approach nearly throughout the whole exploration mission for simulations with 2, 4 and 8 agents. One may assume that this may be due to there being half as many explorers with role-based exploration as there are with our approach. However, Fig. 4.24 suggests otherwise – with our approach, agents spend more time re-sensing parts of the environment that have already been sensed by other explorers compared to the actual first-time sensing time for the team (shown in blue). The amount of re-sensing when using role-based exploration is minimal, as expected. This suggests that the number of active explorers is not what gives our approach the edge. It is possible that one factor could be the selection of rendezvous location in role-based exploration – sometimes, the locations picked can be in areas with small frontiers which can be quickly explored; then, an explorer may move to a different part of the environment and will have to backtrack a lot to navigate back to the meeting location in time for rendezvous. Another factor could be the frequency of communication with the relays and, consequently, with the base station. The number of times the base station receives updates from the agents for various strategies is shown in Table 4.7. When using role-based exploration, the base station receives updates from the agents a lot more frequently than with other strategies, which could explain the overall performance degradation – agents are spending too much time relaying small updates of data to the base station. The reason for this is that in role-based exploration, rendezvous timing is determined by the amount of time required for the relay to navigate to the communication range of the base station and then get to the rendezvous



(a) 2 agents.

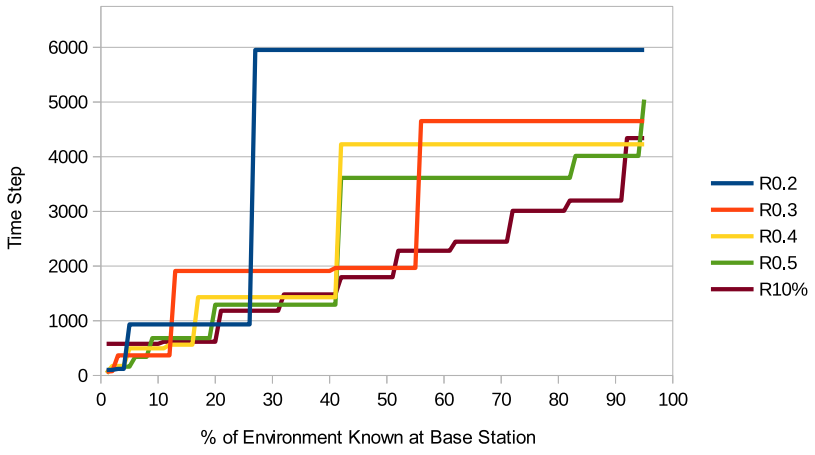


(b) 4 agents.

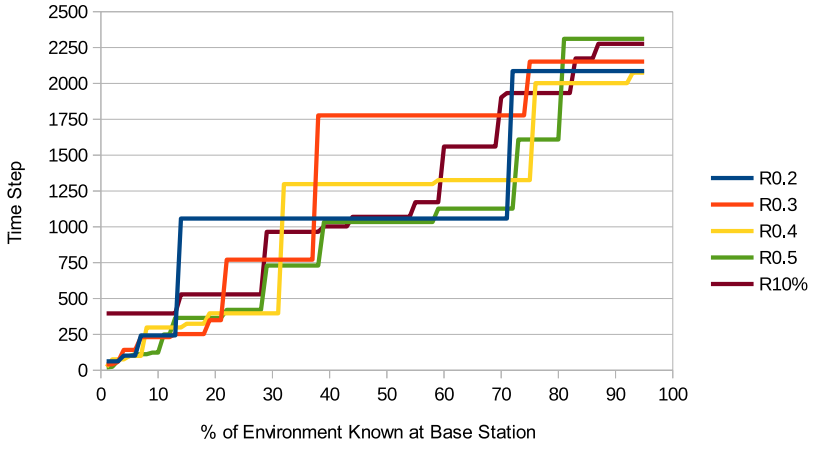


(c) 8 agents.

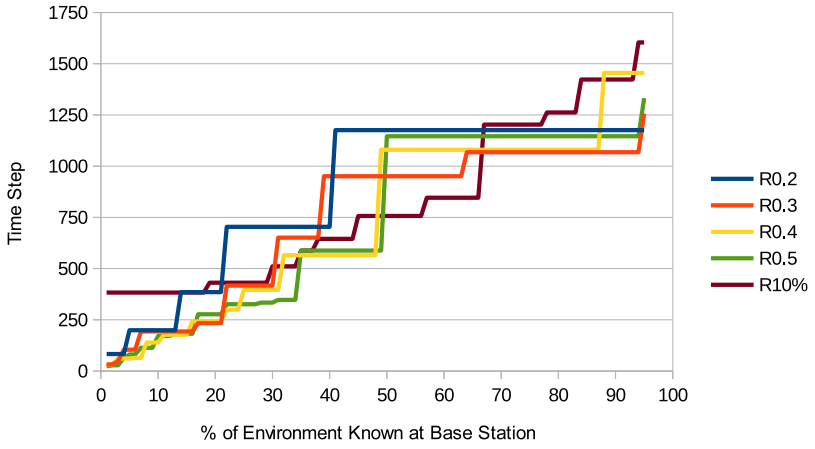
Figure 4.22: Graphs showing the number of timesteps after which information about each percentage of the environment becomes available at the base station. 2, 4 and 8 agents are exploring the "large grid" environment using Greedy, Role-Based, as well as our approach with return ratios of 0.6, 0.7, 0.8 and 0.9. The graphs for return ratios between 0.2 and 0.5 are available in Fig. 4.23.



(a) 2 agents.



(b) 4 agents.



(c) 8 agents.

Figure 4.23: Graphs showing the number of timesteps after which information about each percentage of the environment becomes available at the base station. 2, 4 and 8 agents are exploring the "large grid" environment using our utility return approach with return ratios of 0.2, 0.3, 0.4 and 0.5, as well as with a policy of returning after each 10% of the environment has been explored.

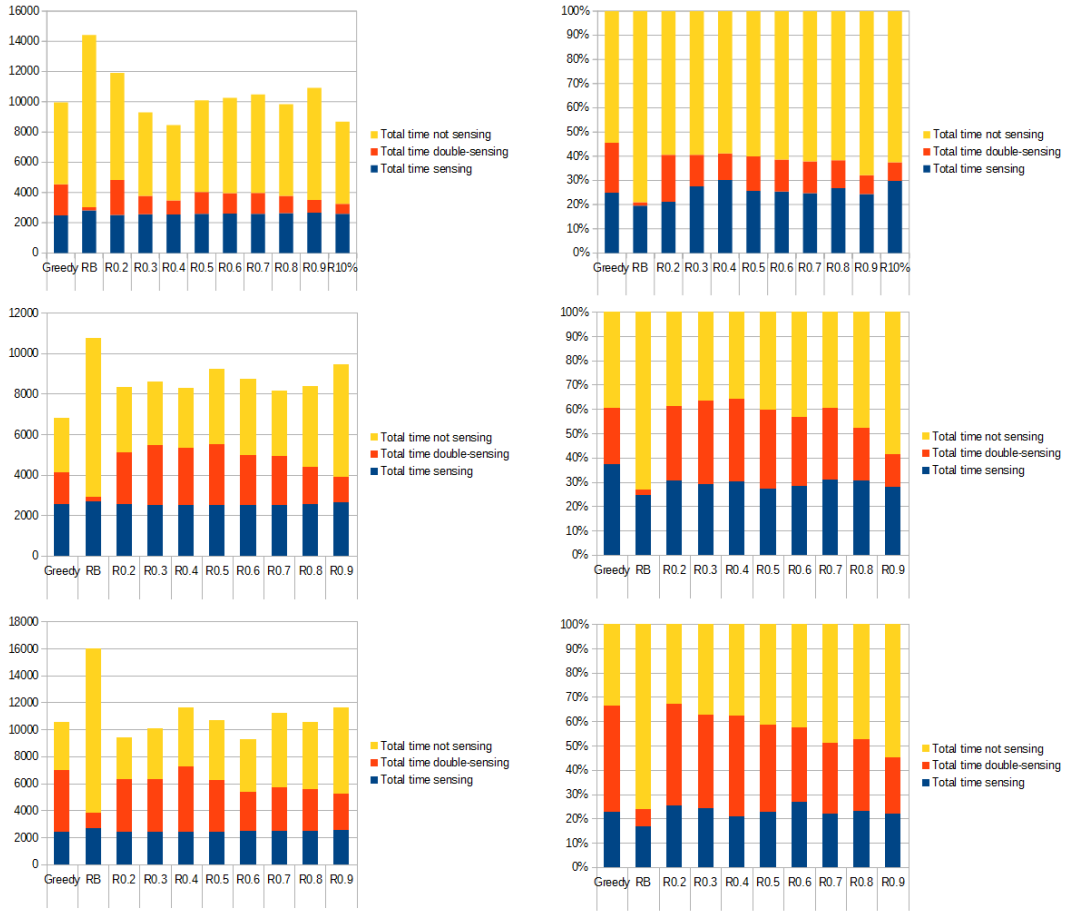


Figure 4.24: Breakdown of overall times spent by the team on sensing new areas, repeated coverage of areas already sensed by other agents, and time spent navigating between areas on the "large grid" environment. Statistics from the run with 2 agents on top, 4 agents in the middle and 8 agents on the bottom.

location. In the large grid environment, paths between any two points in the environment tend to be reasonably short (almost a straight line), which means that rendezvous between relays and explorers happen rather frequently. A way to mitigate this issue could be to impose a minimum limit on how long explorers should explore before heading back to the rendezvous location, however that variable may be environment-specific and require prior knowledge of the environment type. Relays may then have to remain idle while they wait for the explorer at the rendezvous location, but they may use this time to explore frontiers near to the rendezvous point. These ideas are explored in more detail

Exploration strategy	Number of times base station is updated (after 50 simulation steps)	
	4 agents	8 agents
Greedy	4	2
Role-Based	67	80
R0.2	6	6
R0.3	9	9
R0.4	10	11
R0.5	13	13
R0.6	16	16
R0.7	16	19
R0.8	23	28
R0.9	46	48

Table 4.7: Number of times base station knowledge is updated for each strategy on the "large grid" map for 4 and 8 agents.

in the next chapter.

On Fig. 4.25 we can see that role-based exploration does lead to greatly reduced latencies in receiving and propagating messages from the base station to all the agents. An interesting observation here is that as the return ratio is increased from 0.5 to 0.9, the overall speed of exploration is not much reduced, while the communication latency is improved significantly. This suggests that diverting resources to improving communication among the team may be offset by improvements in coordinated behaviour of the team in large open environments, where lack of communication between team members is likely to lead to significant amounts of redundant coverage of the map. According to Table 4.8, using return ratios of between 0.5 and 0.8 appears to yield good overall exploration performance compared to all considered approaches.

	2 agents			
	Max (ratio)	**Avg (ratio)**	**Max (timesteps)**	**Avg (timesteps)**
Greedy	63	5.87	4826	3083
RB	2.23	1.77	3272	1407
R0.2	5.8	2.70	4894	2771
R0.3	3.39	1.73	2407	1082
R0.4	3.08	1.64	2705	1061
R0.5	2.37	1.43	2091	722
R0.6	2.13	1.39	1933	736
R0.7	1.91	1.24	1516	426
R0.8	1.54	1.20	1721	418
R0.9	1.85	1.45	1553	753
R10%	7.41	1.31	500	136
	4 agents			
	Max (ratio)	**Avg (ratio)**	**Max (timestep)**	**Avg (timestep)**
Greedy	70.96	4.87	1679	858
RB	2.5	1.46	759	314
R0.2	4.20	1.71	959	363
R0.3	2.80	1.67	1142	514
R0.4	3.17	1.42	782	292
R0.5	1.84	1.25	661	207
R0.6	1.08	1.34	886	293
R0.7	1.84	1.19	342	128
R0.8	1.85	1.16	485	125
R0.9	2	1.33	473	212
R10%	16.5	1.93	875	377
	8 agents			
	Max (ratio)	**Avg (ratio)**	**Max (timestep)**	**Avg (timestep)**
Greedy	65.85	5.28	1297	764
RB	2.14	1.41	690	203
R0.2	4.15	1.84	652	331
R0.3	3.02	1.45	427	198
R0.4	2.18	1.34	508	193
R0.5	1.87	1.32	533	184
R0.6	1.81	1.14	281	60
R0.7	1.52	1.13	454	70
R0.8	1.75	1.14	373	69
R0.9	1.66	1.20	367	106
R10%	19.15	2.10	536	254

Table 4.8: Comparison of exploration strategies evaluated on the "large grid" environment for teams of 2, 4 and 8 agents. The best value in each column is highlighted in green. Detailed explanation of the values in this table is given in Section 4.4.2.3.

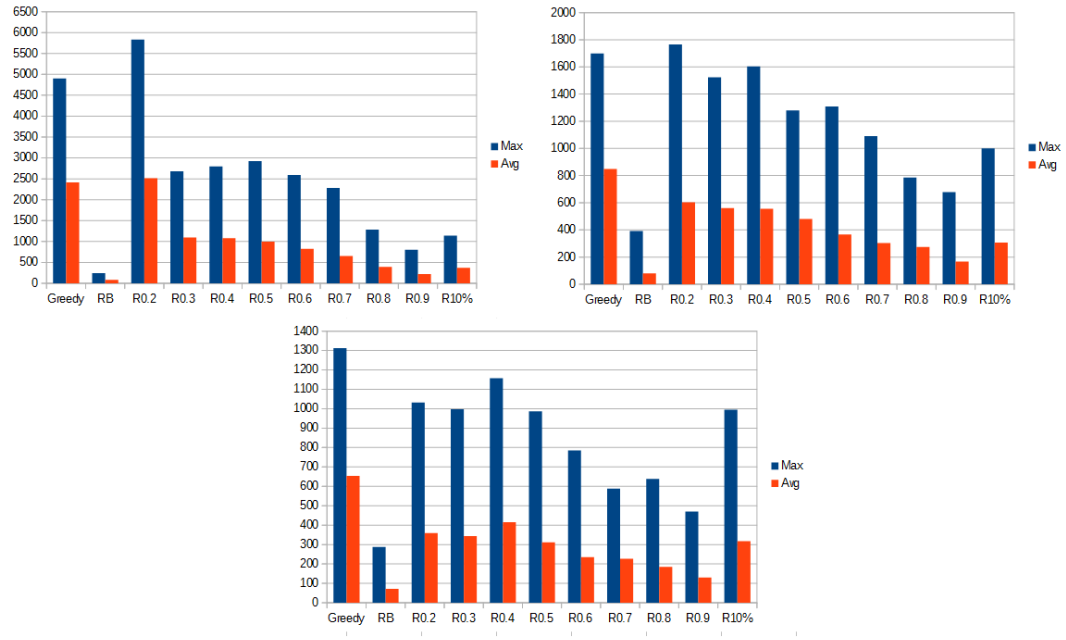


Figure 4.25: The maximum and average latencies of delivering a message sent each timestep to all the agents on the "large grid" map. Top-left is for 2 robots, top-right is for 4 robots, bottom is for 8 robots.

4.5 Discussion of results

4.5.1 Effect of parameters on team behaviour

We have shown in simulation that the approach proposed in this chapter allows to allocate team resources between exploration and communication by tuning a single parameter, the return ratio, between 0 and 1. Values closer to 0 lead to greedier exploration behaviour, while values closer to 1 encourage increased frequency of communication with the base station. Greedier team behaviour tends to result in faster overall exploration at the expense of obtaining information – about *some* of the environment – early on in the mission. Increasing the frequency of communication tends to improve performance early on in the mission, but is usually slower than greedier approaches at getting information to the base station at the later stages of exploration. The approach involves little explicit planning between agents and is therefore very robust, yet it can lead to complex emergent behaviour with agents taking on relaying tasks as becomes necessary, as well as forming relay chains. Lack of explicit rendezvous

planning can also offer increased robustness when operating in dynamic environments. Additionally, the proposed approach allows for flexibility in the 'return' conditions for the agents. For example, in a search-and-rescue scenario, locating a surviving victim would require informing the human responders of this event as soon as possible. By assigning very high utility to this information, agents will be forced to immediately relay this information to base. In different scenarios, it may be beneficial to delay relaying the information. For example, consider an agent that has nearly fully explored a big room or a dead-end corridor (and is aware of it, for example by having prior knowledge of the structure of the environment). In this case, relaying the information to base, and then returning to the area to complete the exploration may waste time and energy. Instead, the agent could finish exploring the area and *then* relay the information. Incorporating such behaviour into our approach would be straightforward compared to approaches with explicit rendezvous planning.

According to the simulation results, using return ratios between 0.6 and 0.7 produces good overall results on all of the environment types considered with teams consisting of between 2 and 8 agents. With these parameter values, the proposed approach seems to adapt very well to different types of environments considered, allocating the appropriate number of explorers, relays as well as adjusting the frequency of communication with the base station without any prior knowledge of the environment being required. These results were obtained for the specific parameters used in the simulated experiments, in particular the ratio of the speed of exploring agents moving through new area to the speed of agents while relaying through known territory. Observations in Section 4.4.2.2 and Fig. 4.5 suggest that these results should hold for speed ratios between around 0.4 and 1.

4.5.2 Failure modes

The coordination approach proposed in this chapter is highly decentralized and robust to the failure of individual agents. There is no explicit reliance of any

agent on other agents in the team. If any exploring agent becomes disabled, other agents in the team will continue their mission as usual, and the frontiers originally assigned to the disabled agent will eventually be explored. If an agent acting as a middle link in an emergent relay chain fails, the chain will automatically adapt to exclude that agent. In both of the cases described above, the team will simply behave as if the failing agent was never a part of the team.

An agent carrying a lot of 'relayed' information failing can cause more disruption to team performance, even though the team will still be able to recover and continue on with the mission. Other agents will still keep a copy of the information that was to be relayed by the failed agent, and share it with other agents and the base, but they will assume that the disabled agent will succeed in relaying the information, and so will not attempt to communicate with the base station until they have obtained enough new information. This means there might be a delay in getting important data to the base station and the sharing of information throughout the team might be temporarily impaired. One way to ensure the flow of information to the base station is maintained is to equip each agent with an additional, more robust means of communicating to their teammates that they are 'alive'. Low frequency, low bandwidth radio can be used for this purpose, as it has a long range and good penetration through obstacles that may be encountered in indoor environments. Then, if an agent goes out of service and its teammates stop receiving the corresponding 'alive' message, they can unmark all of their cells as being 'relayed' as they can no longer assume that that information is actually being relayed to the base station. While this will ensure agent data gets delivered to base in a timely manner even in case of a relay failure, it may temporarily disrupt the exploration effort as multiple agents may attempt to relay the same data to base. Additionally, this introduces some additional risks in case the 'alive' message cannot be sent or received by any agent for any reason.

4.5.2.1 Evaluation in simulation

We evaluate the recovery of the team from agent failure on the RoboCup Rescue Virtual Robot Competition 2013 Preliminary2 map, modified with some additional clutter. The map is shown in Fig. 4.26. In the simulations, we attempt to explore the environment with two agents, starting from random locations on the edges of the map. We script the simulations such that at timestep 1000, which is around the middle of a typical simulation run, the explorer that is carrying the most new information fails. The failing agent is unable to move or communicate with any other agent. A total of 25 simulations are performed. The average total agent knowledge, as well as knowledge at the base station for each timestep are shown in Fig. 4.27, and compared with the averages of 25 simulations without agent failure. For these simulations our approach with return ratio of 0.7 was used, but the results here should give an indication of how our approach is able to deal with failure in general. In particular, as long as at least one agent remains operational, our approach will be able to finish the mission, exploring the full environment and delivering the information to base.

As can be seen from Fig. 4.27b, losing one agent at timestep 1000 leads to correspondingly slower accumulation of total agent knowledge. Fig. 4.27a shows that there is an initial drop in the rate of base station knowledge accumulation, most likely because the remaining agent believes that the failed agent has already relayed some of the data it has to base, and therefore delays its own return trip to base. The team seems to revert to a higher rate of base station knowledge acquisition towards the end of the mission, however, most likely after the remaining agent makes contact with the base station and is able to proceed with the exploration mission normally without making assumptions about the actions of the failed agent.

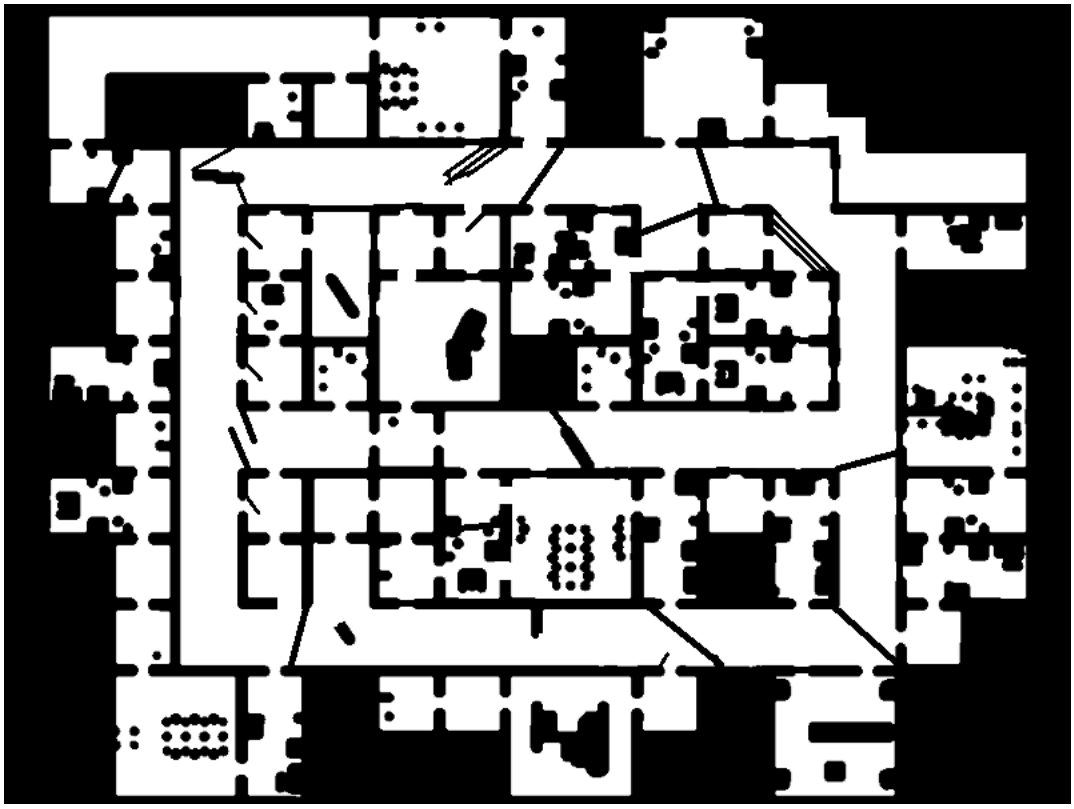
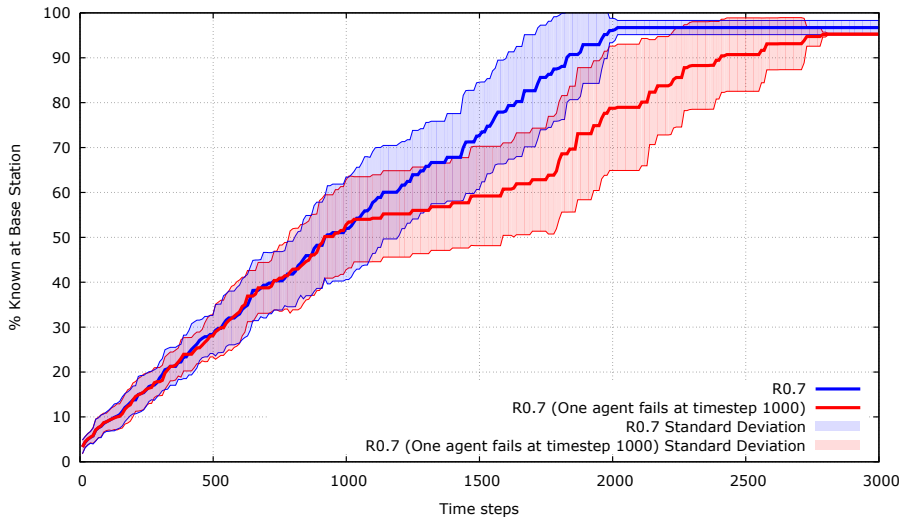


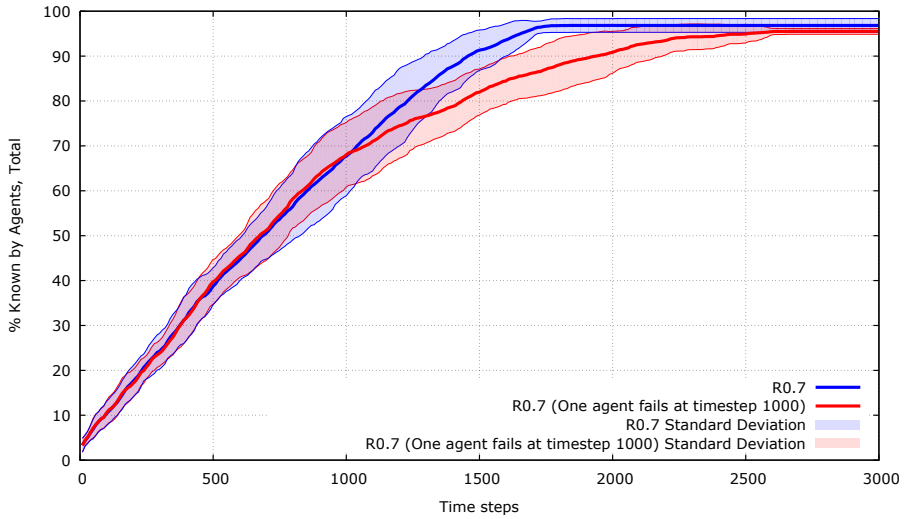
Figure 4.26: RoboCup Rescue Virtual Robot Competition 2013 Preliminary2 occupancy grid map, modified with some additional clutter. Obstacles are shown in black, free space is in white.

4.5.3 Methodology

In this chapter we evaluated the proposed approach against several benchmark strategies in a discrete-time simulator with a carefully chosen level of abstraction. Indoor maps that are currently available for high-fidelity robot simulators are not very realistic; there have been attempts at improving the plausibility of the available simulated environments [5]. Here, we used a number of different maps which are characteristic of environments that may be encountered in an indoor exploration mission – a long corridor, multiple corridors, a small and a large cluttered open space. While simulated experiments in these environments allow us to get an idea of how the proposed approach may perform in different situations, more exhaustive experiments, both in real life and simulation, are required before the proposed method can be applied in real life scenarios.



(a) Average percentage of the environment known at base station for each timestep. Figures for each timestep are averaged across 25 runs with random starting locations for each scenario evaluated.



(b) Average percentage of the environment known by the overall team for each timestep. Figures for each timestep are averaged across 25 runs with random starting locations for each scenario evaluated.

Figure 4.27: Effect of one of the agents failing at timestep 1000 on the speed of exploration when using the R0.7 exploration strategy with 2 robots.

4.5.4 Communication

The simulations in this chapter were conducted with a very short communication range to examine the performance of the cooperation strategies under pessimistic assumptions. Larger communication range will mean that agents will opportunistically communicate with each other more frequently, improving the overall team performance.

The performance of the proposed approach can be impaired in open environments with severely limited communication where agents are unlikely to enter each others' communication range by chance, which can lead to a lot of repeated sensing of the environment. Using a higher return ratio can help mitigate this effect by sharing information through the base station. An additional downside is that the operators have very limited information about the potential whereabouts of the agents (with explicitly agreed rendezvous they will at least know where and when agents are to rendezvous and when to expect them back); it is more difficult to predict or adjust how often communication with agents will take place, and it is more difficult to reliably disseminate messages from the base station to all the agents (such as control commands, for example, or new priority search areas, or even changes in the team exploration strategy). Adding an additional high-range low-bandwidth control communication channel to be used in team communication may help alleviate some of these problems. In addition, opportunistic rendezvous does not allow to fully exploit the topology of the environment and the communication ranges of agents to shorten their travel paths and avoid traversing terrain that may be hard to navigate. For example, when exploring a multi-level building, a staircase connecting two floors can be difficult for robots to navigate, but two robots may be able to establish line-of-sight communication on either end of the staircase, keeping one robot on the ground floor to act as a relay while another agent explores the other level. When using opportunistic rendezvous, the exploring agent assumes that it will need to relay the data for itself and it is only by chance that it may meet

another agent that will assume the relaying task; it will therefore attempt to navigate the staircase. It may happen that it meets another robot after it traverses the staircase on the way to the base station, passes the data to be relayed to the other robot, and then has to traverse the staircase again to get back to exploring the frontiers – an undesirable scenario which slows down the progress of the mission, uses up energy and puts the robots at risk of getting stuck. In the next chapter we attempt to address this problem by proposing a novel approach for rendezvous planning that uses the communication ranges of agents to improve team coordination and performance. In particular, we use the fact that agents are often able to establish wireless communication over thin walls, obstacles or even just rough terrain that is difficult to traverse in order to improve the planning of rendezvous among agents.

Chapter 5

Explicitly Planning for Communication in Team Coordination

Explicitly arranging rendezvous between agents can offer advantages in certain scenarios. We describe some such scenarios and propose the novel concept of multi-point rendezvous to address them, along with a method for generating such rendezvous locations. This is followed by several improvements to the original algorithm with simulated experiments. The discussion of the results and the applicability of the proposed approaches concludes this chapter.

5.1 Rendezvous assumptions and problem statement

While using opportunistic rendezvous for periodic team communication can enhance the robustness of the team and offer good performance in terms of exploration speed and communication frequency as indicated in the previous chapter, explicitly arranging rendezvous can offer several advantages. Some of these potential advantages have already been mentioned in Sections 1.2 and 4.5.4. Carefully selected rendezvous locations can allow agents to take advantage of their communication ranges and the layout of the environment to reduce agent travel times, increase the frequency of communication or avoid navigating through difficult or dangerous areas. Where the communication ranges of agents are much smaller than the size of the environment to be explored, the probability of opportunistically entering the communication range of another agent can be low; explicitly arranging rendezvous can help facilitate communication between agents in such scenarios. If the times and locations of rendezvous are explicitly arranged, human operators may have tighter control over the exploration mission and greater visibility into the potential whereabouts of agents in the team at any given time.

In this chapter we aim to answer the following question. Given two robots in communication range of each other at some time t_0, where and when should they agree to enter the communication range of each other in the future? We make the following assumptions, in addition to the assumptions from the previous chapter:

- Both robots are in communication range of each other at time t_0.
- Both robots use occupancy grid maps with the same resolution to represent the environment. However, it should not be too difficult to adapt the approaches in this chapter to other representations of the environment.
- High bandwidth communication link between the two robots is available

at time t_0, and so both robots share the same occupancy grid maps at time t_0 and can localize each other well in this map.

- Only one of the robots will plan the next rendezvous, and then communicate the information to the other robot. Both robots know in advance which of the two robots should do the planning (for example, the one with the higher ID number).

The primary goal of each rendezvous is to share information among the team. One of the robots may then act as a relay with the base station in order to facilitate the exchange of information between the robot team and the base station. Secondary goals include improving shared localization and arranging subsequent rendezvous.

We select Role-Based Exploration [30] with Beyond Frontier Exploration [129] as a multi-agent exploration framework for the initial implementation and evaluation of our approach in this chapter. The summary of Role-Based Exploration is available in Section 2.4.2.1; a brief description of Beyond Frontier Exploration is given in Section 2.3.1. Role-Based Exploration already includes the concept of an arranged periodic rendezvous among each pair of agents, which provides us with a benchmark and allows for a straightforward integration and evaluation of our rendezvous calculation methods. Beyond Frontier Exploration is an easy to implement and robust frontier exploration method that allows the exploring agents to collaboratively explore the environment. A rendezvous in this context is a periodic meeting between an exploring agent and its assigned relay agent. The methods described can, however, be extended to other or more complex scenarios.

5.2 Method description

5.2.1 Rendezvous locations

Usually, a rendezvous location (RV) for a number of agents is defined as a single point p towards which each agent involved in the rendezvous attempts to navigate at an agreed time. This definition has been used both for ground [32] and aerial [130] vehicles. If the agents use an occupancy grid to represent the environment, then p would be a single occupancy grid cell. In the applications that we consider in the thesis, agents usually have a communication range that is much larger than a single occupancy grid cell, and may even allow for non line-of-sight communication through some obstacles, such as thin walls. Therefore, in practice, communication at a rendezvous usually occurs before all agents actually arrive at the rendezvous location. The effective communication range of robots is not explicitly used in planning the rendezvous; communication between robots before reaching the rendezvous location happens opportunistically. There are advantages to such an approach – it is highly robust to the variability in agent communication ranges in different environments, as well as to localization errors among the agents (as agents do not usually need to reach the rendezvous location exactly in order for rendezvous to take place, thereby relaxing the requirement for robots to localize themselves well relative to the rendezvous location).

However, using a single rendezvous location as described above can be inefficient in some scenarios. Consider, for example, a scenario where two robots can meet and communicate on different sides of a thin wall with minimum travel time, or have to travel a long distance to meet at the same rendezvous location. Fig. 5.1 illustrates such a scenario. Rendezvous location 1 is near the unexplored frontier and near the explorer, but will require the relay to cover a long distance to get there and back to the base station. As a result, the flow of information to the base station will be slow. Rendezvous location 2 is half-way between the explorer and the base station. If this location is selected,

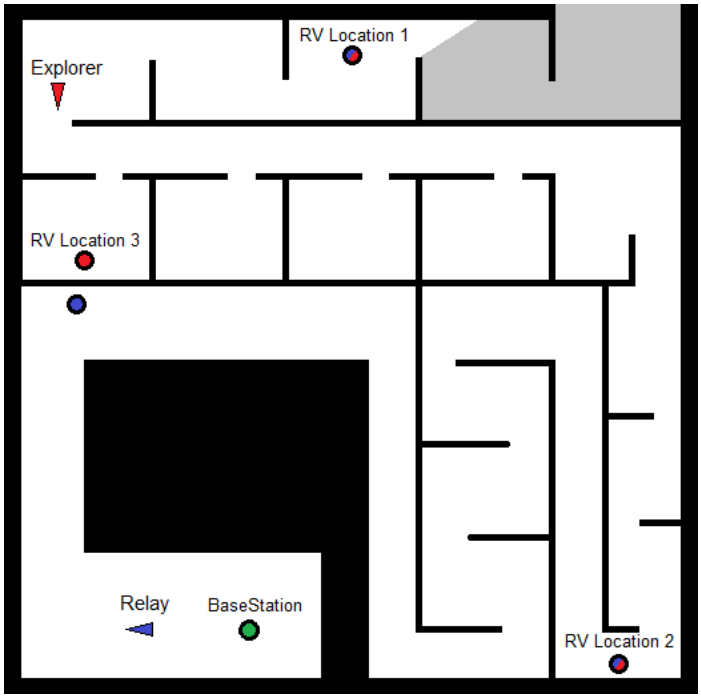


Figure 5.1: An explorer has partially explored an environment (obstacles are black, unexplored space is grey and explored space is white), and needs to deliver the new information to the base station (green) via a relay. Three possible rendezvous locations are shown. Locations 1 and 2 are single points, where both the explorer and the relay head to the same destination. Rendezvous location 3 is split into two points on different sides of a wall – the explorer heads to the red location, while the relay heads to the blue.

the explorer will have to take a lot of time "off" exploring, resulting in slow exploration. Rendezvous location 3 is much more efficient – both the explorer and the relay have to travel relatively short distances to make the rendezvous. Another example could be two robots on different floors of a building trying to rendezvous, where a staircase between the two floors can be very difficult to traverse. With a single rendezvous location, it is very likely that at least one of the robots will have to navigate the staircase, while it would be more efficient for the two robots to meet on either end of the staircase without actually trying to climb it.

In order to address these issues, we define a *rendezvous tuple* as a generalisation of a rendezvous location. A rendezvous tuple is a tuple of points $\langle p_1, p_2, ..., p_n \rangle$, where n is the number of robots planning to attend the ren-

dezvous, such that if each robot i arrives at the corresponding position p_i, all n robots can communicate with each other. Note that a rendezvous location is then simply a special case of a rendezvous tuple, with $p = p_1 = p_2 = p_3 = ... = p_n$. Using a rendezvous tuple has two main advantages over using a rendezvous location:

- It allows us to specify individual robot rendezvous locations p_i that are not necessarily on paths of the robots to the single rendezvous point p, which can reduce robot travel time and energy usage. This can also be useful in situations where a robot is able to communicate with the other robots through obstacles but is unable to physically reach the other robots due to environmental constraints.

- It lets us specify rendezvous points that lie outside of difficult to navigate terrain, making sure that robots do not enter such terrain. For example, it allows us to explicitly arrange for two robots to meet on either end of a staircase. If we use a rendezvous location instead of a rendezvous tuple, we would need to specify that location to be, for example, in the middle of a staircase, and may therefore not be able to avoid one of the robots entering the staircase, which could be very costly in terms of travel time and energy usage (or could even lead to a robot getting stuck) if one of the robots is slightly late to the rendezvous.

The main disadvantage of using a rendezvous tuple is that if the locations in a tuple are not chosen carefully, robots may still be outside of the communication ranges of one another after they arrive at their corresponding locations. Estimating wireless signal ranges can be difficult in indoor environments (see section 5.5.1 for more details). When there is line-of-sight between agents, they may be able to improve the strength of the wireless signal by moving closer to each other in a direct line, in case the signal is too weak to establish communication. This is not always possible when there are obstacles on the line-of-sight

between the two agents. Sometimes they may be able to establish communication by moving towards each other. In some situations, however, they may find themselves in a local maxima with regards to the communication probability with other agents in the rendezvous – that is, they might find themselves unable to increase the chances of communicating with the other agents by moving to a neighbouring location, for example, because of the presence of an obstacle. In that case, robots may have to backtrack in order to reach another rendezvous location from which successful communication is more likely. An illustration of the two situations described above is shown in Fig. 5.2. When using a rendezvous location instead of a tuple, agents are guaranteed to enter the line of sight of each other by the time they reach the rendezvous location.

5.2.2 Selecting rendezvous tuples

A process of selecting good rendezvous tuples should follow the steps below:

1. Obtain a set of candidate rendezvous tuples
2. Evaluate expected utility of each candidate rendezvous tuple given some utility function
3. Select the rendezvous with the highest utility as the next rendezvous

Consider a variant of Role-Based Exploration as described in [32]. Here, single-point rendezvous location P is chosen by the explorer at a prior meeting with the relay to be near the next frontier to be sensed by the explorer, but preferably in an open space or at a junction. [32] shows that choosing a rendezvous location this way tends to improve the exploration performance of the robot team, which means that this is a convenient meeting location for the explorer (that is, the explorer does not need to deviate from its exploration plan too much to make the rendezvous). The relay, however, has to navigate to the same location for rendezvous with the explorer. By using a rendezvous tuple $< P, P_r >$ instead, where the explorer heads to point P as before, but

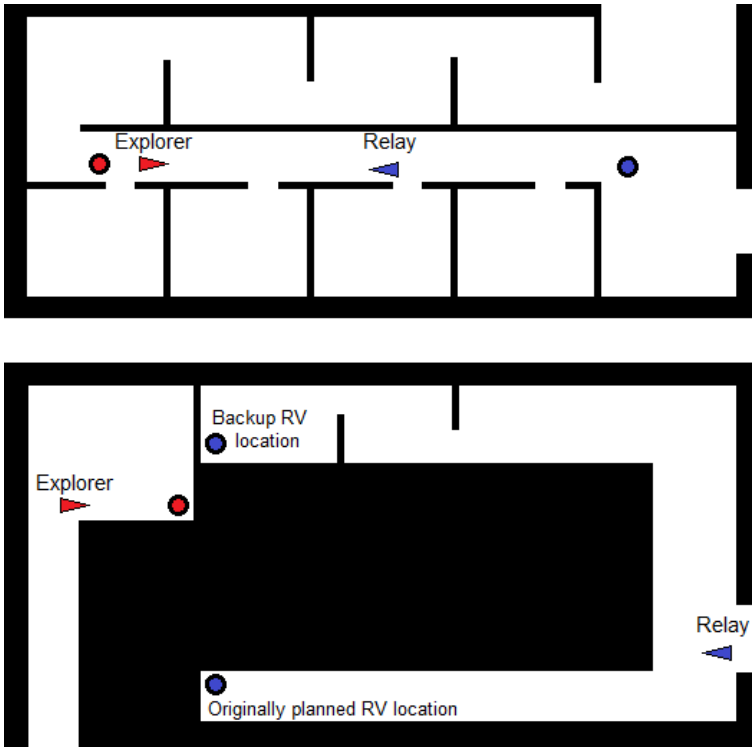


Figure 5.2: Two scenarios where agents have overestimated their communication ranges when planning rendezvous. Obstacles are shown in black, free space is shown in white. The explorer arrives at the red location and the relay arrives at the blue location, but they fail to establish communication as they are outside the communication range of each other. Top: there is line-of-sight between the rendezvous locations, and agents are able to enter the communication range by moving towards each other in a straight line. Bottom: the relay enters area from the right and heads to the originally planned rendezvous location, where it fails to establish communication. Agents are unable to improve signal strength by moving towards each other as they are in a local maxima w.r.t. signal strength; the relay has to backtrack and head to the backup rendezvous location to establish communication.

the relay heads to point P_r which is in communication range of P and is more convenient for the relay to reach than P, we can explicitly take into account the communication ranges of the agents. The outcome is that the relay has to travel less to make the rendezvous, meaning that communication among the team and with the base station can occur more frequently. Given the point P, how do we select a good corresponding point P_r?

5.2.2.1 Obtaining a set of candidate tuples

A set of candidate points for P_r can be obtained by sampling a number of points P_i from the occupancy grid within Euclidean distance r of P, where r is the maximum line-of-sight distance at which communication between two agents is possible. We can then use a communication model, such as a path loss model with wall attenuation factor [11], to estimate the likelihood of communication between points P_i and P. If the likelihood is less than a certain threshold, we discard P_i. Finally, we calculate the utility for each tuple $< P, P_i >$ and select the pair with the best utility as the next rendezvous. One measure of such utility would be the inverse of the distance from P_i to the communication range of the base station, aiming to reduce the distance that the relay has to travel to deliver the information from the rendezvous to base.

In this particular case, the points within communication range of P can also be sampled in a more straightforward way by casting rays of length r from P and sampling points P_i along those rays. In general, however, the assumption that the explorer rendezvous point P is given to us may be relaxed. Consider, for example, the scenario in Fig. 5.1. If location 1 is chosen as the explorer rendezvous location P, the corresponding relay rendezvous candidate locations P_i may all end up being in the top part of the environment, away from the base station. Selecting both points in the tuple from among sampled points may, however, lead us to selecting "RV Location 3" as the rendezvous location, which would yield far greater utility. The algorithm that we propose in this section, which samples points from the whole environment known to the agents and predicts the connectivity among all the sampled points, is more general and can be easily extended to incorporate other utility measures – for example, to allow us to select both rendezvous locations in the pair from among the sampled points as described above.

The full algorithm for selecting the point P_i is presented in Algorithm 5.1. While establishing connectivity among the sampled points, in lines 11 - 13 we

also predict which of the sampled points have connectivity to the base station and add them to a separate list. To reduce the number of costly path calculations in line 19 when looking for the nearest point in base station communication range, only K points with the smallest Euclidean distance to P_i may be considered as a heuristic. In that case, an extra condition can be added in line 11 accordingly. In our implementation, we used $K = 25$.

Other approaches to generating the set of candidate tuples were considered. In [112], we proposed using *thinning* on the occupancy grid map to obtain a set of points that are near obstacles. The candidate locations for the rendezvous location for the explorer were sampled from these points. From each of those points, we would cast rays and sample points along those rays that are within communication range of the candidate rendezvous location for the explorer in order to obtain rendezvous location pairs, which can then be evaluated according to some utility function. We believe, however, that the method described in Algorithm 5.1 is both more efficient and more general.

5.2.2.2 Sampling of points

In the proposed approach, we want to uniformly sample points from the free space of the occupancy grid map. If a pseudo-random sampling method is used and the number of sampled points is small, sampled points are likely to end up clumped together and areas of the environment can end up being under-sampled, as shown on the left side in Fig. 5.3. Using a quasi-random low-discrepancy method for sampling points can help ensure the sampled points are more uniformly spread throughout the environment. Several such methods exists; in this work we chose to use a quasi-random low-discrepancy Sobol sequence [57] for sampling points from the occupancy grid map. Other low-discrepancy sequences can be used in its place. Using a Sobol sequence allows us to generate points that are more equally distributed on the occupancy grid map than the points obtained using a pseudo-random sampling method. Fig. 5.3 shows a comparison between using a Sobol sequence and pseudo-random

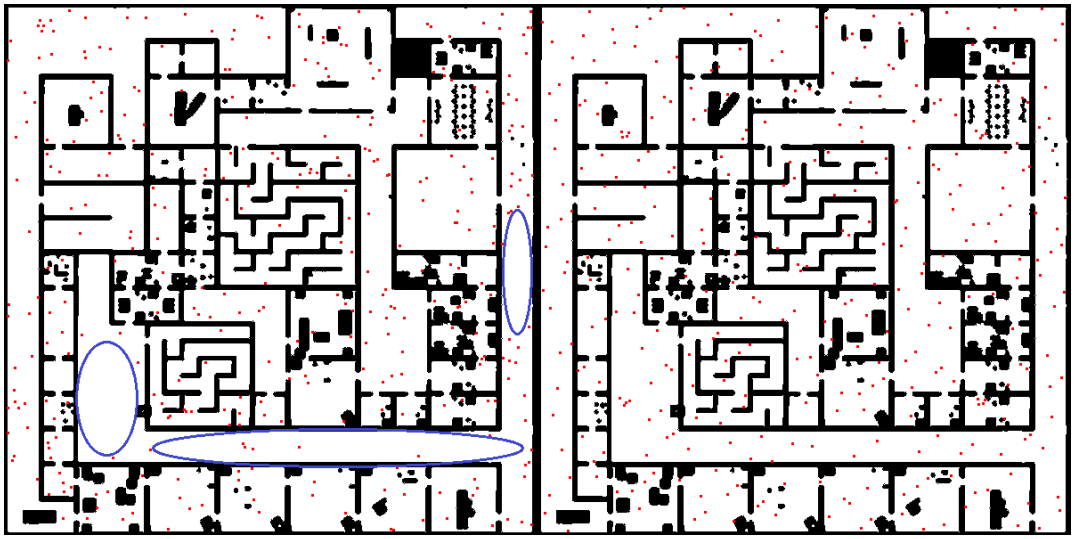


Figure 5.3: On the left, red points are sampled using a pseudo-random number generator. On the right, the same number of points is sampled using a Sobol sequence. On the right, the sampled points are more evenly distributed across the free space of the occupancy grid map (shown in white; obstacles are shown in black). Blue ellipses on the left show areas with low density of sampled points. A Sobol sequence ends up sampling the points more uniformly.

number generator in order to sample points from the occupancy grid map. The algorithm for generating a Sobol sequence is described in detail in [19].

5.2.3 Non-line of sight rendezvous

It is possible that P and P_r may not have line-of-sight between them, however communication is still possible. Having line-of-sight between P and P_r makes the rendezvous configuration robust to errors in the communication model used to predict the possibility of communication between agents in P and P_r. If there is no communication after the agents arrive at their positions, they can move in the direction of each other in a straight line, increasing the probability of sufficient signal strength for communication, and are guaranteed to eventually enter each other's communication range. This is usually not possible when there are obstacles on the straight line between P and P_r – agents moving to the areas of expected better signal strength will find themselves in local maxima, and will need to navigate around obstacles if communication is still

not established at that point. Depending on the environment, they may need to backtrack significantly and spend a lot of additional time navigating to a single rendezvous point P, where communication between the two agents will be guaranteed (see Fig. 5.2). Therefore, it is important to take into consideration the amount of time it would take for the relay to reach P from P_r, especially if the probability of communication at points P and P_r is low according to our communication model.

In addition, it is easier to predict path loss when there is line-of-sight, than it is to predict the attenuation factor of (possibly unknown) obstacles that lie between the agents when there is no line-of-sight. Therefore, we should attach more confidence to our predictions of line-of-sight (LoS) communication than to non-LoS predictions and weigh the risks against the advantages of non-LoS rendezvous in each case. In our implementation we take such risks into account by adjusting the communication model used by the agents to predict signal strength between two points on the occupancy grid map. We set the wall attenuation factor to be 1.1 times higher than the predicted maximum obstacle attenuation factor in the environment. This choice, as well as the problem of estimating communication ranges, are discussed in more detail in section 5.5.1.

Input: Explorer rendezvous location $ExplorerRVLoc$
Occupancy grid $OccGrid$
Density of points to sample $SamplingDensity$
Output: Corresponding relay rendezvous location P_r

```
1  N = OccGrid.GetFreeCellCount()/SamplingDensity;
2  Sample N points P_i from OccGrid;
3  P_N = ExplorerRVLoc;
   // Establish connectivity among sampled points
4  for i = 0 to N + 1 do
5      for j = i to N + 1 do
6          if PropModel.IsConnected(P_i, P_j) then
7              P_i.Connections.Add(P_j);
8              P_j.Connections.Add(P_i);
9          end
10     end
11     if PropModel.IsConnected(P_i, BaseStation) then
12         ConnectionsToBase.Add(P_i);
13     end
14 end
15 MinDistToBase = ∞;
16 P_r = P_N;
   // Locate the point P_i connected to ExplorerRVLoc that has
      the shortest distance to a point within the communication
      range of the base station
17 foreach P_i ∈ P_N.Connections do
18     foreach P_base ∈ ConnectionsToBase do
19         dist = OccGrid.CalculatePath(P_i, P_base).Length();
20         if dist < MinDistToBase then
21             MinDistToBase = dist;
22             P_r = P_i;
23         end
24     end
25 end
26 return P_r;
```

Algorithm 5.1: Choosing a rendezvous location for the relay, given a rendezvous location for the explorer

5.2.4 Initial evaluation

We will evaluate our approach against the benchmark of Role-Based Exploration with single rendezvous locations as described in [32] (we will refer to this benchmark approach as SRL). The evaluation will be performed in the MRESim simulator. The goal of these simulated experiments is to evaluate if, and how well, the proposed approach is able to exploit the topology of the environment to obtain improved team performance, assuming that the communication model used to estimate the communication ranges of agents does not overestimate the real communication ranges. Given this assumption, our approach will perform very similarly to SRL on unfavourable environments where communication through obstacles does not yield significant advantages, as the rendezvous locations selected will be nearly identical. We are therefore only interested here in evaluating performance in an environment where we expect arranging rendezvous through obstacles to yield an advantage. An example of such an environment is the RoboCup Rescue Virtual Robot Competition 2013 Preliminary2 map, with some clutter added to the map. The map is shown in Fig. 4.26, and we will be using it for our initial evaluations.

As the entry point for a team of robots exploring the environment is going to be on the outside edge of the map in many scenarios, we used the areas shown in Fig. 5.4 to randomly select 25 starting locations. In each starting location, we had a static base station and 2 robots with initial locations near the base station (and within its communication range). From each starting location, we performed one run each with SRL and our approach described so far in this chapter, where a rendezvous location consists of multiple points (we will refer to this approach as MP).

Fig. 5.5 shows the average percentage of the environment known at the base station for each time step across all 25 runs from starting locations described above. As we can see, the strategies behave similarly up to around step 1000, after which MP produces slightly worse results. However, the average time it

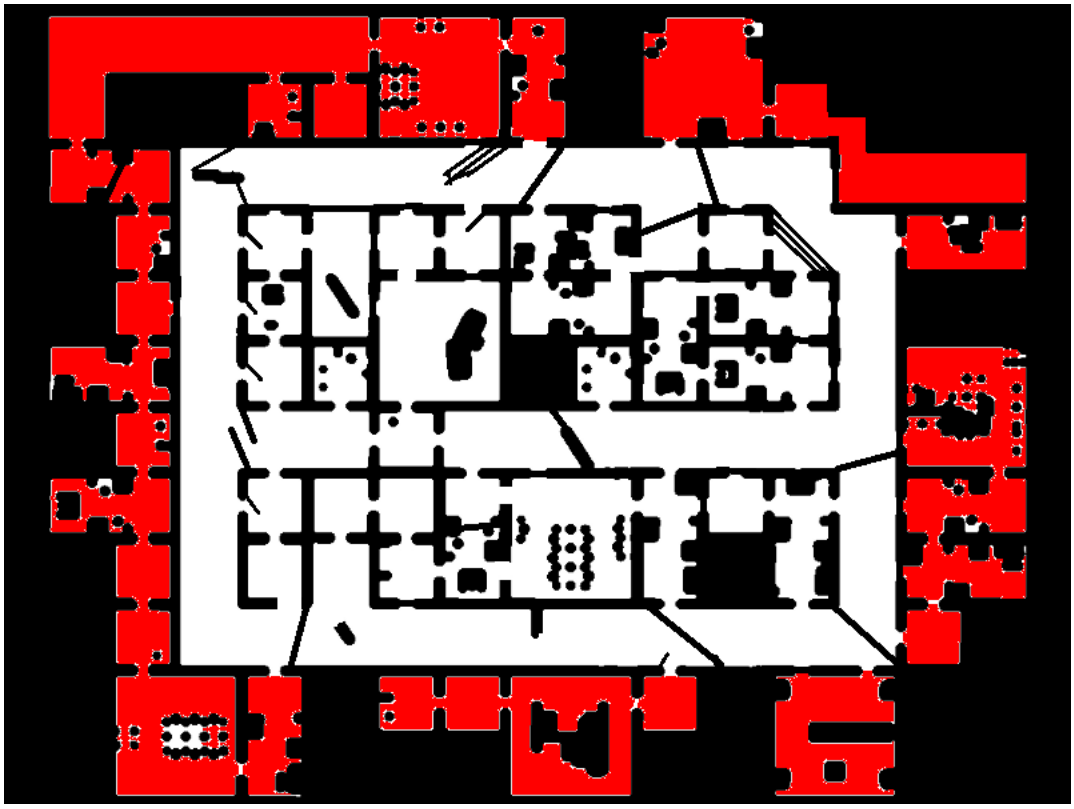


Figure 5.4: Starting locations are randomly selected from the areas shown in red.

takes for a message sent from the base station to reach each agent in the team is much lower when using the MP approach, as shown in Fig. 5.6, particularly after the first 500 time steps. These two results indicate that when using the MP approach the relay is able to deliver information between the explorer and the base station more frequently, but by causing slight disruption to the explorer in the process.

Looking at the number of times the base station gets new information from the agent at each time step, shown in Fig. 5.7, further confirms that when using MP, the frequency of communication with the base station is increased. This is expected – the timing of subsequent RV is decided at the time of the previous RV. It is based on the time when the relay will be able to arrive at the following RV location (but taking into account the minimum amount of time required for the explorer to reach the next frontier and spend some time

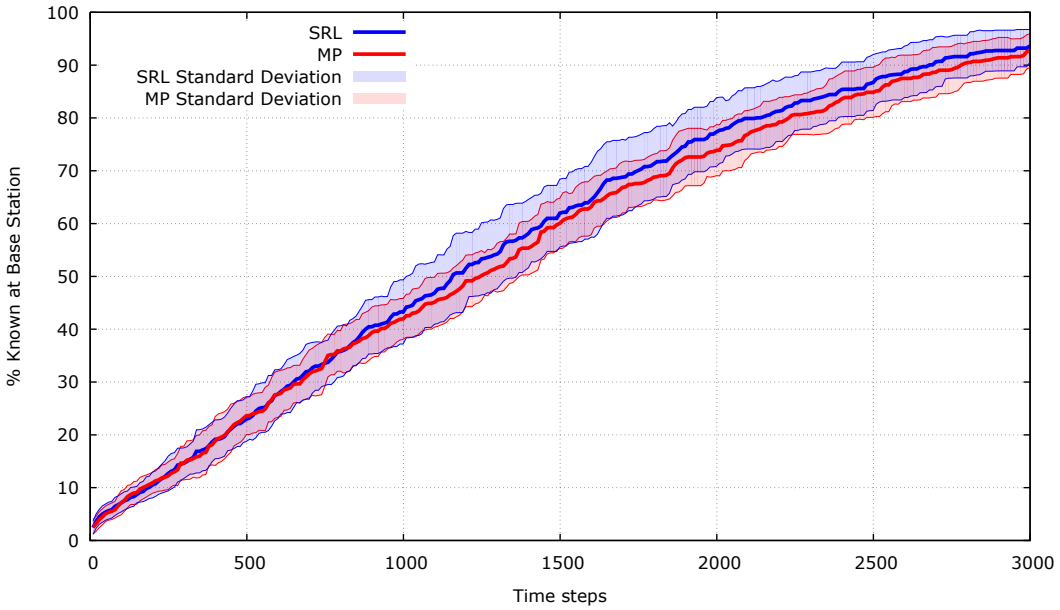


Figure 5.5: Average percentage of the environment known at base station for each timestep.

exploring it). Therefore, shorter travel distances for the relay generally lead to less time between each subsequent rendezvous and therefore more frequent updates of the base station and lower communication latency. When using the MP strategy, the relay has to travel less to get from RV to the base station communication range, and then to the new RV than when using the benchmark SRL strategy, which explains the increase in the frequency of communication with the base station.

Fig. 5.8 shows the average number of timesteps an explorer has spent heading to a rendezvous location (instead of heading to a frontier to sense new areas of the environment). We can see that after around step 500, the explorer does tend to spend more time returning to RV when using the MP approach. At this point approximately 25% of the environment has been explored, and so there is enough information for agents to start exploiting the topology of the environment and their communication ranges with the MP strategy. The frequency of rendezvous should not in general affect the total time the explorer spends returning to RV. The most likely reason why the explorer spends less

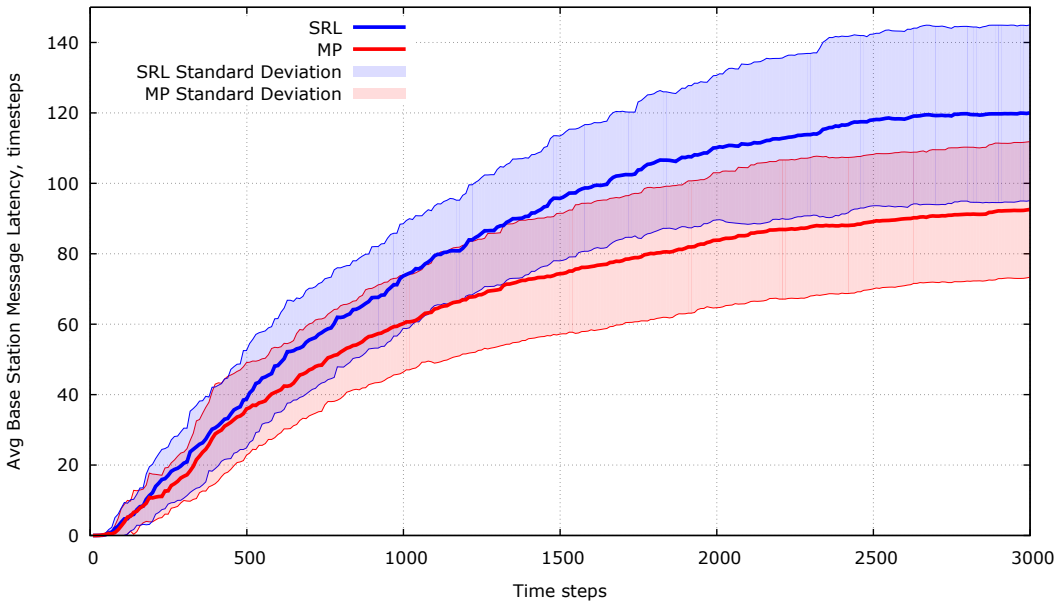


Figure 5.6: Average latency in delivering messages from the base station to all the agents.

time returning to RV when using SRL than when using MP is that when using the SRL strategy, both relay and explorer head to the same location for RV. This means that as the relay is approaching the RV location, especially if it arrives early, the explorer will enter its communication range before it actually reaches the RV location. Using the MP strategy, however, the relay and the explorer head to different locations for the RV (with the assumption that communication between the two locations is possible). Therefore, even if the relay arrives to the RV location early, the explorer will likely have to travel all the way to its RV location (which is the same as the location used for SRL), and therefore will end up spending more time in transit to RV.

According to Fig. 5.9, after around timestep 500 fewer timesteps are spent sensing new areas of the environment on average when using MP compared to SRL. This metric seems to converge for the two strategies after around timestep 2500. When using the SRL strategy, a lot more repeated sensing of the environment happens compared to when using the MP strategy, although the number of timesteps in which repeated sensing occurs is very small compared

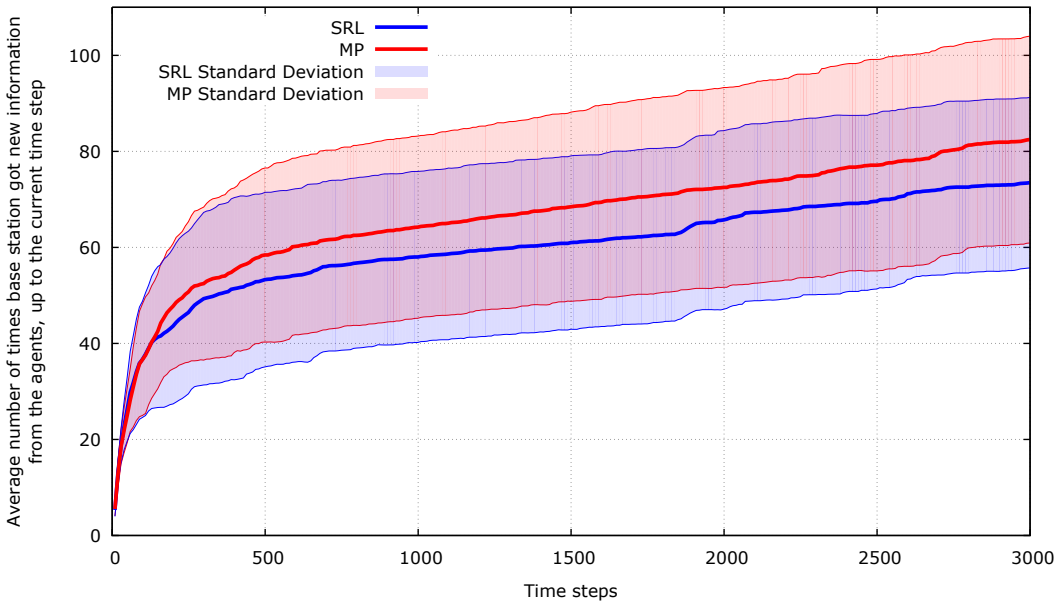


Figure 5.7: Average number of times the base station received new occupancy grid information from the agents up to the current timestep.

to the overall number of timesteps, as can be seen in Fig. 5.10. Since we are only using two robots for the mission, one of which is a relay, the most common case of double-sensing would be when a relay arrives at the RV location, which is usually selected near an old frontier, and therefore senses some of the area it perceives as new (but which has already been sensed by the explorer since the previous rendezvous). This again confirms our explanation for why the explorer spends more time returning to the RV when using the MP strategy and hence why the average percentage of the environment known by the base station is slightly lower when using the MP strategy.

Another interesting observation are the relative contributions of both agents to the sensing of the environment. Because agents can exchange their roles (of Explorer and Relay) during the mission, we track the number of occupancy grid cells sensed while in the agent role, rather than by individual agents. The results are shown in Fig. 5.11. We can see that with both strategies, as expected, the area sensed by relays is negligible compared to that sensed by the explorers. The amount of area sensed by the Explorer with the SRL strategy

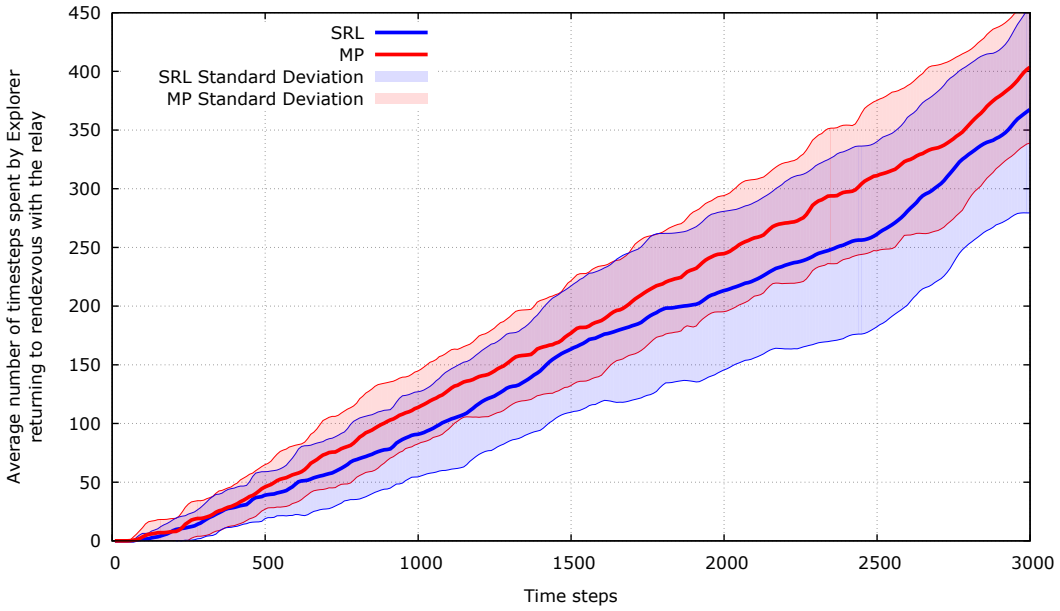


Figure 5.8: Average number of timesteps explorer has spent heading to rendezvous up to the current timestep.

appears slightly higher than that of the MP strategy for most of the mission; this is likely related to our prior observation of explorers spending more time returning to the RV when using the MP strategy.

The number of timesteps that relays remain idle (either due to the explorer being late to the RV, or because the relay is early) for both strategies is shown in Fig. 5.12. After around timestep 500, the total idle time appears to be increasing at a similar rate for both strategies. Initially, the MP strategy has the relay spend a lot more time idle than the SRL strategy. The reason for this is that we require that the explorer spends at least some time (100 timesteps in our implementation) exploring a frontier before heading to RV. The relay has to travel less in the MP strategy, hence more time spent idle. After the explorer moves deeper into the environment, the time it takes for the relay to head to base and return to RV is generally sufficient to give the explorer enough time to explore the frontier, and so rendezvous is scheduled at around the time when the relay is expected to be able to arrive at the RV location, which likely explains why the idle times for relays increase at a similar rate for both strategies.

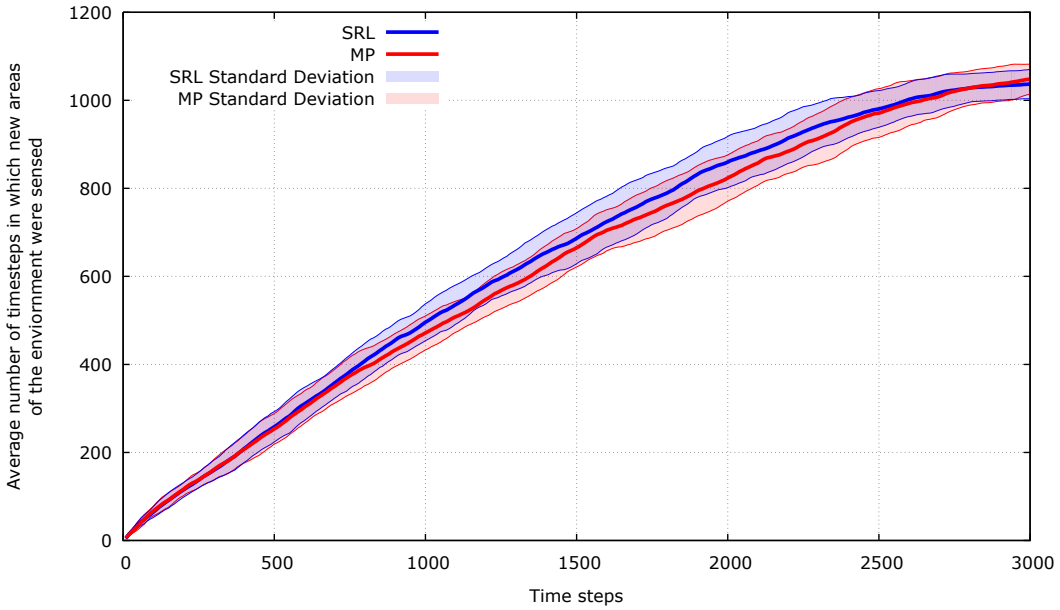


Figure 5.9: Evaluation of MP against SRL with 2 agents. Average number of timesteps agents have spent sensing new areas of the environment, averaged across 25 runs on the modified Robocup Rescue Virtual Robot Competition 2013 Preliminary2 map.

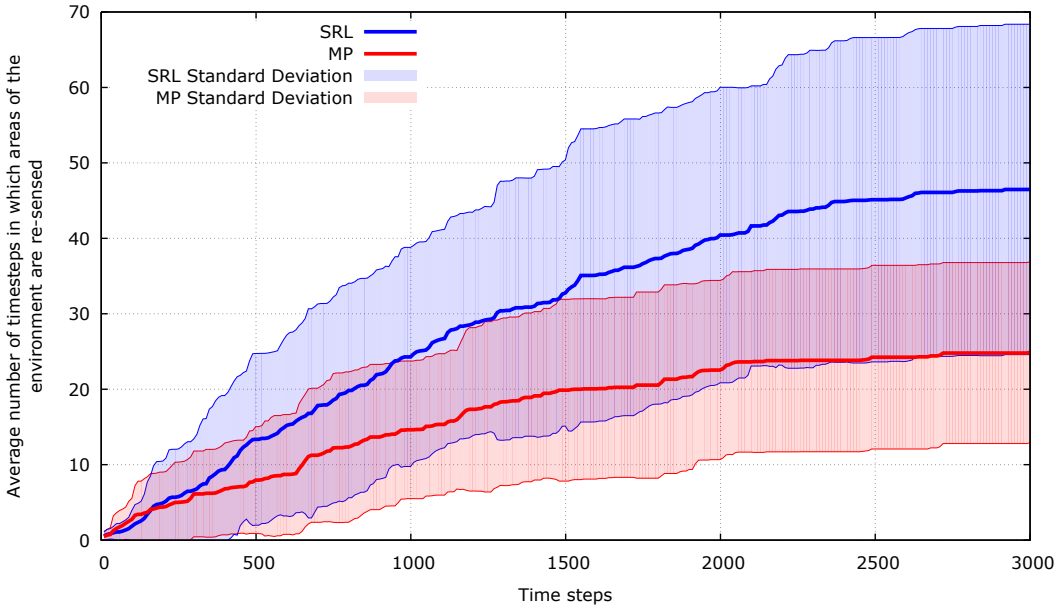


Figure 5.10: Average number of timesteps agents have spent sensing areas of the environment that have already been sensed by another agent before.

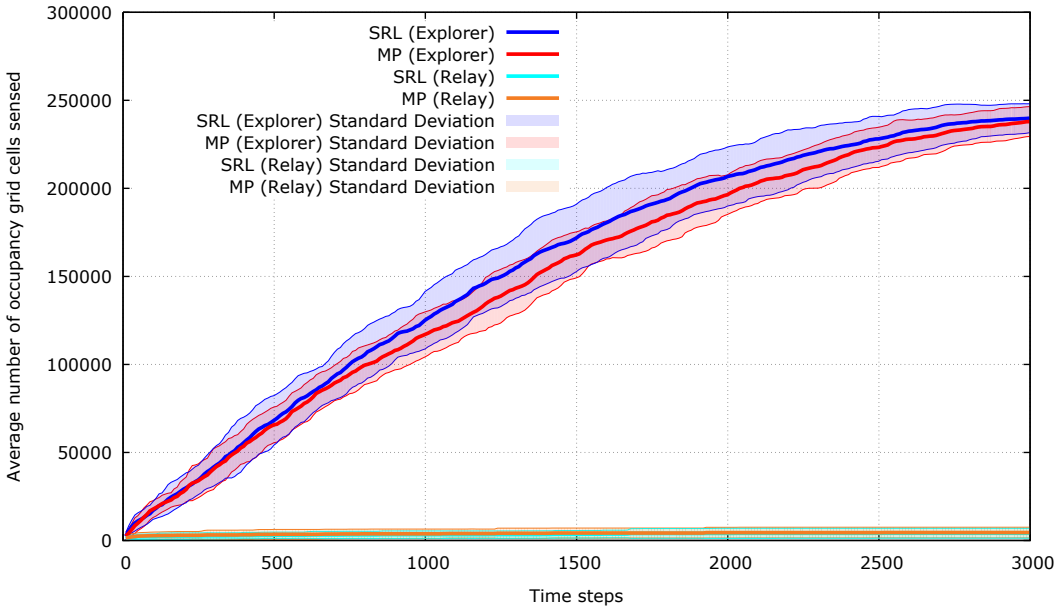


Figure 5.11: Average number of occupancy grid cells sensed by explorer and relay for each strategy.

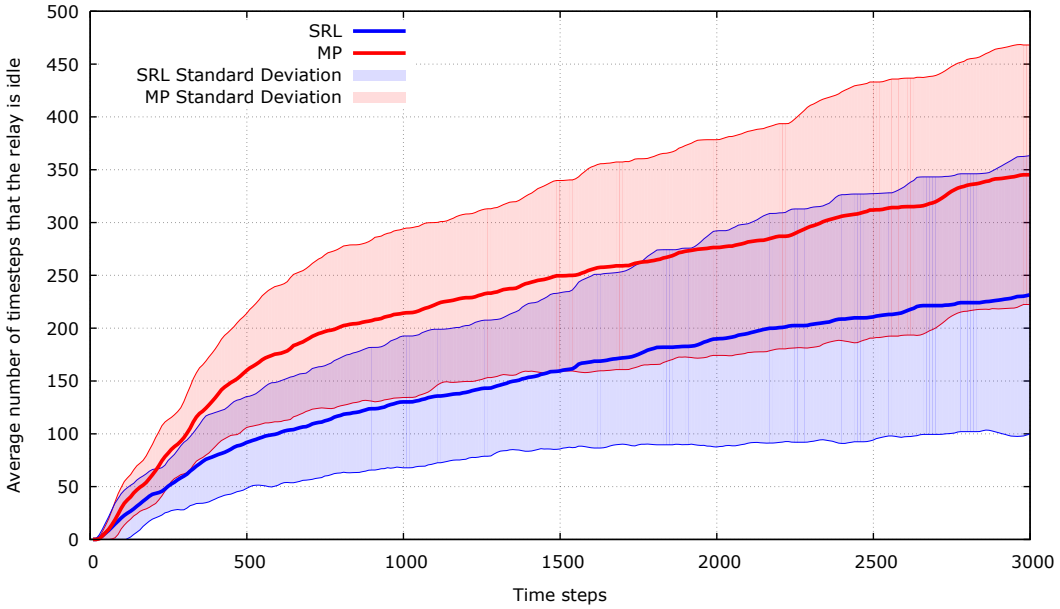


Figure 5.12: Average number of timesteps the relay has spent being idle up to the current timestep.

5.3 Relay exploration

From Fig. 5.12, it appears that with both strategies, relays spend almost 10% of their time idle. Is it possible to use that idle time to use relaying robots to explore more of the environment? To evaluate that, we implemented two new strategies SRL-RELEXP and MP-RELEXP. In these strategies, the explorer continues to behave as before (and so we do not impose any additional constraints on the exploration strategy used by the explorer). The behaviour of the relay, however, is changed and now follows the following steps:

1. When the relay meets its child, the explorer, the two agents agree on the time and location of the subsequent rendezvous. This step is the same as before.

2. After the relay exchanges information with its parent (in our case, the base station), it calculates the path to its rendezvous location.

3. By dividing the path length by its projected average speed, the relay can find out how long it will take for it to reach the RV location. If the relay expects to be at least N timesteps early (we used $N = 15$), then it may be able to take a detour and explore a frontier before heading to RV. Therefore, the relay enters the *Explore* state.

4. When in *Explore* state, the relay only evaluates frontiers that it can reach in time. To do that, it calculates the time it should take to reach the frontier, and the time it may take to reach either its own or the explorer's RV location from the frontier (it picks the shortest distance out of the two). If it can reach the frontier and then make the rendezvous in time, it considers the frontier in the standard frontier selection/allocation algorithm, otherwise it discards the frontier. If there are no frontiers to explore or all of them have been discarded, the relay enters the *GoToChild* state and heads back to RV. Otherwise it continues to explore the frontiers.

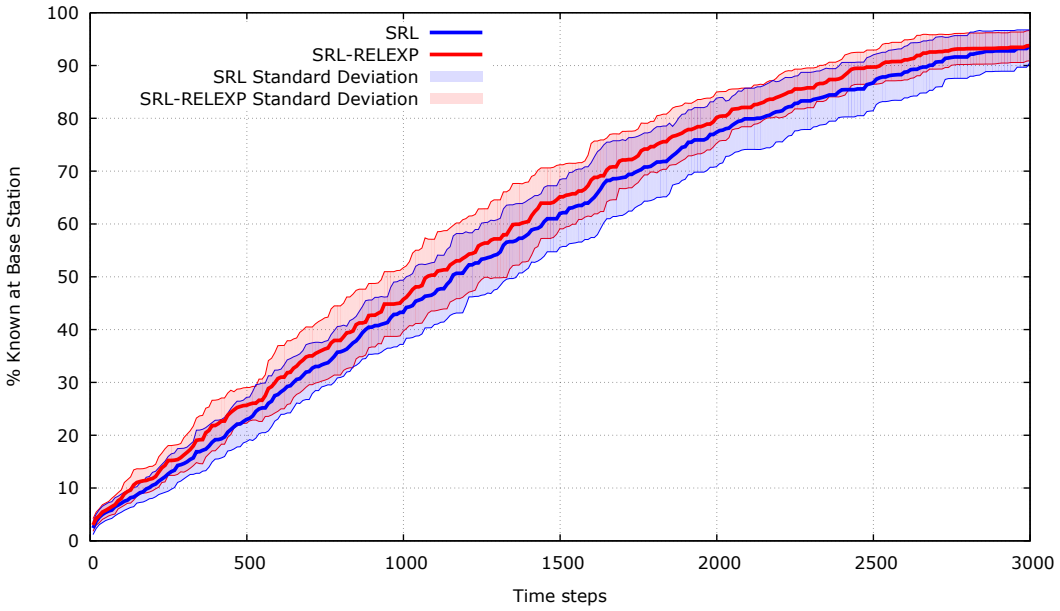


Figure 5.13: Average percentage of the environment known at base station for each timestep for SRL and SRL-RELEXP strategies.

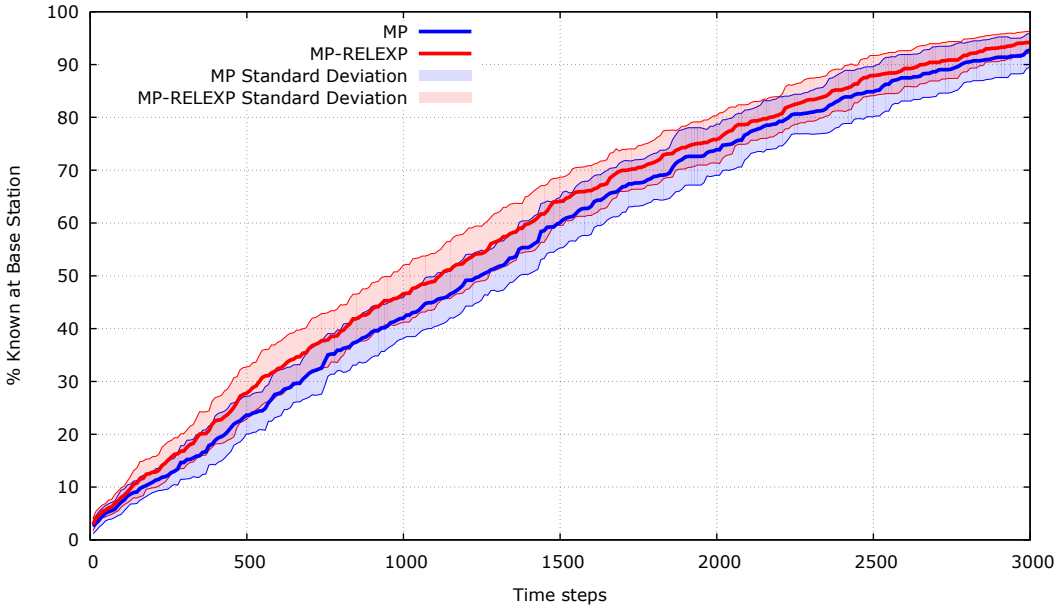


Figure 5.14: Average percentage of the environment known at base station for each timestep for MP and MP-RELEXP strategies.

As we can see from Figs. 5.13, 5.14 both strategies show similar minor improvements in base station knowledge at each timestep over the original ones. Most of the improvements come in the early stages of the exploration – after that, base station knowledge increases at a similar rate to that of the original approaches. At the start of the mission, the relay has a lot of "free time" to

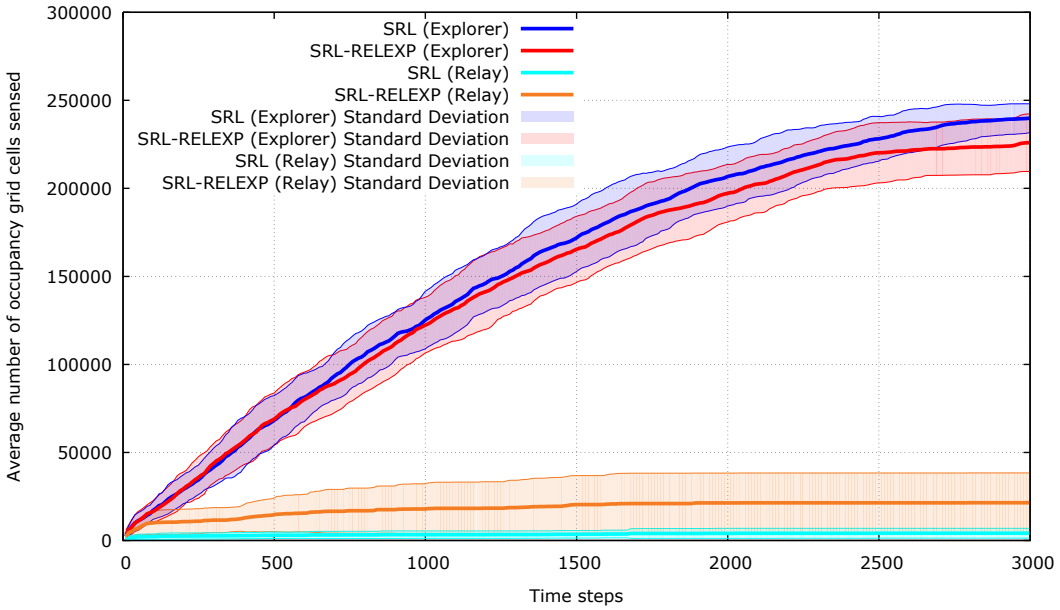


Figure 5.15: Average number of occupancy grid cells sensed by explorer and relay for strategies SRL and SRL-RELEXP.

explore frontiers as it either has to travel little or not at all to perform the relaying functions. At the same time, there are usually still some unexplored frontiers near the base station (and the location of the relay) that can be explored. At the later stages of the mission, the relay has little free time, and so the new strategies behave similarly to the original ones. This behaviour can be seen more clearly by looking at the graph of information sensed by explorer and relay separately, shown in Figs. 5.15, 5.16. We can see that in the RELEXP strategies, as expected, the relay ends up sensing a lot more area than in the original approach, and most of the difference is made in around the first 500 timesteps. The explorer ends up sensing less towards the end of the mission (due to some of the environment already having been sensed by the relay).

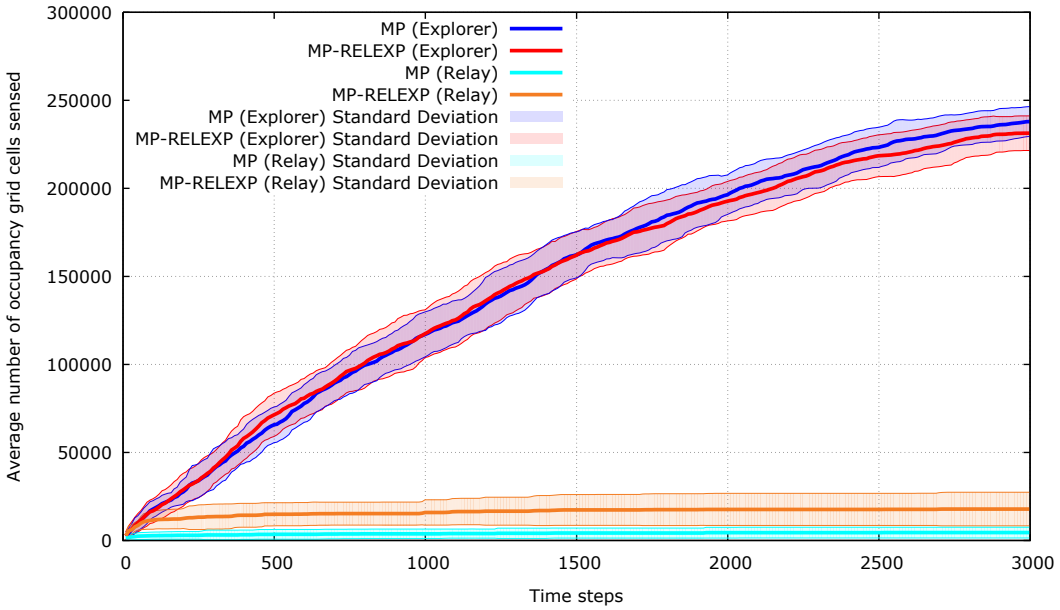


Figure 5.16: Average number of occupancy grid cells sensed by explorer and relay for strategies MP and MP-RELEXP.

5.3.1 Increasing exploration speed

So far we have managed to improve significantly on the communication latency between the agents and the base station by exploiting the communication range of agents in rendezvous planning, while also having a minor improvement in the overall rate of exploration by incorporating the relay in the exploration effort. In many search-and-rescue missions, however, the rate of exploration is the more important parameter. In this subsection, we attempt to improve the rate of exploration significantly with a tradeoff of communication latency. The goal is to have the same communication latency as that of the benchmark, and therefore free up the relay to do more exploration, leading to a significant improvement in the rate at which the environment is sensed. We define the MP-RELEXP-ST strategy to be the same as MP-RELEXP, with the following modifications:

1. When the relay meets with its explorer, the timing of the rendezvous is calculated in exactly the same way as in the benchmark approach, SRL. This is done to keep the frequency of rendezvous to be roughly the same

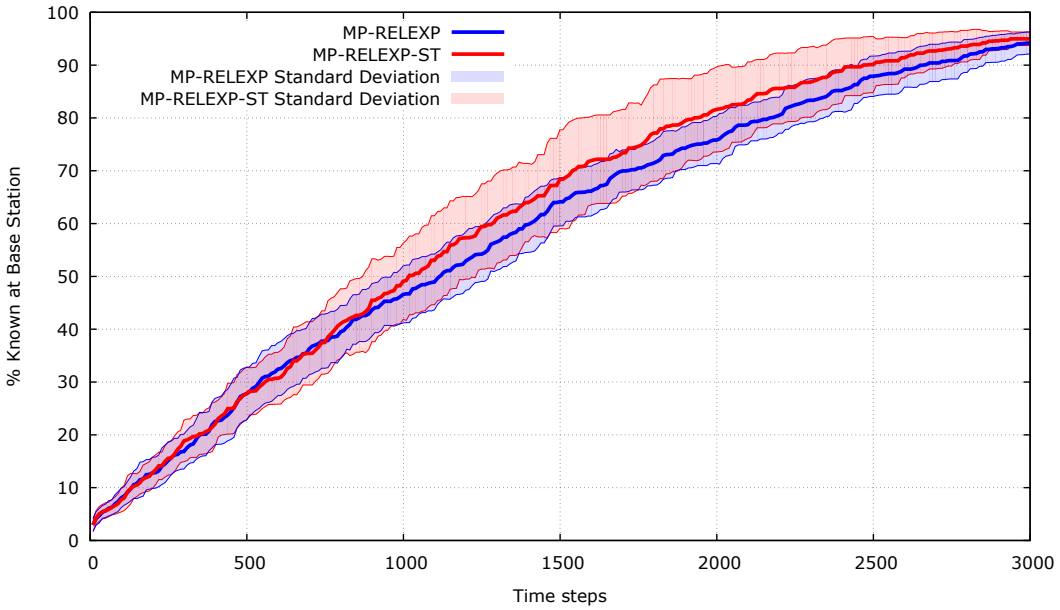


Figure 5.17: Average percentage of the environment known at base station for each timestep for MP-RELEXP and MP-RELEXP-ST strategies.

as in the benchmark approach.

2. Instead of choosing the relay RV location to be nearest to the communication range of the base station, the location is selected to be nearest to the frontier that the relay will plan on exploring (reachable in time).

3. If the relay is unable to reach any frontier in time, the relay RV location will be chosen to be the same as the explorer's RV location. In this case, the rendezvous process will be the exact same as in the benchmark SRL strategy.

As shown in Figs. 5.17, 5.18 MP-RELEXP-ST outperforms the benchmark at nearly every timestep when it comes to information available to the basestation. The main difference in performance with MP-RELEXP is that after the initial timesteps, MP-RELEXP-ST continues to accumulate information at the base station at an accelerated rate relative to MP-RELEXP. The average message latency is slightly worse than the benchmark, and significantly worse than MP-RELEXP, as shown in Fig. 5.19. The relay spends a very similar

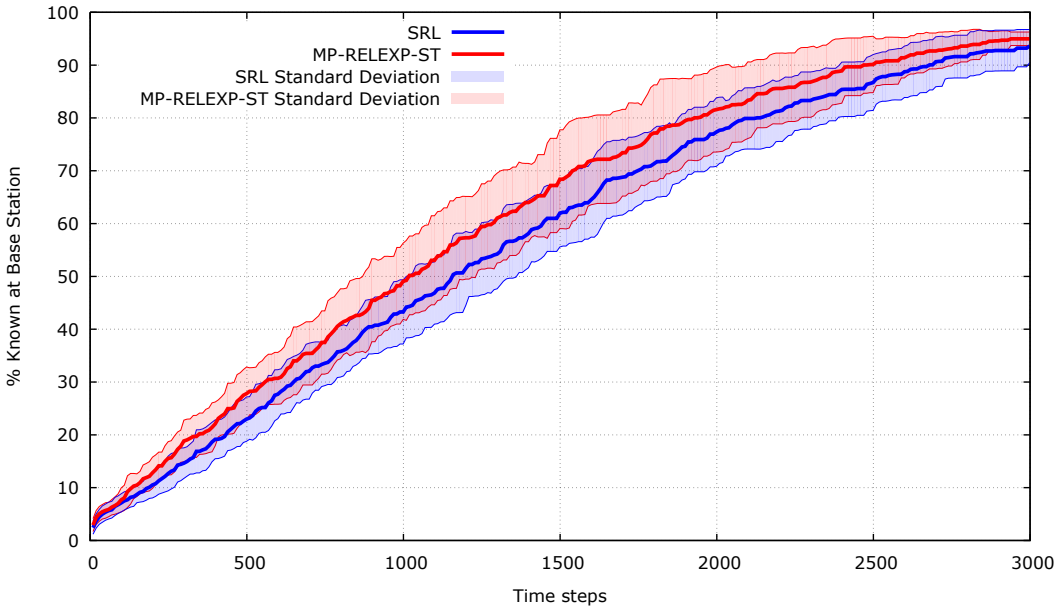


Figure 5.18: Average percentage of the environment known at base station for each timestep for MP-RELEXP-ST and the benchmark SRL strategy.

amount of time idle compared to MP-RELEXP, as can be seen in Fig. 5.20. Figs. 5.21, 5.22 show that after around step 500, the relay gathers more information compared to other approaches, with the corresponding decrease in the amount of information sensed by the explorer towards the end of the mission.

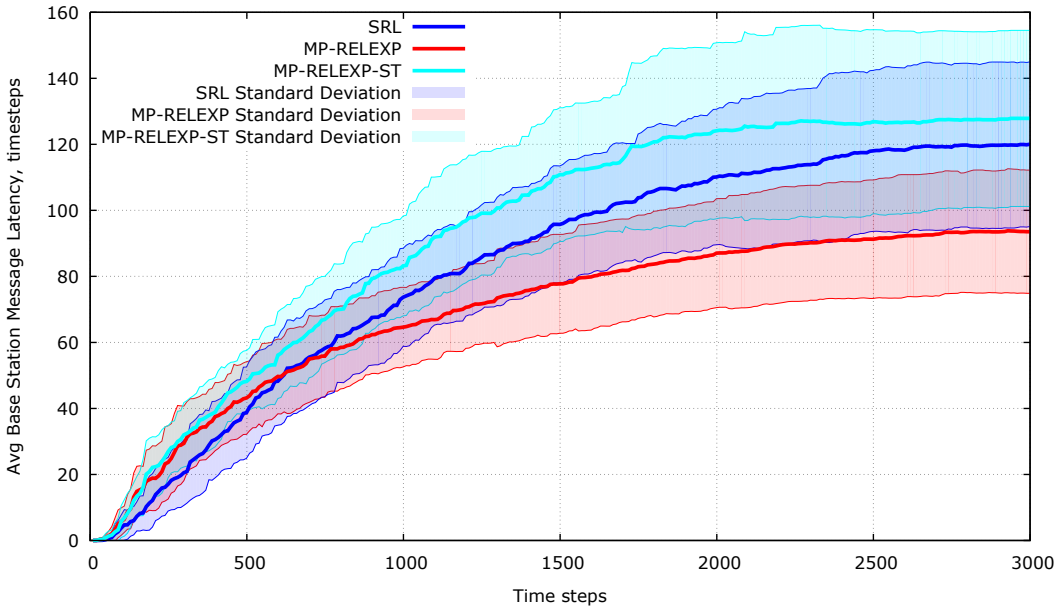


Figure 5.19: Average latency in delivering messages from the base station to all the agents at each timestep.

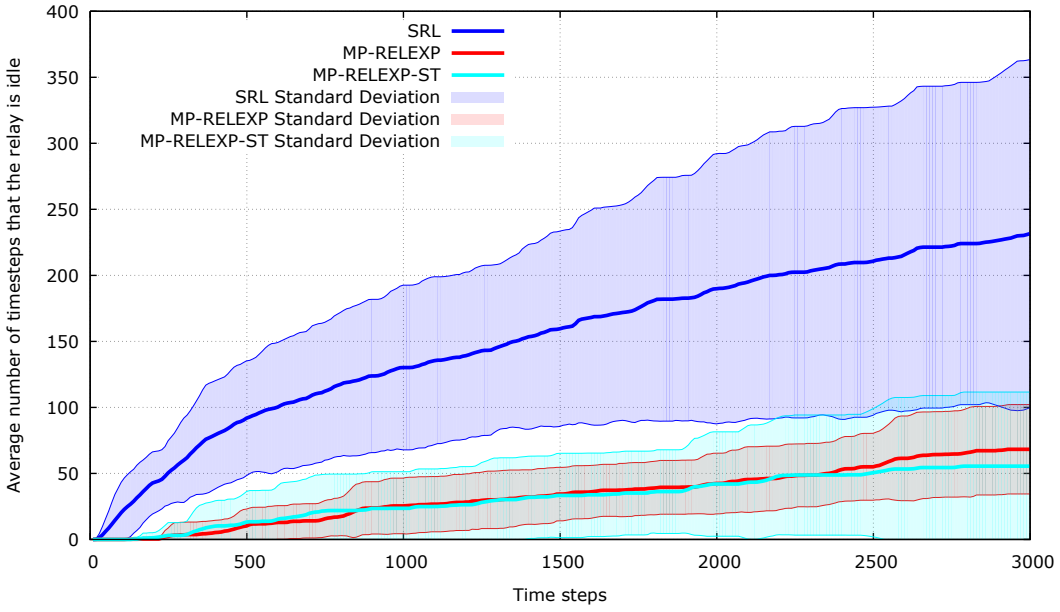


Figure 5.20: Average number of timesteps the relay has spent being idle up to the current timestep.

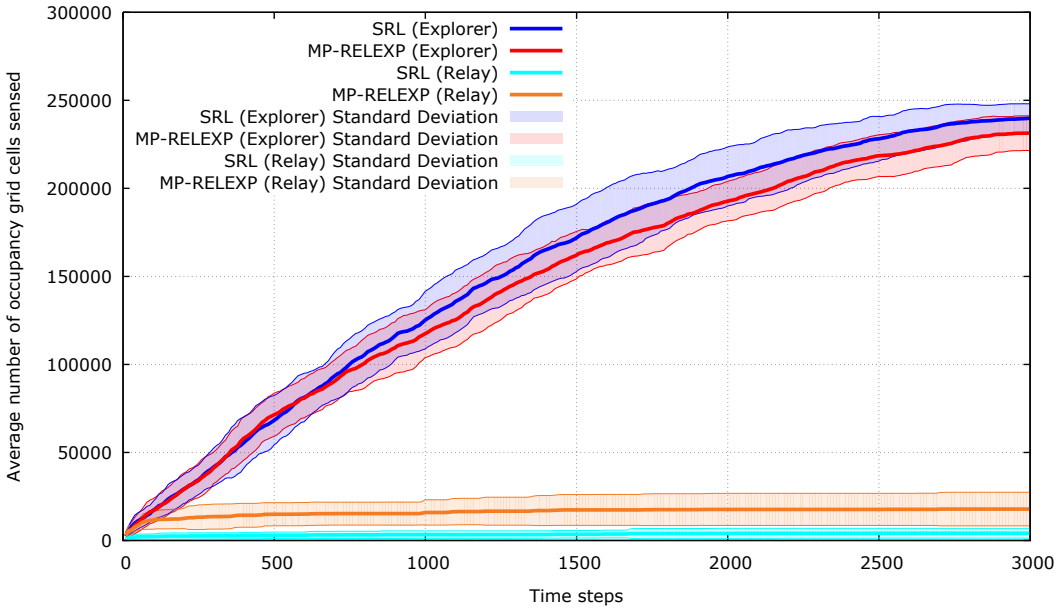


Figure 5.21: Average number of occupancy grid cells sensed by explorer and relay for MP-RELEXP and SRL.

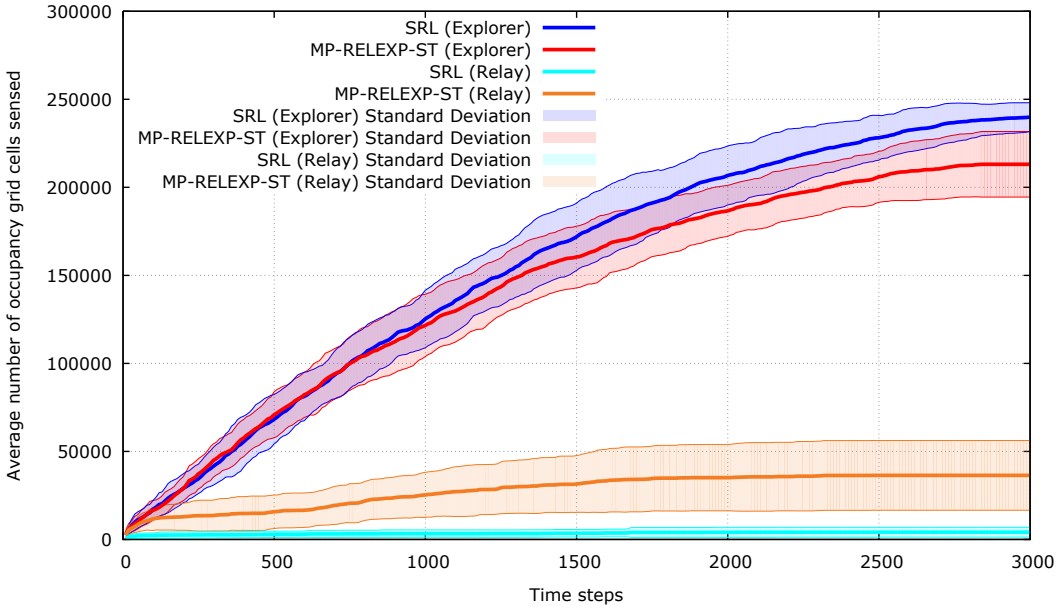


Figure 5.22: Average number of occupancy grid cells sensed by explorer and relay for MP-RELEXP-ST and SRL.

5.4 Scalability

So far in this chapter, we have been discussing the approaches for arranging rendezvous between an explorer and a relay, as well as evaluating their performance when the team consists of only 2 agents. These approaches can be easily scaled up to larger teams consisting of $2N$ agents, where N relay-explorer pairs are formed before the start of the mission. The agents and the base station then form a tree hierarchy, 3 levels deep, with the base station at the root of the tree, the N relays as its children, and each relay having an explorer as their child, forming the leaves of the tree (see Fig. 5.23). Agents can occasionally swap positions in the tree, including between different branches of the tree, if this leads to decreased travel time, according to the role-swap rule [31]. Each relay-explorer pair can then plan their rendezvous independently as described earlier in this chapter. Explorers can still interact with each other to allocate frontiers among themselves, according to the exploration strategy used. Approaches described in this chapter can also be adapted to other tree structures, such as longer chains of relays or multiple explorers served by a single relay, in a relatively straightforward way.

We evaluated the MP-RELEXP and MP-RELEXP-ST approaches introduced in this chapter using the same environment as before, and benchmarked them against the SRL approach. We randomly selected 20 starting locations from areas shown in Fig. 5.4, and ran simulations with 2, 4 and 6 agents. The resulting speed of exploration is shown in Fig. 5.24. From Fig. 5.24a we can see that, as expected, the more agents we have in our team, the faster they are able to explore the environment. It seems that the rate of improvement in the speed of exploration diminishes as we increase the number of agents. Furthermore, the improvement from our approach MP-RELEXP-ST compared to the benchmark SRL appears to get smaller as the number of agents is increased. Fig. 5.24b shows the same effect more clearly. The likely explanation for both of these observations is that as the biggest frontiers are sensed by

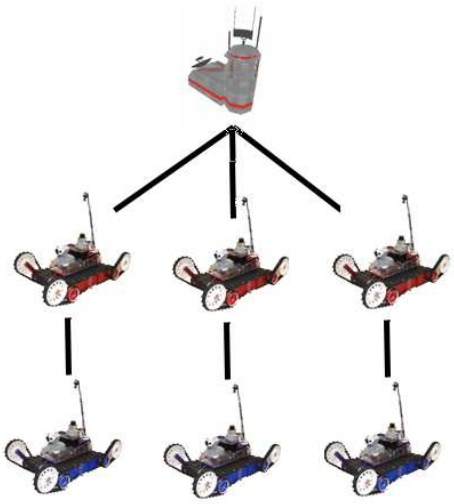
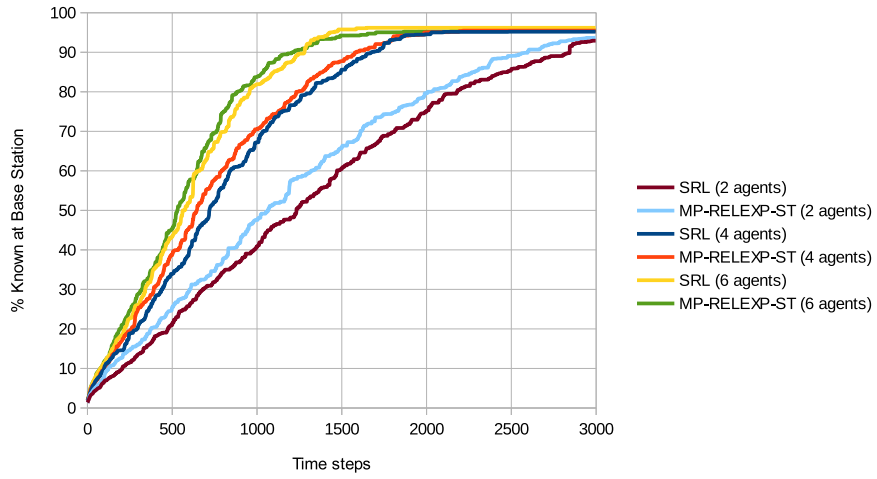


Figure 5.23: Tree hierarchy used for a team of 6 mobile robots (plus a static base station). Base station is the root of the tree, relays are red middle nodes, explores are blue leaves. Based on the image in [31]

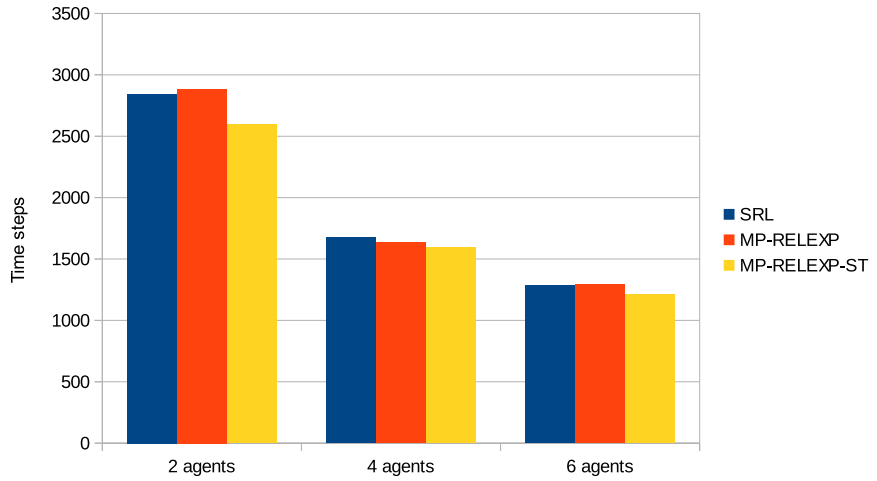
some of the explorers, there are few major frontiers for other agents to explore, including the relays when using the MP-RELEXP-ST strategy, which leads to the diminishing returns when more agents attempt to contribute by exploring the environment.

It would therefore appear that as the number of agents in the team is increased, it could be more effective to use the additional resources to improve team communication. As a reminder, in MP-RELEXP, we exploit the communication ranges of the agents in the team to increase the frequency of rendezvous. In MP-RELEXP-ST we reduce the frequency of rendezvous in order to give relays a chance to explore nearby frontiers. Fig. 5.26 shows the maximum number of timesteps, averaged across 20 simulation runs, that it takes for a message from the base station to reach all agents in the team. As expected, using the MP-RELEXP approach leads to significantly decreased message latency for runs with 2, 4 and 6 agents. These results suggest that as the number of explorers in the team is increased, MP-RELEXP should usually be preferred to MP-RELEXP-ST.

The average results for 20 starting locations selected from the "good" areas



(a) Average percentage of the environment known at base station for each timestep. Figures for each timestep are averaged across 20 runs with random starting locations for each approach evaluated.



(b) Number of timesteps after which the base station has on average information about more than 90% of the environment.

Figure 5.24: Speed of exploration for SRL, MP-RELEXP, MP-RELEXP-ST approaches for 2, 4 and 6 agents.

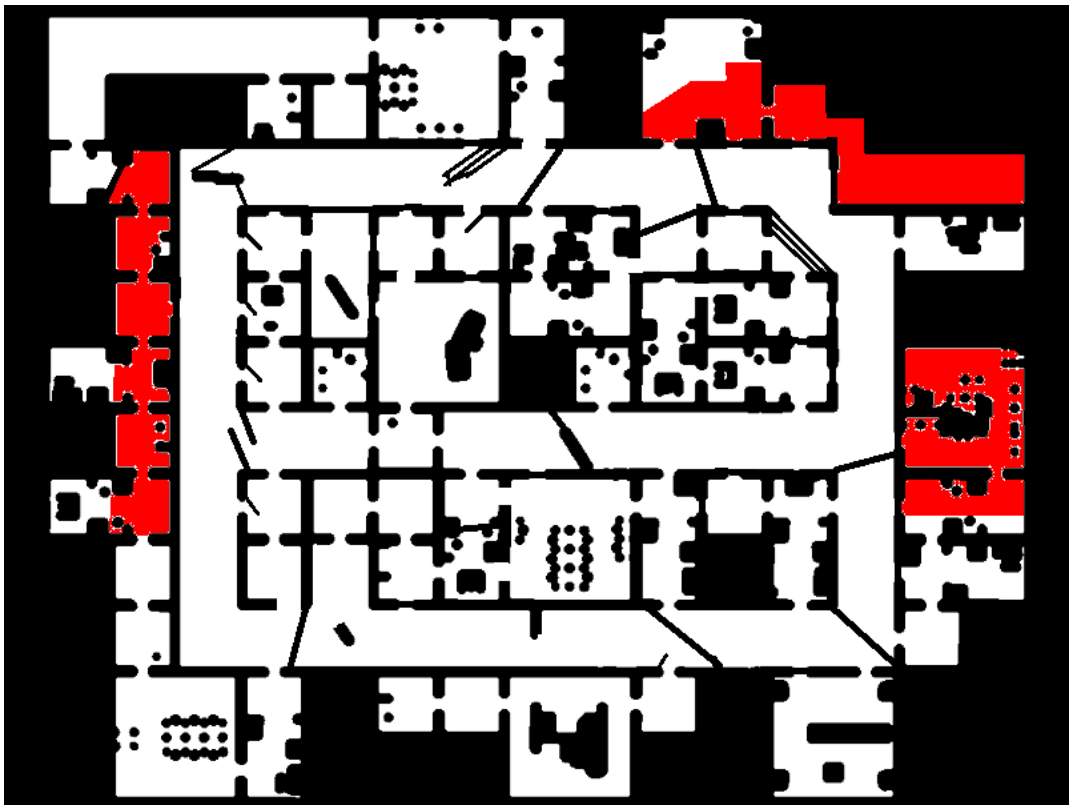


Figure 5.25: Starting locations where MP-RELEXP-ST performs particularly well are selected in the areas shown in red.

in Fig. 5.25 are shown in Figs. 5.28, 5.27. The effects described above still hold, but are emphasized more. It is interesting to observe that MP-RELEXP-ST with 4 agents in the team performs almost as well as the benchmark SRL approach with 6 agents.

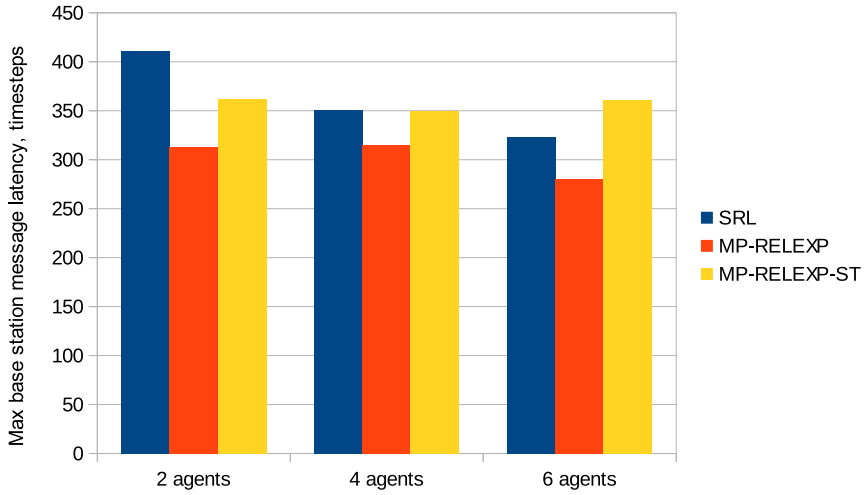


Figure 5.26: Maximum number of timesteps, averaged across 20 simulation runs, that it takes for a message from the base station to reach all agents in the team.

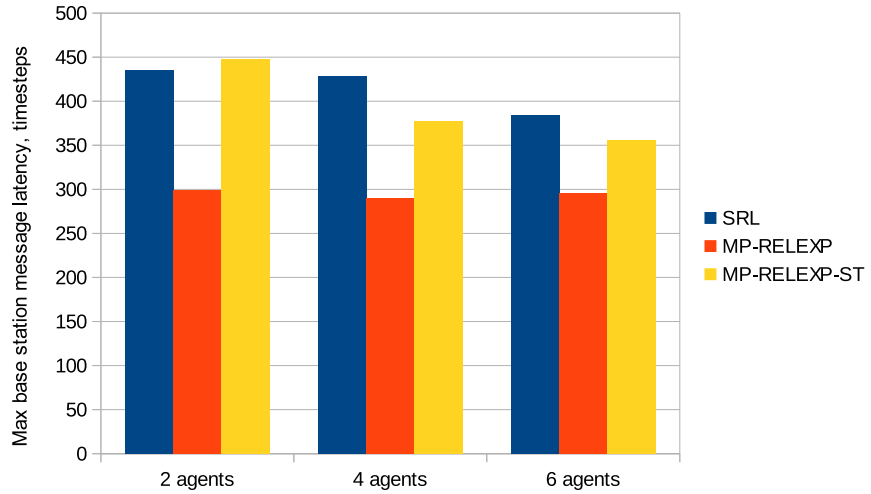
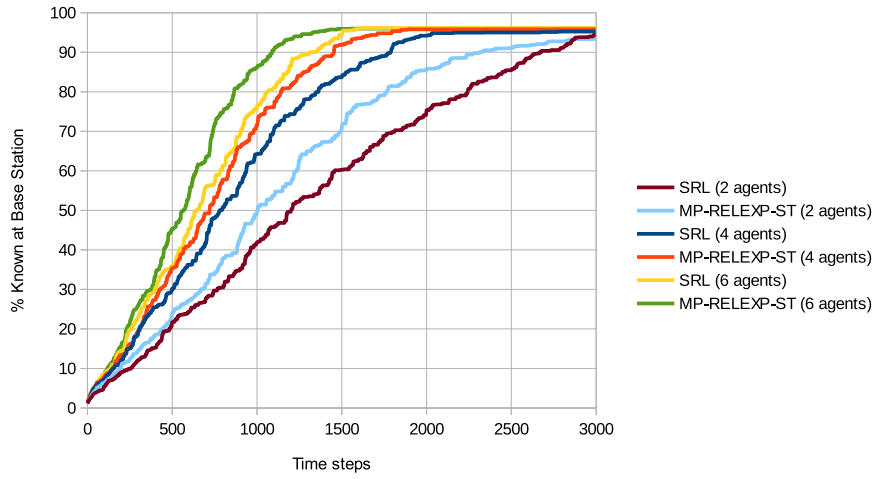
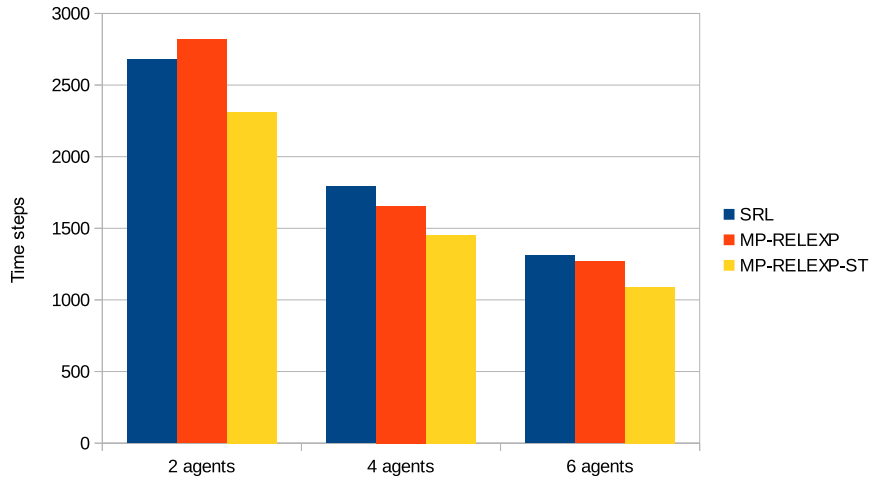


Figure 5.27: Maximum number of timesteps, averaged across 20 simulation runs for the "good" starting locations, that it takes for a message from the base station to reach all agents in the team.



(a) Average percentage of the environment known at base station for each timestep. Figures for each timestep are averaged across 20 runs with random starting locations for each approach evaluated.



(b) Number of timesteps after which the base station has on average information about more than 90% of the environment.

Figure 5.28: Speed of exploration for SRL, MP-RELEXP, MP-RELEXP-ST approaches for 2, 4 and 6 agents for the "good" starting locations.

5.5 Discussion

5.5.1 Estimating agent communication range

The process of selecting good rendezvous tuples depends on the agents having a good model for predicting their communication range at the potential rendezvous locations. If the model used is too conservative, the agents will be unable to exploit their communication range to improve team performance, when compared to using a single rendezvous location (which we use as a benchmark in this chapter). It is important to note that in this case using rendezvous tuples will result in a performance very similar to that of the benchmark, as the rendezvous tuples will simply collapse to either a single point or two points with line-of-sight between them. If the communication model is selected to be too optimistic, agents may end up selecting rendezvous tuples with no communication available between the locations of the agents. If there is no line-of-sight between the locations, the agents will have no way of knowing *why* there is no communication when they arrive at the rendezvous. For a discussion of the failure modes, see section 5.5.2. The more our model overestimates the real communication ranges of the agents, the more often the rendezvous will fail.

In this chapter we use a path loss model with wall attenuation factor (WAF) originally proposed by Bahl and Padmanabhan [11]. This model is also known as the RADAR model. This is the communication model that is used by the Wireless Simulation Server (WSS) [85] to simulate wireless communication between robots in USARSim, a high-fidelity simulator that had been used in the RoboCup Rescue Virtual Robots league between 2007 and 2015 [12]. An equivalent of WSS is going to be used in the league with a different simulation environment Gazebo from 2016 [7]. In this model, communication strength is calculated as follows:

$$S = P_{d_0} - 10 \times N \times log_{10}\frac{d_m}{d_0} \times \min(nW, C) \times WAF$$

where P_{d_0} is the signal strength at reference distance d_0, N is the rate of

Range	Materials	Loss (dB)
Low	Non tinted glass, Wooden doors, Cinder block walls, Plaster	$2-4$
Medium	Brick wall, Marble, Wire mesh, Metal tinted glass	$5-8$
High	Concrete wall, Paper, Ceramic bullet-proof glass	$10-15$
Very high	Metals, Silvering (Mirrors)	>15

Table 5.1: Typical attenuation for building materials at 2.4GHz

path loss, d_m is the distance, nW is the number of obstructing walls, WAF is the wall attenuation factor and C is the maximum number of walls where the attenuation factor needs to be considered. This model has been shown to have good accuracy in indoor environments [25]. Approaches described in this chapter do not depend on the use of any particular communication model, however.

Parameters P_{d_0}, d_0, N, d_m and C can be estimated for a particular loadout of the robots before the rescue mission. However, the wall attenuation factor WAF can be highly variable between different environments (or even within one environment) so it can be difficult to get a good estimate of it before the start of the mission. Prior knowledge about the materials used for the walls and floors of the building as well as for other obstacles that the agents might encounter in the environment might help to get an initial guess of what that value should be. Table 5.1 shows typical attenuation values for some building materials at 2.4GHz [98].

We can then obtain empirical observations of the attenuation factors of the obstacles during the actual exploration mission as follows. When agents are communicating with each other, they are able to measure the signal strength between them. From the occupancy grid map, they will be able to estimate the number of obstacles between them. Knowing the parameters P_{d_0}, d_0, N, d_m and C, they are then able to estimate WAF. As shown in Fig. 5.29, distance

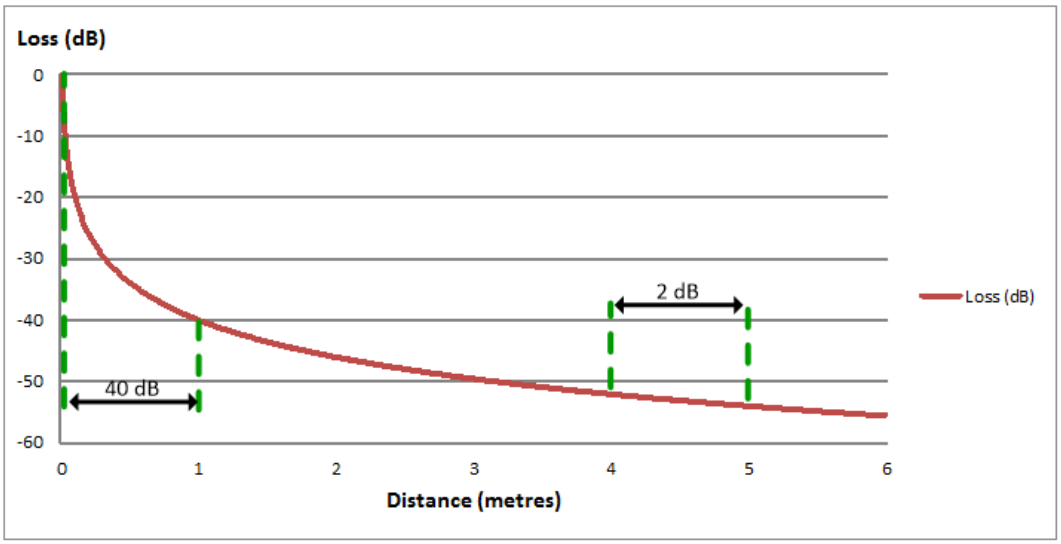


Figure 5.29: Variation in rate of loss over distance (taken from [17])

between the agents has a much greater effect on loss when the distance is small. Therefore, to make the estimate more robust to distance estimation errors, for example due to small localization errors, we only obtain empirical observations when the distance between the agents is greater than 4 meters. In a simple pessimistic model, agents then use the maximal WAF empirically observed during the mission in subsequent range estimations (multiplied by a fudge factor greater than 1 if we want to be even more conservative – we used the factor of 1.1 in this chapter. We leave the empirical studies of the appropriate fudge factor for particular environment types, agent configurations and payloads as future work). Agents also share their observed WAF with other agents during communication.

More interesting methods may include taking into account that different types of obstacles in the environment are likely to have different attenuation factors. Topological data obtained from the occupancy grid, or even raw laser measurements may then be used to infer the appropriate type of some of the obstacles in the environment (load-bearing walls, thin walls, debris and others), and assign appropriate observed WAFs to different obstacles. These estimated attenuation factors could then be stored in the occupancy grid map and used

for signal strength estimation. Furthermore, in some cases the measured attenuation of the obstacles may be valuable in its own right – it might provide the human responders with additional information about the materials and types of obstacles present in the environment. In less structured environments, such as heavily collapsed buildings, such data may be less reliable and the simple pessimistic model described above with WAF assumed homogeneous throughout the environment could possibly perform better than the more complicated approaches.

5.5.2 Failure modes

It is possible that when an agent arrives at the rendezvous location at the pre-agreed time, it is not able to establish communication with its partner. Here we consider two such scenarios – where the agreed rendezvous locations for the two agents had line-of-sight between them and when they did not due to the presence of obstacles between the corresponding locations. If there is line of sight between the two locations, the agent arriving at the rendezvous can improve the chances of communication with its partner by navigating towards the rendezvous location of the partner, as shown in Fig. 5.2. If, after arriving at the rendezvous location of the partner, the agent is still unable to establish communication, it can wait for a designated amount of time in case the partner is late and then either navigate to a backup rendezvous location, continue the exploration mission independently, or head towards the base station.

In case of non-line-of-sight rendezvous scenario, navigating towards the area of greater probability of communication may not be easily possible, as the agent may be stuck in the local maxima with regards to the probability of communication with its partner (see Fig. 5.2). In this case, the following reasons for not being able to establish communication need to be considered:

1. It is possible that the rendezvous partner is simply late, in which case the best course of action could be to wait some more.

2. It could be that the other agent was disabled, in which case we should cancel the rendezvous and re-evaluate the agent hierarchy.

3. While we make the assumption of the environment being static in this thesis, in practice this is often not the case. In a dynamic environment, paths could get blocked, in which case our partner may not be able to make the rendezvous (or may be late, as in case 1). In this situation, we may attempt to meet at a backup location, or, if the agents become irrecoverably separated, re-evaluate the agent hierarchy.

4. If both agents are actually present at their respective rendezvous positions, but they are not able to establish communication due to being outside of the communication ranges of each other, they should proceed to a backup rendezvous location where communication is more likely.

The first three situations described above could happen even when a single rendezvous location is used (and so do not depend on our choice of communication range modelling) [32]. Using rendezvous tuples introduces an additional mode of failure described in the last point above. As the individual agents will not in general be able to distinguish between the four modes of failure, we assume that the agents respond in a way that is consistent with all of them. That is, they should wait for a pre-agreed timeout interval; after that, they should attempt to navigate to a pre-agreed backup rendezvous location where communication is more likely. If the rendezvous at the backup location fails as well, they should re-evaluate the agent hierarchy (by either heading to the base station for new instructions, or changing their role to that of an explorer with the base station as their parent).

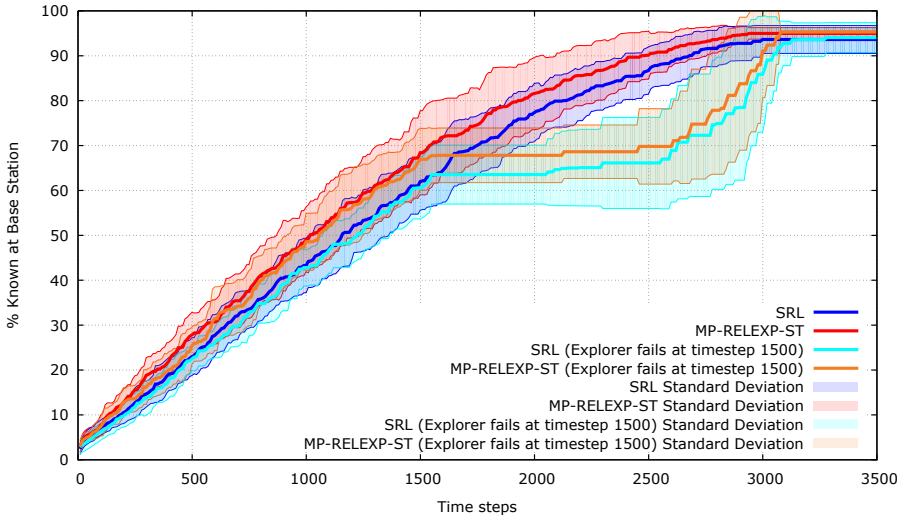
5.5.2.1 Evaluation in simulation

In order to evaluate the potential impact that agent failure can have on the proposed exploration strategies, we ran simulations where two agents explore

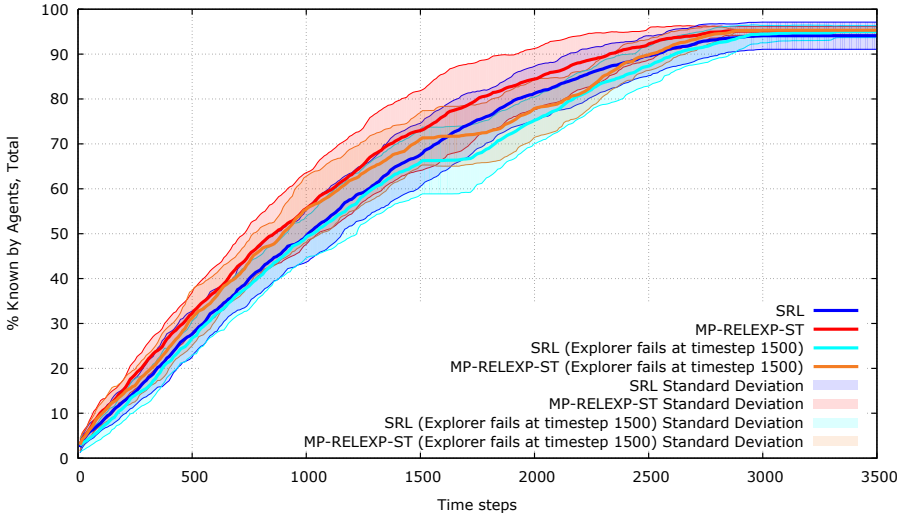
the environment using the MP-RELEXP-ST and the benchmark SRL strategies. In each simulation run, we had one agent fail at timestep 1500, which is around the middle of the mission. We assume that a failing agent is not able to navigate the environment or communicate with its teammates. In all cases we used the recovery procedure described in section 5.5.2 – when using the SRL strategy, the agent arriving at rendezvous waits for its partner for 60 timesteps. With the MP-RELEXP-ST strategy, after waiting for its partner for 60 timesteps, the agent heads to the backup rendezvous location, where it waits for a further 60 timesteps. After that in both strategies the agent finishes exploring the environment using greedy frontier exploration before returning to base.

The effects of the explorer failing are shown in Fig. 5.30. As expected, when using the multi-point rendezvous location, the recovery takes slightly longer, as the relay has to wait for 60 timesteps at the original location, then go to the backup location and wait for an additional 60 timesteps. After that the agent can assume that the explorer has failed and proceed to explore the environment using greedy frontier exploration. Rendezvous in the benchmark SRL strategy guarantees line-of-sight in static environments, and as such the relay can enter exploration mode without the need to head to the backup location. Note that in this scenario the relay has to re-explore the areas that have already been explored by the explorer that failed.

The behaviour is similar when the relay fails instead. However, the team has a slightly faster recovery as can be seen in Fig. 5.31, as the relay would generally have less new information at the time of failure compared to a failing explorer, and therefore the failure may have a lower impact on the overall team performance.

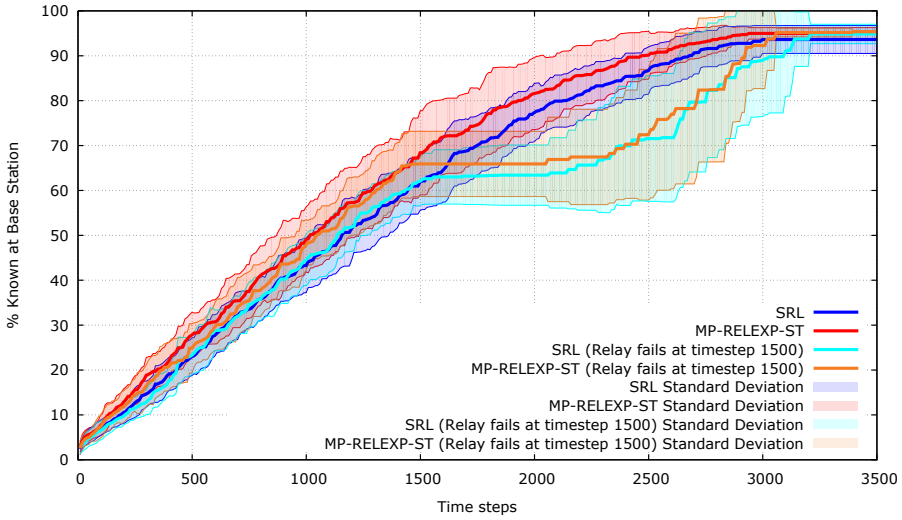


(a) Average percentage of the environment known at base station for each timestep. Figures for each timestep are averaged across 25 runs with random starting locations for each scenario evaluated.

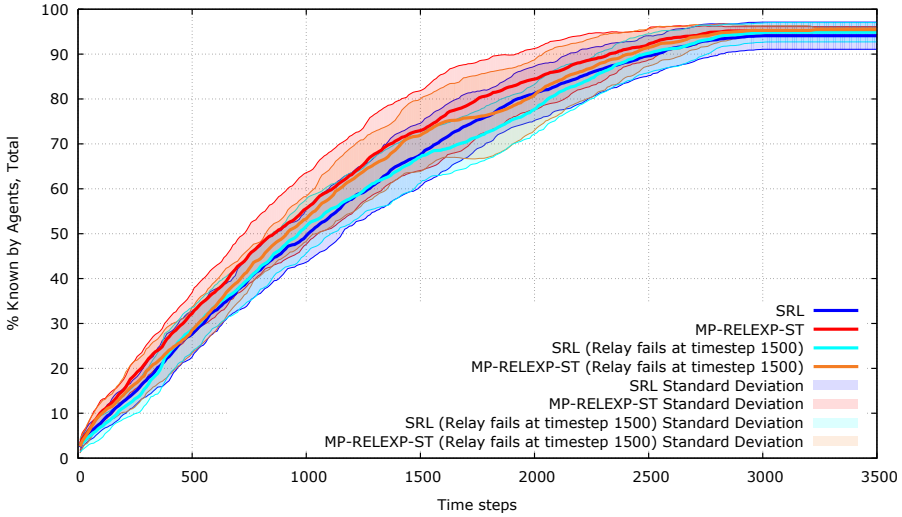


(b) Average percentage of the environment known by the overall team for each timestep. Figures for each timestep are averaged across 25 runs with random starting locations for each scenario evaluated.

Figure 5.30: Effect of the exploring agent failing at timestep 1500 on the speed of exploration when using the SRL and MP-RELEXP-ST strategies.



(a) Average percentage of the environment known at base station for each timestep. Figures for each timestep are averaged across 25 runs with random starting locations for each scenario evaluated.



(b) Average percentage of the environment known by the overall team for each timestep. Figures for each timestep are averaged across 25 runs with random starting locations for each scenario evaluated.

Figure 5.31: Effect of the relay agent failing at timestep 1500 on the speed of exploration when using the SRL and MP-RELEXP-ST strategies

5.5.3 Sampling point density

What effect does the density of sampled points have on the MP-RELEXP-ST approach? In order to evaluate that, we randomly chose 25 starting locations in areas of the map where MP-RELEXP-ST strategy gives the biggest improvements. These areas are shown in Fig. 5.25. We then ran simulations with 3 different sampling densities: on average one point per each 20x20, 70x70 and 120x120 occupancy grid squares (1 sampled point on average per $1.6\,m^2$, $19.6\,m^2$, $57.7\,m^2$). In order to benchmark the performance, we also ran simulations using the SRL-RELEXP approach, which is similar to what MP-RELEXP-ST would perform as if too few points are sampled.

Fig. 5.33 shows how the performance of the MP-RELEXP-ST strategy degrades and approaches that of SRL-RELEXP as we reduce the number of sampled points. The breakdown of area explored by explorer and relay is shown in Fig. 5.34. We can see that as the number of sampled points decreases, the relay is less effective as an explorer which is what leads to the degradation in overall performance shown in Fig. 5.33.

To evaluate how the number of sampled points affects the performance of the algorithms, we randomly positioned the explorer, relay, as well as 5 frontiers of random utility on the free space of the full map, using the same map we have used so far in our evaluations; then, we ran the MP-RELEXP-ST rendezvous location selection algorithm in MRESim on the chosen configuration. The full map contains 261419 free occupancy grid cells. We varied the number of points sampled; for each number of sampled points we repeated the process 25 times on an Intel i7 CPU @ 2.2 GHz, Windows 7 laptop, noting the average times to select the rendezvous locations. When sampled one point per each 20x20, 70x70, 120x120 occupancy grid squares, the numbers of points sampled were 653, 53, 18, and the average processing times were 1008 ms, 439 ms, 407 ms. The SRL rendezvous selection process took 322 ms on average (note that it had to be performed as part of each MP-RELEXP-ST selection process, contributing

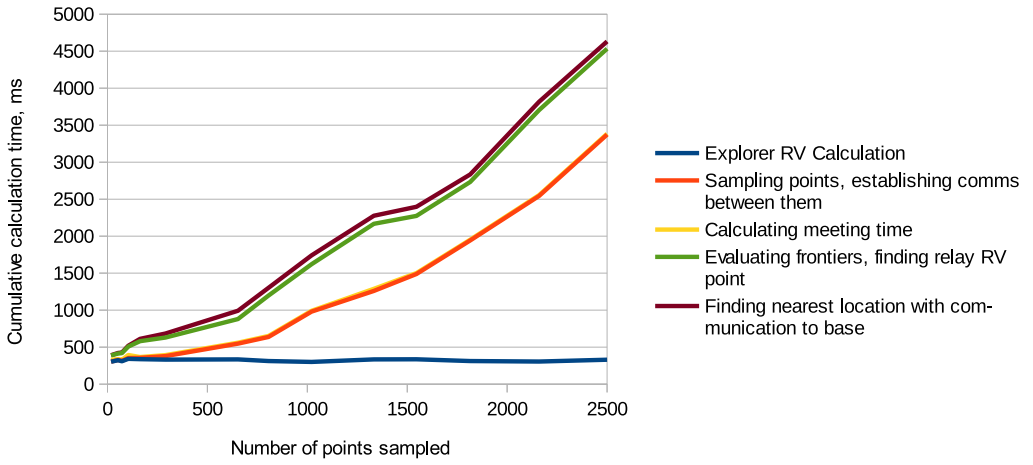


Figure 5.32: Cumulative time per number of points sampled for calculating the rendezvous location using the MP-RELEXP-ST algorithm on an occupancy grid with 261419 free cells.

to the overall processing times). The breakdown of average time spent on each part of the MP-RELEXP-ST algorithm and the relationship of the running time to the number of points sampled is shown in Fig. 5.32.

Given that, with a small number of points sampled, MP-RELEXP-ST behaves similarly to the benchmark, with performance improving as the number of sampled points increases, and that the actual time available to calculate the new rendezvous location could be limited, it could be beneficial to use the MP-RELEXP-ST algorithm as a "just-in-time" method. This way, an agent could start by calculating the new RV location by sampling a small number of points from the environment. Then, more points could be iteratively sampled, improving the resulting generated rendezvous location, until a time limit is reached.

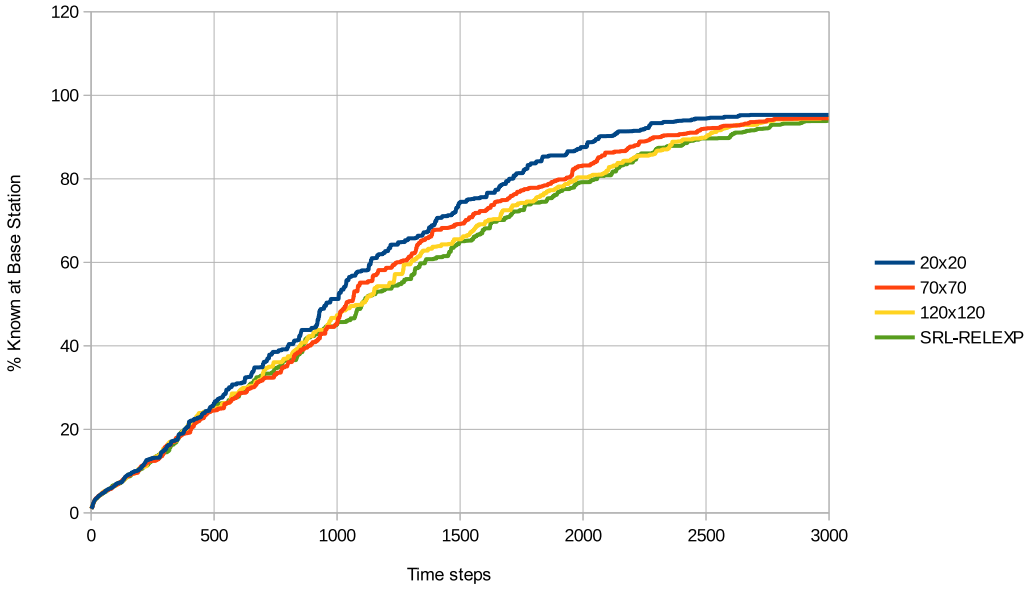


Figure 5.33: Average percentage of the environment known at base station for each timestep in the mission, averaged across 25 runs, for varying number of sampled points.

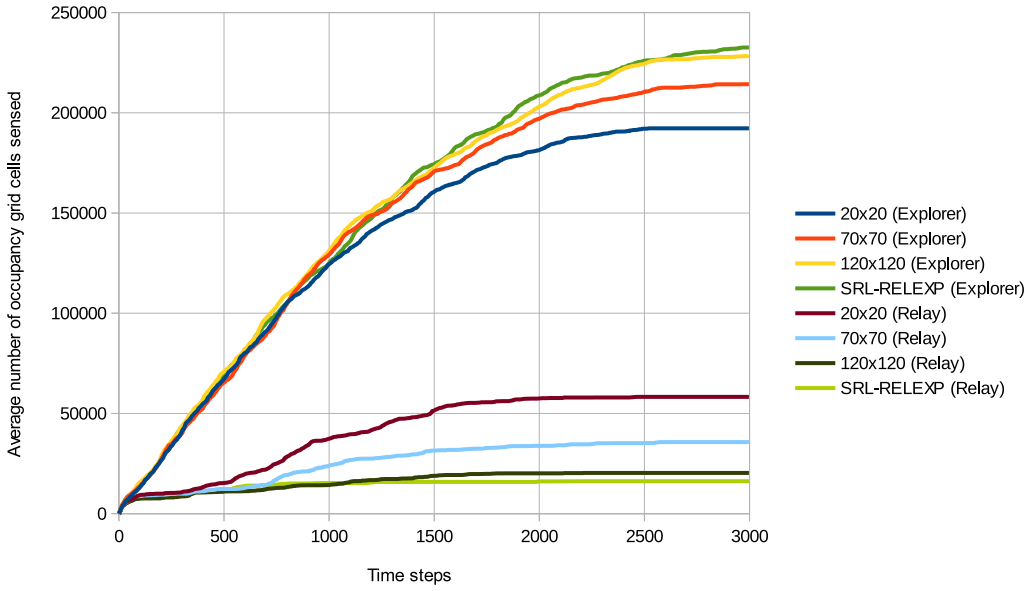


Figure 5.34: Average number of occupancy grid cells sensed by explorer and relay for each timestep in the mission, averaged across 25 runs, for varying number of sampled points.

5.5.4 Methodology

In this chapter, we evaluated the proposed approaches against the benchmark on a single map, shown in Fig. 4.26. The map was selected specifically because its layout was expected to offer significant benefits to team performance if team members were able to plan their communication through obstacles in a sensible way. The goal of the simulations was then to establish that the proposed approaches were able to take advantage of such favourable environment layout, as well as to investigate the potential trade-offs between exploration speed and communication latency that arise from using the proposed approaches. In other types of environments, such as environments consisting of large open spaces or rubble that is easy to navigate around, arranging rendezvous through obstacles would not yield significant benefits for the team. That would mean that the proposed approaches would plan single-location rendezvous and as such behave very similarly to the benchmark as the selected rendezvous locations would be nearly identical. Therefore, we believe that the simulations performed on a single map in this chapter are sufficient to evaluate whether the proposed approaches are able to take advantage of favourable environment layouts and to justify the qualitative results presented in this chapter.

Before the approaches proposed in this chapter are used in real search-and-rescue situations, it would be necessary to perform experiments in environments that are representative of the types of environments where the robots are likely to be used. Such experiments will need to be conducted both in simulation and in the real world. That would require for the appropriate test datasets and standardized benchmarks to be created, similar to the smaller scale standardized arenas used in the Rescue Robot competitions of the RoboCup Rescue League [110]. Such benchmarks will require highly realistic wireless communication models that include models of signal attenuation from obstacles that are consistent with what robots may encounter in a real operating environment.

5.5.5 Applicability

We have shown that the approaches proposed in this chapter, MP-RELEXP and MP-RELEXP-ST, are efficient in exploiting the topology of the environment and the communication ranges of the agents in order to improve the speed of exploration and communication within the team. While in our evaluations we considered primarily the advantages of communicating through obstacles, the approaches can be easily extended to other scenarios. In particular, in the simulated experiments we assumed that the environment can be represented as "free space" and "obstacles", where agents can navigate at a uniform speed through free space and cannot navigate through obstacles. In real search-and-rescue environments, terrain can have various degrees of traversability. The methods in this chapter could then be used to arrange rendezvous in such a way as to avoid navigating through areas of poor traversability (the most straightforward way would be to simply treat such areas as obstacles while planning rendezvous).

Simulated experiments suggest that with few agents in the team, MP-RELEXP-ST should be preferred to MP-RELEXP as it results in faster exploration of the environment and the delivery of information to the base station. With higher numbers of agents the improvement in the speed of exploration becomes smaller, and so MP-RELEXP should be preferred as it offers improved team communication compared to other approaches considered. However, extensive experiments in real physical environments with similar communication constraints as may be encountered in real missions are required before the system can be deployed in the real world.

Compared to the opportunistic rendezvous approaches discussed in chapter 4, the methods proposed in this chapter allow agents to take advantage of the topology of the environment and the communication ranges of agents to reduce agent travel times and improve communication within the team. Additionally, pre-arranged rendezvous gives greater control and awareness of the team to

human operators with the behaviour of the team being more predictable. In particular, the behaviour of the team is less affected by the specific frontier allocation strategies used by the exploring agents as the agents will still aim to meet at the designated locations at pre-agreed times that will be communicated to the base station by the relays. On the other hand, opportunistic rendezvous approaches offer greater flexibility of the team in terms of allocating the resources between communication and exploration and improved robustness of the team due to there being fewer potential points of failure. A combination of the two approaches is possible – for example, agents using opportunistic rendezvous can use the methods described in this chapter to select improved rendezvous locations for communicating with the base station. We leave the in-depth consideration of such combined approaches as future work.

Chapter 6

Conclusions

This chapter gives a summary of contributions in this thesis, followed by directions for future work.

6.1 Summary of contributions

The problem we intended to address in this thesis was this: how should a homogeneous team of robots explore an unknown indoor environment where high bandwidth communication has a very limited range, while maintaining periodic communication with a static base station? We identified the inherent trade-off between the speed of exploration and the provision of communication within the team and proposed two classes of approaches to address this problem: using opportunistic rendezvous to guide the team behaviour and explicitly arranging rendezvous locations.

We proposed an opportunistic approach where the allocation of team resources between exploration and communication, and the corresponding time preference for information by the base station, can be parametrised with a single numerical parameter between 0 and 1, which we call the return ratio. This approach is highly robust and we showed in simulation that it can lead to complex relay chains being formed in different types of environments. Additionally, we showed that this approach is very flexible in adjusting to the desired allocation of resources between exploration and communication, as well as to varying numbers of agents in the team. Simulated results suggest that using return ratios between 0.6 and 0.7 tend to provide a good trade-off between the speed of exploration and the rate of communication on different types of environments for team sizes of between 2 and 8 agents. Furthermore, we showed that the return ratio set to the inverse of the golden ratio means that when a corridor environment is explored by two agents, they will end up meeting at what was the boundary of explored and unexplored space at the time of the previous meeting.

We showed that certain types of environments can benefit from explicitly arranging rendezvous. These types of environments include large open areas where opportunistic communication between agents is unlikely, and environments with areas of low traversability or including thin obstacles through which

communication is possible, where carefully planning rendezvous locations can save agents navigation time, improve team connectivity and keep agents out of dangerous areas. We proposed a novel definition of a rendezvous location as a tuple of points. Using such rendezvous locations could allow agents to exploit the topology of the environment and their communication range. We outlined an algorithm for obtaining such rendezvous locations and proposed exploration strategies MP-RELEXP and MP-RELEXP-ST which can reduce the travel time of the relay in certain environments. We showed in simulation that MP-RELEXP can improve team connectivity and MP-RELEXP-ST can improve exploration speed in some settings. Simulation results suggest that as the number of agents in the team is increased, MP-RELEXP-ST offers diminishing returns and MP-RELEXP should be favoured.

As part of this DPhil, the discrete-time multi-agent exploration simulator MRESim, originally introduced in [30], was extensively developed. The simulator was proposed as a middle ground of abstraction between the Agent and Virtual Robot competitions of RoboCup Rescue. Our work on MRESim as part of this thesis won the 2014 Infrastructure competition of the RoboCup Rescue Simulation League. The simulator is described in detail in Appendix A.

6.1.1 Applicability guidelines

In this thesis two types of exploration strategies were proposed. While more validation is required before these approaches can be applied in real life scenarios (see Section 6.2), here we present guidelines based on the simulated experiments in this thesis on when each of the proposed approaches should be favoured.

Opportunistic approaches utilizing the return ratio should be used when:

1. Teams consist of a relatively large number of homogeneous expendable robots.

2. Environments are corridor-like, where agents are likely to communicate opportunistically.

3. Mission priorities or search areas are unlikely to change throughout the mission, as reliable quick propagation of messages from the base station to the whole robot team may be unavailable.

4. The operating environment is dynamic, making planning for communication difficult.

If the goal of the mission is to complete the full exploration of the environment as quickly as possible, lower values of the return ratio should be preferred. If getting early information about some of the environment is more important, higher values of the return ratio should be used.

Explicit planning for rendezvous using approaches such as MP-RELEXP and MP-RELEXP-ST should be done when:

1. Teams consist of either a small number of robots, or the robots are heterogeneous where some may be more suitable for exploration tasks and some may be more suitable for relaying, or some robots in the team are highly specialized and are critical to the mission, requiring closer control over their whereabouts.

2. Environments consist of large open areas where opportunistic communication between robots is unlikely, or there are areas in the environment where careful planning for communication can yield significant advantages (for example, multi-level environments containing staircases, maps with areas of low traversability or including obstacles through which communication may be possible).

3. Close control over the whereabouts of the robots is required; messages from the base station may need to be distributed to the robot team, for example to change mission priorities.

4. The operating environment is static, allowing robots to arrange rendezvous locations and to reach them at pre-agreed times.

For small robot teams consisting of 4 robots or less, MP-RELEXP-ST should be used to improve the speed of exploration. For teams consisting of 6 robots or more, MP-RELEXP should be preferred, offering improved communication latency.

6.2 Future work

The work presented in this thesis can be extended in many different ways. In this section we discuss some of the possible extensions.

Environment representations

In this work we assumed that operating environments can be represented as a two-dimensional occupancy grid map for the purposes of planning and navigation. In many real-world scenarios this assumption may not hold. Surfaces may be uneven and covered with debris, there may be complex rubble and environments may consist of multiple levels. Some approaches at building 2.5 dimensional maps that incorporate such information have been proposed in [121] and [16]. While we expect the opportunistic rendezvous approach to adapt very well to such scenarios, extensive evaluations under these new assumptions are required. The methods in this thesis which arrange rendezvous explicitly will need to be modified to take the different environment representation into account. A straightforward approach could be to consider the areas of low traversability as obstacles for the sake of planning rendezvous, however more complex and efficient approaches may be possible.

In this work we made the assumption that the operating environments remain static throughout the exploration mission. In practice, in many scenarios it is likely that the environment may change as the exploration effort progresses. Such dynamic environments pose significant challenges for both simultaneous

localization and mapping (SLAM) and planning. In addition, a standardized benchmark for evaluating the performance of multi-agent exploration systems in dynamic environments is required, analogous to the standardized arenas used in the Rescue Robot competition of the RoboCup Rescue League [110].

Prior information

In practical deployments, some information about the operating environment may be available prior to the mission, although this information may not be accurate. Architectural plans of the building to be explored may provide valuable information about the relative locations of rooms and corridors, as well as the potential types of obstacles that may be encountered (including an estimate of their possible attenuation factors). Even just knowing the type of the building that is being explored can offer valuable information about the topology of the environment, as shown in [77]. Making use of this information during the exploration presents an interesting area of future research. For example, rendezvous can be timed in such a way so that an area can be fully explored before the explorer heads back to rendezvous, preventing situations where the explorer leaves a small fragment of an area unexplored, having to then return back to the same area to finish exploring it.

In addition, agents may be able to predict the likely trajectories of their teammates to intercept them in order to facilitate communication. This could greatly improve the performance of both opportunistic and pre-arranged rendezvous approaches. In [52], agents gather statistical information about the trajectories of their teammates in order to intercept them on the way to rendezvous. Agents may also mark the environment that they operate in in order to provide information about their trajectories. Another option could be the use of a low frequency, high range radio channel to exchange small amounts of information that may help agents intercept each other.

Control channel

The use of an additional low frequency (bandwidth), high range radio communication channel may be used to exchange short status messages between the robots, facilitate planning, increase the efficiency of opportunistic rendezvous and further reduce the risks of rendezvous through obstacles vs line-of-sight communication. Loosely speaking, radio signal attenuation through debris increases with transmission frequency, which means that it should be possible to have extended communication range on a low frequency, low-bandwidth channel. That would mean that high-priority short messages can be sent to most of the agents most of the time (such as agent locations, notifications of delays or cancellations to planned meetings, or announcing that an agent is proceeding to the fallback meeting point). In practice it should be easy to transmit such a signal at high power from the outside using (temporary) fixed aerials, so that such an approach could be used by the situation commander to change agent priorities. Conversely, getting such messages from the robots may be harder, as the agents would presumably have their aerials in fixed positions relative to their bodies. Depending on reception conditions (interference) the most practical scheme is probably to transmit to the outside, and have inter-agent messages relayed from there.

The reliability and the range of such communication channels will need to be evaluated. We have looked at demonstration LoRa units from Semtech for proof-of-concept experiments. These operate in the VHF waveband (868MHz), and are sold mainly for remote sensor interrogation applications, with line-of-sight ranges of kilometres. The demo units can be used to test for signal range using hand-held devices through thick walled structures. Our preliminary experiments indicate that these have very good range in indoor environments and offer a lot of potential.

Towards real world deployments

A number of barriers still need to be overcome before the proposed approaches can be used in practical deployments. Emergency responders are generally reluctant to apply autonomous robots in disaster situations as they present an additional risk factor in an already highly unpredictable environment. We believe that verification could play a crucial role in convincing users about the properties of the autonomous multi-robot systems in terms of both robustness and performance. Standardized benchmark scenarios need to be created that bear close resemblance to the operating environments in which these multi-robot systems will need to operate. Using such benchmarks for evaluation will allow to demonstrate adequate reliability of the robotic platforms, accuracy of SLAM approaches in those specific operating environments as well as allow for the collection of accurate quantitative results on the performance of both the approaches presented in this thesis, as well as of competing approaches, that should translate well to real world deployments.

6.3 Summary

Exploration of indoor environments with teams consisting of multiple agents operating under limited communication constraints remains a young field of study. We hope that the contributions in this thesis provide a stepping stone for future work. The improvement of low-level robotic capabilities and the increased use of robots in search-and-rescue missions means there is likely to be a lot more interest in this area in the future.

Appendices

Appendix A

MRESim, a multi-robot exploration simulator

This appendix gives an in-depth description of the multi-agent exploration simulator MRESim. Originally introduced in [30], it was heavily developed as part of this DPhil and was used for the experiments in this work. The contents of this appendix were published in [114].

A.1 Introduction

The RoboCup Rescue competitions provide benchmarks for evaluating robot platforms' usability in disaster mitigation. Research groups should demonstrate their ability to deploy a team of robots that explore a devastated area and locate victims. RoboCup is moving towards long-term goals of sophisticated resource allocation in disaster situations, both at a large scale (the Agents competition) and at a small scale (the Virtual Robots competition). Each year the benchmarks are made more challenging to accommodate and encourage progress by all teams. The Infrastructure competition showcases innovations that have enabled extensions of these benchmarks.

The Infrastructure competition was started in 2004 with the aim of promoting the development of new tools to improve rescue simulation. The simulation of various disaster situations turns out to be complicated and difficult to validate. Therefore, the infrastructure competition was launched to promote the

maintenance and development of simulation environments. For example, the fire simulator [90] was developed by the winner of the infrastructure competition in 2004. Another nice example of a component developed in the infrastructure competition is the flood simulator [108]. Recently, an extension towards flying robots has been proposed [34].

The Amsterdam Oxford Joint Rescue Forces, and their predecessor the UvA Rescue Team, has participated several times in the Infrastructure competition. In 2010 the simulator of the Virtual Robot competition was extended with a realistic response of laser scanners to smoke; a circumstance which is quite common in disaster situations. The response of the laser scanners was validated in a number of experiments in a training centre of the Dutch fire brigade [40]. In 2011 a model of a humanoid robot was introduced in USARSim, which made it possible to model one of the robots in the RoboCup@Home League [122]. In 2012 a validated flying robot was introduced to USARSim and it was demonstrated that such a robot allows for fast exploration of disaster areas, while creating a visual map of the area [34]. This resulted in the UvA Rescue team winning the Infrastructure award in 2012.

What was still missing inside the Rescue Simulation League is an environment which is a common ground for both the Agent and Virtual Robot competitions. Inside the Agent competition the focus is on the coordination of rescue teams [61, 125]. The competition consists of a simulation environment which resembles a city after an earthquake. Teams of fire brigade, police and ambulance team agents try to extinguish fires and rescue victims in the collapsed buildings. In contrast, inside the Virtual Robot competition the focus is on sensor-data fusion on maps automatically generated by teams of robots which explore inside buildings. The multi-robot exploration simulator MRESim[1], described in this appendix, is proposed as a middle ground between the Agent and Virtual Robot competitions.

In the following section the background behind this new infrastructure is

[1] `https://github.com/v-spirin/MRESim`

given, followed by a section which gives an overview of the research performed with this multi-robot exploration simulator.

A.2 Simulator

A.2.1 Simulation cycle

MRESim is a discrete-time simulator. Unlike the existing Virtual Robot competition simulator USARSim, which uses a three-dimensional representation, MRESim works from a two-dimensional grid representation. In each time step, the simulator moves the robots to their new positions, updates their sensor information, simulates communication between the robots and outputs the current state of the simulation. The ordering of the events that happen in each time step are outlined in Algorithm A.1. This is equivalent with the discrete-time step procedure in the simulator environment of the Agent competition. All of the robots process their next steps simultaneously. Since this tends to be the most computationally intensive part of each time step, planning robot movements in parallel allows the simulator to take advantage of modern multi-core CPUs. Other events in each time step are processed sequentially.

A.2.2 User interface and environments

The user interface (Fig. A.1) contains a panel on the right for each robot with toggle buttons to visualize the information which is the basis of each robot's decisions. Example of information that can be made visible is the following: location; path; free space; safe space; frontiers; communication range; skeleton and potential rendezvous points; exact rendezvous points; potential rendezvous points through obstacles; exact rendezvous points through obstacles. There are further toggle buttons for the environment walls and the team hierarchy at the bottom right, and a run may at any time be paused, continued, or stopped using the green buttons at the bottom. This means that it is very easy to examine specific aspects of any supported exploration approach. In addition,

```
   Input: Set R of robots
   // Simulate movement, range data
 1 foreach r_i ∈ R do
   |  // described in Section A.2.5
 2 |  while distance_moved(r_i) < max_speed(r_i) do
 3 |  |  nextStep = r_i.takeStep();
 4 |  |  if isValid(nextStep) then
 5 |  |  |  move(r_i, nextStep);
   |  |  |  // described in Section A.2.4
 6 |  |  |  rangeData = findRangeData(r_i, nextStep);
 7 |  |  |  r_i.receiveRangeData(rangeData);
 8 |  |  else
 9 |  |  |  r_i.setError(true);
10 |  |  |  break;
11 |  |  end
12 |  end
13 end
   // Simulate communication, described in Section A.2.6
14 foreach r_i ∈ R do
15 |  foreach r_j ∈ R, i ≠ j do
16 |  |  if isInRange(r_i, r_j) then
17 |  |  |  r_i.receiveMsg(r_j.createMessage());
18 |  |  |  r_j.receiveMsg(r_i.createMessage());
19 |  |  end
20 |  end
21 end
   // Complete cycle
22 logData();
23 updateGUI();
```

Algorithm A.1: A single simulation cycle

simulation logs produced by MRESim can be replayed in the simulator; this means that each simulation run can be analysed in detail after the run has completed.

Environments in MRESim can be uploaded from text-based or PNG files, as can be found on dataset repositories like Radish[2]. The environments are occupancy grid models, with each cell being either free or an obstacle.

[2] http://radish.sourceforge.net/

Input: Robot r_i, simulator settings $simConfig$

```
// Replan if necessary, then retrieve the next path point
```

1 **if** $r_i.timeToReplan$ ***or*** $r_i.getPath().isEmpty()$ **then**
2 | $simConfig$.getExplorationStrategy().replan(r_i);
3 **end**
4 **return** $r_i.getPath().getNextPoint()$;

Algorithm A.2: Description of the agent takeStep method

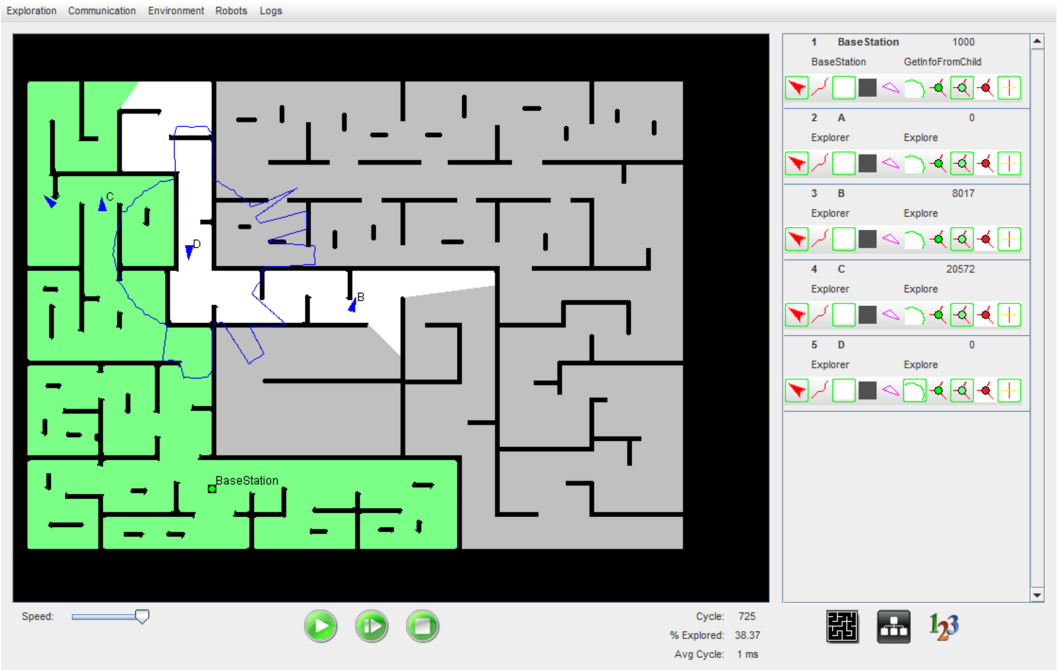


Figure A.1: Screenshot of MRESim. Obstacles are shown in black (▬), unexplored area is shown in grey (▬), area known at base station is shown in green (▬). The blue arrows (►) represent exploring robots, and the communication range of robot D is shown as the blue polygon (▭).

A.2.3 Planning and Movement

In every time step, each robot has to decide where to move next. The next destination is decided by the robot by calling the *takeStep* method (line 3 of Alg. A.1). If the robot has a path to its destination, and has replanned the path recently, then it simply continues on the current path by retrieving the next path point. Otherwise, it calls the corresponding *replan* method of the exploration strategy assigned to the robot team to generate a new path (line 2 of Alg. A.2). The exploration strategy then selects the best destination for the robot, and generates a path to the destination. The exploration strategy

can use all the information known to the robot when making that decision — including the information about last known locations of the robot's teammates and agreements made in previous communication with other robots or the base station. Note that this information is not globally available, but communicated when in range. When no recent information is available, reasonable estimates are made.

Each robot r can move a maximum distance of d_r in each time step (line 5 of Alg. A.1). This is defined as the robot "speed", and can be set up to be different for each robot, allowing for heterogeneous teams of robots to be created. By default, this parameter is set for each robot to $DEFAULT\_SPEED = 3$, configured in the *Constants* class of the *config* package. In each time step, each robot moves a maximum distance of d_r along the path generated in the planning stage (see line 2 in Algorithm A.2). If the robot reaches the planned destination before d_r is exhausted, it will not move any further in that timestep.

A.2.4 Sensing and Mapping

Once a robot has taken a step, the simulator provides it with sensor data at its new location (lines 6-7 of Alg. A.1). This sensor data is generated using raytracing from the robot's location at 1-degree intervals in the 180-degree field of view of the robot (the same field of view as many real laser scanners, as for instance the SICK LMS200). A maximum sensor range can be configured for each robot, allowing for heterogeneity of sensors across the team. An array of 181 measurements is returned to the robot. The measurements are assumed to be noise-free (see §A.2.7 for more detail).

The robot subsequently turns this sensor data into a polygon of free space, detecting obstacles where two points are sufficiently close to one another and below the sensor's range limit. This free space and obstacle detection is maintained in an occupancy grid. When robots are within communication range of one another, they can decide to exchange their local maps in the form of occupancy grids.

A.2.5 Path planning

Several path planning algorithms have been implemented in the MRESim simulator. The most straightforward one is the A* algorithm [50], operating directly on the occupancy grid representation of the map. While A* generates optimal paths, it can be very slow with a sufficiently high resolution of the occupancy grid. In an attempt to overcome this problem, we have implemented several more efficient path planning algorithms. One of them is Jump Point Path search [49], which is a modification of the A* algorithm which significantly improves the performance in environments with large open areas. However, complex paths can still take several hundred milliseconds to compute.

We found that using A* search on a combination of a simple topological map that captures the connectivity between regions and the underlying occupancy grid map can significantly improve performance over the other implemented approaches. In each time step, if the map of the environment has been updated, we can generate a new topological map as follows. First, we perform thinning on the occupancy grid map in order to obtain a skeleton of the free space. Then, we uniformly select a set of nodes from the skeleton. Each point of the occupancy grid is then assigned to the nearest node. The set of nodes and edges connecting the nodes then represents a simple topological map of the environment. An illustration of the process is given in Fig. A.2. We can then plan a path between any two points on the occupancy grid map by using the A* algorithm to find a path in the graph between the two corresponding nodes on the topological map, and using A* on the occupancy grid map to find paths from start and finish locations to their corresponding node locations. Generating paths then becomes very computationally efficient, which improves the speed of running the simulation and can increase the performance of exploration strategies by allowing them to calculate more accurate path lengths, instead of relying on crude approximations.

A.2.6 Communication

MRESim supports a variety of communication models. In principle messages from both robots are exchanged (as indicated in lines 17 and 18 of Alg. A.1), but only when the robots are *inRange* (line 16 of Alg. A.1). Simple models include a straight-line model (any robot within radius x is considered in range, regardless of obstacles) and a line-of-sight model (any two robots that can be connected by a line that doesn't hit an obstacle are in range). A more realistic communication model implemented in MRESim is a path loss model, originally proposed by Bahl and Padmanabhan [11]. This model is also used for simulating communication in the USARSim simulator used at RoboCup until 2015, and is considerably more realistic as it takes attenuation by walls into account. In this model, communication strength is calculated as follows:

$$S = P_{d_0} - 10 \times N \times log_{10}\frac{d_m}{d_0} \times \min(nW, C) \times WAF$$

where P_{d_0} is the signal strength at reference distance d_0, N is the rate of path loss, d_m is the distance, nW is the number of obstructing walls, WAF is the wall attenuation factor and C is the maximum number of walls where the attenuation factor needs to be considered.

A.2.7 Realism of Simulator Results

Clearly MRESim does not take into account a number of factors that any robot system in the real world would have to consider. The most significant ones are:

1. There is no sensor noise. In reality, wheel encoders provide inexact data, and laser range finders often have spurious measurements at either close or maximum range. Localisation remains a significant challenge in robotics, even if a number of techniques such as particle filters and scan matching show great promise.

2. Environments are two-dimensional and flat. In the real world this is almost never the case.

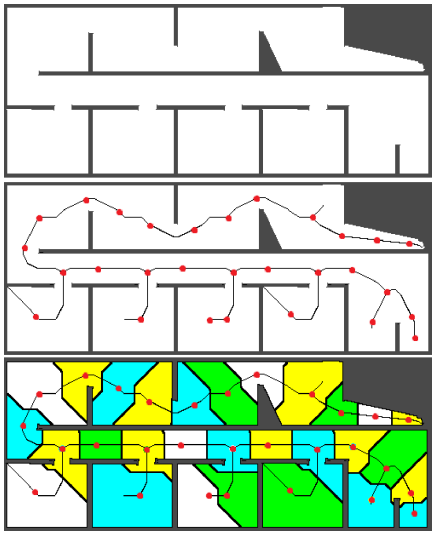


Figure A.2: Illustration of the process used by MRESim to generate topological maps from occupancy grid maps. On top, we see an occupancy grid of a partially explored environment, shown in white. In the middle, we apply "thinning" to obtain a skeleton, and sample uniformly points, shown in red (-•-), from the skeleton to act as nodes in the topological graph. In the bottom picture, we map each point of the occupancy grid map to the nearest node.

3. The simulator is discrete-time. Real robots would each run their own (multi-thread) processes, take different amounts of time to do the required processing, and would not always move the same distance in each time segment.

4. Communication in reality is highly variable and very difficult to predict. Nevertheless, for purposes of quick, simple, controlled, and repeatable comparison of exploration algorithms, MRESim remains a useful tool.

In the next section we give an overview of published work that used MRESim to evaluate exploration strategies. In some of the work, results are compared with those obtained from high-fidelity simulators such as USARSim, as well as with results obtained on real robots. Those experiments suggest that results obtained in MRESim have much in common with the results obtained in real life experiments.

A.3 MRESim as benchmark

A number of exploration strategies have already been implemented in MRESim and are available "out-of-the-box". This makes it a potentially useful tool for benchmarking various exploration strategies against each other. According to [58], the performance of most algorithms is compared using the following three methods, from the best to the worst:

1. Using the same implementation that was used in other work.
2. Using a custom implementation, created from descriptions of the algorithm in published work.
3. Simply taking the results from those in other papers, without re-running the algorithm.

According to [6], most algorithm comparisons in robotics are currently conducted using the second approach. Publishing of the team's source-code, as done inside the Rescue Simulation League, made it possible to reimplement existing algorithms and to make a fair comparison (see for instance [6, 109]). Using MRESim, the first approach can be used more often:

1. Results of using different exploration strategies can be directly compared with each other.
2. Existing implementations of exploration strategies can be used.

A.3.1 Studies based on MRESim

MRESim was first mentioned in [30]. In this study the effect of robots with a relay role was studied, when exploration was needed outside the direct communication range from the base station. This study inspired several other researchers [14, 22, 27] to base the coordination decisions on multiple criteria.

Important for the coordination of the explorer and relay role is the selection of an appropriate rendezvous point [32]. When those points are selected near

gateways, the locations inside indoor environments become important junctions for a navigation algorithm. This work was followed up by several researchers [76, 81, 95] (although the concept of rendezvous points has already been known for a long time [105]). In addition, in [112] the possibility of communicating through obstacles, such as thin walls, was introduced into the planning.

Dividing the work into explorer and relay roles is already valuable, but sometimes the explorer finds a dead-end. This is an example of a situation where it is beneficial to switch roles dynamically using the *role swap rule*, as described in [31]. Other researchers have validated this results with real robots, such as [56].

In the study [134], the exploration indoors is extended to open areas which can be encountered in outdoor scenarios. The robots still use rendezvous points, but no longer near gateways. Instead, the robots divide the work into sectors and meet at the sector boundaries. This work has motivated several other studies [15, 52, 133].

The algorithms implemented in MRESim have been validated with Pioneer robots [33] and TurtleBots [13], although it is difficult to scale indoors to extensive environments with large robot teams. The MRESim environment was discussed in several dissertations [29, 79, 133] and a workshop contribution [124].

The latest extension are robot teams which can adjust their exploration strategy based on the information need of the base station [113]. When it is important that the base station gets timely updates more resources are allocated towards the relay role; when fast exploration is needed more resources are allocated towards the explorer role.

A.4 Discussion and Future Work

MRESim is a simulation environment with a well chosen level of detail, as discussed in A.2.7. When the algorithms developed inside MRESim are applied

in the real world issues will arise, which can be studied in separation inside the simulation. Precisely those issues could be identified as directions for future research.

In addition, the simulator itself can be extended in a number of ways to increase its realism and allow for using it to study a wider range of research problems:

1. **Variable terrain**. In real applications, the terrain of the environment being explored can vary significantly, affecting the speed of robots navigating over such terrain. For example, robots are likely to be able to travel much faster just outside a partially-collapsed building, than inside where there is likely to be a lot of rubble. It may then be possible to improve the exploration speed by reducing the amount of travel over rough terrain, particularly for the relays.

2. **Additional communication channels**. It may be possible to use low-frequency, low-bandwidth, high range radio communication channels in disaster scenarios to transfer some control information between robots, as suggested in [112]. (We are about to test this hypothesis experimentally.)

3. **Robustness against Failing Robots**. Robots may fail for a number of reasons (and often do). In greedy approaches this does not affect the failed robot's teammates, which continue exploring as if nothing happened. In some other approaches, meetings between robots at pre-agreed times and locations can be planned for explicitly. In those cases, the failure of robots is a scenario that must be dealt with carefully. A useful extension to the simulator could therefore be the ability to specify scenarios that include robot failure, making it a useful tool to evaluate the robustness of exploration strategies to individual robot failure.

4. **Dynamic Environments**. In many robotics applications, the environment may change as the exploration effort progresses. A possible exten-

sion to the simulator would be to specify how the environment should change over time, or allow for random changes to the environment. This is precisely where the current Agent competition accelerates and it would be nice if for example part of the fire, earthquake or flood simulation could be incorporated in MRESim.

5. **Prior knowledge**. In practice, some knowledge about the environment may be available to the team before the start of the mission, even though this information may not be accurate.

A.5 Conclusions

The original motivating questions for this research were: How can a team of robots be coordinated to explore a previously unknown and communication-limited environment as efficiently as possible; and how can new information obtained by this team be gathered at a single location as quickly and as reliably as possible?

These are precisely the research questions studied in both the Agent and the Virtual Robot competition of the RoboCup Rescue Simulation League. The studies performed with the simulation environment MRESim are well recognized, both inside the RoboCup community and outside. MRESim is applied to compare several coordination algorithms and could be easily extended with other coordination algorithms. In this paper some ideas for future research are given, but many other extensions are possible.

References

[1] Rescue Robot League. https://en.wikipedia.org/wiki/Rescue_Robot_League#Test_Arena. [Online; accessed 18-July-2016].

[2] RoboCup 2015 Scores. http://wiki.robocup.org/wiki/Robot_League#RoboCup_2015_Scores. [Online; accessed 11-April-2016].

[3] Carlos E Aguero, Nate Koenig, Ian Chen, Hugo Boyer, Steven Peters, John Hsu, Brian Gerkey, Steffi Paepcke, Jose L Rivero, Justin Manzo, et al. Inside the Virtual Robotics challenge: Simulating real-time robotic disaster response. *Automation Science and Engineering, IEEE Transactions on*, 12(2):494–506, 2015.

[4] H Levent Akin, Nobuhiro Ito, Adam Jacoff, Alexander Kleiner, Johannes Pellenz, and Arnoud Visser. RoboCup Rescue Robot and Simulation leagues. *AI magazine*, 34(1):78, 2012.

[5] Francesco Amigoni, Matteo Luperto, and Alberto Quattrini Li. Towards more realistic indoor environments for the virtual robot competition. *RoboCup2014 CD*, 2014.

[6] Francesco Amigoni and Viola Schiaffonati. Good experimental methodologies and simulation in autonomous mobile robotics. In *Model-Based Reasoning in Science and Technology*, volume 314 of *Studies in Computational Intelligence*, pages 315–332. Springer Berlin Heidelberg, 2010.

[7] Francesco Amigoni, Masaru Shimizu, Sanaz Taleghani, and Arnoud Visser. RoboCup 2016 RoboCup Rescue Simulation League Virtual Robot competition rules document. 2016.

[8] Torsten Andre and Christian Bettstetter. Collaboration in multi-robot exploration: To meet or not to meet? *Journal of Intelligent & Robotic Systems*, 82(2):325–337, 2016.

[9] Michael Angermann, Martin Frassl, and Michael Lichtenstern. Mission review of aerial robotic assessment – ammunition explosion Cyprus 2011. In *Safety, Security, and Rescue Robotics (SSRR), 2012 IEEE International Symposium on*, pages 1–6, Nov 2012.

[10] Ronald C Arkin and Jonathan Diaz. Line-of-sight constrained exploration for reactive multiagent robotic teams. In *Advanced Motion Control, 2002. 7th International Workshop on*, pages 455–461. IEEE, 2002.

[11] Paramvir Bahl and Venkata N. Padmanabhan. RADAR: an in-building RF-based user location and tracking system. In *proceedings IEEE INFOCOM*, March 2000.

[12] Benjamin Balaguer, Stephen Balakirsky, Stefano Carpin, Mike Lewis, and Christopher Scrapper. USARSim: a validated simulator for research in robotics and automation. In *Workshop on Robot Simulators: Available Software, Scientific Applications, and Future Trends at IEEE/RSJ*, 2008.

[13] Jacopo Banfi, Alberto Quattrini Li, Nicola Basilico, Ioannis Rekleitis, and Francesco Amigoni. Asynchronous multirobot exploration under recurrent connectivity constraints. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5491–5498, May 2016.

[14] Nicola Basilico and Francesco Amigoni. Exploration strategies based on multi-criteria decision making for searching environments in rescue operations. *Autonomous Robots*, 31(4):401–417, 2011.

[15] Haluk Bayram and H Işil Bozma. Decentralized network topologies in multirobot systems. *Advanced Robotics*, 28(14):967–982, 2014.

[16] Igor Bogoslavskyi, Olga Vysotska, Jacopo Serafin, Giorgio Grisetti, and Cyrill Stachniss. Efficient traversability analysis for mobile robots using the Kinect sensor. In *Mobile Robots (ECMR), 2013 European Conference on*, pages 158–163. IEEE, 2013.

[17] Nigel Bowden. Wi-Fi planning, walls and dBs measuring obstruction losses for WLAN predictive modelling. `http://www.ekahau.com/wifidesign/blog/2015/09/07/wi-fi-planning-walls-and-dbs-measuring-obstruction-losses-for-wlan-predictive-modelling/`, Sep 2015. [Online; accessed 10-June-2016].

[18] Peter Brass, Flavio Cabrera-Mora, Andrea Gasparri, and Jizhong Xiao. Multirobot tree and graph exploration. *IEEE Transactions on Robotics*, 27(4):707–717, 2011.

[19] Paul Bratley and Bennett L. Fox. Algorithm 659: Implementing Sobol's Quasirandom Sequence Generator. *ACM Transactions on Mathematical Software*, 14(1):88–100, March 1988.

[20] Wolfram Burgard, Mark Moors, Dieter Fox, Reid Simmons, and Sebastian Thrun. Collaborative multi-robot exploration. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, pages 476–481. IEEE, 2000.

[21] Lucian Busoniu, Robert Babuska, and Bart De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, And Cybernetics-Part C: Applications and Reviews, 38 (2)*, 2008.

[22] Jesus Capitan, Matthijs TJ Spaan, Luis Merino, and Anibal Ollero. De-

centralized multi-robot cooperation with auctioned POMDPs. *The International Journal of Robotics Research*, 32(6):650–671, 2013.

[23] Luca Carlone, Miguel Kaouk Ng, Jingjing Du, Basilio Bona, and Marina Indri. Rao-Blackwellized Particle Filters multi robot SLAM with unknown initial correspondences and limited communication. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 243–249, May 2010.

[24] Stefano Carpin. Merging maps via Hough transform. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1878–1883, Sept 2008.

[25] Stefano Carpin, Todor Stoyanov, Yashodhan Nevatia, M Lewis, and J Wang. Quantitative assessments of USARSim accuracy. In *Proceedings of PerMIS*, 2006.

[26] Stefano Carpin, Jijun Wang, Michael Lewis, Andreas Birk, and Adam Jacoff. High fidelity tools for rescue robotics: results and perspectives. In *RoboCup 2005: Robot Soccer World Cup IX*, pages 301–311. 2005.

[27] Jesús S Cepeda, Luiz Chaimowicz, Rogelio Soto, José L Gordillo, Edén A Alanís-Reyes, and Luis C Carrillo-Arce. A behavior-based strategy for single and multi-robot autonomous exploration. *Sensors*, 12(9):12772–12797, 2012.

[28] Micael S Couceiro, Rui P Rocha, and Nuno MF Ferreira. A novel multi-robot exploration approach based on particle swarm optimization algorithms. In *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, pages 327–332, 2011.

[29] Julian de Hoog. *Role-Based Multi-Robot Exploration*. PhD thesis, University of Oxford, May 2011.

[30] Julian de Hoog, Stephen Cameron, and Arnoud Visser. Role-based autonomous multi-robot exploration. In *Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, 2009. COMPUTATION-WORLD '09. Computation World:*, pages 482–487, Nov 2009.

[31] Julian de Hoog, Stephen Cameron, and Arnoud Visser. Dynamic team hierarchies in communication-limited multi-robot exploration. In *Safety Security and Rescue Robotics (SSRR), 2010 IEEE International Workshop on*, pages 1–7, July 2010.

[32] Julian de Hoog, Stephen Cameron, and Arnoud Visser. Selection of rendezvous points for multi-robot exploration in dynamic environments. In *Workshop on Agents in Realtime and Dynamic Environments, International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, May 2010.

[33] Julian de Hoog, Adrian Jiménez-González, Stephen Cameron, José Ramiro Martínez de Dios, and Anibal Ollero. Using mobile relays in multi-robot exploration. In *Proceedings of the Australasian Conference on Robotics and Automation*, December 2011.

[34] Nick Dijkshoorn and Arnoud Visser. Urban Search and with Micro Aerial Vehicles. In *Proceedings CD of the 16th RoboCup International Symposium*, June 2012.

[35] Edsger W Dijkstra. A note on two problems in connexion with graphs. *NUMERISCHE MATHEMATIK*, 1(1):269–271, 1959.

[36] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part I. *IEEE Robotics Automation Magazine*, 13(2):99–110, June 2006.

[37] Russ C Eberhart, James Kennedy, et al. A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on*

micro machine and human science, volume 1, pages 39–43. New York, NY, 1995.

[38] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, June 1989.

[39] Ettore Ferranti, Niki Trigoni, and Mark Levene. Brick& Mortar: an on-line multi-agent exploration algorithm. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 761–767. IEEE, 2007.

[40] Okke Formsma, Nick Dijkshoorn, Sander van Noort, and Arnoud Visser. Realistic Simulation of Laser Range Finder Behavior in a Smoky Environment. In *RoboCup 2010: Robot Soccer World Cup XIV*, volume 6556, pages 336–349. June 2011.

[41] Dieter Fox, Jonathan Ko, Kurt Konolige, Benson Limketkai, Dirk Schulz, and Benjamin Stewart. Distributed multirobot exploration and mapping. *Proceedings of the IEEE*, 94(7):1325–1339, July 2006.

[42] Stephen Friedman, Hanna Pasula, and Dieter Fox. Voronoi random fields: Extracting the topological structure of indoor environments via place labeling. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, volume 7, pages 2109–2114, 2007.

[43] Jorge Fuentes-Pacheco, José Ruiz-Ascencio, and Juan Manuel Rendón-Mancha. Visual Simultaneous Localization and Mapping: A survey. *Artificial Intelligence Review*, 43(1):55–81, January 2015.

[44] Mohammad Ghavamzadeh and Sridhar Mahadevan. Learning to communicate and act using hierarchical reinforcement learning. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 1114–1121. IEEE Computer Society, 2004.

[45] Hector H Gonzalez-Banos and Jean-Claude Latombe. Navigation strategies for exploring indoor environments. *The International Journal of Robotics Research*, 21(10-11):829–848, 2002.

[46] Giorgio Grisetti, Rainer Kummerle, Cyrill Stachniss, and Wolfram Burgard. A tutorial on graph-based SLAM. *Intelligent Transportation Systems Magazine, IEEE*, 2(4):31–43, 2010.

[47] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improving grid-based SLAM with Rao-Blackwellized Particle Filters by adaptive proposals and selective resampling. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 2432–2437, April 2005.

[48] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improved techniques for grid mapping with Rao-Blackwellized Particle Filters. *IEEE Transactions on Robotics*, 23(1):34–46, Feb 2007.

[49] Daniel Harabor and Alban Grastien. Online graph pruning for pathfinding on grid maps. *In Proceedings of the 25th National Conference on Artificial Intelligence (AAAI '11), San Francisco, California, USA*, pages 1114–1119, 2011.

[50] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, July 1968.

[51] Geoffrey A Hollinger and Sanjiv Singh. Multirobot coordination with periodic connectivity: Theory and experiments. *Robotics, IEEE Transactions on*, 28(4):967–973, 2012.

[52] Hamido Hourani, Eckart Hauck, and Sabina Jeschke. Serendipity rendezvous as a mitigation of exploration's interruptibility for a team of

robots. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2984–2991. IEEE, 2013.

[53] Andrew Howard, Lynne E. Parker, and Gaurav S. Sukhatme. Experiments with a Large Heterogeneous Mobile Robot Team: Exploration, Mapping, Deployment and Detection. *International Journal of Robotics Research*, 25(5-6):431–447, May 2006.

[54] Adam Jacoff, Raymond Sheh, Ann-Marie Virts, Tetsuya Kimura, Johannes Pellenz, Sören Schwertfeger, and Jackrit Suthakorn. Using competitions to advance the development of standard test methods for response robots. In *Proceedings of the Workshop on Performance Metrics for Intelligent Systems*, pages 182–189, 2012.

[55] Elizabeth A Jensen, Ernesto Nunes, and Maria Gini. Communication-restricted exploration for robot teams. In *AAAI Workshops*. Citeseer, 2014.

[56] Adrián Jiménez-González, José Ramiro Martínez-de Dios, and Aníbal Ollero. An integrated testbed for cooperative perception with heterogeneous mobile and static sensors. *Sensors*, 11(12):11516–11543, 2011.

[57] Stephen Joe and Frances Y. Kuo. Constructing Sobol sequences with better two-dimensional projections. *SIAM Journal on Scientific Computing*, 30(5):2635–2654, 2008.

[58] David S. Johnson. A theoretician's guide to the experimental analysis of algorithms. *Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges*, page 215–250, 2002.

[59] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1):90–98, 1986.

[60] Chanki Kim, Rathinasamy Sakthivel, and Wan Kyun Chung. Unscented FastSLAM: A robust and efficient solution to the SLAM problem. *IEEE Transactions on Robotics*, 24(4):808–820, Aug 2008.

[61] Hiroaki Kitano, Satoshi Tadokoro, Itsuki Noda, Hitoshi Matsubara, Tomoichi Takahashi, Atsuhi Shinjou, and SusuInu Shimada. RoboCup Rescue: Search and Rescue in large-scale disasters as a domain for autonomous agents research. In *Proceedings of IEEE Conference on Man, Systems, and Cybernetics (SMC-99)*, volume 6, pages 739–743. IEEE, 1999.

[62] Alexander Kleiner, Johann Prediger, and Bernhard Nebel. RFID technology-based exploration and SLAM for search and rescue. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 4054–4059. IEEE, 2006.

[63] Jens Kober and Jan Peters. *Reinforcement Learning in Robotics: A Survey*, pages 579–610. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[64] Stefan Kohlbrecher, Johannes Meyer, Thorsten Graber, Karen Petersen, Uwe Klingauf, and Oskar von Stryk. Hector open source modules for autonomous mapping and navigation with rescue robots. In *RoboCup 2013: Robot World Cup XVII*, volume 8371, pages 624–631, 2014.

[65] Stefan Kohlbrecher, Johannes Meyer, Thorsten Graber, Karen Petersen, Oskar von Stryk, and Uwe Klingauf. RoboCupRescue 2014 - Robot League Team Hector Darmstadt (Germany). Technical report, Technische Universitt Darmstadt, 2014.

[66] Stefan Kohlbrecher, Oskar Von Stryk, Johannes Meyer, and Uwe Klingauf. A Flexible and Scalable SLAM System with Full 3D Motion Estimation. In *Proceedings IEEE International Symposium on Safety, Security*

and Rescue Robotics (SSRR), pages 155–160, Kyoto, Japan, November 1-5 2011. IEEE.

[67] Zeid Kootbally, Stephen Balakirsky, and Arnoud Visser. Enabling code-sharing in rescue simulation with USARSim/ROS. In *RoboCup 2013: Robot World Cup XVII*, pages 592–599. 2013.

[68] Yoram Koren and Johann Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1398–1404, 1991.

[69] Gerhard K Kraetzschmar, Guillem Pages Gassull, Klaus Uhl, Guillem Pags, and Gassull Klaus Uhl. Probabilistic quadtrees for variable-resolution mapping of large environments. In *Proceedings of the 5th IFAC/EURON symposium on intelligent autonomous vehicles*. July, 2004.

[70] Geert-Jan M Kruijff, Fiora Pirri, Mario Gianni, Panagiotis Papadakis, Matia Pizzoli, Arnab Sinha, Viatcheslav Tretyakov, Thorsten Linder, Emanuele Pianese, Salvatore Corrao, et al. Rescue robots at earthquake-hit Mirandola, Italy: A field report. In *Safety, Security, and Rescue Robotics (SSRR), 2012 IEEE International Symposium on*, pages 1–8, Nov 2012.

[71] Harold W Kuhn. The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[72] Haye Lau and A NSW. Behavioural approach for multi-robot exploration. In *Proceedings of the 2003 Australasian Conference on Robotics and Automation, Brisbane, Australia*, 2003.

[73] Steven M LaValle. Rapidly-Exploring Random Trees: A new tool for

path planning. *Iowa State University, Computer Science Department, TR 98-11, Technical Report*, 1998.

[74] Steven M LaValle and James J Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, May 2001.

[75] Alberto Quattrini Li, Riccardo Cipolleschi, Michele Giusto, and Francesco Amigoni. A semantically-informed multirobot system for exploration of relevant areas in search and rescue settings. *Autonomous Robots*, 40(4):581–597, 2016.

[76] Chunbo Luo, Paul Ward, Stephen Cameron, Gerard Parr, and Sally McClean. Communication provision for a team of remotely searching UAVs: A mobile relay approach. In *Globecom Workshops (GC Wkshps), 2012 IEEE*, pages 1544–1549. IEEE, 2012.

[77] Matteo Luperto, Alberto Quattrini Li, and Francesco Amigoni. A system for building semantic maps of indoor environments exploiting the concept of building typology. In *RoboCup 2013: Robot World Cup XVII*, pages 504–515. 2013.

[78] Lino Marques and AT de Almeida. Finding odours across large search spaces: A particle swarm-based approach. In *Climbing and Walking Robots*, pages 419–426. Springer, 2005.

[79] Adrian Martin. *A Framework for the Development of Scalable Heterogeneous Robot Teams with Dynamically Distributed Processing*. PhD thesis, University of Toronto, 2013.

[80] Eric Martinson and Ronald C Arkin. Learning to role-switch in multirobot systems. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 2, pages 2727–2734. IEEE, 2003.

[81] Malika Meghjani and Gregory Dudek. Combining multi-robot exploration and rendezvous. In *2011 Canadian Conference on Computer and Robot Vision (CRV)*, pages 80–85, 2011.

[82] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Eighteenth National Conference on Artificial Intelligence*, pages 593–598, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.

[83] Michael Montemerlo, Sebastian Thrun, Daphne Roller, and Ben Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, IJCAI'03, pages 1151–1156, 2003.

[84] Robin R. Murphy, Satoshi Tadokoro, Daniele Nardi, Adam Jacoff, Paolo Fiorini, Howie Choset, and Aydan M. Erkmen. *Springer Handbook of Robotics*, chapter Search and Rescue Robotics, pages 1151–1173. Springer, Berlin, Heidelberg, 2008.

[85] Yashodhan Nevatia. Ad-hoc routing for USARSim. *Jacobs University Brement, Networks and Distributed Systems Seminar*, 2007.

[86] Hoa G Nguyen, Hobart R Everett, Narek Manouk, and Ambrish Verma. Autonomous mobile communication relays. In *AeroSense 2002*, pages 50–57. International Society for Optics and Photonics, 2002.

[87] Hoa G Nguyen, Nathan Farrington, and Narek Pezeshkian. Maintaining communication link for tactical ground robots. *Presented at the AUVSI Unmanned Systems North America 2004, Anaheim, CA, August 3-5*, 2004.

[88] Hoa G Nguyen, Narek Pezeshkian, Anoop Gupta, and Nathan Farrington. Maintaining communications link for a robot operating in a hazardous environment. In *Proceedings of 10th International Conference on Robotics and Remote Systems for Hazardous Environments, Gainesville, FL*, pages 28–31, 2004.

[89] Hoa G Nguyen, Narek Pezeshkian, Michelle Raymond, Anoop Gupta, and Joseph M Spector. Autonomous communication relays for tactical robots. In *Proceedings of 11th International Conference on Advanced Robotics (ICAR)*, pages 35–40, 2003.

[90] Timo A Nüssle, Alexander Kleiner, and Michael Brenner. Approaching urban disaster reality: The ResQ firesimulator. In D. Nardi, M. Riedmiller, C. Sammut, and J. Santos-Victor, editors, *RoboCup 2004: Robot Soccer World Cup VIII*, volume 3276, pages 474–482, 2004.

[91] Mattia Ornaghi, Alberto Quattrini Li, Jacopo Banfi, Nicola Basilico, and Francesco Amigoni. Multirobot exploration with communication constraints: An experimental comparison. In *AAMAS2015 (International Conference on Autonomous Agents and Multiagent Systems) Workshop on "Autonomous Robots and Multirobot Systems (ARMS)"*, 2015.

[92] Yuanteng Pei and Matt W Mutka. Steiner traveler: Relay deployment for remote sensing in heterogeneous multi-robot exploration. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1551–1556. IEEE, 2012.

[93] Johannes Pellenz, Adam Jacoff, Tetsuya Kimura, Ehsan Mihankhah, Raymond Sheh, and Jackrit Suthakorn. *RoboCup 2014: Robot World Cup XVIII*, chapter RoboCup Rescue Robot League, pages 673–685. 2015.

[94] Charles E Perkins. *Ad hoc networking*. Addison-Wesley Professional, 2008.

[95] Viet-Cuong Pham and Jyh Ching Juang. An improved active SLAM algorithm for multi-robot exploration. In *SICE Annual Conference (SICE), 2011 Proceedings of*, pages 1660–1665. IEEE, 2011.

[96] Gill Pratt and Justin Manzo. The DARPA Robotics Challenge [competitions]. *Robotics & Automation Magazine, IEEE*, 20(2):10–12, 2013.

[97] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*, 2009.

[98] Steve Rackley. *Wireless networking technology: From principles to successful implementation.* Elsevier, 2011.

[99] Adrian Ratter and Claude Sammut. Local map based graph SLAM with hierarchical loop closure and optimisation. *In Proceedings of Australasian Conference on Robotics and Automation 2015 (ACRA 2015)*, pages 224–233, December 2015.

[100] Jing Ren, Kenneth A McIsaac, and Rajnikant V Patel. Modified newton's method applied to potential field-based navigation for mobile robots. *IEEE Transactions on Robotics*, 22(2):384–391, 2006.

[101] Alessandro Renzaglia and Agostino Martinelli. Potential field based approach for coordinate exploration with a multi-robot team. In *2010 IEEE Safety Security and Rescue Robotics*, pages 1–6. IEEE, 2010.

[102] Martijn N Rooker and Andreas Birk. Combining exploration and ad-hoc networking in RoboCup Rescue. In *RoboCup 2004: Robot Soccer World Cup VIII*, pages 236–246. 2004.

[103] Martijn N Rooker and Andreas Birk. Communicative exploration with robot packs. In *RoboCup 2005: Robot Soccer World Cup IX*, pages 267–278. 2005.

[104] Martijn N Rooker and Andreas Birk. Multi-robot exploration under the constraints of wireless networking. *Control Engineering Practice*, 15(4):435–445, 2007.

[105] Nicholas Roy and Gregory Dudek. Collaborative robot exploration and rendezvous: Algorithms, performance bounds and observations. *Autonomous Robots*, 11(2):117–136, 2001.

[106] Hanan Samet. The quadtree and related hierarchical data structures. *ACM Computing Surveys (CSUR)*, 16(2):187–260, June 1984.

[107] Noritaka Sato, Fumitoshi Matsuno, Tatsuhiro Yamasaki, Tetsushi Kamegawa, Naoji Shiroma, and Hiroki Igarashi. Cooperative task execution by a multiple robot team and its operators in search and rescue operations. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 2, pages 1083–1088. IEEE.

[108] Hamed Shahbazi, Abbas Abdolmaleki, Sajjad Salehi, Mahdi Shahsavari, and Mostafa Movahedi. RoboCup Rescue 2010 Rescue Simulation League Team Description Paper - Brave Circles - Infra-Structure Competition. In *Proceedings CD of the 14th RoboCup International Symposium*, July 2010.

[109] Mohammad.H Shayesteh, Mahdi Salamati, Sanaz Taleghani, Atoosa Hashemi, and Sara Hashmi. MRL Team Description Paper for Virtual Robots. Team Description Paper for the 2014 RoboCup competition, July 2014.

[110] Raymond Sheh, Tetsuya Kimura, Ehsan Mihankhah, Johannes Pellenz, Soren Schwertfeger, and Jackrit Suthakorn. The RoboCup Rescue robot league: guiding robots towards fieldable capabilities. In *Advanced*

Robotics and its Social Impacts (ARSO), 2011 IEEE Workshop on, pages 31–34. IEEE, 2011.

[111] Weihua Sheng, Qingyan Yang, Jindong Tan, and Ning Xi. Distributed multi-robot coordination in area exploration. *Robotics and Autonomous Systems*, 54(12):945–955, 2006.

[112] Victor Spirin and Stephen Cameron. Rendezvous through obstacles in multi-agent exploration. In *Proceedings of the IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR 2014)*, October 2014.

[113] Victor Spirin, Stephen Cameron, and Julian de Hoog. Time preference for information in multi-agent exploration with limited communication. In *Proceedings of 14th Towards Autonomous Robotics and Systems conference (TAROS 2013)*, pages 34–45, August 2013.

[114] Victor Spirin, Julian de Hoog, Arnoud Visser, and Stephen Cameron. MRESim, a multi-robot exploration simulator for the Rescue Simulation League. In *RoboCup 2014: Robot World Cup XVIII*, pages 106–117. 2014.

[115] Cyrill Stachniss, Óscar Martínez Mozos, and Wolfram Burgard. Efficient exploration of unknown indoor environments using a team of mobile robots. *Annals of Mathematics and Artificial Intelligence*, 52(2-4):205–227, 2008.

[116] Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

[117] Satoshi Tadokoro. *Rescue Robotics: DDT Project on Robots and Systems for Urban Search and Rescue*. Springer Science & Business Media, 2009.

[118] Satoshi Tadokoro, Hiroaki Kitano, Tomoichi Takahashi, Itsuki Noda, Hitoshi Matsubara, Atsushi Shin Joh, Tetsuo Koto, Ikuo Takeuchi, Hironao

Takahashi, Fumitoshi Matsuno, et al. The RoboCup-Rescue project: A robotic approach to the disaster mitigation problem. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 4, pages 4089–4094. IEEE, 2000.

[119] Sebastian Thrun and Arno Bü. Integrating grid-based and topological maps for mobile robot navigation. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, AAAI'96, pages 944–950, 1996.

[120] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT press, 2005.

[121] Maarten van der Velden, Wouter Josemans, Bram Huijten, and Arnoud Visser. Application of traversability maps in the Virtual Rescue competition. In *Proceedings of the RoboCup IranOpen 2010 Symposium (RIOS10)*, 2010.

[122] Sander van Noort and Arnoud Visser. Extending Virtual Robots towards RoboCup Soccer Simulation and @Home. In *RoboCup 2012: Robot Soccer World Cup XVI*, volume 7500, pages 332–343. 2013.

[123] Arnoud Visser, Francesco Amigoni, and Masaru Shimizu. The Future of Robot Rescue Simulation Workshop - An initiative to increase the number of participants in the league. Technical Report. University of Amsterdam, Politecnico di Milano & Chukyo University. Retrieved from `https://staff.fnwi.uva.nl/a.visser/publications/RCRS2016_paper_13.pdf`. [Online; accessed 19-June-2016].

[124] Arnoud Visser, Julian de Hoog, Adrian Jiménez-González, and José Ramiro Martínez de Dios. Discussion of multi-robot exploration in communication-limited environments. In *Workshop "Towards Fully*

Decentralized Multi-Robot Systems: Hardware, Software and Integration" at the ICRA Conference, May 2013.

[125] Arnoud Visser, Nobuhiro Ito, and Alexander Kleiner. RoboCup Rescue Simulation innovation strategy. In *RoboCup 2014: Robot World Cup XVIII*, pages 661–672. 2014.

[126] Arnoud Visser and Bayu A Slamet. Balancing the information gain against the movement cost for multi-robot frontier exploration. In *European Robotics Symposium 2008*, pages 43–52. Springer, 2008.

[127] Arnoud Visser and Bayu A Slamet. Including communication success in the estimation of information gain for multi-robot exploration. In *Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks and Workshops, 2008. WiOPT 2008. 6th International Symposium on*, pages 680–687, April 2008.

[128] Arnoud Visser, Bayu A Slamet, Max Pfingsthorn, et al. Robust weighted scan matching with quadtrees. 2009.

[129] Arnoud Visser, Xingrui-Ji, Merlijn Ittersum, Luis A. González Jaime, and Laurenţiu A. Stancu. Beyond Frontier Exploration. In Ubbo Visser, Fernando Ribeiro, Takeshi Ohashi, and Frank Dellaert, editors, *RoboCup 2007: Robot Soccer World Cup XI*, volume 5001 of *Lecture Notes in Computer Science*, pages 113–123. 2008.

[130] Paul Ward. *Coordinated Search with Unmanned Aerial Vehicle Teams.* DPhil thesis, University of Oxford, 2012.

[131] Paul Ward and Stephen Cameron. Coordination in multi-tiered robotic search. In *Towards Autonomous Robotic Systems - 12th Annual Conference, TAROS 2011, Sheffield, UK, August 31 - September 2, 2011. Proceedings*, pages 384–385, 2011.

[132] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[133] Briana Lowe Wellman. *Cooperation paradigms for overcoming communication limitations in multirobot wide area coverage.* PhD thesis, The University of Alabama, 2011.

[134] Briana Lowe Wellman, Julian de Hoog, Shameka Dawson, and Monika Anderson. Using rendezvous to overcome communication limitations in multirobot exploration. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pages 2401–2406, October 2011.

[135] Kai M Wurm, Armin Hornung, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems. *In Proceedings of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation*, 2010.

[136] Kai M Wurm, Cyrill Stachniss, and Wolfram Burgard. Coordinated multi-robot exploration using a segmentation of the environment. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 1160–1165. IEEE, 2008.

[137] Brian Yamauchi. A frontier-based approach for autonomous exploration. In *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*, pages 146–151. IEEE, 1997.

[138] Brian Yamauchi. Frontier-based exploration using multiple robots. In *Proceedings of the second international conference on Autonomous agents*, pages 47–53. ACM, 1998.

[139] Tomoaki Yoshida, Keiji Nagatani, Satoshi Tadokoro, Takeshi Nishimura, and Eiji Koyanagi. Improvements to the rescue robot Quince toward future indoor surveillance missions in the Fukushima Daiichi nuclear power plant. In *Field and Service Robotics - Results of the 8th International Conference, Tohoku University / Matsushima, Japan, 16-19 July 2012*, pages 19–32, 2012.

[140] Robert Zlot, Anthony Stentz, M Bernardine Dias, and Scott Thayer. Multi-robot exploration controlled by a market economy. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '02)*, volume 3, pages 3016–3023, 2002.