Multi-Robot Exploration and Planning in Limited Communication Environments

Vibhav Inna Kedege

MSc Thesis Report
October 2020 - July 2021

TU Delft

Multi-Robot Exploration and Planning in Limited Communication Environments

by

Vibhav Inna Kedege

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday July 5, 2021 at 10:00 AM.

Student number: 4998596
Project duration: October 1, 2020 – July 1, 2021
Thesis committee: Dr. ir. Frans. A. Oliehoek, TU Delft
Dr. ir. Arjan van Genderen, TU Delft
MSc. Ludo Stellingwerff, Almende B.V.

Abstract

Multi robot exploration is an area coverage strategy where a team of robots work together to explore and discover the contents of an environment. This is particularly useful in search and rescue missions. During such missions, a team of robots can be deployed into a disaster environment to map the surroundings and provide valuable information like the location of victims, heat sources and gas leak points to rescue personnel. Disaster environments however are most of the time completely unknown, cluttered, radio-hampered and GPS-denied. Thus robot teams need to deal with an environment that is previously unknown to them and has limited communication avenue. Limited Communication leads to robots not being able to share information with one another, which can lead to multiple robots exploring the same region of the environment. This redundant exploration results in increased time for search and rescue, that could prove to be costly.

Providing a robust solution to the above mentioned problem is the aim of this thesis. In this thesis, the planning algorithm of Monte Carlo Tree Search (MCTS) is utilised by multiple robots of the exploration team in a distributed manner to cooperate and explore an environment effectively. The environment is completely unknown to the team of robots and has restricted communication. Further, each robot cannot observe the state of the entire exploration environment and is limited to only observing its nearby/local surroundings based on its sensor range. Therefore exploration strategies that can deal with an unknown environment, limited communication and local observibility are studied.

For the purpose of exploration performance comparison, a multi-robot team is deployed into an environment under various scenarios that is based on the level of communication which is possible - No, Full and Partial. In addition to this, a fourth scenario is created where robots with partial communication are given the capability to predict the behaviour of team-members. From extensive simulation tests, it will be shown that partial communication can recover a significant amount of performance lost by removing the full communication dependency. Robots with peer prediction ability are found to help increase this recovery further in very specific situations. Further, it will also be shown that giving prior information about the environment obstacle locations to robots does not effect the exploration performance in a limited communication environment. Instead, increasing the number of chances of robots meeting one another and sharing information positively effects the exploration performance.

Contents

Acknowledgements

Ever since the start of my masters curriculum back in August 2019, I have been fascinated with the application of Embedded Systems in the domain of Artificial Intelligence and Robotics. I sincerely believe that a combination of all these areas can help create autonomous systems that support human activity. Particularly, frequently occurring natural disaster events like floods, earthquakes or forest fires have shown that using mobile robots to help rescue personnel can expedite victim rescue efforts and locate the cause of such disasters much faster. Thus there is a huge potential of using autonomous mobile robots in such an application. Upon further research, I found that Almende B.V. as well as the Interactive Learning & Decision Making (ILDM) research group of TU Delft were working on this topic and so I took up the challenge and started to pursue research in this domain as part of my masters thesis.

First and Foremost, I am thankful to all those involved from both Almende B.V. as well as the ILDM research group for their invaluable support throughout the process of my thesis. The regular weekly meetings with Aleksander Czechowski from the ILDM department gave me perspective on state of the art planners from previous research that have been used in the design of sequential decision planners. The regular meetings with Ludo Stellingwerff, Reka Berci-Hajnovics and Snehal Jauhri from Almende BV were insightful in reinforcing the idea of practical constraints that had to be kept in mind while designing distributed algorithms. I am also thankful to Ludo Stellingwerff and Reka Berci-Hajnovics for tips on how to design software that is segmented and easy to read. Further, the supervision provided by Frans A Oliehoek was very useful and particularly the seminars conducted during his COMARL and ELLIS-Delft Seminar Series gave me the opportunity to listen to state of the art research in the field of multi-agent systems from researchers from all over the world. Overall, the guidance provided by each supervisor mentioned kept me focused on the essential problem of limited communication environments and has shaped this thesis beyond words can describe.

Finally, I would like to also thank my friends and family for their belief and constant support. Particularly I would like to thank my mother Sushma Indrajit for her upbeat attitude, my late father Dr I K Indrajit for philosophies that have guided me till date and my late dog Ginger whose support cannot be left unmentioned!

Vibhav Inna Kedege
Delft, July 5th 2021

1

Summary

A team of robots working together to explore an environment is at the core of multi robot exploration. This is particularly useful in applications like region surveillance, space exploration and disaster management missions. In the case of disaster management or search and rescue missions, a team of robots usually encounter a situation where the area to be covered is completely unknown. Further, being a disaster zone there is high possibility that the area has restricted communication. An environment with restricted communication makes it difficult for robots to share information with one another. When robots do not share information then this hampers the cooperation between robots to achieve the overall exploration goal. This leads to an increase in the overall exploration time, with the same sub-region of the environment being explored multiple times by many robots. This is referred to as redundant coverage. Redundant coverage is particularly problematic in search and rescue operations, as an increase in the time for complete exploration would delay any further rescue work. A restricted communication environment also leads to robots having no information about the global state of the environment and robots can only observe the contents of the sub-regions of the environment that are within its sensing range (local observibility) . There is also no possibility to have reliance on any centralised unit of the environment and thus distributed approaches of exploration are required. Thus, when deployed into a disaster environment, robots of a team are required to be distributed, deal with local observibility and move around freely to explore larger portions of the environment. The environment itself has restricted communication and is unknown to robots.

After going through existing literature on the topic, it was found that there were many previous multi-robot exploration methods that solved the exploration problem through a distributed approach with cooperative agents having local observibility and the ability to move around freely in the environment. However among the two environment constraints of an unknown region and limited communication, most methods considered only one of these. The aim of this masters thesis however is to consider both these environment constraints and study the coverage capability of locally observable, distributed & cooperative robots on maps of various sizes.

To perform the exploration, robots planned their paths using the Monte Carlo Tree Search (MCTS) algorithm. This algorithm was selected due to its ability to plan paths by looking ahead into the future for a predefined number of steps. This could help direct the robots to move towards unknown regions for larger number of steps and could also prevent robots from getting stuck within regions that were cluttered with obstacles. The algorithm was implemented in a distributed manner where robots had information only of its nearby local surroundings. A significant focus was placed on how to deal with limited communication. This was done through the creation of various scenarios. The first scenario was of full communication and was the starting step where the works of previous authors was replicated. In this scenario robots could share information with one another irrespective of the distance between them. During the data sharing stage robots shared their position information, their belief of the map coverage status and planned paths. This information was used by each robot to update its belief of the environment and to plan its own paths. Considering the plans of the other robots resulted in cooperation between robots during exploration.

After implementation of the existing work was complete, in the next phase a scenario was created where the communication dependency was completely removed from the replicated work. This led to a scenario where robots could not share information with one another at any point of time. This scenario is referred to as No Communication and there was no cooperation between the robots. Following this scenario the aim was to bring in some form of cooperation between robots, while taking into account the environment physical limitations. With this idea in mind, robots were given the capability to communicate and share information only when they were within a predefined communication range. This scenario is referred to as partial communication. By using partial communication the number of steps required to explore an environment reduced and moved closer to the Full communication case. The next step was to test techniques that could further lessen the gap between partial and full communication. To do this, robots were given the capability to reason out and predict the position of peers robots that were beyond the communication range. This was done by utilising known information about peer robots as well as using computationally cheap heuristics. This is referred to as the partial communication with prediction scenario. Thus there were four scenarios that were created based on communication, namely full, no, partial and partial with prediction.

The total number of steps required for coverage for each communication scenario was compared, for various sizes of robot teams. Every experiment was performed for at least 10 times on 5 different maps. From the results it was found that each scenario with MCTS, including no communication could perform faster than a previous method on frontier based exploration which also assumed the same constraints as listed earlier. By introducing partial communication there was significant recovery of the lost performance when compared to completely removing communication. This lost performance was further recovered by using predictions, in very specific situations. Particularly in smaller team sizes exploring large environments, when robot communication range was increased then partial communication with prediction could further recover lost performance. It was also found that feeding in information of the contents of the map to robots before the start of exploration did not have any significant effect on the exploration performance. Instead, increasing the number of instances of robots meeting one another and sharing important information positively effects the exploration performance of the team.

2

Introduction

Mobile robots have been conceptualised to complement and help human activities ever since the beginning of the 21st century. The authors of the Journal of Unmanned Vehicle Systems in their work [10] mention that ever since 2004, there have been applications of mobile robots in crop survey, atmosphere data collection, forest fire investigation, radiation contaminated area monitoring, power lines inspection and pipeline surveillance. Following this, a number of survey papers to study these applications have been produced, like Dunbabin et al [17] and Perry et al [19] in the field of environment monitoring and Zhang et el [43] in agriculture. There have also been large efforts in evaluating the various algorithms that have been used in order to control the coordination between large teams of such mobile robots, like Schranz et al [37], Brambilla et al [8] and Liew et al [24]. Due to this potential of unmanned systems in various application areas, there are several organisations that have shown an interest in developing the efforts of collaborative robots to perform tasks. One such organisation is the COMP4DRONES consortium which is an Electronic Components and Systems for European Leadership Joint Undertaking (ECSEL JU) and brings together 50 partners from around Europe (indicated in figure 2.1) to provide a framework for safe and autonomous drones to customize their working for civilian services [32]. The objective of this consortium is to:

1. Ease the integration and customization of drones.
2. Enable drones to take safe decisions in an autonomous manner.
3. Ensure development that is protected against cyber-security threats.
4. Minimize the design and verification efforts for drone applications.
5. Ensure sustainable impact.
6. Create an industry-driven community

The consortium aims at applying the above objectives in use cases like transport, construction, logistics, surveillance & inspection and agriculture [32]. The use case of transport is to use drones to monitor devices, road traffic and infrastructure in order to perform faster detection and early response of incidents. In construction, the aim is to use drones during the construction phase of civil infrastructure to keep track of the progress without causing interference to the development of activities undertaken in the work. The logistics use case will test drones in its capability to deliver equipment in hard to access areas like forests. It also deals with the transportation of equipment, drugs and blood samples inside large hospital territories. Under the Surveillance & Inspection use case, the aim is to utilise drones in inspecting off-shore infrastructures like windmills, and also in mapping disaster sites. In the use case of agriculture, the focus is on crop monitoring and crop and health management.

As seen from figure 2.1, one of the partners of the consortium is Almende B.V and TU Delft. The use case that the 2 have focused on is the surveillance & inspection aspect and more specifically in disaster management application. Due to this, the focus of this thesis is on the inspection use case. The use case is focused on using a team of drones in the mapping and assessment of an indoor disaster

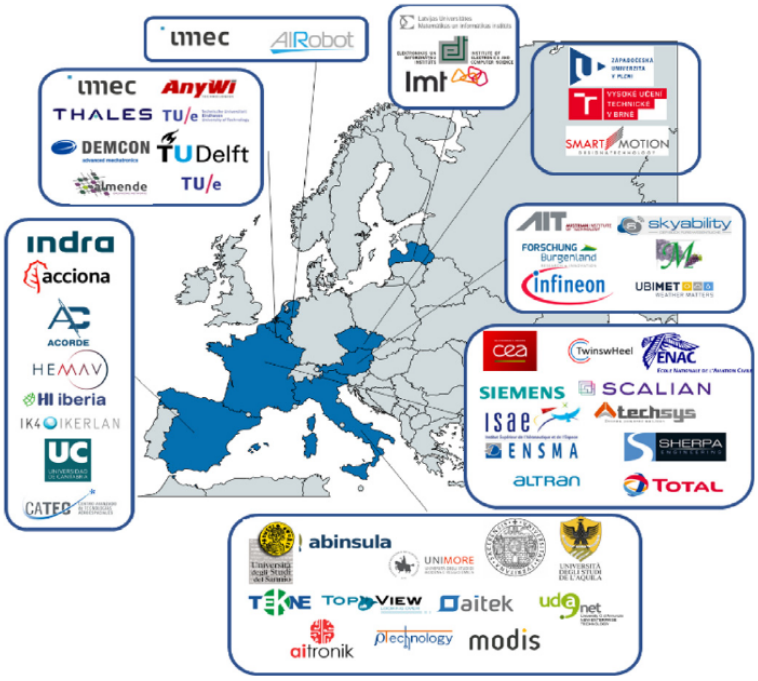


Figure 2.1: COMP4DRONES Partners [32]

environment. Such indoor disaster environments can be unsafe for humans and thus having a team of robots perform exploration can be a safer option for rescue personnel.

The exploration scenario defined by the COMP4DRONES consortium consists of a fleet of ground and aerial robots navigating and mapping an unknown environment where there is no GPS signal available [32]. The scenario consists of a mobile control station near an unknown, cluttered, radio-hampered and GPS-denied site. From the mobile control station a wheel based rover consisting of aerial drones is sent to the site. The rover acts as the reference point for all aerial drones exploring the environment. The entire system, rover, control station and aerial drones are used to create a common map of the environment consisting of various points of interest such as victim locations and heat sources. This map, created from the point of view of various drones is used to find a safe route to access these points of interest for time critical purposes like victim rescue. The system is also required to provide a live view of the exploration to the control station and must be flexible to cope with a changing environment or failure of a part of the system. From this the objectives of the use case are to reduce the cost of surveillance, increase the reliability via an automated process, increase the frequency of surveillance, detect anomalies automatically and reduce the manned entries to the confined/hazardous spaces.

2.1. Research Focus

From the use case specified in the previous section, it can be summarised that the system to be studied is a fleet of drones used to map an unknown area to monitor hazardous regions, find safe passageways and identify victims to be rescued which will provide rescue workers vital information that would help in rescue efforts. The challenge however is that the environment is GPS denied with limited communication. In such GPS denied cluttered environments drones would not know the exact position of peers in the environment. Thus, limited communication constraint raises two major issues for the system [3]. The first issue relates to the joint knowledge of the environment during the exploration mission. This joint knowledge includes map coverage status by the agent team and the positions of the agents in the environment. With stable communication across the environment, this joint knowledge is accessible by each robot at any time. However in a limited setting, robots will be unable to access the information of all neighbours and will possibly only be able to exchange data with peers under very specific situations. The second issue is in cooperative path planning algorithms of each individual robot that maybe re-

quired to track the movements of peer robots. Due to this tracking inability, any cooperation method will be affected and the amount of redundant coverage increases thereby increasing the number of steps to explore an environment. Such an increase in the number of exploration steps of the team leads to a delay in the rescue efforts of victims and search for points in the disaster zone to contain. Multi-robot cooperation and safe autonomous navigation in an unknown & communication limited environment is thus a challenging and urgent problem that requires to be solved and is the focus of this masters thesis. The Research focus for the thesis can thus be summarised as,

> To design distributed cooperative planning strategies for multi-robot exploration systems that can be deployed in disaster management situations, by considering the lack of prior environment information and restricted communication that exists in such time crucial missions.

From this objective we can extract the main aspects of the system that gives a direction for the literature survey, implementation and experiments.

Multi-Robot System - Exploration using a single agent can be time consuming and especially as the exploration area becomes larger. In a search and rescue mission where time is of utmost importance, having multiple robots is extremely beneficial.

Cooperative Strategies - As there are multiple agents present, each robot must choose paths based on the strategies of its peers. This leads to robots executing actions based on team behaviour rather than executing actions that would only benefit itself.

Restricted Communication - As the environment is in a disaster zone, robots are required to deal with no communication with one another.

Unknown Environment - Robots are not given a map of the environment prior to exploration and thus need to discover paths as they move around the arena.

Distributed Approach - There is no centralised system that assigns robots goals and also no leader that robots can follow. Thus, robots need to follow a peer-to-peer network and perform exploration.

2.2. Research Questions

Considering the research focus as well as the main aspects from the focus, we can frame the primary research questions that will be answered at the end of the thesis. These are:

1. **RQ1**: What distributed strategies can cooperative agents utilise to explore unknown environments with limited communication, in minimal number of steps?

2. **RQ2**: What is the effect of limited communication on the cooperative exploration strategies?

3. **RQ3**: What strategies could help in replicating the performance of agents as in an arena of full communication?

2.3. Thesis Structure

In terms of the structure, the thesis report has already laid the focus of research in this chapter. The next chapter, chapter 3 summarises important background knowledge that is required to understand the remaining parts of the thesis. Chapter 3 also describes various methods used previously to solve the problem of multi-robot exploration. In chapter 4, the algorithm design approach to implement the baseline as well as additional scenarios of this study are explained. Chapter 5 is the section on experiments that explains the simulations that were performed to answer the previously defined research questions. The chapter ends with a discussion page that summarises important observations. Chapter 6 concludes the thesis by answering the research questions and lays down a direction for further research. The thesis also presents an Apppendix in chapter 7, that presents the statistical significance tests for certain data populations and the maps that were used for the experiments. At the end of the thesis, chapter 8 presents the thesis research in a concise scientific paper format.

2.4. Naming Conventions

It is important to note that throughout the thesis, the terms robot, drone, agent and rover is used interchangeably, and is used to refer to a single robot. Further, as it is a case of multiple robots there are many parts where the interaction between multiple agents is explained. In such explanations, the term current agent is used to refer to an agent that is being considered or explained about, while the term peer agent is used to refer all the other agents except the current agent.

3

Background and Related Work

In order to study existing methods that have been used to approach multi-robot exploration, a literature survey was performed first. In this literature survey, the study was narrowed down to algorithms in which the agents of multi-robot teams adhered to one of the following constraints of having no prior information of the environment or dealing with limited possibility for communication & data sharing. As mentioned in the summary in chapter 1, most methods only consider one of these constraints. The algorithms/methods studied had either one of these constraints present and these can be grouped into 5 broad classes:

1. Robot Dispersion
2. Cost and Utility
3. Scene Partitioning
4. Game Theory
5. Sequential Decision Planning

Each class of methods have been described in the following sections. Before explaining each method, section 3.1 lays the foundation by highlighting the advantages of using autonomous multi-agent systems that follow a distributed architecture. In section 3.2 robot dispersion techniques are discussed where robots having no prior information about the area, scatter around and explore it. In section 3.3, cost and utility approaches are discussed where robots first discover certain positions in the area that are more beneficial in terms of exploration and subsequently negotiate with one another for the assigning of positions to team members. Section 3.4 summarises techniques where the entire region is divided into sub-regions before any exploration takes place. Each sub-region is assigned to a single robot in order to reduce redundant exploration of same regions by multiple team members. Section 3.5 relates the field of game theory to the task of exploration and summarises algorithms where different robots cooperate and achieve equilibrium positions that is beneficial to the overall goal of environment exploration. In Section 3.6 sequential decision planners have been discussed, along with how fundamental Artificial Intelligence (AI) ideas like Markov Decision Processes (MDPs) have been used in path planning algorithms previously. In such techniques, robots plan their current actions by looking at the consequences of their actions for a certain horizon of time into the future. In section 3.7 a review on peer modelling methods, where agents model one another have been described. In section 3.8 a discussion section that reviews the algorithms in terms of their ability of handle the constraints mentioned in section 2.1 is discussed.

3.1. Background

In this section, the motivation behind utilising autonomous multi-agent systems have been described. As previously mentioned, robotic exploration has been defined as the problem of a robot navigating through an unknown environment and maximising the knowledge or representation of a particular

area [25]. Area exploration is crucial for applications like space exploration, investigation of biological species, disaster warning systems and search rescue missions. In most of these scenarios however the environment is highly dynamic and reliable communication with human operators is not always guaranteed. Therefore the robot needs to be autonomous in order to facilitate crucial functionalities like safe navigation, map building and victim detection. In the specific case of search and rescue operations, an autonomous robot equipped with sensors like camera and LIDAR can be deployed in the environment in order to find objects/locations of interest, for example fire spots in a forest fire, concentration of hazardous materials and victims in disaster areas [39]. In very large environments however using a single robot to scan the entire area or volume is not enough. This becomes a matter of concern, especially in search and rescue operations where reducing search time is important. Therefore in such applications where minimum exploration time of larger regions of search is desirable, using multiple robots is appropriate and beneficial. Most robots have wireless communication capabilities and therefore in a multi-robot setting there is possibility of having important information being communicated amongst members of the robot group. This leads to the following advantages that multi-robot systems provide [30]:

1. Reduced Latency - If tasks are decomposed in a parallel manner, then the overall goal can be performed in a lesser amount of time.

2. Distributed Sensing - The range of sensing of a group of robots is wider than that of a single robot and there is also the possibility of having multiple views of the same environment.

3. Distributed Action - A group of robots can actuate in different places at the same time.

4. Fault Tolerance - Certain system architectures (which will be discussed) provide system redundancy where the failure of any single entity does not lead to failure of the entire system.

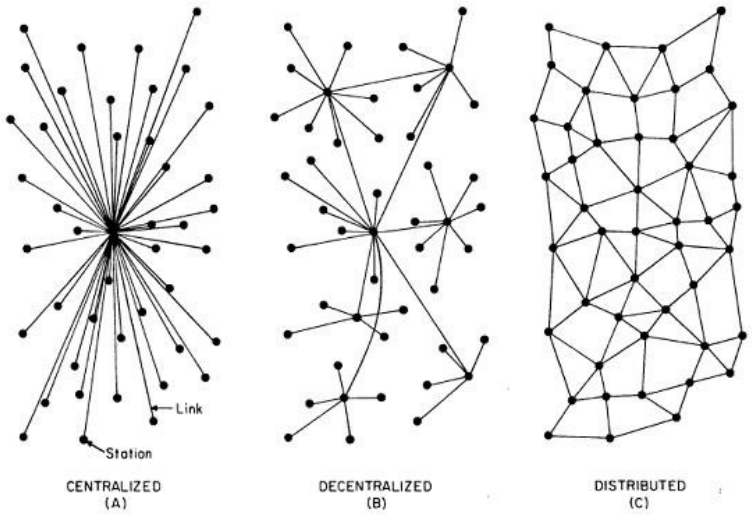


Figure 3.1: Centralised, Decentralised and Distributed Architectures [4]

On a high-level, multi-robot systems can be modelled as nodes that can follow three possible architectures - Centralised, Decentralised or Distributed. This can be seen in figure 3.1, where the nodes represent robots and links between the nodes represent the existence of a communication channel. The centralised architecture has a single master node onto which multiple nodes are connected to via a link. While it is easier to implement this in software for small robot teams, it causes high dependence on the single central node. If the single central node fails, then it results in the collapse of the entire system. This is in contrast to the distributed architecture in which there is no centralised master unit, but instead there is peer-to-peer communication. This gives following advantages [**Brambilla12swarmrobotics:**]:

1. Robustness - The ability to cope with the failure or loss of an individual node.

2. Scalability - The ability to perform well with different group sizes and not be affected by the introduction or removal of individuals.

3. Flexibility - The ability to cope with a broad spectrum of environments and tasks.

A middle ground between the two architectures is the decentralised architecture where there are multiple master nodes, with no complete reliance on a single unit.

Thus it can be seen that autonomous multi-robot teams that follow a distributed architecture are highly beneficial. The benefits lead to a robust, scalable and flexible system that reduces the time for exploration, which is very useful in a search and rescue scenario.

3.2. Robot Dispersion

In robot dispersion, robots are treated as nodes that are deployed into an environment, that is unknown to the robots. These robots begin the search at the same location and spread out during the exploration process. Each robot spreads out while maintaining communication and tries to maximise the amount of area searched [20]. This can be understood from figure 3.2. In the method presented by Howard et al in [20], this has been done by considering an artificial potential field algorithm where each node exerts a virtual force that causes nodes to move away from each other. The maximum distance between the nodes is limited by the communication range of each robot, thus ensuring that all the nodes are connected. The process of moving away from one another is also possible by considering WiFi intensity signals as has been done in by Ugur et al in [41]. In this method, wifi signal intensity depends on the distance and the orientation of the robots. Robots move away from one another when their distance is lesser than a minimal threshold value.

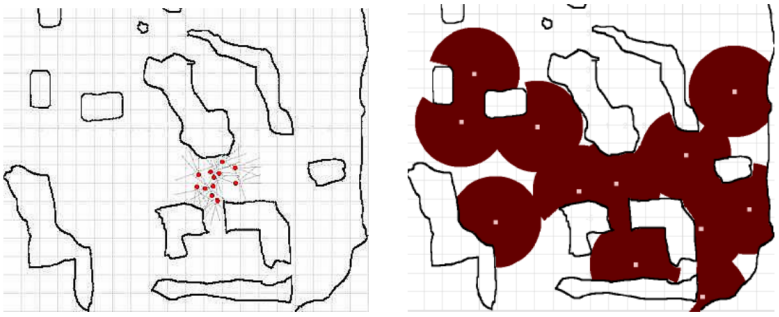


Figure 3.2: Robot Dispersion [16]

In some methods instead of having robots perform the task of only exploring new regions, robots also have the possibility to relay communication signals which maintains inter-robot connectivity. In the method presented by Steven et al in [16], 2 methods namely the Clique-Intensity Algorithm and the backbone dispersion algorithm achieve this.

Clique-Intensity Algorithm

This algorithm is specifically designed for a distributed homogeneous swarm. Each robot has position information about its peers in the form of a graph where the nodes represent robots and the signal intensities between the robots act as weights. This type of representation is referred to as a connectivity graph. A clique is defined as a graph or a sub-graph where every node is connected to one another. In addition, a maximal clique is defined as a clique which is not a sub-graph of another clique. Figure 3.3 shows an example of a connectivity graph along with the cliques. Note that each clique is also the maximal clique in this figure. While dispersing around the area, robots form maximal cliques of sizes 2 or 3. For each clique a single robot is chosen to remain stationary in movement. The remaining robots in the clique continue to disperse around the area while maintaining connectivity with the stationary robot. The dispersion continues to occur until a situation of equilibrium is reached where no further robot movement can take place.

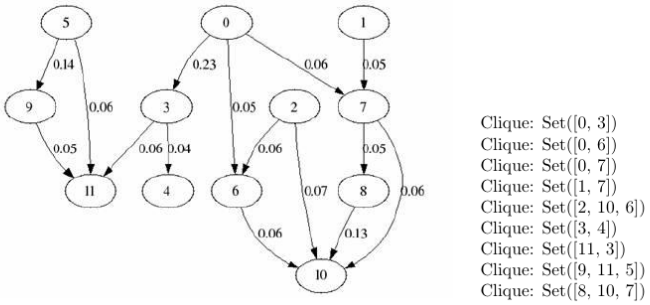


Figure 3.3: Connectivity Graph [16]

Backbone dispersion
In this algorithm each robot can take up the role of a stationary or wanderer robot. The idea is to have a group of closely connected stationary robots, called backbone onto which every wanderer robot maintains direct contact. Wanderer robots are allowed to move around freely. There is always possibility to increase the size of the backbone, by assigning shifting the role of a wanderer robot to stationary. This is done primarily to cover further areas. Wanderer robots thus need to decide when to move freely and when to shift roles. The manner in which this is done is shown in figure 3.4. The core idea behind the actions as presented in the table is to keep each wanderer robot connected with the backbone network. Thus in the cases where the wanderer robot is not in contact with any robot in the backbone, the robot itself becomes part of the backbone network.

Robot is in direct contact with:	Action	Goal
Two or more robots, at least one in the backbone	Wander randomly, avoiding obstacles	Find uncovered areas
Exactly one robot which is in the backbone	Remain stationary until another robot wanders into range	Avoid losing contact with the backbone
No robots in the backbone and at least one not in the backbone	Join the backbone, and send a message to one neighbor telling it to join the backbone	Extend the backbone to an uncovered area
No robots at all	Wander randomly, avoiding obstacles	Regain contact with the backbone

Figure 3.4: Wanderer robot behaviour in Backbone dispersion [16]

The authors ran the Clique-Intensity Algorithm in a 25 square metre arena with 12 robots having a maximum clique-distance as 5 metres, and a robot sensing radius of 2.5 metres. The experiment was run for 10 times and the robot reached the equilibrium position each time (as seen in figure 3.2) in 60 to 100 seconds [16]. The agents in the same arena were also run with backbone dispersion algorithm and they reached an equilibrium position in approximately the same amount of time. However in this case, there was always a possibility for a robot that could not communicate with its teammates to remain stationary forever.

In another class of methods under this umbrella, the length of the path that each robot travels is modified based on the environment and neighbors. In the method proposed by Bao Pang et al in [34], a robot varies its path length based on the density of the robots in the neighborhood that it perceives. The aim of doing this is to reduce the chances for redundant searches in the environment by multiple robots. When the robot density is high, then each robots searches a small area around it by taking smaller steps. When the density is low however, each robot searches a larger area with a large step size. In order to ensure that the robots explore their own separate areas a robot that encounters another robot in front moves away in the opposite direction. The robots have no information of the environment and do not communicate with one another. In order to detect surrounding density each robot considers its peers to be physical interference from which it must maintain distance. Each robot estimates the density by considering the time delay value from physical interference. A smaller value of this time delay indicates a higher density. The authors utilised this method in an arena without any obstacles

to locate and reach target objects. There were a total of 6 robots and 6 targets. On performing the experiment the proposed method was able to search for targets in 5 minutes which was faster compared to previous methods on random movement, namely the Levy Distribution method (that took 7 minutes) and Brownian Motion Foraging (that took 15 minutes). The authors mention that though the method empirically shows to work faster than other prior methods, the proposed method still requires theoretical analysis that is part of a future work.

3.3. Cost and Utility

In the previous section, methods on how to disperse robots around the surrounding were discussed. However, none of these methods took the environment contents into consideration. In this section cost and utility based approaches are considered that direct robots towards points that are beneficial for the overall system. In the case of multiple robots, many robots can detect such beneficial points. This leads to a situation where robots are required to negotiate and choose points to explore. Such an idea is the motivation for the implementation of centralised Multiple Robot Task Allocation (MRTA) methods in works like [42] and [38], where cost and utility of important points are calculated and assigned in a centralised manner. The same can also be implemented in a distributed setup where robots share their detected points with one another and each robot runs its own task allocation method. This has been done in [6]. In each case, the negotiation is done through the computation of cost and utility of points. The idea of using cost and utility is to compute the revenue (as given in equation 3.1) of a particular point/task for every robot and to assign the robot the point/task where the revenue is the highest.

$$Revenue = Utility - Cost \tag{3.1}$$

Usually the cost is computed as the distance of a particular point of interest from the robot [42], [38] [6], while the utility gives the measure of benefit of a point. The benefit can be computed has been computed in prior works in many ways. For instance utility is computed as the amount of information the robot discovers by Umari et al in [42]. In another method by Benavides et al [6], points where robots are able to meet and share information are given a higher reward. The points of the region that is negotiated is common in these methods. The methods make use of finding frontier points. Frontier points are those parts of the environment that lie on the boundary of known and unknown regions. Thus, moving to such points is highly beneficial for the exploration system to discover new regions.

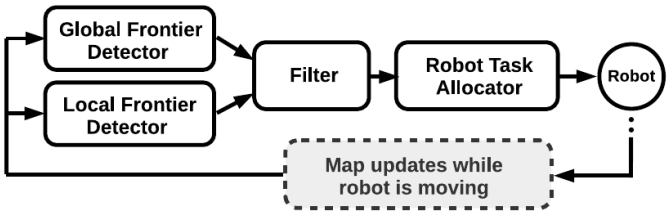


Figure 3.5: RRT Exploration overview [42]

The method proposed by Umari et al [42] follows the idea of looking for frontier points. There are 4 important blocks in the method that are as shown in figure 3.5. In the figure, the Global Frontier Detector, filter and assigner run on a centralised unit that is not a robot. The Local Frontier Detector runs on each robot. Each robot only knows the size of the region it needs to explore. This is represented as a bounding box. The bounding box is divided into 2D occupancy grid cells which can take the values of free, obstacle, unknown or frontier cell. This 2D grid representation can be seen in figure 3.6 where the region is divided into small squares. Prior to exploring the environment, robots assume that each grid cell is unknown. The authors then perform a novel method of Rapidly Expanding Random Tree search technique to find frontier points. This method consists of the robot being at a stationary point and expanding/growing tree to look for unknown points which are then used as targets/tasks for further processes. The process as in figure 3.5 starts with a local and global frontier detector, where frontiers are detected first. The number of frontier points detected is usually very high in number. Therefore a 2nd

phase is present that consists of a filter to extract frontier centroid points. Once the points have been filtered, a fourth phase called the assigner assigns goal positions to each robot. This assignment of goal positions is done by the assigner through a market based approach where the assigner computes the revenue (R) of a goal for each robot (equation 3.1). In equation 3.1, the information gain is computed by calculating the area of the region that is expected to be explored for a frontier point as in figure 3.6, while the cost is the navigation cost and is the euclidean distance of robot and the task's position. Each frontier point is then assigned to the robot that gives the highest revenue. The authors mention that such RRT based exploration methods are useful in 3D environments where camera based navigation is likely to fail. However this is left as a future work by the authors.

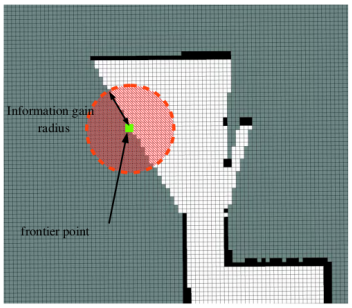


Figure 3.6: Information Gain [42]

This method of frontier exploration using RRT has also been extended to a leader-follower case by Nair et al in [29], where the global frontier detector, filter, task allocator and map-merger are run on one of the robots (called robot 1). The remaining follower robots run local detectors, move around and share their maps. This can be seen in figure 3.7. The work studies the impact of frequency of communication between robots on the overall exploration. It was found that beyond a certain frequency (0.5Hz) there was no significant impact on the exploration time of 3 robots.

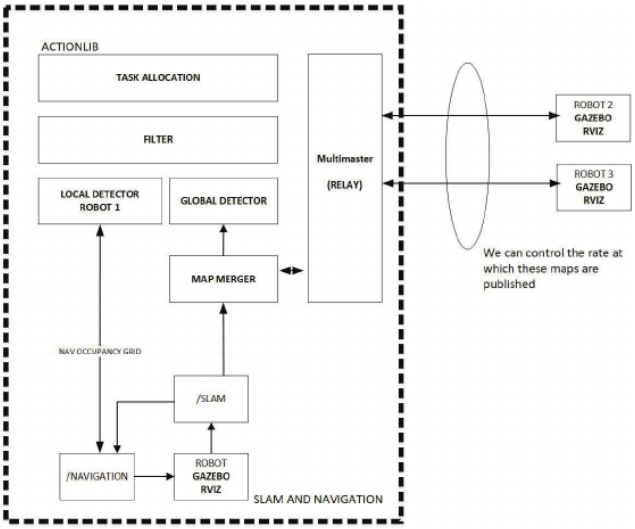


Figure 3.7: Leader-Follower RRT [29]

In each of the frontier methods mentioned above, communication occurs at all times and robots communicate with one other irrespective of distance or number of obstacles in between. Some other methods however consider the impact of these physical constraints while measuring the revenue of a particular task. Once such method is proposed by Benavides et al in [6] where multi-robot cooperative exploration systems in constrained communication environments have been discussed. Similar to the

previous two methods, in this method the size of the bounding box W is known by each robot of the team. Further W is divided into an occupancy grid structure where each cell has a probability value of $0, 1, 0.5$ corresponding to the status of $free, occupied, unknown$. Each robot also detects frontier points which are the tasks that are allocated based on a revenue equation similar to equation 3.1. The specific equation used here is given by,

$$\text{utility}_i\left(T_j\right) = -\alpha\text{pathCost}_i\left(T_j\right) + \beta\text{connectivity}_{T_j}\left(\bigcup_{k\neq i}\{R_k\}\right) \tag{3.2}$$

In the above equation α and β are the tuning parameters such that $\alpha, \beta \in [0, 1]$ while $pathCost$ and $connectivity$ act as the cost and utility respectively for a given task location. $pathCost$ is defined to be the Euclidean distance of the present location of the robot to the task location. $connectivity$ is the measure of how connected a robot would be with the rest of the members of the group at a particular task location. The authors conducted exploration experiments by using 2 robots in a 16x30 grid arena.

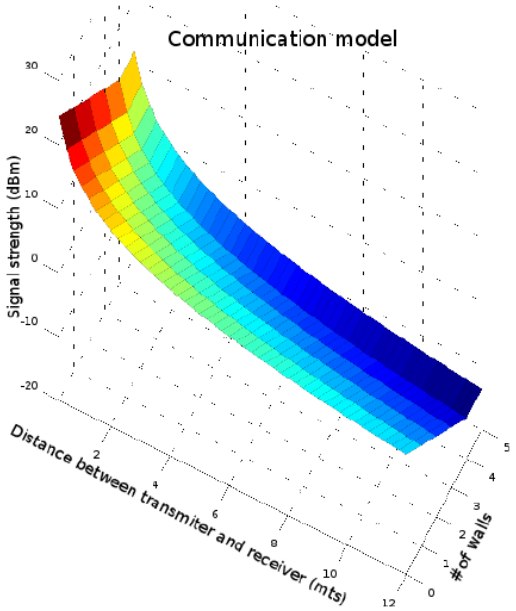


Figure 3.8: Communication Constraint [6]

The authors found that by their method robots could maintain connectivity for information exchange more than previous benchmarks. However they also acknowledge that increasing the chances to keep agents connected increases the total exploration time. As a future work, they mention that keeping separate relay robots to facilitate communication could help increase exploration time.

The previous method of Benavides et al [6] focuses on maintaining communication at every moment of time. In some other methods however, the approach taken does not require constant communication between teammates. An example of this is the method proposed by Dadvar et al [15], where communication only occur at very specific moments of time and not between all the robots. The authors propose 2 teams of robots. The first team of UAV's (called Hunters) are used to search for task locations while a team of UGV's (called Gatherers) are used to move towards the detected task locations. Communication can only occur between these two complementary teams using a communication platform that is referred to an "online board" through which gatherers notice the location of the new task detection's. The new task locations are announced by the hunters. The authors mention that the coordination through the use of the "online board" between the 2 complementary teams is necessary for effective exploration and especially in environments with complex obstacles and narrow corridors. As a future work, a definition of the cost function that considers the communication burden between agents is proposed by the authors.

3.4. Scene Partitioning

The idea behind the scene partitioning technique is to divide the exploration environment into sub-regions and assign a sub-region to each robot to explore. This can be seen in figure 3.9. The aim of doing this is to reduce the redundant coverage. Further, having the robots explore sub-regions also allows robots to move around a part of the arena independently of the other robot neighbours for a moment of time. This can lead to the reduction of the need for communication between the robots at all time steps [25]. In most techniques once the sub-regions have been allocated, robots explore it using techniques like frontier detection.

Lopez et el in their method [25] have also followed this method with the motivation of reducing com-

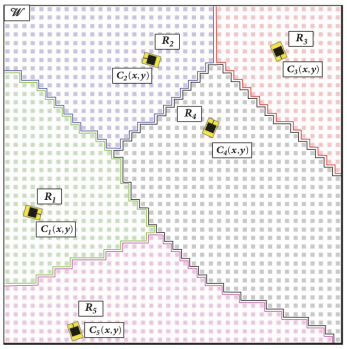


Figure 3.9: Scene Partitioning [25]

munication and coverage redundancy. The method considers the space to explore as a bounding box W that is divided into grid cells. Every grid cell in the map is associated with a sub-region. There is no communication constraint between the robots and each robot is able to share its location and map with one another irrespective of the presence of obstacles or distance within the map. There are 4 major modules that are utilised in the exploration process. These are:

Information Sharing module
A robot shares information about its position and exploration map with other robots. The map information is shared as occupancy grid data and the locations of all robots are sent as a list $V_i = \{C_1, C_2 ...., C_r\}$.

Zone Assignment module
Each robot i is assigned a list $E_i = \{c_1, c_2 ...., c_n\}$ of the grid-cells that it needs to explore. The list consists of those unknown grid-cells that are closest in terms of euclidean distance to robot i.

Data Acquisition module
Each robot uses a laser range finder (LRF) to acquire information about the state of each grid cell. Each grid cell can take one of the values of $\{unexploredcell, exploredcell, staticobstaclecell, mobileobstaclecell\}$.

Exploration module
Figure 3.10 shows the overall working of the exploration module. The module is used to guide the robots towards unknown zones of the scenario and consists of the goal assignment, path planning and trajectory execution sub-phases. In the goal assignment sub-phase, goal cell positions from the set E_i that are nearest to the robot is assigned as the next goal position. In the path planning phase, A^* algorithm is used to plan a global path P_i from the current position to the goal assigned in the previous sub-phase. In the trajectory execution phase, a genetic algorithm is used in order to control the linear velocity and angular velocity required to move the robot along the planned path P_i.

A high level working of the algorithm that includes the above phases can be seen in figure 3.11. The exploration process starts by first sharing map and position information between the robots. Following this step, zones or grid cells that the robots are required to explore are assigned to the robots. The

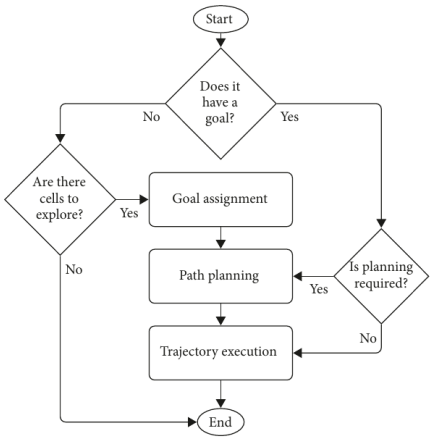


Figure 3.10: The exploration module [25]

robots then begin the exploration phase and data acquisition phase by moving around and using their Laser Range Finders to better understand the environment. The data acquisition and exploration phase occurs until all the goals assigned to a robot are explored. When the zone assigned to the robot is explored, then the robot is assigned a new zone and when the entire bounding region W is explored, the exploration by the robots stop.

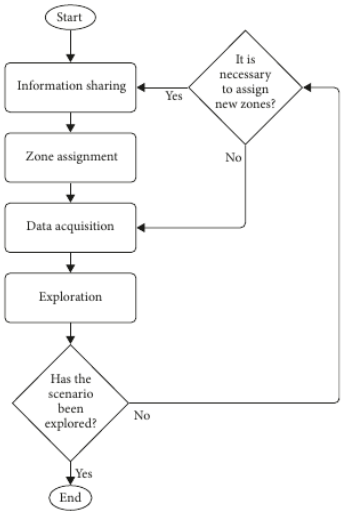


Figure 3.11: Four phases of Scene Partitioning in [25]

The above method was verified by the authors to work and it performed better than previous methods like random frontier and nearest frontier methods when the number of robots was greater than 6. The method is decentralised, modular and can deal with dynamic obstacles. As a future work, the authors aim to work on running the robots on a scenario where the dimensions of the environment are unknown and also aim to include localisation uncertainty caused by the sensor noise of the robots.

According to the authors of upma et al in [22] however, the above method has a drawback of requiring high computational cost as the distance of every cell from each robot needs to be computed in the zone assignment phase [22]. Instead, upma et al use a process called Ad-hoc partitioning in their work [22]. In this work, the environment boundary (known by the robots prior exploration) is divided into sub-regions that are equal in number to the robots. This is shown in figure 3.12. This division is done by the first robot that enters into the arena and the robots search for frontiers within their sub-region that they explore and do an exploration in this sub-region.

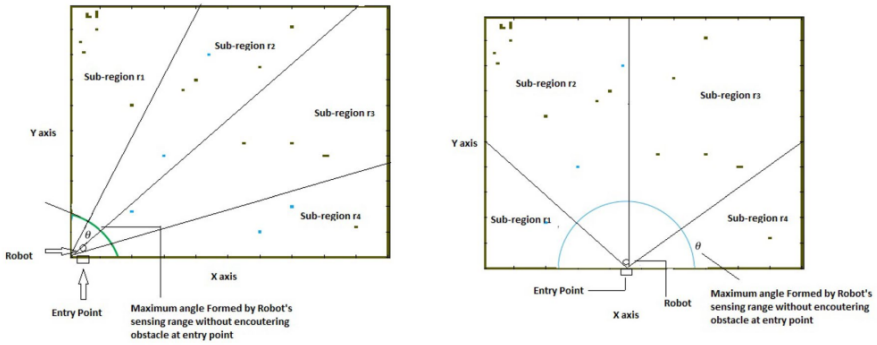


Figure 3.12: Ad-hoc Partitioning [22]

3.5. Game Theory

Ideas from the field of game theory have also been used in surveillance and target detection using multiple robot systems in previous literature. This is due to the similar underlying ideas in multi-robot systems and game theory. In coordinated Multi-robot systems, each robot can be modelled as an individual decision maker that moves based on the strategies of teammates. In game theory, strategic interaction between multiple decision-makers is described and analyzed [28]. Further, strategic interaction activities (commonly called games) give maximum possible outcome for self while at the same time predicts the rational decision taken by the others. This idea can be useful in a scenario when there is limited avenue for communication.

3.5.1. Game Theory Fundamentals

Some important fundamentals of game theory are summarised in this section and have been taken from work of the authors Mkiramweni et al in [28]. The fundamental terms have been summarised in table 3.1. In any game, players are the entities or individuals who make decisions and perform actions.

Term	Description
Players	Entities or individuals who make decisions and perform actions
Actions	Moves taken by players in a particular game
Strategies	a complete plan of actions throughout the game
Payoff	Reward that the player receive at the end of game

Table 3.1: Game Theory Terms [28]

Actions are the moves taken by the players in a particular game. Strategies are the description of how a player could play a game and is a collection of the plan of actions. These strategies can be mixed or pure. In a mixed strategy, the probability distribution for all possible actions of a player in a situation are specified while in a pure strategy, a player takes a unique action in a given situation. The Payoff is also the reward that the players receive at the end of the game which is dependent on the actions of all other players. In addition to these terms another key terminology is the Nash Equilibrium. Nash equilibrium is defined as the solution of the game that provides the highest (or most beneficial) payoff to a given player while keeping the strategies of other players constant. In such an equilibrium condition there is no benefit or incentive for the agent to change its action, as it would lead to a lower payoff.

3.5.2. Game Theory in Multi-robot exploration

In the context of multi-robot exploration, one of the earliest techniques involving game theory is of Meng et al in the work [27]. In this algorithm the authors assume an Urban Search and Rescue (USAR) case where there is unreliable communication. The region to be explored is divided into sub-regions before exploration and each sub-region has an associated probability of the existence of a target. A target is the location of a possible victim or source of the disaster (As in figure 3.13). The authors mention that this target probability can be supplied by the human operators or another heuristic function and can reduce the time of search. Thus given a coarse map of the searching area and prior knowledge of the

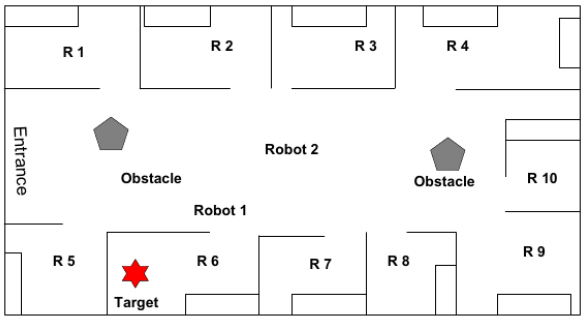


Figure 3.13: Prior Target in Region information [27]

target distribution, the method aims to find an efficient and robust searching strategy for multiple robots so that the expected average searching time is minimized [27]. In order to perform this, 2 states (busy and free) are defined where busy is when a robot is searching inside a region, otherwise it is free. A new event is defined as the instance when the robot enters a new region or finishes searching a current region and triggers the update of a robot state. Only when an event occurs is when robots communicate with one another. This reduces the communication overhead. This communication is independent of distance and the robots can communicate at all locations within the map. Further, whenever a robot finishes searching a location and it finds no target, then it updates this in its prior probability tracker. The data that is communicated amongst the robots are target selection decision, sub-region target existence probability, time taken to travel from the current location to each sub-region and the time taken required to cover each sub-region. Each robot then uses a utility function that considers this data. The authors then propose a greedy strategy as well as a Nash Equilibrium strategy that assigns sub-regions to each robot from the individual robot's perspective. In the case of a greedy strategy, each robot chooses a sub-region with the highest utility value given by the utility function. In the Nash Equilibrium case however sub-regions are selected by the agent based on overall benefit given by the nash equilibrium. The authors implemented each strategy in a distributed manner and found that the nash equilibrium approach had better search performance and was more robust to handle environment uncertainty [27]. However, the authors also mention that as the computation of the utility involves all the peer agents, therefore with increase in the number of agents the computation also increases.

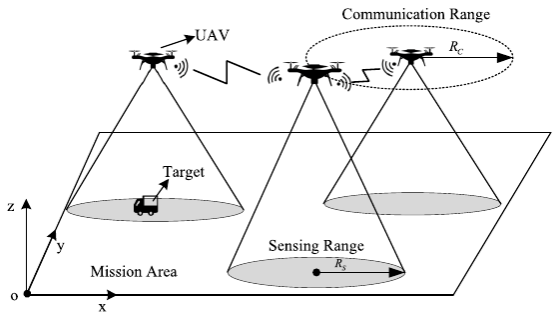


Figure 3.14: Framework used in the method of [27]

In another method by the Ni et al in [31], the problem of performing coordinated surveillance of the region using a team of UAV's has been performed. Similar to the previous approach [27], the assumption here is that each robot has knowledge of the prior distribution of targets in the environment, that is divided into smaller 2D grid-cells. Each UAV has limited sensing & communication range R_{C_i} and is assumed to move on a fixed plane above the search space, as shown in figure 3.14. The position of each UAV is projected onto the 2 dimensional environment and is recorded as the UAV position. A network of the UAV's is then formed as an undirected dynamic graph which is given by $Net(t) = (E(t), V)$, where the robot represent the vertices and an edge exists between 2 UAVs if the distance is smaller that the communication range R_{C_i}. For all the robots present in this graph, a

utility function is computed. The utility function computes the effect of choosing action i on the overall coverage of cells in the 2D grid-cell region. The strategies of the neighboring peer agents is also known and thus a nash equilibrium is computed. To select a suitable action, the authors select a UAV at random and utilise a modified binary log linear learning algorithm that they mention can find the nash equilibrium quickly. The modified binary log learning algorithm is used to select the next possible action for a UAV. The process of collision avoidance and selecting actions is performed in many iterations and in every iteration one of the UAV is selected randomly while the other UAV's retain the action from the previous iteration. The iterations continue to occur until the mission space Ω is completely searched or the number of iterations of the algorithm reaches a maximum predefined limit. The authors conclude that this method can achieve the Nash equilibrium faster and can also prevent UAV's from moving to zero-utility areas. They however also mention that the search task in environments with moving obstacles is yet to be tested.

3.6. Sequential Decision Planning

In the previous chapter, game theory methods were discussed where agents would choose actions based on the computation of a utility value. The utility value in the methods was only computed for that step alone. This section builds upon this idea of using utility or reward functions and discusses sequential decision planners. The utility is computed based on the sequence of actions that the agent performs [36]. In the multi-agent case, individual agents consider the plans of the other robots and can plan out the sequence of paths. Before moving onto the algorithms that have specifically been used for multi-robot exploration, we first present some important definitions related to Sequential Decision Planners.

3.6.1. Markov Decision Process

Consider a single agent present in an environment. At each time step t the agent takes an action. The action influences the environment and causes a transition in the state of the environment. This state transition is usually modelled using probability, and is formally referred to as the transition probability. At time step t+1, the agent receives an observation of the new state of the environment. If we consider the situation to be a sequential process then at every time step t, the agent needs to choose an action. The way in which the agent chooses these actions is through a reward function/model that is modified as per the goal that the agent is required to achieve. Thus in the above scenario there are states, actions, transition probabilities and a reward metric. This can be understood by also referring to figure 3.15 where the agent executes an action a, which results in the transition of the environment state from s to s'. Through this transition of the environment state, the agent receives a reward r. The sequential

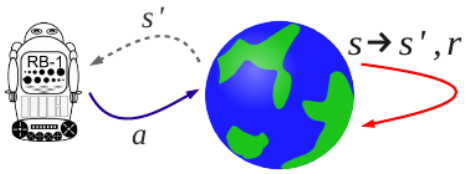


Figure 3.15: Environment and Transitions [33]

decision problem described in this figure has three important properties:

1. The agent can fully observe what happens in the environment and there is no ambiguity in its observation.
2. The rewards obtained at every time step can be added and thus are additive rewards.
3. The transition model follows a first order Markovian property, which means that the current state only depends on the previous state and not on any earlier states.

Such a sequential decision problem model is referred to as a Markov Decision Process (MDP). It is mathematically defined by a set of states (with initial value s_0), a set of possible actions, a transition

model $P(s^{'}|s,a)$ and a reward function $R(s)$ [36]. In an MDP, the sequence of actions can be grouped together into a list. This list of actions is usually referred to as a policy (π). Amongst the possible sequences of actions or policies, there exists an optimal policy (denoted by π^*) which directs the agent to a desired behaviour/goal [33]. The goal can be to maximise a total reward, minimise a cost or reach a particular state. If we consider that the agent interacts with the environment for H number of time-steps, then an example of a policy π can be the list of actions $\pi = [a_1, a_2 ... a_H]$. The search over the entire space of policies in order to obtain the optimal policy, by computing the expected reward for each policy is referred to as **Planning**.

The optimal policy is also referred to as the trajectory. This can be computed before the motion begins or incrementally during the motion. The former trajectory planning strategy is called offline planning while the latter is called the online planning. Usually offline trajectory planning is associated with the computation of the entire trajectory form the initial state until the goal. On-line planning however is the incremental computation of action values that lead an agent from its current position to the goal. Offline planners usually produce globally optimal solutions if the environment is fully known and static, while online planners produce locally optimal results and are suitable in situations where the environment is changing during motion or is partially known [18]. The differences have also been concisely shown in figure 3.2.

Sl. No	Attribute	Offline Planner	Online Planner
1	Solution nature	global optimal	local optimal
2	Targeted Environment	static	dynamic
3	Environment Knowledge	Full	Partial

Table 3.2: Differences between Offline and Online Planning [18]

In the next subsections, we delve more into techniques that can be used to perform planning and thus obtain the optimal policy.

3.6.2. Exact Planning - Value and Policy Iteration

In the field of Artificial Intelligence, planning is defined in many ways. One of the definitions of planning which is also applicable in this discussion of solving an MDP is that planning is any computation process that takes a model as input and produces or improves a policy for interacting with the modeled environment [40]. For an MDP, this computation of policy can be done in either an exact manner that obtains the exact optimal solution, or though an approximation using search based techniques. Consider the process shown in figure 3.16. In this figure, an agent performs an action a_t at time t. At the next time-step $t+1$, the agent observes the environment state s_{t+1} and obtains a reward r_{t+1}. All these

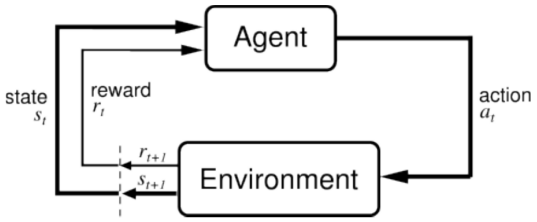


Figure 3.16: Planning in MDP's [40]

variables can be grouped together and used to represent the MDP as a tuple (S, A, T, R, H), where,

1. S is the set of states of the environment.
2. A is the set of actions that the agent can execute.
3. $T : S \times A \times S \times [0, 1, ...H] - > [0, 1]$ is the transition function that consists of the probabilities $T_t(s, a, s^{'}) = P(s'|s, a)$ of moving to a state $s^{'}$ when in state s and executing an action a.

4. $R : S \times A \times S \times [0, 1, ...H] -> \mathbb{R}$ is the reward at $(s^{'}, s, a)$.
5. H is the horizon of planning

Mathematically the goal is to obtain $\pi : S \times [0, 1, ...H] -> A$ that maximises the expected sum of rewards. ie,

$$\pi^* = \arg\max_{\pi} \mathrm{E}\left[\sum_{t=0}^{H} R_t\left(S_t, A_t, S_{t+1}\right) \mid \pi\right] \tag{3.3}$$

The computation of the exact value of policy π is done using exact methods. Exact methods to solve MDP's utilise another popular equation called the Bellman Equation. The Bellman equation is based on the idea that the utility of a state is the immediate reward for that state plus the expected discounted utility of the next state, assuming that the agent chooses the optimal action [36]. The Bellman equation is given by equation 3.4.

$$V(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P\left(s' \mid s, a\right) V\left(s'\right) \tag{3.4}$$

One example of an exact method that is used to solve an MDP is the Value iteration algorithm. In this iterative algorithm, a value function/utility value is computed for all states and is updated until equilibrium is reached. Once this equilibrium condition is reached, the optimal value function is recorded as U^* and the optimal policy is obtained using the equation 3.5 [36] [1]:

$$\pi^*(s) = \arg\max_{a \in A} \sum_{s'} T\left(s, a, s'\right)\left[R\left(s, a, s'\right) + \gamma V^*\left(s'\right)\right] \tag{3.5}$$

Another exact algorithm is Policy iteration that consists of policy evaluation and policy iteration steps. In the policy evaluation step, the current policy is fixed and the value function is obtained using Bellman updates until convergence [36] [1].

$$\begin{aligned} &V_0^{\pi}(s) = 0 \\ &V_{i+1}^{\pi}(s) \leftarrow \textstyle\sum_{s'} T\left(s, \pi(s), s'\right)\left[R\left(s, \pi(s), s'\right) + \gamma V_i^{\pi}\left(s'\right)\right] \end{aligned} \tag{3.6}$$

In the policy iteration step, the value function is fixed and the best action (or best policy) is computed for the next iteration.

$$\forall s \quad V^{\pi}(s) = \sum_{s'} T\left(s, \pi(s), s'\right)\left[R\left(s, \pi(s), s'\right) + \gamma V^{\pi}\left(s'\right)\right] \tag{3.7}$$

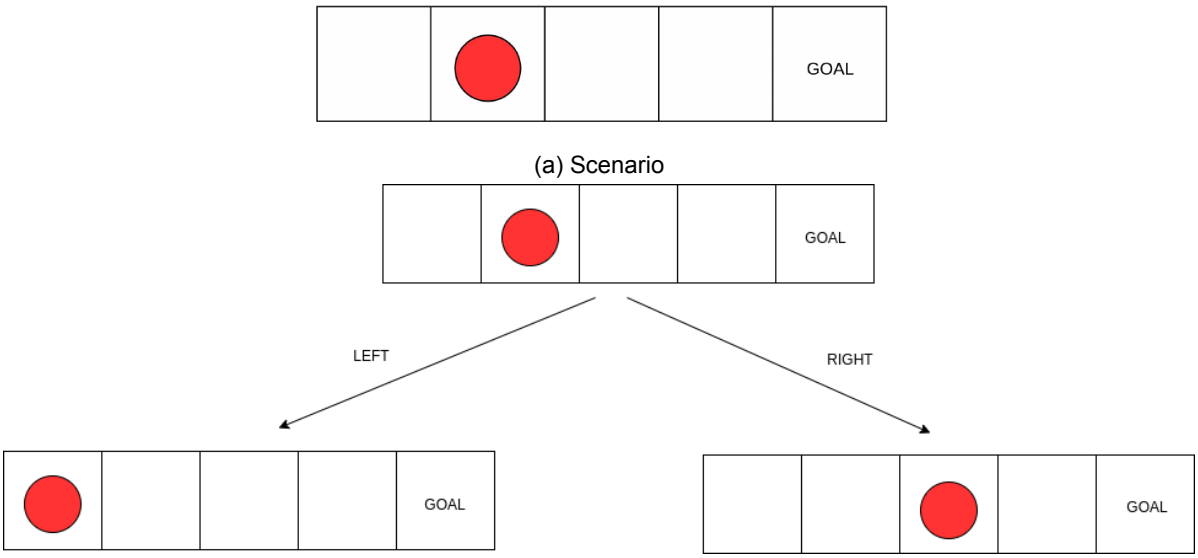
The 2 steps are repeated until the policy converges.

3.6.3. Approximate Planning - Monte Carlo Tree Search

One drawback of the previously described methods is the amount of computation required, as many iterations need to be done for every possible state of the agent. If the state space is very large, then a single iteration over all the states is expensive. Instead of finding the exact policy, there are other methods that can obtain a near optimal solution, referred to as an approximate policy.

This can be done with the help of trees where the nodes of the tree represent the environment state and the branches of the tree represent the actions that can be selected. This can be seen in figure 3.17 where the scenario of a red robot and its goal are shown in a 5 grid environment in sub-figure 3.17a. The corresponding tree representation with actions and resulting states can be seen in sub-figure 3.17b.

Even in most tree solving methods, computations due to branching might be an issue and the search exponentially increases with the increase in action space [1]. Due to this, there are various techniques that have been used to reduce the amount of computation. One technique is to find the approximate policy based on random sampling of possible future scenarios. This type of strategy comes under the umbrella of Monte Carlo methods. In Monte Carlo methods, repeated random sampling is performed to obtained numerical results. Apart from reducing the branching factor through the reduction of states to search, Monte Carlo Methods have the advantages of [40]:



(a) Scenario

(b) Scenario solved by a tree

Figure 3.17: Approximate Planning using Trees

1. Learning optimal behaviour from interaction with the environment, with no model of the environment dynamics.

2. Planning can be performed by running simulations on a simulator of the environment, that can also be used to model other agents in multi robot systems.

3. There is no requirement to update the value estimate on the basis of value estimates of successor states (also called a bootstrapping), as was done in previous exact methods.

Combining the Monte Carlo Method with the tree search over MDP processes, we get a sequential online path planner which is the Monte Carlo Tree Search (MCTS) algorithm. A basic working of the algorithm has been shown in figure 3.19. The main intuition behind MCTS is that by using Monte Carlo simulations to quickly sample thousands of possible trajectories, we can achieve good approximations of the values of possible actions from the root node (the node from which the search started) [12]. There are 4 major steps that are repeated in every search iteration, which have been described below [9]:

1. Selection - Starting at the root node, the child selection policy is recursively applied to descend the tree until an expandable node is reached. An expandable node is a non terminal state that still has unvisited/unexpanded children.

2. Expansion - Based on the available actions, more child nodes are added to the tree in this step.

3. Simulation - A simulation is run from new nodes according to a default policy, thus leading to an output.

4. Back Propagation - The simulation result is backed up through the tree in order to update their statistics.

Further, at each iteration there exists 2 rules/policies that must be followed. These are:

1. Tree Policy: Used in the selection or expansion step to select or create an expandable node from the nodes already present in the search tree

2. Default Policy: Used in the simulation step to play out actions for a predefined number of times or until a non-terminal state is reached.

Algorithm 1 General MCTS approach.

```
function MCTSSEARCH(s_0)
    create root node v_0 with state s_0
    while within computational budget do
        v_l ← TREEPOLICY(v_0)
        Δ ← DEFAULTPOLICY(s(v_l))
        BACKUP(v_l, Δ)
    return a(BESTCHILD(v_0))
```

Figure 3.18: MCTS PsuedoCode [9]

Figure 3.18 shows a sample psuedo code on how the above mentioned steps and terms come together. Consider $v0$ to be the root node having a state $s0$. vl is the last node reached during the tree policy stage corresponding to state sl. Δ refers to the reward for the terminal state that is reached by applying the default policy from state sl. The value of Δ is backed up from the terminal node to the root node. This process is repeated a number of times and at the end of the entire process, the action value that leads to the best child from the root node is obtained. There are many ways to extract the best child and this will be explained further.

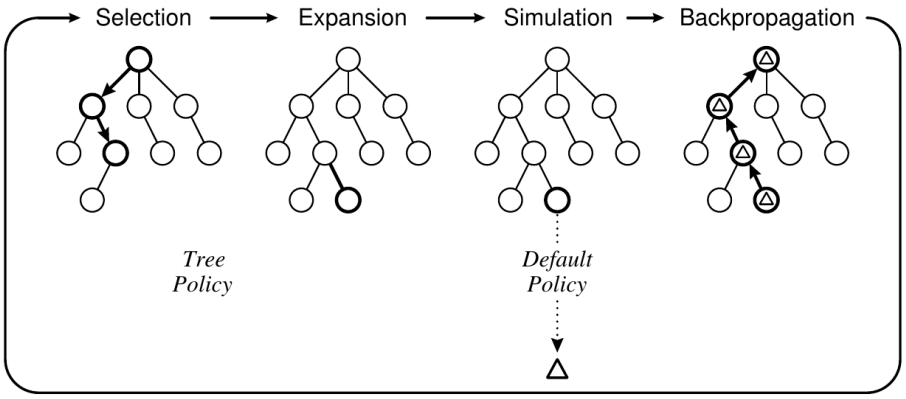


Figure 3.19: MCTS Basic [9]

One iteration of the MCTS process has been shown in figure 3.19. The first stage involves the tree policy step where starting at the root node $v0$, child nodes are recursively selected until a node vn is reached that is either a terminal state or is not fully expanded. At such a node an action a, that has not been executed yet is selected from the action list. This action is executed, which leads to a new child node that is added to the list. Following this the simulation step starts where simulations are run from the newly created node according to the default policy. After a predefined number of times, an objective function is evaluated which produces a reward value Δ. The reward is then propagated up the sequence of nodes in order to update each node's visit count and reward value. These steps continue to occur until the computation budget is met or the search is interrupted. Following this, the best action corresponding to the root node is selected. Some ways of selecting the best possible action are:

1. Max child: Selects the child with highest reward
2. Robust child: Selects the child that is most visited
3. Max-Robust child: Selects the child with highest visit count and highest reward

Usually the default policy used in the above tree is a sequence of random actions. The tree policy however follows an exploration-exploitation trade off where at each node either the best possible action at that moment is chosen (exploitation) or an action is chosen that maybe optimal/superior in the long

run (exploration) [9]. A popular relation to perform this is the Upper confidence bound for trees (UCT) algorithm and is described by the following equation:

$$\mathrm{UCT1} = \bar{X}_j + 2C_p\sqrt{\frac{2\ln n}{n_j}} \tag{3.8}$$

In this equation, $\bar{X}_j$ refers to the average reward obtained by taking the action j, n_j refers to the number of times that action j / child j has been selected and n refers to the number of times the current parent node has been visited. It is important to note that the $\bar{X}_j$ influences exploitation while the latter term influences the exploration. From the equation it can be seen that for a particular action j, n_j will be equal to 0 initially which leads to a UCT value of $\inf$. This leads to every child being visited at least once, which is essential given the random nature of playouts. Further, the value of $\bar{X}_j$ lies in the range $[0, 1]$ and it is found that keeping $C_p = 1/\sqrt{2}$ causes the rewards from exploration to be in this range too [9].

3.6.4. Partially Observable Markov Decision Process

Partially Observable Markov Decision Process, abbreviated as POMDP is an extension of an MDP process that incorporates the notion of observations and their probability of occurrence. The probability of occurrence is usually conditional to the state of the environment. The reason for having separate probability over the observations is because there is always possibility of agents having noisy and limited range sensors that could prevent it from observing the true state of the environment. This can be further understood by considering figure 3.20, where a robot is interacting with the world. As compared to the earlier case (figure 3.15), the agent does not get information about the state of the environment (due to sensor limitations) but instead gets information as observations (represented as $o(.|s, a)$). As

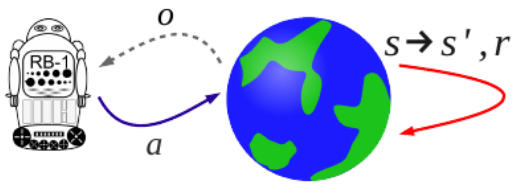


Figure 3.20: Environment and Transitions in a POMDP [33]

the agent continuously interacts with the environment, the agent usually stores these observations and uses this history of observations to estimate the probability of the occurrence of states. The agent then decides upon a suitable action using this probability measure. This probability measure of states is referred to as the belief of the agent and is mathematically represented as,

$$\forall_{s_t} \quad b\,(s_t) \triangleq \Pr\,(s_t \mid o_t, a_{t-1}, o_{t-1}, \ldots, a_1, o_1, a_0) \tag{3.9}$$

where, s_t, o_t and a_{t-1} are the estimated state of the environment at time t, observation of the obtained state at t and the action executed at time $t-1$. Thus using the history of observations an agent can predict a particular state as a belief. Using this belief information the agent can select a suitable action.

3.6.5. Decentralised Partially Observable Markov Decision Process

Decentralised Partially Observable Markov Decision Process, abbreviated as Dec-POMDP is a framework that is used to generalise the POMDP to multiple agents in order to model a team of cooperative agents that operate in a stochastic, partially observable environment [33].

Figure 3.21 illustrates this with 2 robots that each have separate observations about the environment and also execute separate actions. Mathematically, this can be represented as a tuple $M =< \mathbb{D}, \mathbb{S}, \mathbb{A}, T, \mathbb{O}, O, R, h, b_0 >$ where,

1. $\mathbb{D} = 1, 2, ..n$ is the set of n agents

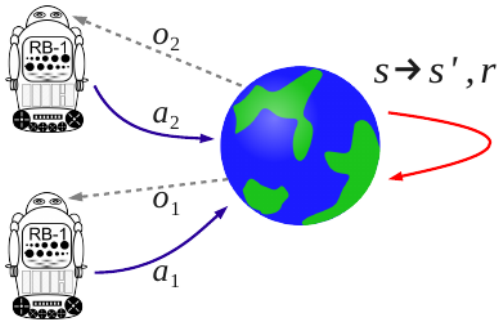


Figure 3.21: Environment and Transitions in a Dec-POMDP [33]

2. $\mathbb{S}$ is the finite set of states
3. $\mathbb{A}$ is the set of joint actions
4. T is the transition probability function
5. $\mathbb{O}$ is the set of joint observations
6. O is the observation probability function
7. R is the immediate reward function
8. h is the finite time horizon, for computations
9. b_0 is the initial state distribution at time t=0

The above model thus, extends the single agent POMDP models by incorporating joint actions and observations. The above tuple model is used to model the sequence decision making from observation of the environment to emitting actions. At every time step the environment emits a joint observation $\mathbb{O}$ which consists of n individual observations. This is obtained from $O = P(o|a, s^{'}$ which represent the probabilities of joint observations. Each agent observes its own component i from $o_i \in \mathbb{O}$. Each agent uses its own belief function on the observation o_i in order to estimate the state and consequently the action. Each agent i takes an action a_i that together form one joint action $a =< a_1, a_2 ... a_n >$. Subsequent to this the transition probability function $T = P(s^{'}|s, a)$ is used to obtain the next state corresponding to the current action and current state. At this time, the agent also receives the immediate reward R that is awarded for each join action. In the next time step, the same process repeats again. The finite time horizon h is used to obtain rewards over certain time steps and obtain a cumulative reward value. This is used to obtain a joint policy that maximises the expected cumulative reward. The joint policy is a tuple of individual policies for each agent.

Using the above framework helps model POMDPs for a multi-agent setting in a stochastic environment. It can be used as a starting step for agents to make decisions in an uncertain environment. There are three broad categories of uncertainty that must be dealt with:

1. Outcome Uncertainty - The outcome or effects of actions may be uncertain. The outcome of wheel on different friction surfaces is a good example of this.
2. State Uncertainty - Due to limited/noisy sensors the agent may not be able to determine the exact state of the environment exactly.
3. Uncertainty over peer agents - This deals with the difficulty for each agent knowing the other agents choice actions as well as states.

3.6.6. MCTS in Distributed Teams

In the context of exploration of a region using multiple robots, MCTS has been used in some algorithms. One example of this is in the work by Hyatt et al in [21]. In this work, Monte Carlo Tree Search has been used in order to perform coverage path planning (CPP). The authors make some important assumptions like all robots having uninterrupted communication between each other, robots having the capability to only chose amongst three actions that is, left, right, or straight. The nodes of the tree are the grid cells that the robots can encounter, the UCT tree policy is used and the default policy is of moving in a straight line, until the grid square ahead is either occupied or has been covered. The function X that is calculated at the end of the simulation step is given by equation 3.10.

$$X = \sum_{k=1}^{T} \left[\frac{1}{(t_k + 1)^2} (p_k - C_{hit} q_k) \right] \tag{3.10}$$

In the above equation, $p_k = 1$ if the robot lands up in a newly discovered grid cell at time step k and "covers" it, else $p_k = 0$. $q_k = 1$ if the robot hits a wall or a robot at time step k, else it is 0. T is the final time step for the simulation horizon. The normalisation using the $(t_k + 1)^2$ term is used to decay the reward overtime and takes into account the objective values calculated further in the future as being less certain. The C_{hit} parameter is used to apply a higher penalty for states that are explored or in collision. Each robot executes its own version of a search tree thus leading to a decentralised architecture and computation. The robots are only required to share their discovered map, the best path at the last solve and their covered grid squares such that in the simulation phase, each robot first simulates the action of the other robots based on last known best path. This allows the planner to take into account the plans of the other robots and then simulates its own action. The authors ran experiments of exploration on previously unknown maps with teams of 1 until 10 robots and found that for large team sizes, the MCTS algorithm performs better than a previous boustrophedon planner, in which robots explored the arena by moving repeatedly in an UP to DOWN, and DOWN to UP movement. As a future work the authors mention that the effects of varying the default policy, horizon length and exploration coefficient on the planning needs to be studied.

The above method assumes full communication between agents. However MCTS has also been used in cases that assume intermittent communication along with a dynamic environment. One such method by Claes et al is presented in [12] in which a team of robots is used in a warehouse commissioning task where robots gather a deliver items in an efficient manner while adhering to their own capacity constraints. The authors mention that conventionally this problem is solved by a Multi-Robot Task Allocation (MRTA) approach. However such an approach is centralised and assumes a static environment. Therefore a new formal framework called Spatial task allocation problem (SPATAP) is introduced that is distributed and considered dynamic reallocation of tasks. SPATAPS is a subset of a Multi-agent MDP (MMDP) in which multiple decision makers are present who observe the full state of the environment [13]. This is in contrast to the previously defined Dec-MDP where each agent is locally fully observable and can only observe the local state or its own component in the vector of states. The authors mention that solving SPATAPS is difficult due to the large order of the state space that grows exponentially with the number of agents and the number of task locations. To solve the commissioning task, the paper adopted a distributed approach that built on sample-based methods that was independent on the size of state space. This is done by enabling the agents to predict the behaviour of other agents, given the common observed global state. To predict the behaviour of teammates, the paper uses a heuristic approach. The heuristic consists of a utility/reward function which is given by equation 3.11.

$$\mathrm{NV}(n_x, i) = \begin{cases} -\infty & \text{if } \tau_{n_x} = \mathfrak{T}_{\text{empty}} \\ \frac{\mathrm{TV}(\tau_{n_x, \iota_i})}{\mathrm{dist}(\lambda_i, n_x)} & \text{otherwise} \end{cases} \tag{3.11}$$

where n_x is the location of the tasks, i refers to the agent, T_{empty} implies that no task is present, $dist(\lambda_i, n_x)$ is the length of the shortest path from agent i to the node n_x and $TV(\tau_{n_x, \iota_i})$ is the evaluation of the task τ_{n_x}, given the current inventory status of the agent ι_i. The paper considers this to be the largest sum of the cost values of the tasks at a particular node that can still be stored by the the

agent i. The aim is to maximise the value of NV as it is a reward function. The evaluation of this is done by the agents in 3 different ways:

1. Greedy with Social Law
   (a) In this approach all agents always try to move towards nodes that has the highest evaluation of the NV function.
   (b) Mathematically this can be written as, $a_i = \text{GoTo}\left(\max_{n_x \in \mathcal{N}} \text{NV}\left(n_x, i\right), i\right)$, where the $GoTo$ operator results in the next action that helps the agent i move along the shortest path to the task location n_x
2. Reverse Greedy Allocation
   (a) In this approach the problem is seen from the perspective of the nodes and each graph location is assigned to the agent that has the highest value for the value function given in equation 3.11
3. Iterative Greedy Allocation
   (a) In this approach, equation 3.11 is evaluated for all locations and for all agents. Agents are then selected and then assigned to the node values having the highest reward value.
   (b) The agent is then removed from the list and the process is continued for all the remaining agents in an iterative manner.

It is important to note that as each agent can observe the global state of the environment, each agent has information on the positions of the other agents at all times. Thus, each agent is able to evaluate equation 3.11 individually. In the MCTS step, the heuristics are evaluated during the selection, expansion and simulation stages. In the selection and expansion stage the action prescribed using the UCT method usually overrides the action given out by the heuristic. However, the heuristic is completely utilised in the simulation stage. The authors also note the importance of having a Do it yourself reward in the MCTS process which gives more weight to movements where the agent is rewarded a higher reward if it performs a task itself. The authors performed experiments on teams of 2 until 8 agents and noted that the planning time is better than previous greedy methods of warehouse commissioning. As a future work, the authors propose to modify the heuristic NV function to include some aspect on task appearance probabilities and also suggest to improve the MCTS search by including prior knowledge of node positions.

In the method Best et al in [7], the authors propose an MCTS solution for a multi-robot scenario for generalised team orienteering and active object recognition. The approach assumes that communication is intermittent with small amount of data being sent over the network between robots. The data being sent over (whenever possible) is a highly compressed version of their local search trees that is probabilistic in nature. More specifically this data is the best selected action strategy. In the algorithm, the objective function that is evaluated by each agent is a local reward f that consists of the action of the current agent as well as the best shared actions of all other agents. The authors use an algorithm

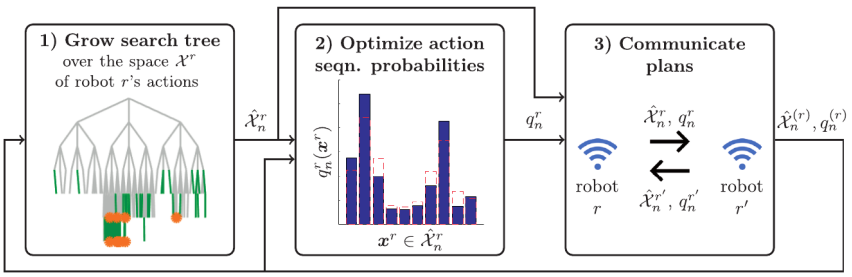


Fig. 1. Overview of the algorithm running on-board robot r. (1) The search tree is expanded by adding new actions (green). Periodically, the set of best nodes (orange) is selected as the domain $\hat{\mathcal{X}}_n^r$. (2) The probability distribution q_n^r is optimized (from dotted red to solid blue). (3) If possible, the domains and distributions are communicated between robots.

Figure 3.22: Decentralised MCTS [7]

called Dec-MCTS whose steps can be seen in figure 3.22. There are three main phases:

1. Growth of the search tree using MCTS, while taking into account the information about other robots.

2. Updating of the probability distribution over possible action sequences.

3. Communication of the probability distribution with other robots.

The 3 phases continue regardless of successful communication, until a computation budget is met. In the first phase a tree search is run that takes into account the plans of other robots during the simulation step. This results in an inherent coordination aspect that occurs within the process. The tree policy used is called discounted UCT which is a modification of the UCT algorithm which accounts for the non-stationary reward distribution by biasing each sample by a weight γ which increases at each simulation stage. In the second phase, the probability distribution q^i is updated where $q^i(\mathbf{x^{(i)}})$ defines the probability that robot i will select $\mathbf{x^{(i)}} \in \chi_n^i$. χ_n^i is the list of the most promising action sequences $\{\mathbf{x_1^{(i)}}, \mathbf{x_2^{(i)}} ...\}$ for robot i found by the first MCTS phase. In the last phase, robots that are able to communicate, exchange the distribution $q^i(\mathbf{x^{(i)}})$ and the best action sequence χ_n^i to one another. Each robot then replaces prior information of $(q^i(\mathbf{x^{(i)}}), \chi_n^i)$ with the new obtained information. Using the above defined Dec-MCTS algorithm the authors perform a generalised team orienteering problem where 8 robots aim to visit a maximal number of goal regions, that are weighted by importance. This can be seen in figure 3.23. The authors studied the effect of communication loss on the algorithm and found that when around 97% of the packets are degraded, the performance is lost but its still better than the no communication case.

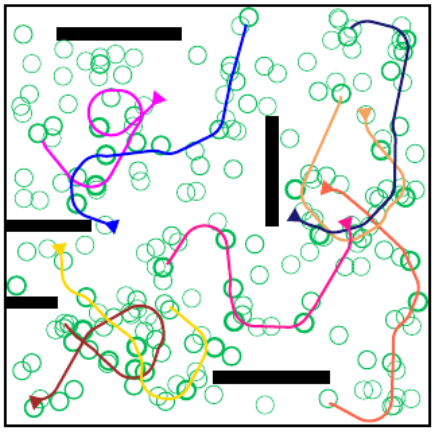


Figure 3.23: A team of 8 Robots moving into goal locations (green circles) using the method described in [7]

Combining the above two strategies of sharing a probability distribution and modelling of other agents has been performed by the Minglong Li et al in their work [23]. In this paper, the authors use a method of decentralised sharing of paths and prediction of the trajectories based on an MCTS tree search which they refer to as Dec-MCTS-SP. The method has been specifically used for multi-agent information gathering under threat and uncertainty, which is modelled as a Decentralised Partially Observable Markov Decision Process (Dec-POMDP). The authors mention that their algorithm (the Dec-MCTS-SP) is inspired by humans playing a game of soccer, where several players playing soccer do not need not talk frequently to let fellow teammates know what their next move is. Instead, given an overall goal, a cooperation occurs by the prediction of the behaviour of others.To perform the information gathering, the environment is modelled as an undirected graph $G = (V, E)$ with number of vertices $N = |V|$ and each vertex has 2 states, one of information and the other for threat level. It is assumed that the agent can only observe the information and threat at the node at which it is located. Thus the authors formulate the planning problem as a constrained Dec-POMDP. Thus, there exists a joint set of actions that agents can select their actions from. The planning objective of the agents is to choose the joint movement actions that would help maximise the total expected reward accumulated over h time

steps represented by the below equation,

$$R(h) = \sum_{t=0}^{h} r(t) \tag{3.12}$$

where, r(t) is the immediate reward at time step t. In Dec-MCTS-SP, each agent runs an MCTS algorithm with its predictions of the peer agents as well as information obtained from teammates. During the simulation stage, the tree is grown up to a maximum depth D and the reward is computed for the current planning agetn as well as the peers. In the computation of peers, action sequences are required and for this the authors use the actions of other robots that are sampled from the shared probability distributions of action sequences as done in the previous method of Best et al in [7] and is denoted by D_s. Along with this the authors also use the previous heuristic prediction of Claes et al in [12], denoted by D_p. The idea of D_p is to predict the actions of other robots in a computationally cheap manner, reduce the communication pressure and get better coordination.

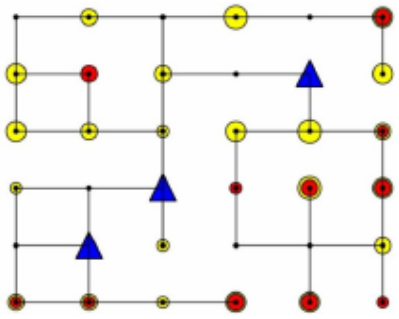


Figure 3.24: Graph Environment in Dec-MCTS-SP [23]

The authors ran simulations on the communication uncertain environment shown in figure 3.24 that has 3 agents (blue triangles) patrolling an area of 36 locations (black dots). The information is shown in yellow circles while the threat levels are shown in red. The size of the circles denotes the information or threat levels. On comparing the results it was found that with the Dec-MCTS-SP algorithm performs path planning around 1-2 seconds faster than Dec-MCTS, and with the increase in the number of agents the planning the gap between the 2 increases and with Dec-MCTS-SP performing path planning faster. As a future work, the authors propose to extend the working under more complex environments and also experiment with different communication models like broadcasting or sequential communication.

3.7. Peer Modelling

In the field of multi agent systems, there have been various techniques that can be used by agents to model the behaviour of peer members and predict the actions that peer agents would take. This approach not only gives the advantage of cooperating actions in order to achieve team goals in a faster manner, but also reduces the dependency on full-fledged communication in many applications. The ability to model other agents in these applications becomes a key component for effective collaboration. A comprehensive survey on autonomous agents modelling one another can be found in Albrecht et al in [2], in which there exists a modelling agent and a modelled agent. The model is defined to be a function that takes as input some portion of the observed interaction history, and returns a prediction of some property of interest regarding the modelled agent. In order to better understand the classifications of techniques mentioned in the paper, the paper lays down a framework for the possible assumptions about the modelled agent that has been listed in table 3.3. The categorisation on Deterministic vs Stochastic is based on the action choices. If the modelled agent chooses an action for a given action history with complete certainty then it is deterministic, else stochastic. Fixed and changing refers to behaviour of the modelled agent. If an agent keeps its behaviour of choosing actions, or its decision making ability unchanged, then it is termed as fixed. A Markovian agent that chooses actions based solely on its current state is an example of this type of agent. An Adaptive/learning agent is example of changing behaviour where a modelled agent learns the behaviour of other agents and chooses actions

Sl. No	Attribute	Type1	Type2
1	Action Choice	Deterministic	Stochastic
2	Agent Behaviour	Fixed	Changing
3	Decision factors	Known	Unknown
4	Action choice	Independent	Correlated
5	Goals	Common	Conflicting
6	Action Strategy	Simultaneous	alternating
7	State Representation	Discrete	Continuous
8	Environment Observability	Full	Partial

Table 3.3: Assumption Categorisation [2]

based on these models. Decision factors are attributes that influence the decision of the agent and are usually based on history (like the most recent n observations), or based on abstract features which were calculated from the history. The modelled agent can either know all these factors completely or not know about some or any of them. In terms of action choice, if the modelled agent chooses actions independently from another agent, then it is independent. When there are multiple agents then the goals can be common or conflicting and the strategy of action execution plays an important role. It can be simultaneous like the game of robot soccer, or it can be alternating as in the case of 2-player poker.

Using the above assumptions, the authors of [2] broadly categorised peer modelling methods into the following:

1. Policy reconstruction
2. Type-based reasoning
3. Classification
4. Plan recognition
5. Recursive reasoning
6. Graphical models
7. Group modelling

Policy Reconstruction is the modelling techniques where the modelling agent predicts the action probabilities of the modelled agent, by assuming a particular model structure and learning the parameters based on observed actions. In Type-based reasoning, the action probability of the modeled agent is predicted by assuming that the modelled agent has one of the several known types of models. Based on observed actions, the relative likelihood of each of these types is computed and the the one giving the highest score is selected. Classification is the technique in which the modelling agent assumes a particular structure of the model and uses machine learning in order to fit the model parameters based on information from the environment. Plan recognition is the technique by which the model predicts the goal and future actions of the modelled agent. In recursive reasoning a model is used to predict the next action of the modelled agent. Graphical models predict the action probabilities of the modelled agent and use a graphical model to represent the agent decision process. The last category of group modelling is a technique used to predict the joint properties of the group of agents.

3.8. Comparisons

Having explored the various techniques that have been utilised in multi robot exploration, we now turn to compare the methods on criteria that is required for the application of this research study. It is emphasised that the specific scenario that is being targeted is to have a team of multiple robots that can cooperate effectively in order to explore an unknown region that has a limited communication. Thus, the constraints that we are interested in are listed below along with an abbreviation which is used in a following table:

1. Unknown Map (UM) - Agents have no prior knowledge of the region that is to be explored.

2. Limited Communication (LC)- Agents cannot communicate and share information beyond a communication range.

3. Local Observability (LO) - Agents can only observe their nearby surroundings, but not the entire environment.

4. Distributed Team (DT) - Agents follow a peer-to-peer strategy and do not have any dependency on any centralised entity.

5. Free Movement (FM) - Agents move around the entire region continuously and are not restricted to any position.

Sl. No	Method	Class	UM	LC	LO	DT	FM
1	Clique-Intensity Algorithm [16]	Robot Dispersion	✓	✓	✓	✓	-
2	Backbone dispersion [16]	Robot Dispersion	✓	✓	✓	✓	-
3	Improved Walk Dispersion [34]	Robot Dispersion	✓	✓	✓	✓	-
4	RRT Exploration [42]	Cost and Utility	✓	-	✓	-	✓
5	Leader Follower RRT Exploration [29]	Cost and Utility	✓	-	✓	-	✓
6	Communication Revenue [6]	Cost and Utility	✓	✓	✓	✓	-
7	Hunter Gatherer [15]	Cost and Utility	✓	-	-	✓	-
8	Lopez et Uriel [25]	Scene Partitioning	✓	-	✓	✓	✓
9	EVSA [22]	Scene Partitioning	✓	-	✓	✓	✓
10	Meng et al [27]	Game Theory	-	-	✓	✓	✓
11	Binary Log Learning [31]	Game Theory	-	-	✓	✓	-
12	CPP MCTS [21]	SDP	✓	-	✓	✓	✓
13	SPATAPS [12]	SDP	✓	✓	-	✓	✓
14	ABC Deep Learning [14]	SDP	✓	✓	-	✓	✓
15	Greame et al [7]	SDP	-	✓	-	✓	✓
16	Minglong Li et al [23]	SDP	✓	-	-	✓	✓

Table 3.4: Method Comparison

Each of the methods described in this chapter have been compared on these constraints and can be found in table 3.4. A tick mark indicates that the method considers the constraint in its implementation. It can be seen that in the Class of methods on Robot Dispersion ([16], [34]) all the methods consider the constraints. However, all these algorithms have minimal cooperation and are utilised in applications where after distributing themselves in the environment, they remain stationary. Therefore in order to maximise the exploration, more robots are required rather than more movement of team member. Amongst the Cost and Utility based approach, the Rapidly exploring Random Tree (RRT) based methods ([42], [29]) has free movement available and is biased towards regions that it does not know yet. Each method does not assume limited communication and relies on a centralised node. In the Communication Revenue method [6], most of the constraints are adhered to, however agents are required to continuously maintain positions where they can be in communication with one another. This restricts their movement to discover new areas individually. This along with no limited communication constraint is also the drawback with the Hunter Gatherer method [15] where sub-groups of agents can talk to one another via an online board. The existence of the online board however removes the distributed aspect of the system because a failure in the online board will lead to a disruption in the entire exploration. In the case of both the scene partitioning methods ([25], [22]), the system is completely distributed and has agents that can move freely. However the main drawback is that the agents do not consider the constraint of limited communication. Methods under the class of game theory involved a surveillance system where a team performs continuous monitoring of a region([27], [31]). The methods do consider a distributed approach with team members negotiating to find a Nash Equilibrium. However, both methods assume a target probability map to be available prior to the exploration that gives a probabilistic estimate on the existence of a target within each region. Further, the agents in this method

also consider unrestricted communication which allows agents to access the plans of one another at all times. In the case of sequential decision planners (SDP) , all methods follow a distributed approach with agents have free movement. In the method on Coverage Path planning [21] agents have the possibility to perform exploration in an unknown area. However the drawback is that the method assumes unrestricted communication to be available. In the case of the methods related to Heuristic Prediction ([12],[14]) agents do satisfy most of the constraints that are present. However the only constraint not satisfied is solely having local observibility. The agents in both these methods instead can observe the global state of the environment at all times. The method of sharing a probability of actions in the work of Greame et al in [7] is also similar to the heuristic methods in terms of constraint satisfaction, where agents are modelled to deal with intermittent communication. The additional constraint that this does not satisfy is working in a setup of a completely unknown system. Instead the region is modelled as a graph with predefined locations that the team of agents must move through within a given time. The approach by Minglong Li et al in [23] combines the work of Greame et al with the heuristic prediction in unseen maps for information gathering. However in this method the communication is unrestricted and each agent can observe the global state of the environment.

4

Approach

In the previous chapter, various methods utilised to perform multi-robot exploration were discussed and compared. In this section, the approach taken as per the research objective mentioned in section 2.1 will be described. As a starting step, the MCTS Planner of Hyatt et al [21] was implemented first. The reason for selecting this particular method will be explained in section 4.1. Following this, section 4.2 describes the problem setting, provides motivation for algorithm design choices and highlights the assumptions that have been made in the implementations. In the case of the method by Hyatt et al [21] the scenario was an environment of full communication, where agents were allowed to communicate important information like their state, position and plans with one another. This has been further elaborated in section 4.3. Once tested to be working the communication dependency of the algorithm was dropped in order to have no communication between agents. In this situation agents could not share any information with one another, but could only sense nearby agents for the purpose of collision avoidance. This scenario has been explained in section 4.4. In order to relax the constraint of communication while keeping physical limitations in mind, agents were then allowed to communicate important information only when within a predefined communication range from one another. This is referred to as partial communication and has been described in section 4.5. In order to bring the behaviour of agents having partial communication closer to the full communication case, agents were then given the capability to predict the strategies of peers. Agents performed this by using the most recently known information of peers and through the use of computationally cheap heuristics. This has been further elaborated in section 4.6. All the MCTS methods have certain hyper-parameters that can be tuned. The motivation for the specific choice of certain values for these hyper-parameters is given in section 4.7. The MCTS based methods are compared to another existing algorithm that considers the same environment constraints. This algorithm is referred to as frontier communication and is explained in section 4.8.

4.1. Baseline Selection

In this section we discuss the selection of the Baseline for further research to be conducted. Each major method of the class of methods has been compared in table 3.4 based on the ability to deal with completely unknown maps, work in environments of limited communication, possess only local observibility, support a distributed team of agents and facilitate free movement. Amongst the methods discussed, the method on Coverage Path Planning by Hyatt et al [21] using MCTS satisfied 4 out of 5 requirements. The method was also used by the authors to tackle a search and rescue exploration scenario. Further the ability of an agent to select an action by looking ahead a certain number of steps and computing the resulting reward could help agents avoid moving towards cluttered regions where it could get stuck. The method also involved the sharing of individual map coverage status, agent positions and path plans between agents. Being a simulation based method, each agent had the ability to consider the plans shared by the other robots in the computation of its own plans, which brought in the aspect of cooperation. The only drawback however was that the method assumed unrestricted communication.

In order to deal with the limited communication constraint, methods discussed in table 3.4 that

adhered to the constraint of Limited Communication (LC) were analysed. This included all the methods of the Robot Dispersion class, the Communication Revenue method [6] of the Cost Utility class and the SPATAPS [12] ABC Deep Learning [14] methods of the SDP class. The Robot Dispersion class and the Communication Revenue method however did not satisfy the constraint of Free movement. This could possibility restrict the time for movement around the arena and increase the exploration time for the team. The SDP methods did not have this restriction and could move around freely. Each of these methods utilised computationally cheap heuristics that could predict/estimate the next actions of team members. This could potentially be useful in the current application where there is limited communication and such heuristics could be used to predict the next positions along with the planned actions of the peer agents. This however is still an unanswered question because the heuristics have previously only been used in a setting where each agent could observe the global environment.

Due to the various merits of MCTS mentioned previously, the method of Coverage Path Planning by Hyatt et al [21] that used MCTS for exploration was selected as the baseline. In this algorithm agents could share map coverage, position and the best planned paths at every moment of time, without any restriction. The algorithm was then modified to remove the full communication assumption to a situation of completely no communication. In order to include the idea of cooperation between agents the case of partial communication was implemented, where agents could share information only when present within a predefined communication range. With the goal of narrowing the difference between Partial Communication and Full Communication scenarios, every agent was given the capability to predict the paths of peer agents that were outside the communication range. Agents made such predictions using known path information, as well as computationally cheap heuristics that was placed on top of the partial communication setup. It was possible to do this, because the goal of each agent is the same, that is to maximise the area of coverage. By approaching the study in the above manner important contributions that have been made are:

1. Extending the study on Coverage Path Planning by Hyatt et al [21] to an environment of limited communication.

2. Extending the heuristics dealt with in SPATAPS [12] and ABC Deep Learning [14] towards applications where the global state of the environment is unknown.

3. Utilising partial communication to move agents freely in an area thereby removing the need for restricted movement in previous multi-robot exploration work like the communication revenue method [6] and frontier based communication [5].

4.2. Algorithm Design

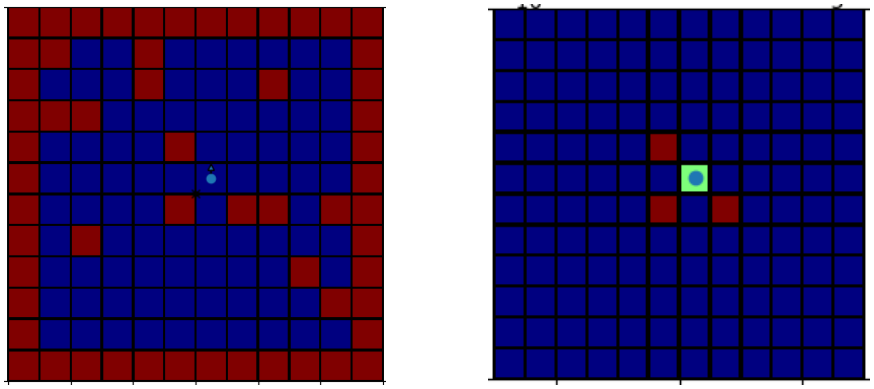
At the beginning of this chapter, it was explained that there are 4 different scenarios that have been considered in this study - Full Communication, No Communication, Partial Communication and Partial Communication with Prediction. The problem setting agents in each scenario perform exploration in has been explained in section 4.2.1. The stages in the design of the MCTS planner are explained in subsection 4.2.2. Within the MCTS planner there are a number of design choices and the motivation for particular design choices is described in subsection 4.2.3.

4.2.1. Problem Setting

Every scenario, irrespective of the communication level involves spawning or deploying multiple robots into a grid-world. The agents explore a grid-world that consists of a number of grid-cells. Each grid-cell can take one of the value of covered, unexplored or an obstacle. The state transitions of the grid-world environment occur based on the actions of the agents. This state transition occurs in discrete time-steps and is deterministic.

Each agent has the option to execute one of the actions $\{UP, RIGHT, LEFT\}$ and can take up positions in the grid-world which is defined by the tuple (x, y, yaw). The position (x, y) refer to the x and y coordinates in the grid-world, while yaw refers to the heading angle of the agent with respect to the horizontal axis. The heading angle indicates the orientation of the agent and can take the values $\{0, 90, 180, 270\}$ corresponding to pointing right, up, left and down respectively. From the action list, the action UP moves an agent 1 grid-cell ahead based on the yaw value. $RIGHT$ and $LEFT$ actions only turn the agent clockwise or anti-clockwise and do not shift the agent by any grid-cell.

Prior to spawning, each agent knows the size of the area to explore and maintains a separate belief of the environment as a grid-map. In the belief map of each agent, the grid-cells on the map can take one of the values of $\{Covered, Unexplored, Obstacle\}$. Each agent assumes that all the grid-cells are $Unexplored$ at the beginning of the exploration. As an agent moves around the environment, it acquires more knowledge about the environment and uses this to update its own belief map. Each agent however has locally observability and can only observe the state of a small segment of grid-cells of the entire grid-world. The extent of local observability is defined by the agent sensor range. For example, with a sensor range 1 the agent only discovers the values of the 8 adjacent grid-cells present around it. This is indicated in figure 4.1. The colour combination is that $Unexplored$ grid-cells are marked in blue, $Covered$ grid-cells in green and grid-cells with $Obstacle$ are marked in red. In the figure a 12x12 grid map along with a single spawned agent can be seen. Sub-figure 4.1a shows the position of the agent in the environment while sub-figure 4.1b shows the agent's belief map. When a



(a) Agent position in Environment (b) Agent's Environment Belief

Figure 4.1: Map Discovery by a Single Agent

team of multiple agents are spawned into the environment, then each agent plans out on its own paths. This maintains the distributed approach of the problem and there is no dependency on any central entity. There are also certain assumptions that each agent of the team makes about its team and the environment. These are:

1. Each agent knows the total number of agents that are present in the exploration team.
2. Each agent knows the size of the map and the origin as a frame of reference.
3. An agent considers a grid-cell to be covered when it is directly over it.

Thus, by utilising the above setting agents of a team are distributed, locally observable, can deal with limited communication and an unknown environment.

4.2.2. Design Stages

As discussed in the previous subsection, the problem setting is a team of distributed agents having local observibility spawned into an unknown environment. Each agent can individually sense the environment which helps it update its own belief of the environment. When multiple agents are required to cooperate and explore an environment, then a fundamental aspect that each agent is required to know is the actions of peer agents. To do this, agents need have the possibility to share information with one another. By doing this each agent can get to know the plans of the peer agents. In most of the scenarios of this study (except No Communication), it is possible for agents to share information with one another. Agents share their environment belief or map coverage status, position in the arena and their best planned paths. This shared information is used by the agent to update its own belief of the environment and also help it make cooperative plans of actions to execute. Overall this leads to cooperative area coverage where agents explore the area such that a maximum number of gridcells are covered. This entire coverage algorithm can be grouped into four stages which are **Environment Sensing**, **Data Sharing**, **MCTS Planning** and **Action Execution**. The working of the algorithm during the full communication baseline scenario is explained in Algorithm 1.

Algorithm 1: Full Communication

Result: Area Covered by N agents upto user defined Coverage Goal
while *Coverage Goal is Not Met* **do**
  $t \leftarrow t + 1$;
  for $i \leftarrow 1$ **to** N **do**
    $agent_i.Map \leftarrow agent_i.EnvironmentSensing()$;
    for $j \leftarrow 1$ **to** N *and* $j \neq i$ **do**
      $agent_i.Map, agent_i.Plan_j(t-1) \leftarrow agent_i.DataSharing(agent_j)$;
    end
    $agent_i.A, agent_i.Plan_i(t) \leftarrow MCTSPlanning(agent_i.Map, agent_i.Plan_{-i}(t-1))$;
  end
  for $i \leftarrow 1$ **to** N **do**
    $ActionExecution(agent_i.A)$;
  end
end

Algorithm 1 delineates the way a team of N agents covers an arena. The exploration begins with a team of N agents spawned into an environment that supports full communication. The team has a predefined coverage goal that is known before the exploration begins. The coverage goal specifies the number of grid-cells that the team is required to explore. A high level working of the algorithm is that while the coverage goal is not yet met, each of the stages of Environment sensing, data sharing and MCTS planning is performed by each agent. Once each agent has performed all these stages, the planned action of each agent is executed during the Action Execution step.

Each agent maintains its own belief of the environment as a map $agent_i.Map$. At time-step t, $agent_i$ runs the Environment Sensing stage and updates $agent_i.Map$. Following this, the Data Sharing stage occurs for each peer agent j. During the Data Sharing stage, each agent i obtains the Map belief of each peer agent j. $agent_i$ uses this information to update its own map, $agent_i.Map$. In addition to the belief map, agent i also obtains the best plan of peer $agent_j$ that was planned at the previous time-step $t-1$. This information is then used during the MCTS Planning stage to compute the best possible action $agent_i.A$. It is important to note that at time step t, $agent_i$ uses the plans of the peer agents of a previous time step $t-1$. While this information is 1 time-step old, it is assumed that the peer agent will execute actions as per this plan at the current time step t also. From time-step t, the plan to be executed will be of length $t-1$. For the last final time-step, the agent computes the missing action by utilising the default policy strategy, that will be explained. This results in a complete plan of size t. After the completion of the MCTS Planning stage, each agent records the action $agent_i.A$ which it is supposed to execute. The MCTS planning stage also gives the plan that the agent takes at t and which it will share with other peer agents in the next time step t_{k+1}. Once the above 3 stages are complete for all agents, each agent executes their assigned action. Following this, the current coverage status of the global environment at timestep t is compared to the coverage goal. If the coverage goal is still not yet met then the time-step is incremented and the stages of Environment Sensing, Data Sharing, MCTS Planning and Action Execution are run again.

Restricted Communication Scenarios:
Algorithm 1 explains the working of the area coverage algorithm for a full communication setup. In the case of scenarios with limited communication, the algorithm differs in the Data Sharing Step. In the case of No Communication, there is no possibility of communication and hence there is no Data Sharing stage. In the case of Partial Communication, the Data Sharing stage is replaced by a Neighbourhood Data sharing stage. This will be discussed in subsection 4.5. In the case of partial communication with prediction in addition to the Neighbourhood data sharing stage, there is also an additional stage called the Peer Prediction stage. These stages are explained further in subsection 4.6.

4.2.3. Monte Carlo Tree Search Planner Design

An MCTS planner has been used in planning the paths of agents. The planning begins from the root node. At the root node, the agent utilises the environment state, its own position and the position

plans of the known peer agents. Using this information the agent needs to choose the most promising action from its action set ($\{UP, RIGHT, LEFT\}$). As mentioned in section 3.6.3, an MCTS planner has 4 major steps which are selection, expansion, rollout and back-propagation. These 4 steps are repeated multiple times and the final output from the entire computation is the action that gives the highest reward path. Along with the 4 steps, there are 2 important policies that help narrow down the search towards more promising choices of actions. These are the Tree Policy and the Default Policy. The Tree policy is used to select nodes or create expandable ones. In the implementation, the tree policy utilises the Upper confidence bound for trees (UCT1) which is given by equation 4.1.

$$\text{UCT1} = \bar{X}_j + 2C_p\sqrt{\frac{2\ln n}{n_j}} \tag{4.1}$$

In this equation, $\bar{X}_j$ refers to the reward obtained by taking the action j, n_j refers to the number of times that action j has been selected and n refers to the number of times the current parent node has been visited. The $\bar{X}_j$ term drives selecting actions that maximise rewards while the latter term enables exploring other possible actions. The equation when put in words, translates into directing the search towards regions that give a higher reward, while also ensuring that every action from the available list of actions is executed at least once. The UCT tree policy is able to narrow down the search towards nodes with larger rewards and gives a fair chance to all actions right in the beginning. It is also the policy that was utilised by the Hyatt et al [21] for robot exploration. Due to these 2 reasons, equation 4.1 has been chosen.

The default policy is used during the simulation stage to roll-out the consequence of the selected action for a predefined number of steps. Usually in most applications, the default policy used is a sequence of random actions. However in this case, the default policy selects actions that directs agent planning towards unknown grid-cells. By directing the roll-outs towards unknown grid-cells, agents can maximise the detection of unknown grid-cells in its plans. This helps create plans that covers larger number of unknown grid-cells.

Upon completion of the default policy roll-out after a predefined number of steps, the agent is required to compute a reward. The reward computed is called the $TotalReward$ and is the weighted sum of 2 other reward, which are the $LocalReward$ and the $GlobalReward$. The $LocalReward$ is given by equation 4.2,

$$LocalReward = \sum_{k=1}^{T}\left[\frac{1}{(t+1)^2}\left(C_{cov} - C_{hit}\right)\right] \tag{4.2}$$

In the above equation $C_{cov} = positivenumber$ if the robot lands up on a newly discovered grid cell at time step k and "covers" it, else $C_{cov} = 0$. $C_{hit} = positivenumber$ if the robot encounters an obstacle or another robot at time step k, else it remains 0. Due to the negative sign, the C_{hit} parameter is used to apply a penalty for instances of collision. T is the final time step for the simulation horizon. The normalisation using the $(t+1)^2$ term is used to decay the local reward overtime. The decay overtime is done to give a higher weight to rewards obtained at the nearer time-step t. The decay overtime is also referred to as discounting. In general, discounting can be performed by using a general expression $y = \frac{1}{(t+1)^n}$, where $n \geq 1$. Figure 4.2 shows the trend of $y = \frac{1}{(t+1)^n}$ for $n = 1, 2, 3$, and with 30 time steps. 30, as will be seen is the value that the authors of the baseline [21] have considered in their work. When $n = 1$, it can be seen that y takes up a fairly large value (0.10) even when $t = 10$. This results in reward values 10 steps ahead being given a sizable weight. If along this path at say $t = 2$, there are obstacles, then the robot is likely to move into the obstacle due to the larger weight for all subsequent time-steps. This behaviour of the robot moving into an obstacle was seen for $n = 1$ and thus higher values of n were tested. When $n = 3$, then the value of y becomes 0.00 at $t = 6$ itself. This prevents plans from further time-steps to be taken into account and the planning cannot look ahead beyond $t = 6$. When $n = 2$, then y becomes 0.00 at $t = 15$ and the planning horizon increases when compared to the previous case. Thus, $n = 2$ results in a discounted weight that is not as large as the $n = 1$ case and also that does not become 0.00 at smaller time-steps. Further, $n = 2$ has also been

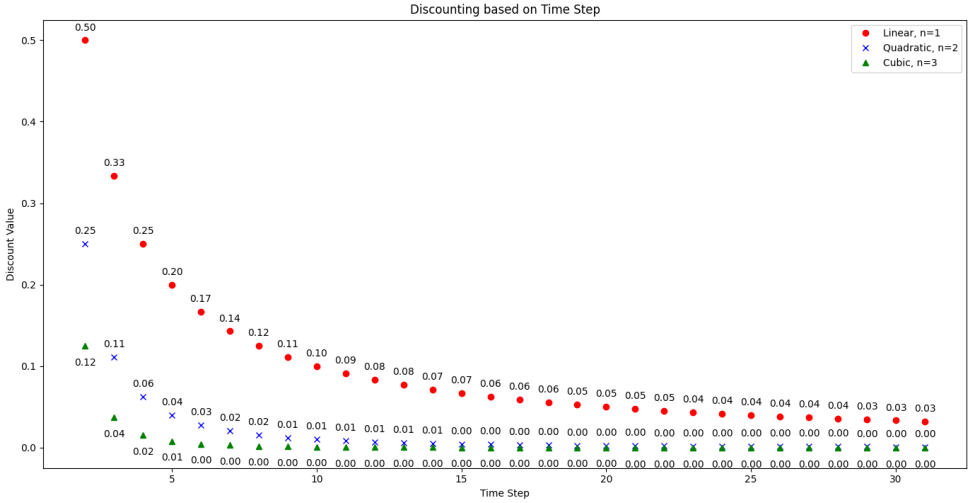


Figure 4.2: Discounting by time-step

utilised by the baseline method [21]. Due to these reasons, discounting using the $(t+1)^2$ term has been chosen for every communication scenario.

Apart from the $LocalReward$, the agent also computes a $GlobalReward$. The $GlobalReward$ is computed after the agent simulates the plans of its peers as well as its own actions. The aim of doing this is to include a reward for team behaviour and bias the agent to choose actions that do not lead it to redundant coverage. The $GlobalReward$ is given by,

$$GlobalReward = -\frac{NumberofUnexploredCellsintheGridMap}{TotalNumberofCellsintheGridMap} \tag{4.3}$$

The $GlobalReward$ is such that it is inversely proportional to the number of unexplored cells, thus giving higher reward to strategies that result in lesser number of unexplored cells. A lesser number of unexplored cells implies that there are more cells that have been discovered as covered or an obstacle. This discovery is made both by the current agent as well as the peer agents and thus using the Global Reward gives rewards to cooperative team behaviour. Utilising both these rewards, the $TotalReward$ is then computed as,

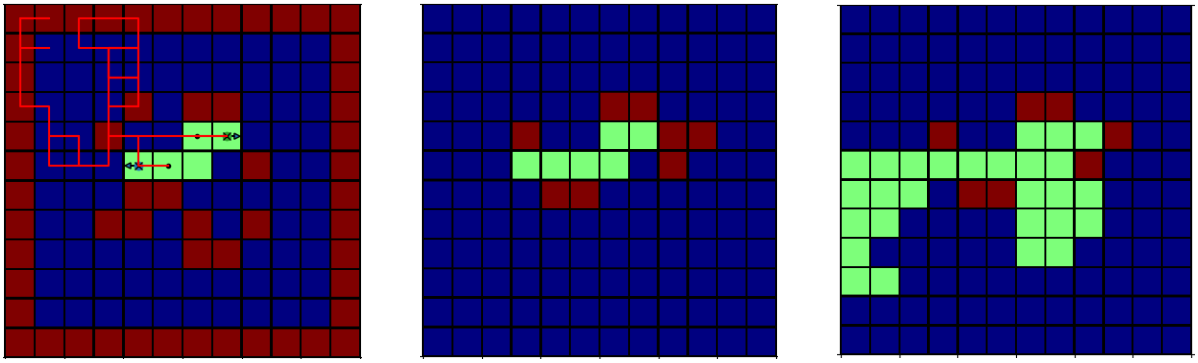
$$TotalReward = w_{LR}LocalReward + w_{GR}GlobalReward \tag{4.4}$$

Here w_{LR} and w_{GR} are the weights given to the 2 rewards. In the original work [21], w_{LR} = w_{GR} = 1. This $TotalReward$ is then utilised during the back propagation step, where the computed reward is first added to the existing reward of the active node. Following this the parent of each node is updated with the average of the reward of the child nodes. This propagation of Total Reward occurs until the root node. It is to be noted that the above terms of $TotalReward$, $LocalReward$ and $GlobalReward$ have been adopted as per the baseline implementation by Hyatt et al [21]. In actual, it can be seen that the $LocalReward$ is in fact a Value that is computed over many steps and the $GlobalReward$ is a score that is given after the completion of the rollout phase.

As mentioned in the beginning of this subsection, the default policy used directs the search towards unknown grid-cells. This can be done through 2 different approaches. The 2 design approaches are referred to as default policy 1 and default policy 2. There are 3 important phases during the default policy stage. These are **Simulation Path Planning**, **Peer Plan Execution** and **Reward Computation**. In the Simulation Path Planning phase, an agent begins running the computation of the plans on its belief map. In the Peer Plan Execution phase, an agent executes the plans of peer agents on its belief map. Reward Computation is the phase in which an agent computes the Total Reward (equation 4.4) of its planned actions.

The main difference between default policy 1 and default policy 2 is the method by which agents consider the plans of peer agents. In Default policy 1, an agent follows the sequence of **Simulation Path Planning -> Peer Plan Execution -> Reward Computation**, whereas in Default Policy 2 the sequence of **Peer Plan Execution -> Simulation Path Planning -> Reward Computation** is followed. Further, the agent performs the planning on its own belief map of the environment. In the below section, each policy has been explained in more detail. It is important to note that in the explanations, a full communication case is considered where the environment supports unrestricted communication.

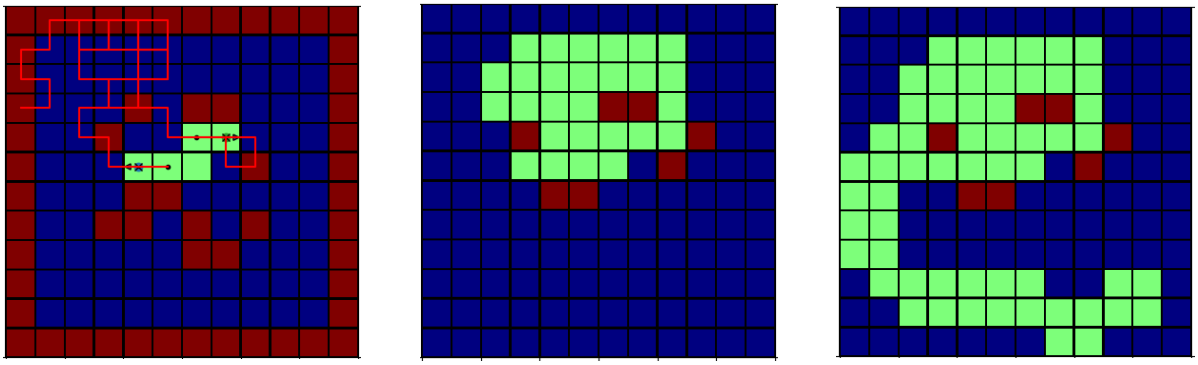
Default Policy 1
This is the default policy implemented in the original paper [21]. As mentioned in the previous paragraph, each agent in the simulation step first executes its own actions in simulation in the Simulation path planning stage. Subsequent to this the plans of peer agents is executed and marked by the agent on its belief map. This is possible as data sharing happens before path planning and prior to the execution of default policy 1 each agent obtains the path plans of each peer team member. The agent then moves onto reward computation, where it computes the $LocalReward$ and $GlobalReward$. This can be further understood by considering figure 4.3. Sub-figure 4.3a shows the current position of 2 agents in an environment. The left agent is the current agent (agent0) and its peer is on the right (agent1). Sub-figure 4.3b shows the status of the environment at the starting when agent0 begins computing the simulation paths. Once the simulation paths have been made, these paths along with the peer agent paths are executed and marked on the belief map of agent0, in the Peer Plan Execution step. In the Reward Computation step, the $TotalReward$ is computed.



(a) The Environment (b) Simulation Path Planning (c) Reward Computation

Figure 4.3: Default Policy 1 Steps

Default Policy 2
Default Policy2 differs from the earlier case in the execution steps of the 3 processes. At the very beginning, the agent performs the Peer Plan Execution stage and executes the path of peer agents. Following this, the simulation path planning stage is executed where agents plan their own paths in the simulator. The advantage of doing this compared to the above is that the paths of peer agents are marked as covered on the planning agent's map. Thus, plans made by the current agent would not move onto these grid-cells. Once the simulation path plans have been created by the agent, the agent moves onto computing the rewards in the reward computation step. The method can be better understood by considering figure 4.4. The positions of the 2 agents in the environment can be seen in sub-figure 4.4a with current agent (agent0) on the left and the peer agent (agent1) on the right. Sub-figure 4.4b shows the step of peer plan execution where agent0 executes the plans of agent1 in its simulation. Following this, the simulation path planning step is run where the agents move towards unknown grid-cells. Finally, at the end the reward computation step is run where the $TotalReward$ is computed.



(a) The Environment (b) Peer Plan Execution (c) Reward Computation

Figure 4.4: Reward Computation

4.3. Full Communication

The first scenario implemented is the reproduction of the work by Hyatt et al in [21] where agents share information in an environment that facilitates unrestricted communication. The working has already been explained in subsection 4.2.2 in terms of the major stages of the design approach. In such an environment agents can exchange any amount of data irrespective of their position and distance from one another. The data that agents share at each time step are:

1. Their current position
2. Their computed best path using MCTS
3. Their map coverage status

The position as mentioned in subsection 4.2.1 is the tuple (x, y, yaw). The computed best path is the list of actions that leads to the path with the highest reward, in the previous time step. The map coverage status contains the value ($\{Covered, Unexplored, Obstacle\}$) of all the grid-cells in the environment, as perceived by the agent.

4.4. No Communication

In this case of No Communication, agents have no capability to communicate any information to peer agents. The scenario is used as a starting step to remove the communication dependency from the original algorithm. While no robot can communicate with one another, agents can still sense the presence of peers that are within the sensing range. This is done for the purpose of collision avoidance. When planning the next action, the MCTS planner of the agent only simulates its own actions in the simulation step. Thus, there is no difference in using any of the default policies as mentioned previously. For the sake of uniformity in reward computation, each agent computes the $TotalReward$ as before by using equations 4.2, 4.3 and 4.4. A high level working of this algorithm can be seen in algorithm 2.

It can be seen that there are only 3 stages in the working which are, Environment Sensing, MCTS Planning and Action Execution. Further, MCTS Planning only uses its own map belief of the environment $agent_i.Map$. Once the most suitable action has been assigned to every agent ($agent_i.A$), each $agent_i$ executes its own action and the major stages are repeated if the coverage goal is still not met.

Algorithm 2: No Communication

Result: Area Covered by N agents upto user defined Coverage Goal
while *Coverage Goal is Not Met* **do**
  $t \leftarrow t + 1$;
  for $i \leftarrow 1$ **to** N **do**
    $agent_i.Map \leftarrow agent_i.EnvironmentSensing()$;
    $agent_i.A \leftarrow MCTSPlanning(agent_i.Map)$;
  end
  for $i \leftarrow 1$ **to** N **do**
    $ActionExecution(agent_i.A)$;
  end
end

4.5. Partial Communication

Most robots do have the capability to communicate in a physical environment. The communication between robots however is usually limited by factors like range, environment noise and energy consumption [11]. In order to incorporate this type of behaviour in the system, the scenario of partial communication based on the distance between agents is created. In this scenario agents share information when within a predefined communication range. The data that is shared is the same as mentioned in the full communication case, that is the individual coverage maps, most recent best paths computed, and positions. Once data is shared, each agent starts the planning process and considers the peer's plans during the simulation/rollout stage. When 2 agents are outside the communication range of one another, then each agent makes plans without considering the plans of the other. The working of the major stages of this scenario can be seen in Algorithm 3.

Algorithm 3: Partial Communication

Result: Area Covered by N agents upto user defined Coverage Goal
while *Coverage Goal is Not Met* **do**
  $t \leftarrow t + 1$;
  for $i \leftarrow 1$ **to** N **do**
    $agent_i.Map, agent_i.Neigh \leftarrow agent_i.EnvironmentSensing()$;
    for $j \leftarrow 1 \in agent_i.Neigh$ **do**
      $agent_i.Map$, $agent_i.Plan_j(t-1) \leftarrow agent_i.NeighborhoodDataSharing(agent_j)$;
    end
    $agent_i.A$, $agent_i.Plan_i(t) \leftarrow MCTSPlanning(agent_i.Map, agent_i.Plan_{Neigh}(t-1))$;
  end
  for $i \leftarrow 1$ **to** N **do**
    $ActionExecution(agent_i.A)$;
  end
end

Algorithm 3 is similar to Algorithm 1. The difference however is that during the Environment Sensing stage, $agent_i$ keeps a track of the neighbours that it senses to be within the communication range in a list $agent_i.Neigh$. $agent_i$ then moves onto obtain the plans for only these Neighbours. If there are no neighbours around then this list of plans is empty. At the MCTS Planning stage, $agent_i$ executes the plans for only the $agent_i.Neigh$ number of neighbours, denoted in the figure as $agent_i.Plan_{Neigh}(t-1)$. The best possible action ($agent_i.A$) is then computed using MCTS Planning. Following this, the actions of all agents are executed and the above steps repeat until the coverage goal is met by the team of agents.

4.6. Partial Communication with Prediction

In the case of partial communication, agents simulate the actions of peer agents that are within the agent's communication range. When peer agents move outside this range, the current agent does not consider it in its planning. If agents however are given the possibility to predict peer member plans, then it can lead to a situation where agents can still simulate and consider the plans of peer agents. This could possibly lessen the gap and make the situation approach the full communication scenario. This is the idea of Partial Communication with Prediction which is an extension of the Partial Communication case. The major stages of this scenario are explained in Algorithm 4. Algorithm 4 is similar to Algorithm

Algorithm 4: Partial Communication with Prediction

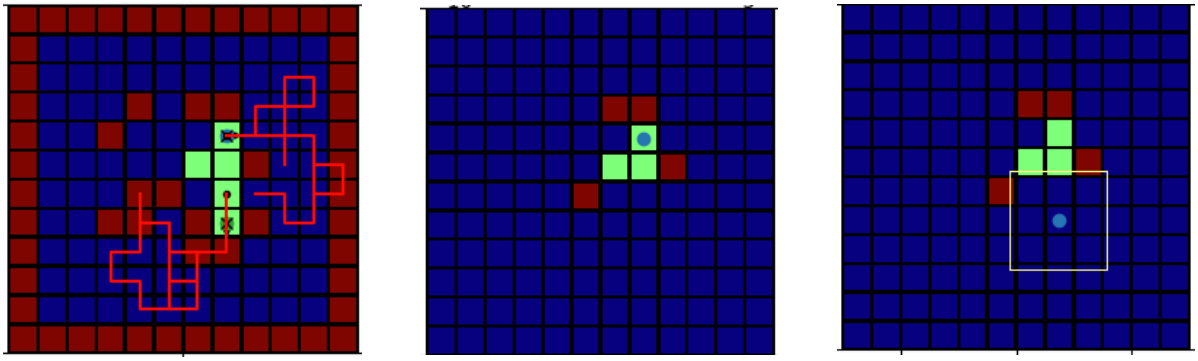
Result: Area Covered by N agents upto user defined Coverage Goal
while *Coverage Goal is Not Met* **do**
  $t \leftarrow t + 1$;
  for $i \leftarrow 1$ **to** N **do**
    $agent_i.Map, agent_i.Neigh \leftarrow agent_i.EnvironmentSensing()$;
    $agent_i.NonNeigh \leftarrow N - agent_i.Neigh$;
    for $j \in agent_i.Neigh$ **do**
      $agent_i.Map,\ agent_i.Plan_j(t-1) \leftarrow agent_i.NeighborhoodDataSharing(agent_j)$;
    end
    for $j \in agent_i.NonNeigh$ **do**
      $agent_i.Plan_j(t-1) \leftarrow agent_i.PeerPrediction(agent_j)$;
    end
    $agent_i.A,\ agent_i.Plan_i(t) \leftarrow MCTSPlanning(agent_i.Map, agent_i.Plan_{-i}(t-1))$;
  end
  for $i \leftarrow 1$ **to** N **do**
    $ActionExecution(agent_i.A)$;
  end
end

3. At the beginning, $agent_i$ computes the list of peer agents that are not in the communication range. This is denoted by $agent_i.NonNeigh$. As the total number of agents in the team (N) is known to each agent, therefore $agent_i.NonNeigh$ can be computed from the information of the number of neighbours $agent_i.Neigh$. Once $agent_i.NonNeigh$ is computed, the most recent plans of the agents outside the communication range is predicted using the methods that will be described in the below sub sections. The plans of all peers, within or outside the communication range is used by $agent_i$ during the MCTS planning stage. Once the most suitable action $agent_i.A$ is computed in this manner for every agent, then the actions are executed and the environment makes a transition. These steps continue until the coverage goal is met.

As the goal of the entire team is to cover grid-cells, each agent's individual behaviour is the same, that is to be biased towards covering more numbers of unknown grid-cells. Each agent can then use this fundamental rule to predict what its peer agent performed. These predictions will be based on the agents own map. The agents own map is not the most accurate representation of the environment as it is continuously getting updated through the sensing mechanism of the agent and map merging with maps of peer agents within communication range. As it is not an accurate representation, therefore this may lead to imperfect predictions. However, agents also know the best planned paths of peer agents that it obtained when within the communication range. This list of best planned paths, though finite in number can be used to improve the accuracy of predictions. As an example, consider 2 agents, $agent_i$ and $agent_j$. When the 2 agents are not in communication range, then $agent_i$ can predict the path of $agent_j$ using the last shared best planned path of $agent_j$. This information is finite, and thus for other instances $agent_i$ needs to make predictions of the path of $agent_j$. The method by which agents predict the paths of the peer agents is based on heuristic computation. The heuristic used is explained in subsection 4.6.1. Further, there are a number of corner cases that must be solved during the prediction. These will be explained in subsections 4.6.2 and 4.6.3. The entire working of the prediction algorithm, including the corner cases is explained in subsection 4.6.4.

4.6.1. Coverage-by-step Heuristic

When agents are within communication range, then they share the best planned paths. The best planned paths is a list of actions that the peer agent computed in the last time step. For agents that are outside the communication range, the current agent makes predictions of peer paths as a list of actions. These predictions are based on the last shared information by the peer agent. However, as time-steps progress this information cannot be used used by the current agent at all times. Thus, for such instances when the actions of peers is completely unknown, the coverage-by-step Heuristic is used. In the coverage-by-step heuristic the current agent predicts the next action of the peer agent by computing a reward for the 8 grid-cells around the predicted/last seen position of the peer agent. This can be understood by considering figure 4.5a. In the scenario shown, there are 2 agents (agent0 and agent1) which are present outside communication range. Sub-figure 4.5b shows agent0's belief of the environment at the current time step while sub-figure 4.5c shows the estimated position of its peer agent, agent 1 at the current time step. It is important to note that this figure shows the position of agent 1, as predicted by agent 0. The yellow box shows the 8 zones/grid-cells that agent0 uses in order to predict the next position and next action of agent1, using the heuristic reward.



(a) The Environment (b) Agent Belief (c) Agent Peer Belief

Figure 4.5: Heuristic h1 computation

The heuristic reward ($Reward_{heur}$) is computed using the reward value for each zone. This reward is computed based on equation 4.5 where R_{cov} denotes the coverage reward of the particular zone and N_{min_steps} gives the minimum number of steps that the belief agent is required to move from the current position to the zone position.

$$Reward_{heur} = \frac{R_{cov}}{N_{minsteps}} \tag{4.5}$$

The value of R_{cov} is given by the equation

$$R_{cov} = \begin{cases} +2, & \text{if the zone is unknown} \\ +1, & \text{if the zone is covered} \\ 0, & \text{if the zone is an obstacle} \end{cases} \tag{4.6}$$

The above values of R_{cov} were tested and found to give satisfactory results. Initially the value of R_{cov} was set as +1 for moving to an unknown gridcell, 0 for moving to an already covered grid-cell and -1 for moving into an obstacle. However, it was found that this set of R_{cov} did not work well and especially when the agent's peer belief was surrounded by covered regions. In such a case, the reward zero was given to all the surrounding grid-cell regions and the peer belief position would remain in the same position or move around randomly, thus reducing accurate predictions of the peer agent positions.

The value of N_{min_steps} can be understood by considering figure 4.6. The figure shows the orientation of the agent (indicated in orange) in every direction and denotes the corresponding values N_{min_steps} and the first step in that list as a tuple format (N_{min_steps},First step). As an example consider the top left case where the agent faces the upward direction. The grid-cell right above the agent is only 1 move away from the agent and the first action that moves the agent to this grid-cell is UP.

Therefore the tuple is given by (1,UP). Within the same orientation figure, if the case of the grid-cell just below the agents location is considered, then there are minimum 3 steps that would be required to move to the grid square. These steps are [RIGHT,RIGHT,UP] or [LEFT,LEFT,UP]. Thus the tuple is given by (3,LEFT) or (3,RIGHT). The minimum number of steps and the first action of the list can be computed in the same way for other grid cells. Further the steps are dependent on the orientation of the agent and therefore the same values apply in all possible orientations when rotated by 90 degrees, which can be seen in the remaining 3 cases of figure 4.6. Overall using equation 4.5, the agent uses $Reward_{h1}$ to obtain the first action that moves it to the zone with the highest value of $Reward_{h1}$. In most cases a higher reward indicates an uncovered zone that can be reached with minimum number of steps.

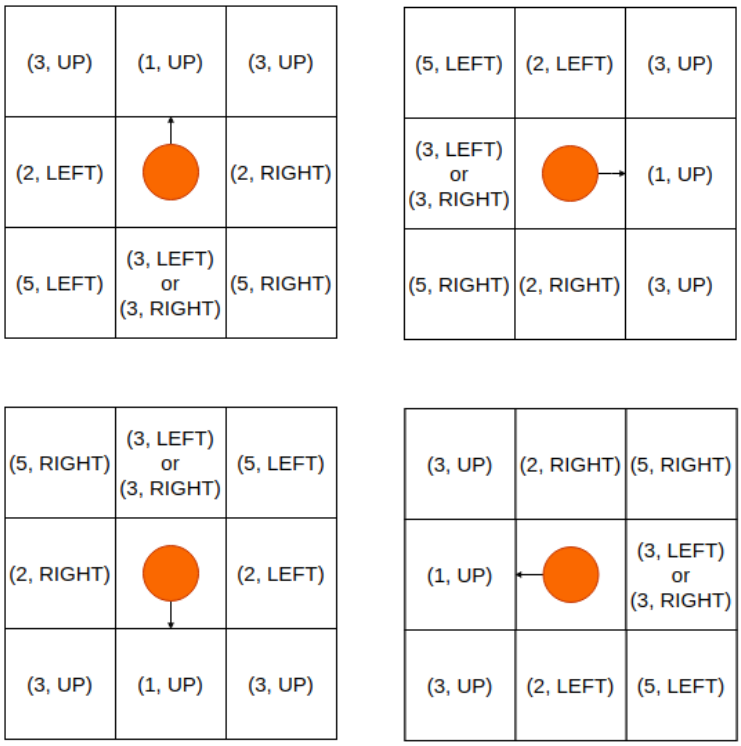
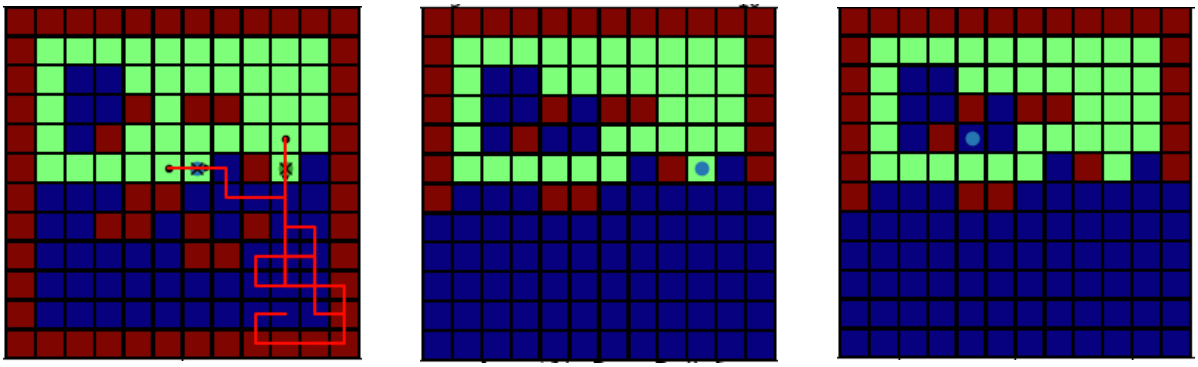


Figure 4.6: Step Computation with direction

4.6.2. Special Cases

While the coverage-by-step heuristic appears to be a straightforward way in predicting the peer agents position by biasing it towards unknown regions with minimum number of steps, it does have some limitations. There were 2 limitations that have been found and additional measures to deal with them have been put into place. The first limitation is that the coverage-by-step heuristic can sometimes choose zones that is present behind obstacles. Due to this, agents have no direct path to such zones and no movement takes place. The second limitation is that agents only have the capability to look one step ahead to plan paths. As will be mentioned further, looking beyond one step can better mimic the paths planning of peer agents.

The first limitation about no direct path being present usually occurs when a goal position is blocked by obstacles. This can be understood by considering the scenario in figure 4.7. In this scenario there are two agents, agent0 and agent1 that are outside the communication range. Therefore each of them utilise the coverage-by-step heuristic to predict the location of one another. Figure 4.7b shows the



(a) The Environment (b) Agent Belief (c) Agent Peer Belief

Figure 4.7: Obstacle Lock

location of agent1 in the map, as well as what it perceives the environment to be at the current time step. Figure 4.5c displays what the agent0 currently predicts the location of agent1 to be. It is important to note that this position of agent1 is obtained after the predicted position of agent1 has moved over the 2 grid cells that are on its right and top right corner. Further, the orientation of agent1 is $2*\pi$, indicating that it is pointing leftwards. Thus, agent0 assumes that these 2 grid-cells are already explored. Therefore while running the heuristic, agent0 takes this into account and amongst the surrounding 8 cells of agent1, it only has the top left grid-cell as an unknown grid-cell. As the orientation of the belief pose of agent1 is $2 * \pi$, in such a scenario the top left grid-cell which is the only unknown location is 3 steps away (according to figure 4.6) with the first action being UP. However due to the presence of an obstacle on both the sides, the agent cannot move to this position and keeps attempting to move straight (UP). Due to this the position of agent1 remains constant and the belief of agent1 appears to be stuck for all further time steps. This type of blocking by obstacles is what we refer to as the **Obstacle Lock Condition**. In order to deal with this scenario, a separate obstacle lock detector is introduced that adds higher number of steps to such positions. In doing so, the reward for moving to such zones decreases as per equation 4.5. In addition to observing 2 obstacles that block the movement of a peer agent, another way of blocking also occurs when a single obstacle is present. This is shown in figure 4.8. In this case the position of the agent has an orientation of $\frac{3\pi}{2}$ and is pointing downwards. When the

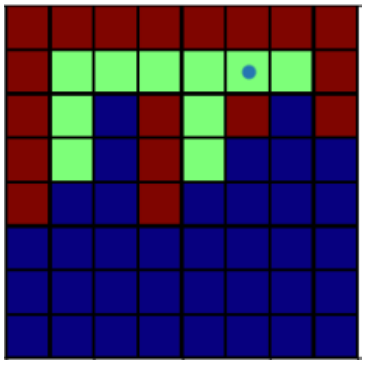


Figure 4.8: Single Obstacle Lock

coverage-by-step heuristic is run, then the unknown region on the bottom right (top-right for the agent) is selected which is the only unknown grid-cell and requires an action UP to be executed. However right in front of the agent, there exists an obstacle and therefore the agent would be unable to move ahead. Due to this the agent continues to select an UP action, but the peer belief remains in the same position and gets stuck. In order to tackle this scenario, the agent is given the ability to turn or execute the action RIGHT/LEFT in order to move out of this position.

The 2nd limitation of planning only 1 step ahead is explained here. While in most cases the coverage-by-step heuristic manages to perform satisfactorily by only sampling grid-cells that are 1

grid-cell away from it, at many positions there are decision points where the agent can choose actions randomly. Such positions are present especially when the agent is in fully covered zones or when the agent is surrounded by obstacles. To tackle this, whenever agents are in a scenario where the surrounding 8 grid-cells around it give the same value of $Reward_{heur}$, then the agents samples N steps ahead to look for additional difference on the benefit of actions. When looking N steps ahead, the agent calculates the reward of the grid-cell that it encounters according to the R_{cov} value obtained from equation 4.6. As an example consider a simple case shown in figure 4.9, where the peer belief of an agent is estimated to be present close to the boundary position. The agent is pointing towards the right direction that is, its yaw angle is 0. If we solely rely on the coverage-by-step heuristic, then the agent obtains the highest reward of $Reward_{heur}$ for moving UP (rightwards in the grid-world), as it only requires 1 step to move in this direction. However, it can be seen in the figure that if the peer belief state agent chooses to turn RIGHT (downwards in the grid-world), then within 2 steps there is a higher possibility of covering an unknown state. Thus if the peer belief agent is able to look ahead further than one step in positions of uncertain decisions, then it has the possibility to make a better estimates of the future positions of its peers. This increases the closeness of its prediction to the actual peer agent, that uses MCTS and hence is biased towards unknown grid-cells that are even located further than 1 step.

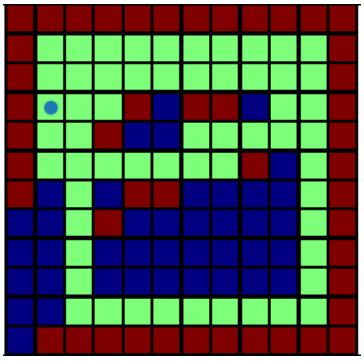


Figure 4.9: Look ahead heuristic for decision resolution

...

4.6.3. DIY Rewards and default policy

In the previous sub-sections it is mentioned that the agent utilises 2 deterministic pieces of information, that is the position and the best planned paths of the peer agents. Using this, the heuristic prediction can be used by the agent to predict the actions of the peer agents. However as the environment is itself unknown and is being discovered by each agent as it moves around the area, agents would not be able to make perfect predictions about the movement of peers at all times. The number of accurate predictions decreases with the increase in time since the last meeting between agents. Now, the idea of using predictions is to enable agents to know the path taken by agents beyond the communication range. Using this the agent can distribute itself better in the surroundings and avoid redundant coverage. When the predictions are right, this potentially leads to the coverage of more unknown grid-cells in the map by the team. However when the predictions are wrong, the number of coverage steps can increase. This is because, in a case when the agent assumes that its mispredicted peer agent would cover a particular region (say $region_r$), then it would move elsewhere. However, as the peer agent is mispredicted, none of the agents move towards $region_r$ and the grid-cells in this region remain unexplored. This effects the overall planning with an increase in the number of steps.

This type of problem of mispredicted peer agents was also noticed by Claes et al in [12] and the approach used by the authors was to introduce a do-it-yourself (DIY) reward for planning agents. The DIY reward tackles such uncertainties by giving preference for the agent to perform tasks themselves. In the case of exploration if agents compute a DIY reward for those agents whose paths are being predicted, then the number of chances when grid-cells remain unexplored would be minimised. In the order to mimic this type of Do-it-yourself strategy in this thesis, the default policy utilised by each agent is modified.

Default Policy 3
In subsection 4.2.3, there were 2 default policies described namely default policy 1 and default policy 2. Default policy 1 follows the strategy of **Simulation Path Planning -> Peer Plan Execution -> Reward Computation** while Default policy 2 follows the strategy of **Peer Plan Execution -> Simulation Path Planning -> Reward Computation**. It can be seen from above that Default policy 1 involves computing the simulation path plans before executing the paths of the other agents. Thus by doing this the current agent does not consider the plans of the peer agents while planning paths. In order to mimic the do-it-yourself strategy, agents utilise this strategy for agents that are outside the communication range and for whom predictions are being made. For the agents that are within communication range, the agent computes default policy 2. By doing this, the current agent is able to override the plans of peers for which it predicts paths and has less information about, whereas it considers the plans of agents within communication range in its plans. It is important to note that the reward computation remains the same and the peers for which predictions have been made only effects by the computation of the $GlobalReward$ of the agent. This ensures that the current agent computes plans by taking teammate behaviour into account. In summary, Default policy 3 is thus a hybrid policy used by current agent in the simulation step and runs default policy 1 for agents outside the communication range and default policy 2 for agents within the range.

4.6.4. The Working Algorithm

In the partial communication with prediction scenario, various functions/blocks that have been discussed, namely -

1. Coverage-by-Step Heuristic
2. Obstacle lock detection
3. One step ahead planning
4. Peer agent best path execution
5. DIY reward and Default Policy 3

Algorithm 5 shows how these stages come together. The algorithm considers 2 agents, $agent_i$ & $agent_j$ and shows how $agent_i$ predicts the path of $agent_j$. $agent_j$'s path prediction is represented by a list of actions. When $agent_i$ detects $agent_j$ in its communication range then $agent_i$ obtains the best planned paths of $agent_j$ as the list. The last meet time counter which is used to track the time when the agents last met is reset to 0. When the $agent_i$ detects that $agent_j$ is outside the communication range, it begins the prediction of $agent_j$'s path. The first step however is to update the current location based on the last known location of the peer agent. This is indicated by the upper part of the algorithm. At the lower part of the algorithm is the actual computation of the list of actions of the peer agent. In the case of the upper part of the peer action state update, the algorithm begins by first checking if the action corresponding to the current time as indicated by last_meet_time in the $peer\_best\_action\_tracker$ is known. If this is known, the action is executed and the new position is recorded ($next\_position$). In the case when the agent does not know this action, $agent_i$ begins to compute the predicted action and uses an obstacle lock filter to filter out grid-cells in its surrounding that do not have an obstacle lock. Following this the coverage by step heuristic is calculated for the cells under test. The heuristic also includes the case of one step ahead planning which samples N steps ahead for additional information that is required by grid-cells of equal computation value. The action is recorded as the next action, which is then used to update the position ($next\_position$). Once the state has been updated, the phase of computing the action list of peer agents begins. The plans of agents is computed for $T\_Horizon$ number of steps using the same method. The difference is that at each iteration, the $next\_action$ is recorded into the $predicted\_action\_list$. After $T\_Horizon$ number of steps, the $predicted\_action\_list$ of $agent_j$ as predicted by $agent_i$ is obtained. This list will then be used in the simulation stage of the MCTS planner.

Algorithm 5: Heuristic h3, used by $agent_i$ to model $agent_j$ movement

Result: $predicted\_action\_list$
while *Coverage Goal is Not Met* **do**
  ...;
  if $agent_i$ *and* $agent_j$ *are in communication range* **then**
    $peer\_best\_action\_tracker$ = Data Sharing($agent_i$,$agent_j$);
    last_meet_time = 0;
  else
    // First update the position of the agent (Upper part)
    if $peer\_best\_action\_tracker[last\_meet\_time]$ *known?* **then**
      $next\_action$ = $peer\_best\_action\_tracker[last\_meet\_time]$;
      $next\_position$=Motion($next\_position$,$next\_action$);
    else
      $cells\_to\_test$ = Obstacle_Lock_Filter($agent_i.peer\_position_j$);
      $next\_action$ = Coverage_by_step($agent_i.peer\_position_j$,$cells\_to\_test$);
      $next\_position$=Motion($next\_position$,$next\_action$);
    end
    // Compute the action list of the peer agent (Lower part)
    last_meet_time += 1;
    t = last_meet_time ;
    $position\_tracker$ = $next\_position$;
    while *t<=T_Horizon* **do**
      if $peer\_best\_action\_tracker[t]$ *known?* **then**
        $next\_action$ = $peer\_best\_action\_tracker[t]$;
        $position\_tracker$=Motion($position\_tracker$,$next\_action$);
        $predicted\_action\_list$.Append($next\_action$);
      else
        $cells\_to\_test$ = Obstacle_Lock_Filter($agent_i.peer\_position_j$);
        $next\_action$ = Coverage_by_step($agent_i.peer\_position_j$,$cells\_to\_test$);
        $position\_tracker$=Motion($position\_tracker$,$next\_action$);
        $predicted\_action\_list$.Append($next\_action$);
      end
      t += 1;
    end
  end
  ...
end

4.7. Hyper-parameter Selection

From the above sections, it can be seen that there are a number of hyper-parameters that can be tuned. These, along with the selected values are listed in table 4.1. In this section, the motivation for the selection of the values for the listed hyper-parameters will be explained.

Sl No	Hyper-parameter	Description	Value
1	C_cov	Coverage Reward for the MCTS Planner	5
2	C_hit	Collision Penalty for the MCTS Planner	2
3	lw	Weight for the local reward	1
4	gw	Weight for the global reward	1
5	T_horizon	Number of rollout steps in the default policy	30
6	C_p	Exploration Coefficient for the MCTS Planner	$\frac{1}{\sqrt{2}}$
7	comp_budget	Computational budget for the MCTS Planner	200

Table 4.1: Hyper-parameter Values

The first two hyper-parameters seen in table 4.1 are C_cov and C_hit. These were set as 1 and 2 by Hyatt et al [21]. However, other values of C_cov and C_hit were also tested by spawning a single agent into a 20x20 gridcell arena with a goal to explore 95% of the arena. 95% coverage on a 20x20 gridmap, as will be explained in the next section, corresponds to exploring all 400 grid-cells but leaving 3 grid-cells unexplored. In each case C_cov and C_hit values were varied, simulations were ran for 50 times and the exploration steps were recorded. The results can be seen in figure 4.10. From the figure, it can

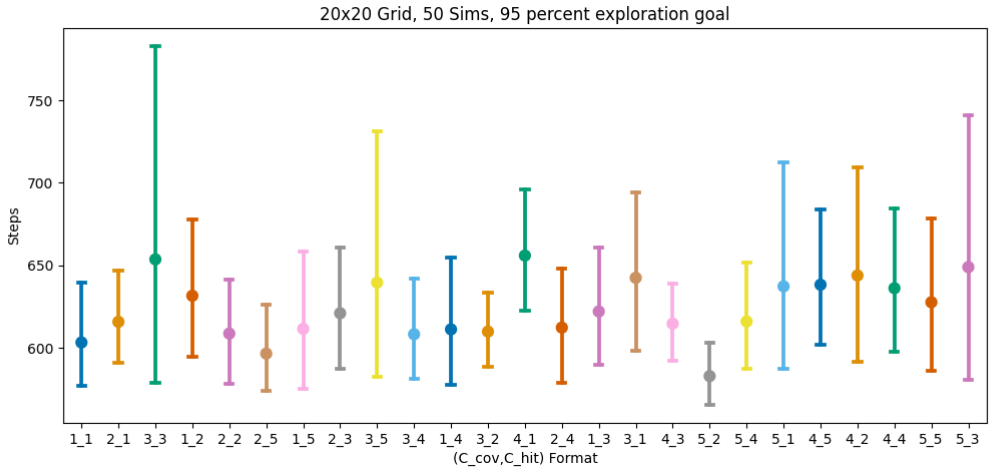


Figure 4.10: (C_cov,C_hit) Exploration Comparison

be seen that the combination of C_cov=5 and C_hit=2 results in the lowest number of exploration steps that has a relatively lower confidence interval overlap when compared to other possible values. This may possibly be due to a high enough value for C_cov that is constrained by a suitable penalty value. Due to this observation, these values of C_cov and C_hit were utilised. In terms of the the weight for the local and global rewards, it was found that setting lw=0 and solely using the global reward resulted in the agent moving in random directions. This is because of the nature of the global reward that is given by equation 4.3. The global reward gives a score that is based on the number of covered cells without discounting the actions taken at later time-steps, which is done in the local reward(equation 4.2). Initially when the entire region is unexplored, moving in any direction gives the same reward value. If agents initially move around randomly in the arena, then the overall number of steps also increases. Due to this the local reward was definitely included in the computation of total reward. It was found that the weights of lw=1 and gw=1 gave comparable numerical values of local and global rewards. It was also used by Hyatt et al [21] in their experiments. T_horizon = 30 and C_p = $\frac{1}{\sqrt{2}}$ was also used by Hyatt et al [21] and found to work satisfactorily. The comp_budget gives the number of times an agent performs the basic MCTS steps of selection, expansion, rollout and back-propagation. Initially 500 was set. However this led to high number of computations that made simulations very slow. A value of 200 however was found to bring the simulation time lower and hence was chosen.

4.8. Frontier Communication

In order to compare the various scenarios having MCTS with an existing work in literature, the frontier communication algorithm as presented in the work of Antoine et al [5] and adapted by Benavides et al [6] was implemented. This specific work was selected as it also worked with the constraints of limited communication, a distributed setup and an unknown environment. Further the method was applied to the environment exploration problem. In the approach a team of multiple agents are spawned into the environment. As it is a limited communication environment, agents have the capability of sharing information between one another, when within a communication range. The information shared is the same as the MCTS partial communication case, namely the map coverage status, the position and best selected path plans in the last time step. Each agent then executes an algorithm that consists of 3 main phases - task identification, task allocation and task execution. A task is defined to be a frontier grid-cell. A frontier grid-cell is a type of grid-cell that is located at the boundary between a known and

an unknown region. Figure 4.11 shows an example of frontiers that is marked in yellow. The known region is marked in white, while the unknown region is marked in grey. The task identification phase

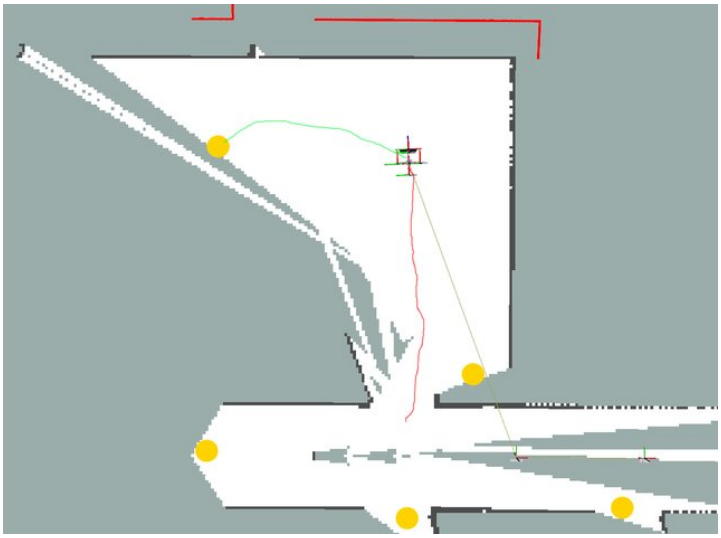


Figure 4.11: Environment Frontier [35]

involves detecting such points in the environment. To detect such points, an agent samples surrounding grid-cells and adds such frontier points to a task list. The number of surrounding grid-cells sampled depends on the sensor range of the agent. It is important to note that this process is done by multiple agents and if another agent is within communication range, then the agent also adds the peer agent's target list to its own list. Once the task list is set, the agent moves onto selecting a task in the Task allocation phase. In the task allocation phase, each agent runs the minPos algorithm [5] in a distributed manner. The algorithm is shown in figure 4.12. As seen in figure 4.12, the algorithm requires a list of

Algorithm 3: MinPos — Complexity $O(nm)$

Input: $\mathcal{R}_i$, $\mathcal{C}$ cost matrix

Output: α_{ij} assignment of robot $\mathcal{R}_i$

foreach $\mathcal{F}_j \in \mathcal{F}$ **do**

$$\mathcal{P}_{ij} = \sum_{\forall \mathcal{R}_k \in \mathcal{R},\ k \neq i,\ \mathcal{C}_{kj} < \mathcal{C}_{ij}} 1$$

end

$\alpha_{ij} = 1$ with $j = \underset{\forall \mathcal{F} j \in \mathcal{F}}{\operatorname{argmin}} \mathcal{P}_{ij}$

In case of equality choose the minimum cost among min $\mathcal{P}_{ij}$

Figure 4.12: Minimum Position Algorithm

robot positions R_i and a cost matrix C. The cost matrix C_{ij} has the robot number as the row values while the columns indicate the task/frontier number. Each entry of the cost matrix is the cost of agent i of selecting the task j. The cost in this case is the Euclidean distance. It is important to note that each robot has a cost matrix of its own, whose entries depend on the peer agents that are present around it. For each task j, the current agent computes the number of robots that have a lower cost value. The number of such robots are added together and recorded in a matrix P_{ij}. The current agent then assigns itself to task j for which it has the lowest P_{ij} value. A low P_{ij} value indicates that the current agent is closer to a task j than its peers. The current agent does this for itself and nearby peers. Overall, by using this algorithm the current agent can select frontier zones that are nearest to itself while predicting the allocation made by nearby peer agents.

5

Experiments

In this chapter the experiments utilised to answer the research questions framed in subsection 2.2 will be discussed. The term population in this chapter is used to refer to data that have some common attributes. For example a data-set obtained for the exploration of a 20x20 map with 2 agents having Partial Communication is one population, and a data-set for the exploration of a 20x20 map with 2 agents having Full Communication is another. The methodology used in each case is to plot the mean of each population along with a 95% confidence interval overlap. For each population, exploration scenarios were run 10 times on 5 different maps, thereby leading to 50 simulations each. Table 5.1 gives the abbreviations that are used to refer to various scenarios.

Sl No	Communication	Prediction	Default Policy	Abbreviation
1	Full	-	1,2	FULLCOMM
2	No	-	1,2	NOCOMM
3	Partial	-	1,2	PARCOMM
8	Partial	Yes	3	PARCOMM_PRED

Table 5.1: Scenarios and Abbreviations for MCTS methods

The categorisation is based on the type of communication, prediction possibility and the default policy strategy used. Apart from the scenarios mentioned in the table, the scenario of agents performing frontier based exploration is denoted by the abbreviation FRONTCOMM. Exploration in all scenarios have been performed on 20x20 or 40x40 grid maps. The obstacle density of each map is set as 10% and the maps can be seen in Appendix 7.3. The specific size of a 20x20 gridmap with an obstacle density of 10% was chosen as it is the same size that the authors of the baseline method, Hyatt et al [21] chose in their experiments. Apart from using a 20x20 gridmap, experiments were also performed on a 40x40 dimension gridmap with an same obstacle density of 10%. This was done to test the working of the algorithms on larger environments. Further, all explorations were run with a team of agents having a 95% coverage goal. Initially a 100% exploration was given to the team of agents. However in many cases consisting of smaller number of agents (upto 3), there was high variance of results. This was due to situations in which agents would cover a large portion of the arena but leave out exploring far off lying single gridcells. Due to finite computation, agents would take more number of steps to make plans that reach such single unexplored gridcells. Thus, a 95% coverage goal was used instead. A 95% exploration goal on a 20x20 map translates to exploring all the 400 grid-cells but leaving 3 grid-cells unexplored, while a 95% exploration goal on a 40x40 map leaves out 12 grid-cells from the 1600 gridcells present.

By default, the sensor range and communication range of each agent were set as 1. With a sensor range as 1, agents could sense the status of nearby 8 grid-cells as explained in subsection 4.2.1. A communication range of 1 implied that agents could share information with agents when present in the surrounding 8 grid-cells. Further, termination of experiments/simulations in each case took place based on the coverage of the actual environment with what the team of agents covered and not based on the status of coverage of what an agent perceived.

To understand the motivation behind the experiments conducted, we refer to the 3 research questions present in subsection 2.2. The first research question **RQ1** considers agent strategies that can lead to minimal number of exploration steps of unknown environments with limited communication. The second research question **RQ2** lays emphasis on the effect of limited communication on cooperative exploration strategies. The last research question **RQ3** puts the focus on techniques that agents in a limited communication arena can utilise to replicate the performance as in a full communication scenario. The first experiment performed studied the effect of the MCTS default policy on exploration. This will be explained in section 5.1. This experiment was done primarily to motivate the default policy design choice and provide part of the answer to RQ1. Based on the results, a default policy was selected for all subsequent experiments. A study was then performed to measure the impact of agent team size on the exploration performance, for each MCTS scenario (section 5.2). The experiment answers RQ1, RQ2 and RQ3 because it studies the effect of agent team size, as well as the communication scenarios on the exploration performance. While one major constraint considered was a completely unexplored arena, a study was conducted to observe the effect of having additional knowledge of the location of obstacles in the arena in the PARCOMM and PARCOMM_PRED scenarios. This will be explained in section 5.3. In this study, agents were given prior information of the arena with the aim of moving the performance closer to FULLCOMM, thus aiming to answer RQ3. Another experiment that tries to answer this question is the study on the effect of varying the communication range on the exploration performance for PARCOMM and PARCOMM_PRED. This will be described in section 5.4. The final experiment conducted was to compare the MCTS based algorithms with the frontier exploration (FRONTCOMM) method. The experiment aims at studying the effectiveness of the look-ahead planning capability of MCTS algorithms in the task of exploration, and thus is used to give part of the answer to RQ1. This will be explained in section 5.12. At the end of the chapter, section 5.6 summarises the results obtained from each experiment, draws key points and lays the foundation for the conclusion to be made in the following chapter, chapter 6.

5.1. Exploration Performance - Default Policy Comparison

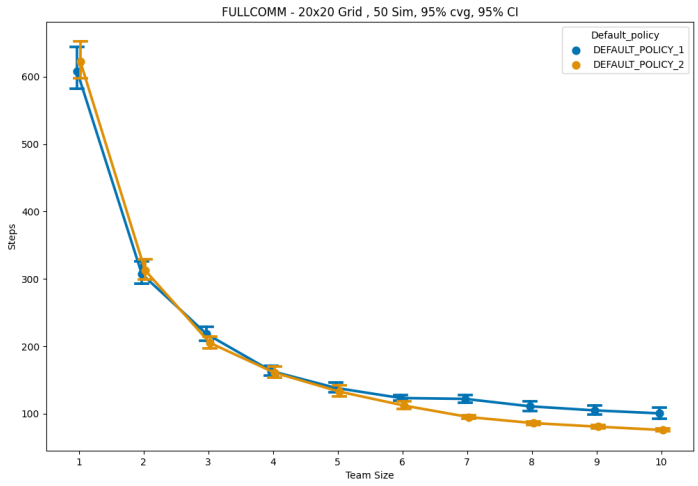
As discussed in chapter 4, there were 2 default policy methods that were used - Default Policy 1 and Default Policy 2. In order to compare the working of each method, simulations were run in the FULLCOMM scenario for 1 until 10 robots. 50 simulations were run for each population in 20x20 & 40x40 arenas and in each case a coverage goal of 95% is given to the team. The results obtained can be seen in figure 5.1 and figure 5.2.

In each case, the graph consist of the mean values of the steps along with a 95% Confidence interval. Tables 5.1b and 5.2b show the exact value of the means as well as the size of the confidence intervals for 20x20 and 40x40 maps. Some key observations from the graphs and tables are as follows:

1. It can be seen from figure 5.1a, that the downward trend of the exploration with increasing number of agents is more smoother for the case of Default Policy 2, in a 20x20 arena.

2. From table 5.1b it is observed that in a 20x20 arena, agents with default policy 2 take lesser number of steps than default policy 1 for 8 out of 10 times. In the case of a 40x40 grid (table 5.2b), agents with default policy 2 take lesser number of steps than default policy 1 for 6 out of 10 times.

The above observations can be reasoned by considering the core difference between Default policy 1 and Default policy 2 which is in the way the agent plans out its current path and compute the reward. As discussed in section 4.2.3, in Default Policy 1 an agent starts making plans before executing the policies of peer agents on its own belief map. This is different to what happens in the case of default policy 2 where the agent executes policies of peer agents on its own belief map. Only after this is when the agent plans the path and computes the total reward. This implies that the agent considers the movement of peer agents while computing its own plans. As it is a full communication case, the current agent can receive information at every point of time. Thus in the case of default policy 2, the agent is able to give lower local rewards to those grid-cells that are already covered by peer agents. This results in the current agent planning paths towards locations that are away from the peer agents which results in agents moving away from each other and avoiding redundant coverage. In the case of the default policy 1, agents plan their actions before considering the plans of peer agents. Due to this, an agent has already planned the paths and these paths may cross the paths of the peer agents. Due to this,



(a) 20x20

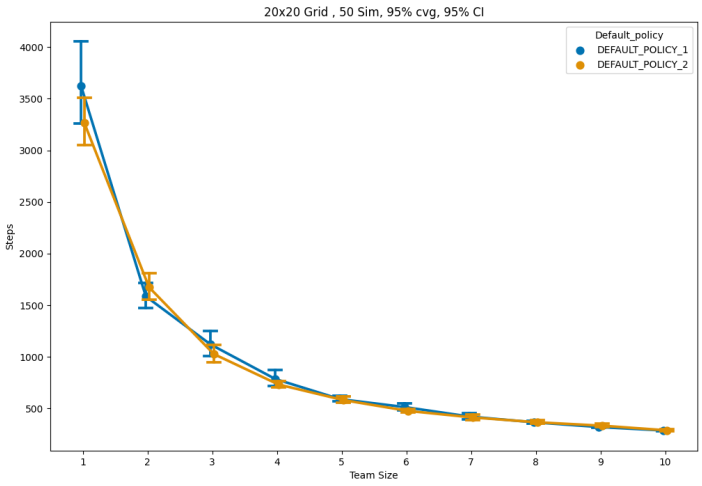
Team Size	DP1 mean	DP1 CI	DP2 mean	DP2 CI
1	608	32	622	28
2	307	18	313	16
3	218	10	205	9
4	163	7	161	9
5	138	7	133	9
6	123	4	112	6
7	122	6	95	3
8	111	8	86	3
9	105	7	81	2
10	101	9	76	2

(b) 20x20 Data

Figure 5.1: Coverage steps versus Default Policy - 20x20

the agents do not spread out and hence have a higher number of steps as compared to Default Policy 2.

As the number of steps of coverage was found to be lower in the case of default policy 2 than default policy 1 for FULLCOMM in a majority of team sizes for both 20x20 and 40x40 maps, for the remaining experiments Default Policy 2 has been used in the MCTS planner.



(a) 40x40

Team Size	DP1 mean	DP1 CI	DP2 mean	DP2 CI
1	3623	416	3265	233
2	1586	127	1674	138
3	1121	133	1028	91
4	786	79	733	31
5	591	29	582	30
6	514	36	477	17
7	423	33	414	28
8	367	14	367	16
9	321	10	334	19
10	288	8	289	10

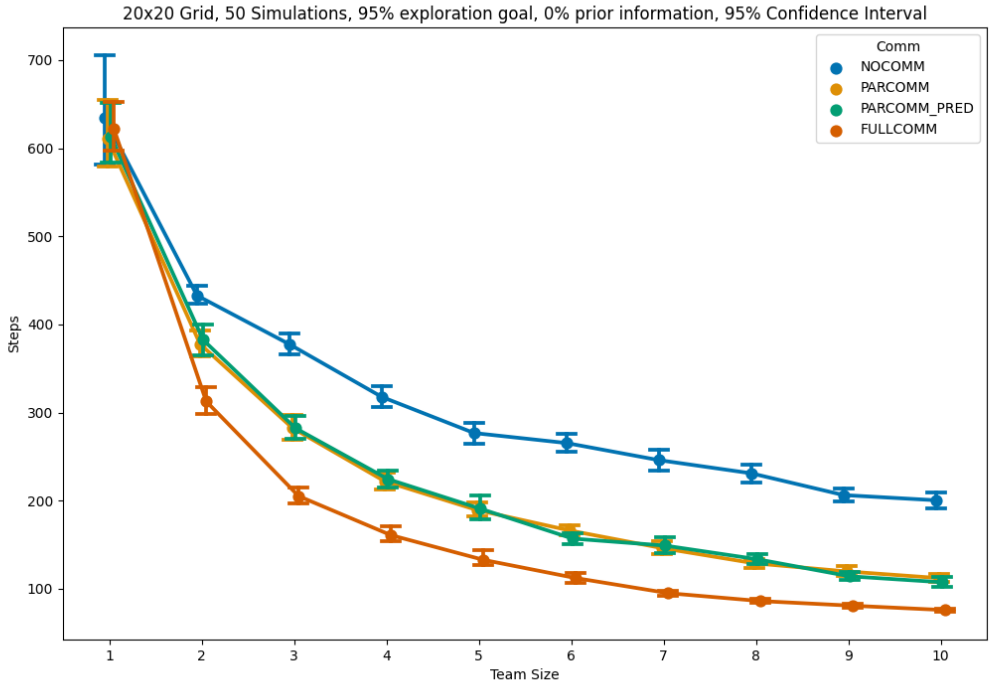
(b) 40x40 Data

Figure 5.2: Coverage steps versus Default Policy - 40x40

5.2. Exploration Performance v/s Team Size

As mentioned in the beginning of chapter 4, FULLCOMM was implemented first. This scenario is based on an existing work by Hyatt et al [21]. After dropping the communication dependency from FULLCOMM, this led to the NOCOMM scenario. Distance based communication was then added to the NOCOMM case leading to PARCOMM. As an extension of this, agents were given the ability to predict the paths of peers in the PARCOMM_PRED scenario. In this section, the ability of PARCOMM and PARCOMM_PRED in recovering the lost performance is studied. The study is performed on the number of exploration steps required for coverage in each scenario and for varying number of agents. Each exploration scenario is run on 20x20 and 40x40 maps with varying team sizes. The size of the teams are varied from 1 until 10 and the team has a coverage goal of 95%. The results are shown in figure 5.3 and figure 5.4.

In Figures 5.3a and 5.4a, the mean value of the steps for coverage along with the confidence interval are plotted for each team size. In tables 5.3b and 5.4b, quantitative data on the number of exploration steps is shown. Each table consists of 6 columns. Column 1 shows the number of agents, while column 2 and column 3 show the mean value of the number of coverage steps for FULLCOMM and NOCOMM cases, respectively. Column 4 and Column 5 show the recovery that is obtained from using PARCOMM or PARCOMM_PRED. The computation of this recovery can be understood by considering figure 5.5. The figure shows an example of the general trend between the steps of the NOCOMM, PARCOMM and FULLCOMM scenarios that can be seen in figures 5.3a and 5.4a. Each horizontal marker denotes the number of steps to explore an area. As expected FULLCOMM usually takes the least number of



(a) 20x20 Map

Team Size	FULLCOMM	NOCOMM	PARCOMM recovery	PARCOMM_PRED recovery
1	622	634	197	175
2	313	433	46	42
3	205	378	55	55
4	161	318	61	59
5	133	277	61	60
6	112	265	65	71
7	95	246	66	64
8	86	231	70	67
9	81	206	69	73
10	76	201	71	75

(b) 20x20 Map Data

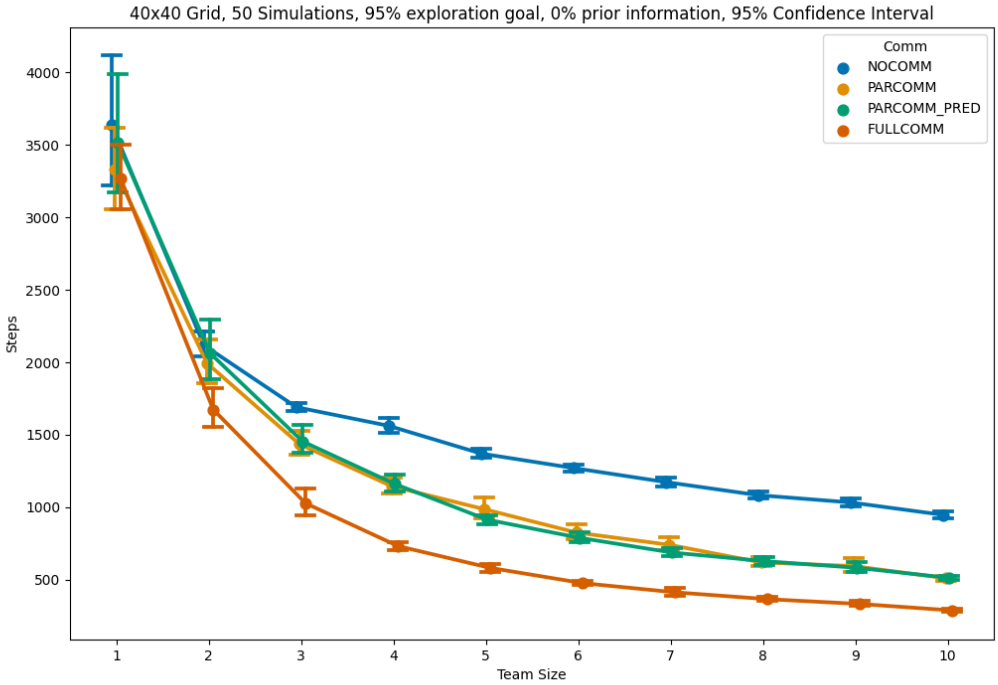
Figure 5.3: Coverage steps versus Team Size - 20x20

steps, while NOCOMM takes the highest. PARCOMM usually lies in between the two. The percentage recovery that PARCOMM provides can be measured by using equation 5.1, where a is the performance lost when moving from FULLCOMM to NOCOMM and b is the performance lost when moving from FULLCOMM to PARCOMM. Thus $a - b$ gives the performance recovered and when this is divided by a, it gives the percentage recovery. This is indicated in figure 5.5.

$$Recovery = \frac{a - b}{a} \tag{5.1}$$

A smaller value of b indicates that PARCOMM is close to FULLCOMM and when b is small the recovery provided is larger. Thus a larger value of recovery indicates higher amount of performance regained. The same equation can also used to compute the recovery for PARCOMM_PRED. On observing the values of the graphs, we draw some important observations:

1. From figures 5.3a and 5.4a the performance of PARCOMM lies in between that of NOCOMM and



(a) 40x40 Map

Team Size	FULLCOMM	NOCOMM	PARCOMM recovery	PARCOMM_PRED recovery
1	3265	3637	83	33
2	1674	2118	28	12
3	1028	1691	38	36
4	733	1562	50	48
5	582	1371	49	58
6	477	1271	56	61
7	414	1174	57	64
8	367	1084	65	63
9	334	1034	63	64
10	289	948	66	66

(b) 40x40 Map Data

Figure 5.4: Coverage steps versus Team - 40x40

FULLCOMM with the former being larger in number.

2. From figure 5.3a and 5.4a, the performance of PARCOMM and PARCOMM_PRED are almost similiar and lie between NOCOMM and FULLCOMM.

3. In each gridmap case, as the team size increases the performance of PARCOMM and PARCOMM_PRED move closer to the performance of FULLCOMM.

4. From tables 5.3b and 5.4b, it can be seen that the recovery provided by the team increases with the increase in the size of the team. For instance, in the case of PARCOMM in the 20x20 gridmap the recovery moves from 46% for 2 agents upto 71% in the case of 10 agents. In the case of PARCOMM_PRED, this moves from 42% upto 75%.

5. From table 5.3b, PARCOMM_PRED appears to give additional recovery over PARCOMM in many cases. However except for the case of a team with 6 agents, none of the differences were found

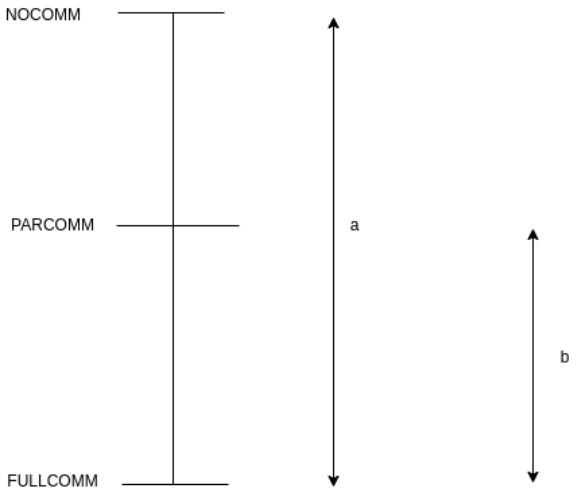


Figure 5.5: Recovery Computation

to be significant based on the significance test results of table 7.1.

6. From table 5.4b, in the case of a 40x40 graph the recovery provided by PARCOMM_PRED can be seen to be higher than the recovery provided by PARCOMM for 4 cases. However the significance test shown in table 7.2 indicate that the difference between PARCOMM_PRED and PARCOMM is not significant in any case.

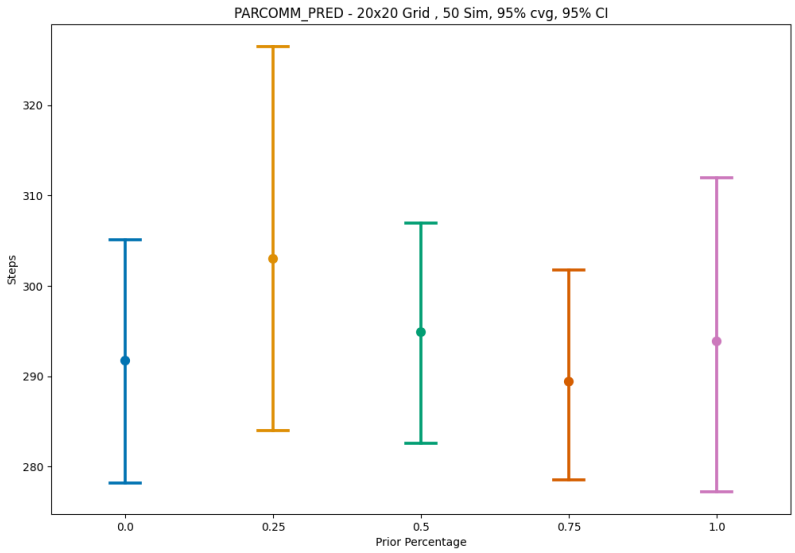
The first observation of NOCOMM having the largest number of steps is due to each agent being able to only sense its peers and not retrieve any information about the environment. Thus each agent explores regions already explored by its peers leading to increased redundant coverage and subsequently a larger exploration time. In the case of FULLCOMM, at each time step agents communicate with one another and hence agents can update their simulator at each moment of time. This leads to agents planning paths that are away from each other at each time, which reduces the amount of redundant coverage. This leads to faster coverage of the entire arena by the team and thus FULLCOMM gives the lowest number of coverage steps amongst the scenarios for all team sizes. In the case of PARCOMM, agents can only share information within a communication range. When agents are outside communication range, agents do not simulate the path of the other agents. As there is still some amount of information sharing, the performance moves away from the NOCOMM case in each of the team sizes. For higher team sizes the probability of agents meeting each other and sharing information increases, due to which the gap between PARCOMM and FULLCOMM decreases. This explains the 3rd observation. This also leads to the explanation of the 4th observation. A decrease in the number of steps leads to an increase in the recovery as per equation 5.1. When PARCOMM_PRED is considered, then in the case of a 20x20 grid the additional recovery provided is not significant in most cases, except a team with 6 agents. However there appears to be no trend followed when compared to any other team size. Thus, having a significant difference in only 1 case does not indicate any additional benefit of PARCOMM_PRED. In the case of 40x40 maps also, PARCOMM_PRED performs in a similar manner as PARCOMM. Thus, other scenarios where PARCOMM_PRED provides additional benefit requires to be studied.

If larger maps are considered, then the probability of agents meeting one another is low. This leads to lesser chances where agents can share information and correct the mispredictions made about its positions by peers. As mentioned in section 4.6.3 of chapter 4, in such an unknown environment where agents are still discovering the area, the chances of such mispredictions are quite high. If agents are able to correct one another's mispredictions, then this may lead to accurate predictions for more number of time and hence better performance of PARCOMM_PRED. Such corrections can happen through sharing of information. Given that the agent only has 1 grid-cell communication range, the probability of sharing information and correcting the mispredictions of the peer agents is low. However the predictions may improve if the communication range is increased. This is another study which has been conducted in section 5.4. Another way is to feed the agent with some idea about the contents of

the environment. This could possibly enable the prediction of the paths of peers agents while avoiding obstacles and keeping it close to the true position of the peer. The next section studies this idea further.

5.3. Exploration Performance v/s Prior Information

As seen in the previous section, in both 20x20 and 40x40 maps the gain by using PARCOMM_PRED was not significant. This was speculated to be because of the increased number of mispredictions of the positions of the peer agents. The proposal was to check if this could be solved on larger maps by increasing the communication range or by giving the agent additional information about the environment. In this section, the effect of prior environment information on the exploration performance is studied. This has been done by running the PARCOMM_PRED scenario with 3 agents on a map with a coverage goal of 95%. Before the exploration however each agent is given the same prior information of obstacle positions in the environment. This is given as the percentage of known obstacles in the environment to each agent. The percentages used in the study are 0%, 25%, 50% and 100%. The percentage corresponds to the percentage of obstacles whose position the agent knows before entering the arena. Thus 0% corresponds to the agent knowing nothing about the obstacle positions while 100% meant that the agent knows the location of all obstacles in the arena. The exploration has been performed on a 20x20 as well as a 40x40 map and the results are indicated in figure 5.6 and figure 5.7. Both figures also consist of subgraphs that show the mean coverage steps along with the confidence interval for each prior information case. From both figures it can be seen that irrespective



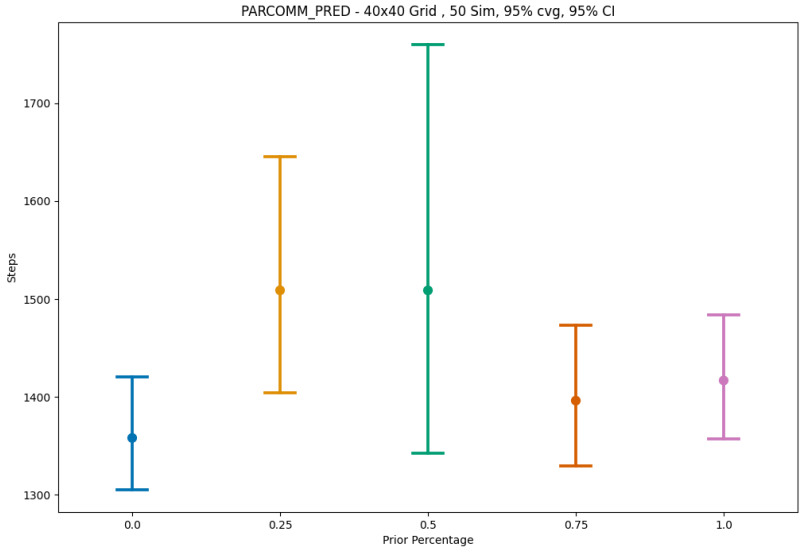
(a) 20x20 Map

Prior	Steps mean	Steps CI
0	292	14
0.25	303	24
0.5	295	13
0.75	289	13
1	294	19

(b) 20x20 Data

Figure 5.6: Coverage steps versus Prior Information - 20x20

of the gridmap size, having prior information does not effect the exploration performance. Figures 5.6a and 5.7a also show no trend of the results. In the case of the 20x20 map, from table 5.6b it can be seen that irrespective of the prior percentage the team takes roughly around 300 steps for coverage. While the confidence intervals for the 40x40 grid case of table 5.7b are higher, it can be seen that the



(a) 40x40 Map

Prior	Step mean	Step CI
0	1358	62
0.25	1509	125
0.5	1509	234
0.75	1396	77
1	1417	69

(b) 40x40 Data

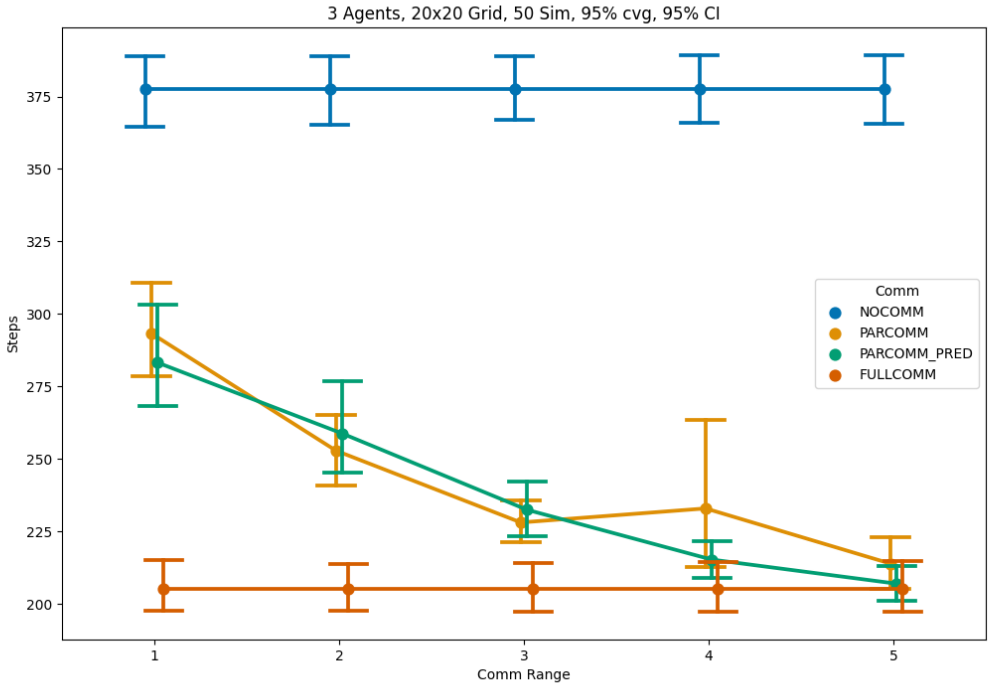
Figure 5.7: Coverage steps versus Prior Information - 40x40

step means roughly lie between 1300 until 1500. The same experiment of spawning 3 agents into an environment and varying the prior information was also performed in the PARCOMM scenario. The results are shown in figure 7.1 for a 20x20 gridmap and figure 7.2 for a 40x40 gridmap, and the same observation is seen that there is no effect of having prior information on the exploration performance. Thus, the presumption that giving agents prior information in order to improve the exploration does not hold well for the PARCOMM_PRED as well as PARCOMM scenarios.

Instead we move onto the other presumption, that making the agents share information with one another more often to correct mispredictions may help further increase the performance of PARCOMM and PARCOMM_PRED. The difference between sharing information and knowing about the parts of the environment is that in the former case the agents also get to know the coverage status of the environment in addition to the obstacle positions. An agent can update its own belief with the coverage maps of peers and can make plans in the updated environment. With more information about the environment, the amount of redundant coverage by the agent also decreases. To check the hypothesis we increase the chances of communication by increasing the communication range of each agent. The study has been further explained in the next section.

5.4. Exploration Performance v/s Communication Range

In this section, a team of a certain number of agents with the no information of the environment are spawned into an arena. As before the exploration goal is to perform 95% coverage. The communication range is varied to take one of the values in the set $\{1, 2, 3, 4, 5\}$. Each of the scenarios of NOCOMM, FULLCOMM, PARCOMM and PARCOMM_PRED have been simulated and the recovery provided by the PARCOMM and PARCOMM_PRED has been measured using equation 5.1. In the first experiment 3 agents were spawned into a 20x20 arena. The results obtained can be seen in figure 5.8



(a) 20x20 Map

Comm Range	FULLCOMM	NOCOMM	PARCOMM recovery	PARCOMM_PRED recovery
1	205	377	49	55
2	205	377	72	69
3	205	377	87	84
4	205	377	84	94
5	205	377	95	99

(b) 20x20 Data

Figure 5.8: Coverage steps versus Communication Range - 20x20, 3 agents

Figure 5.8a shows the trend of the steps as a function of the communication range. The corresponding table in figure 5.8b shows the value of the average number of steps for FULLCOMM and NOCOMM in column 2 and column 3. The recovery of performance by PARCOMM and PARCOMM_PRED are shown in column 4 and column 5 for the other 2 cases. From the figure we obtain the following observations:

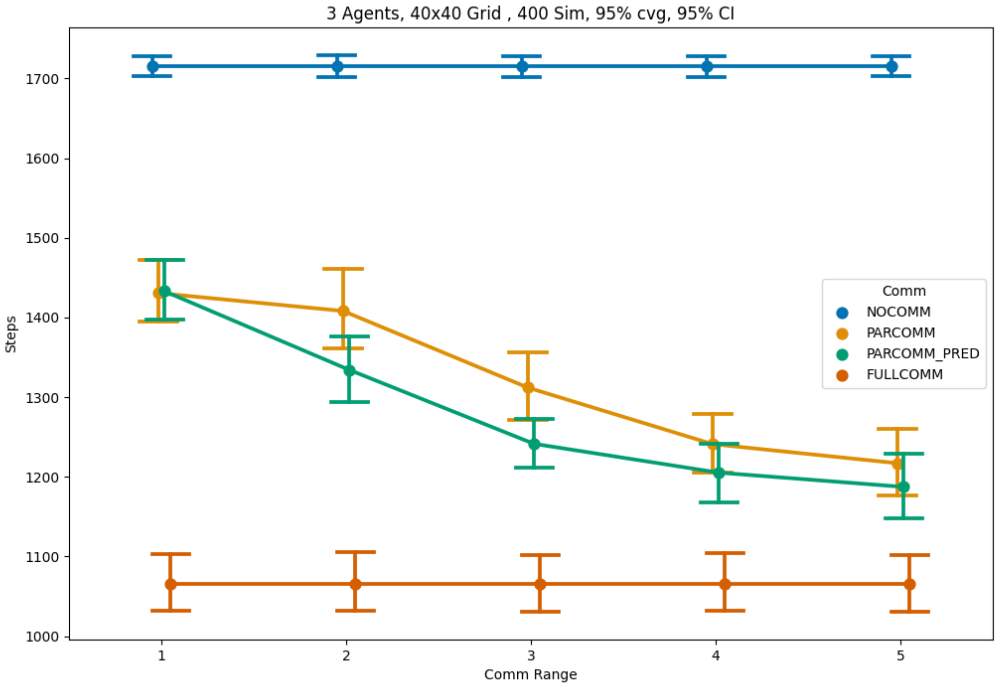
1. With an increase in communication range, both PARCOMM and PARCOMM_PRED approach the behaviour of FULLCOMM. At a communication range of 5, the recovery provided is as high as 99% for PARCOMM_PRED.

2. When running significance tests on the steps of PARCOMM_PRED and PARCOMM, it is found that there is no significant difference. The results can be seen in figure 7.3.

In a 20x20 arena when agents are given the capability to communicate upto a width of 5 grid-cells around it, then at every moment of time the agents can communicate with agent positions that are in upto (5+5+1) ×(5+5+1) = 121 number of gridcell positions. This is about 30% (121/400) of the entire environment. With 3 agents, the chances of information sharing is even higher. This explains the cause of PARCOMM_PRED and PARCOMM reaching above 95% of coverage recovery. Along with more information on the coverage status of the map, more number of communications also correct the mispredictions in PARCOMM_PRED. Further, whenever agents move out of the communication range of

one another then the predictions are made on a map that has more amount of certain information. This results in agents cooperating better, avoiding redundant coverage and achieving almost FULLCOMM behaviour. As seen from the significance tests, this is true for both PARCOMM & PARCOMM_PRED cases and there is no effect of using predictions, due to the increased number of communication moments.

The situation can however change in 40x40 gridmaps. In such a map, a 5 grid-cell communication range will imply that at every point of time the agent can communicate with only about 7.5% (121/1600) of the gridmap cells. Thus, it is possible that the percentage recovery may not be as high as what is observed in 20x20 even with PARCOMM_PRED. In order to test this, 3 agents were spawned into a 40x40 grid with a team coverage goal of 95%. The resulting graph are shown in figure 5.9a and the percentage recovery data is shown in table 5.9b. It is important to note that at first, 50 simulations



(a) 40x40 Grid, 3 agents - Graph

Comm_Range	FULLCOMM	NOCOMM	PARCOMM_Rec	PARCOMM_PRED_Rec
1	1066	1715	44	43
2	1066	1715	47	59
3	1066	1715	62	73
4	1066	1715	73	79
5	1066	1715	77	81

(b) 40x40 Grid, 3 agents - Data

Figure 5.9: Coverage steps versus Communication Range - 40x40, 3 agents

were performed. However due to the high value of variance, comparisons were difficult to perform and therefore additional data was added which raises the number of simulations to 400. We draw the following observations from the 2 figures:

1. In a 40x40 gridmap the performance of PARCOMM and PARCOMM_PRED approach the coverage steps of FULLCOMM with an increase in the communication range.
2. Significance tests from table 7.4 indicate that the difference between the values obtained in PAR-

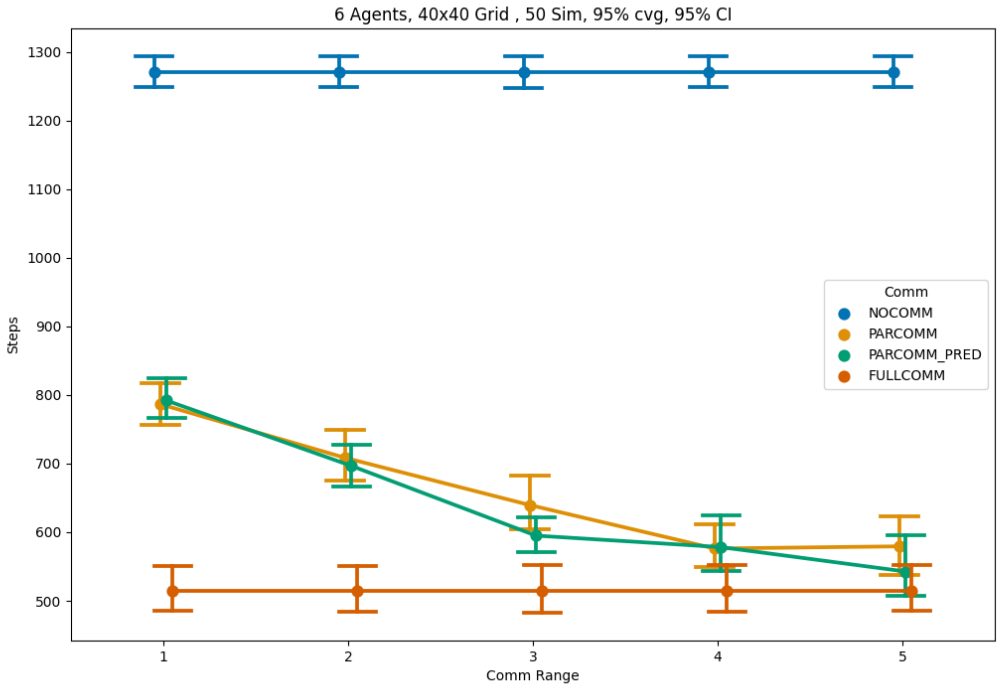
COMM and PARCOMM_PRED is significant for communication range values of 2 and 3. At communication range 2, the additional benefit provided by PARCOMM_PRED is about 12%, while it is 9% in the case of a communication range of 3.

The percentage recovery provided by 5 grid-cell communication reaches a maximum of 96% in the case of PARCOMM_PRED in a 40x40 gridmap as compared to the 99% recovery in case of a 20x20 gridmap. However in spite of this the general trend of the increase in communication range positively impacting the coverage steps remains. Increasing the communication range by 1 grid-cell causes PARCOMM_PRED to add upto a 12% benefit. This can be attributed to the higher chances of misprediction corrections, cooperation and minimal redundant coverage. As the communication range increases, this percentage of added benefit decreases. With increasing communication range, there are more chances of communication between agents and this leads to agent not being given enough chances to make predictions. Thus, with increased communication range PARCOMM and PARCOMM_PRED tend to become equivalent, and together approach the performance of FULLCOMM.

From the above results, it can be seen that the size of the arena causes a limit in the benefit that PARCOMM_PRED can provide. It can also be seen that there is potential to improve the PARCOMM performance further. This can be done by increasing the number of sharing instances between agents. Sharing of information as discussed in this section can come from increasing the communication range. However as seen from section 5.2, it can be seen that sharing is also influenced by the size of the team. In order to test the size of the team and its effect on the PARCOMM and PARCOMM_PRED for varying communication range, the same experiment has been conducted with 6 agents and 9 agents in a 40x40 map with a coverage goal of 95%. The results can be seen in figure 5.10 and figure 5.11.

From figures 5.10 and 5.11, it can be seen that there is a lot of overlap in the behaviour of PARCOMM and PARCOMM_PRED. Statistical tests that can be seen in table 7.5 and 7.6 indicate that the difference is not significant. This is due to the increased number of chances where agents can meet one another and exchange information. This leads to similiarity in the behaviour of PARCOMM and PARCOMM_PRED cases. What remains definite is that in the cases of 6 and 9 agents, PARCOMM and PARCOMM_PRED is able to reach the performance level of FULLCOMM.

Thus, in large gridmaps PARCOMM_PRED can provide additional benefit to PARCOMM for smaller team sizes and increased communication range. However the additional benefit is only upto a certain communication range, after which the performance of PARCOMM_PRED and PARCOMM are similiar. In both 20x20 and 40x40 gridmaps, with an increase in the communication range the behaviour of both PARCOMM and PARCOMM_PRED approach the behaviour of FULLCOMM. This indicates that having agents that communicate and share information is more beneficial for agents to maximise cooperative coverage, than giving prior information about the environment.

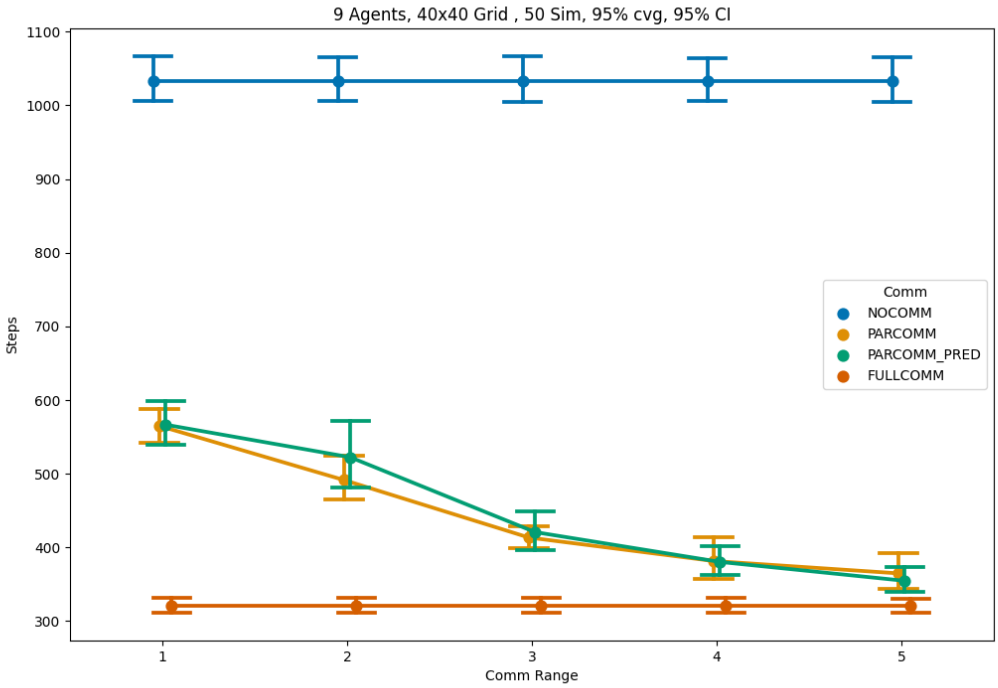


(a) 40x40 Grid, 6 agents - Graph

Comm Range	FULLCOMM	NOCOMM	PARCOMM Rec	PARCOMM_PRED Rec
1	514	1271	64	63
2	514	1271	74	76
3	514	1271	83	89
4	514	1271	92	92
5	514	1271	91	96

(b) 40x40 Grid, 6 agent - Data

Figure 5.10: Coverage steps versus Communication Range - 40x40, 6 agents



(a) 40x40 Grid, 9 agents - Graph

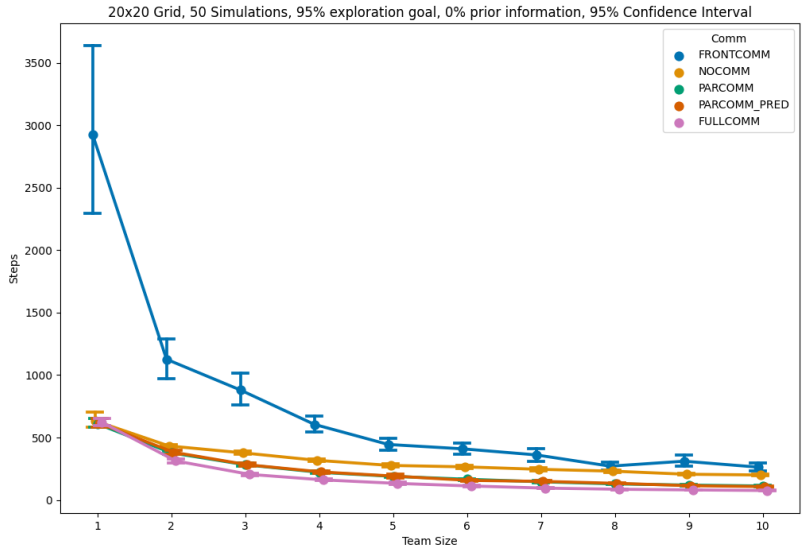
Comm_Range	FULLCOMM	NOCOMM	PARCOMM Rec	PARCOMM_PRED Rec
1	321	1034	66	66
2	321	1034	76	72
3	321	1034	87	86
4	321	1034	92	92
5	321	1034	94	95

(b) 40x40 Grid, 9 agent - Data

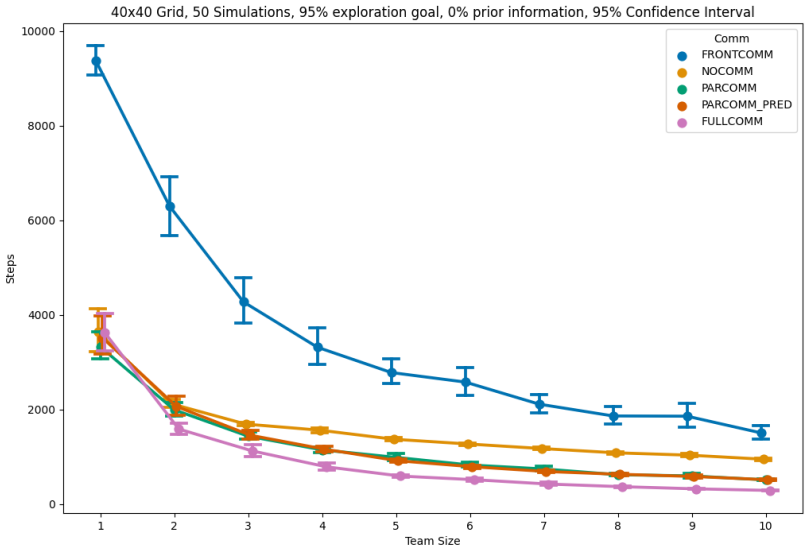
Figure 5.11: Coverage steps versus Communication Range - 40x40, 9 agents

5.5. Exploration Performance - FRONTCOMM Comparison

In this section each MCTS scenario is compared with the FRONTCOMM algorithm. The working of FRONTCOMM has been explained in section 4.8. For comparison, the number of steps required for each scenarios has been compared for 1 until 10 agents. In each case agents have been spawned into 20x20 and 40x40 arenas with a 95% team coverage goal. The results have been shown in figure 5.12.



(a) 20x20



(b) 40x40

Figure 5.12: Scenarios v/s FRONTCOMM

It can be observed from the figure, that the MCTS method performs area exploration much faster than the frontier communication method. This can be attributed to the look ahead planning capability of MCTS where it plans paths based on the reward obtained for a sequence of future actions. FRONT-

COMM has no such computation possible and only looks at the action for the current time step. This also means that FRONTCOMM would make it difficult for an agent to come out of cluttered zones of the environment. It is important to note that FRONTCOMM does have partial communication and can share map coverage information when within communication range. Due to this, the number of coverage steps decreases with an increase in the number of agents in the environment, However, the aspect of look ahead planning makes even NOCOMM perform faster than FRONTCOMM. From each sub-graph of figure 5.12, this type of behaviour is true for both 20x20 and 40x40 maps.

5.6. Discussion

In this section, the results from various experiments conducted have been summarised and discussed. The chapter began with the experiment to compare default policy 1 and default policy 2 (in section 5.1). It was found that using default policy 2 was giving faster exploration for large team sizes in the 20x20 map than default policy 1, which was the default policy implemented by Hyat et al in [21]. Once the MCTS algorithm working was verified and default policy 2 was selected, the MCTS scenarios were compared for their exploration steps as a function of the team size in section 5.2. In this case, the key observations made were that as per expectations the order of number of coverage steps in decreasing order was NO_COMM, PARCOMM & PARCOMM_PRED and then FULL_COMM. It was found that the recovery of performance provided in the 10 agent case by PARCOMM & PARCOMM_PRED was greater than 70% for 20x20 maps and 60% for 40x40 maps. It was also noticed that there was no significant benefit being provided by PARCOMM_PRED to PARCOMM in this experiment. Therefore, the exact scenario where predictions could provide additional benefit was to be tracked. The first speculation was that giving prior knowledge would help the predictions. However from the study in section 5.3, it was found that giving prior information did not have much effect. Instead, the next speculation of increasing the number of misprediction corrections by increasing the communication range was tested in section 5.4. It was found that with increase in communication range, the performance of both PARCOMM as well as PARCOMM_PRED comes closer to FULLCOMM and the performance recovery provided by each was greater than 90% for both 20x20 and 40x40 maps in the 5 grid-cell communication range case. Further, it was found that for 3 agents in a 40x40 gridmap, increasing the communication range results in PARCOMM_PRED providing additional benefit over PARCOMM by upto 12%. This difference was also found to be statistically significant. The additional benefit however decreased with an increase in the communication range. This study also found that in the case of 6 and 9 agents in a 40x40 gridmap, PARCOMM and PARCOMM_PRED performed in a similar manner for 1 until 5 number of communication ranges. The experiments ended with section 5.5 where MCTS based algorithms were found to perform faster than the frontier communication method.

Each finding in the above steps can be grouped into the following broad points:

1. **Look Ahead Planning Benefit** - From figures 5.12a and 5.12b that compare the MCTS and Frontier Communication algorithms, it can be inferred that the look ahead capability of MCTS is highly beneficial for exploration. This not only helps the agent move out of cluttered environments, but can help in good collaborative behaviour due to its simulation based approach.

2. **Significance of Default Policy** - Contrary to what is mentioned by Hyatt et al [21], it was found from figure 5.1a where the default policies have been compared, that the default policy does impact the coverage of results. Particularly the manner in which the current agent makes predictions of the movement of its peers makes a difference in the coverage performance. It was found that agents must first simulate the plans of peer agents in belief maps before executing its own actions in order to get better performance which also increases with the size of the agent team.

3. **Performance Recovery by Partial Communication** - NOCOMM is the scenario of removing the communication dependency from FULLCOMM, which is the original baseline work of Hyatt et al [21]. From figures 5.3a and 5.4a that compared the coverage steps with the team size, it was found that PARCOMM can recover the lost efficiency by moving from FULLCOMM to NOCOMM. The recovery provided by partial communication rises with the increase in the number of agents.

4. **Unprofitable prior map information on predictions** - From figures 5.6 and 5.7 that compared the coverage steps with the amount of prior information of the environment, it was found that

having prior information of the map contents did not help improve the prediction for 20x20 and 40x40. This was also found for the PARCOMM scenario from figures 7.1 and 7.2.

5. **Positive impact of increased communication range on partial communication** - From figures 5.8, 5.9, 5.10 and 5.11 that compared the coverage steps with the communication range, it was observed that increasing the communication range is helpful to reduce the coverage steps of PARCOMM and PARCOMM_PRED and bring it to approximately the same level as FULLCOMM.

6. **Positive impact of communication range on predictions for small number of agents on larger maps** - From table 5.9b that compares the data on the percentage recovery by PARCOMM and PARCOMM_PRED for 3 agents in a 40x40 map, it is observed that increasing the communication range upto a limit can help a team of smaller agents with predictions cover a large arena with lesser number of steps as compared to only using partial communication. Beyond the limit, PARCOMM performance also becomes better and agents in both the scenarios perform almost similar.

6

Conclusion

In this thesis, strategies that enable multi-robot exploration in limited communication environments have been studied. The first phase was to perform an in-depth literature study on existing methods. Monte Carlo Tree Search (MCTS) algorithm was selected from it due to its advantages of being a look ahead planner and its ability to simulate the actions of peer agents which could help with a cooperative strategy. An existing work that used MCTS by Hyatt et al [21] was implemented first. The communication dependency of this work was completely removed and strategies to recover the lost efficiency were studied. As seen from chapter 5 the strategy of providing robots with the ability communicate when within a communication range could regain the lost efficiency by a percentage as high as 75%. By increasing the communication range and number of robots in the team, this percentage could be increased upto 99%. It was also found larger maps, that increasing the communication range of the agents up-to a limit could enable peer prediction to bring an additional benefit to the exploration for a small teams size. Apart from these important findings, there were also 4 other key inferences made that are discussed in section 5.6 of chapter 5. In this section the research questions mentioned in chapter 2 have been answered following which the thesis is concluded by mentioning directions of further research.

6.1. Research Questions Revisited

RQ1 - What distributed strategies can cooperative agents utilise to explore unknown environments with limited communication, in minimal number of steps?
The strategy utilised by each agent was that of MCTS. From section 5.6 of the previous chapter it has been observed that this strategy helps a team of agents perform faster than the existing frontier communication method by effectively avoiding cluttered regions. In a limited communication environment, including partial communication helps decrease the exploration steps and the number of exploration steps decreases with the increase in the number of agents and communication range.

RQ2 - What is the effect of limited communication on the individual strategies?
Limited Communication, especially of useful information like map coverage and best plan information can increase the number of steps required by the team for area coverage. If the case of completely removing communication is considered then the No Communication takes the maximum number of steps. This can be seen in the graphs of chapter 5. Partial communication improves this behaviour and reduces the number of steps for exploration. The number of steps can be further reduced with an increase in the number of agents or communication range. Thus limited communication impacts the individual strategy of agents in a team in its ability to plan effective paths and leads to an increased number of coverage steps.

RQ3 - What strategies would help in replicating the performance of agents as in an arena of full communication?
As seen from the graphs in section 5, full communication behaviour can be replicated upto a significant extent by introducing partial communication. In addition to this, giving the agents the ability to predict

the behaviour of peer agents can help further reduce the steps of coverage in very specefic situations. It has also been found that increasing the number of agents in the team as well as the communication range can help replicate the behaviour full communication. This is because of the increased number of chances for agents to share important map coverage information and also correct the mispredictions of their paths made by peer agents, in the case of partial communication with prediction. Further, it has also been found that giving prior information of the arena in terms of the locations of obstacles does not have much of an effect on improving exploration, for the scenarios of partial communication and partial communication with prediction.

6.2. Recommendations for future work

The research conducted in this thesis focuses on creating distributed algorithms for robots that explore an unknown map with limited communication, with no information on the global positions of the peer agents. Some directions for further research while considering these constraints are:

1. **100% area coverage** - Currently the simulation terminates even when certain grid-cells are not explored. This has been done as the MCTS planner tends to be limited by the finite T_horizon and computation budget and not plan paths towards far off and missed out grid-cells. Increasing the computation budget can possibly help in this process. However, the amount of computation possible by the system must be taken into account.

2. **Distributed Termination** - Currently, the simulation terminates when an external entity perceives that 95% of the region has been covered. This however brings in a type of centralised termination into the system. Instead, if simulations can be terminated when each agent perceives that 95% of the region has been covered, then it enhances the distributed approach to the problem.

3. **Line of Sight Communication** - Currently agents share information based on the communication range. This however does not take into account if obstacles are present between agents. Thus, extending partial communication to a situation where agents can only communicate if there are no obstacles or are in line of sight communication is required to be explored.

4. **Including a Meeting Reward** - Currently the default policy computes a local and reward. However as seen in the thesis, the ability of peer agents meeting one another to share information is very beneficial. Thus, adding a weighted meeting reward can help with reduced value of exploration steps. A meeting reward can also be used by the agents to create more chances for misprediction corrections.

5. **Dynamic Obstacles** - Currently the maps and study have only considered static obstacles. However being a search and rescue operation, there is quite a high possibility of obstacles shifting around at various times that could change the belief of agents. While the MCTS approach is independent on the type of arena it is deployed it, studying the effect of dynamic obstacles on the scenarios is yet to be explored.

6. **3-Dimensional Case** - Currently the thesis focuses on a 2D exploration case. The same exploration is yet to be tried out in a 3D environment, where the region can be divided into a 3D gridworld with cubes and action space of MCTS would increase from 3 to 5 (additionally including TOP and BOTTOM actions).

7

Appendix

7.1. Appendix A - Significance Checks

Statistical Significance tests have been presented in this Appendix. This has been done for Partial Communication, abbreviated as PC and Partial Communication with Prediction, abbreviated as PCP. In statistical tests, the mean, standard deviation and number of data-points are taken for each of the populations under test. From these measures, a t-value is computed. The t-value computed in this case is the Welch's t-test [26]. This specific test has been used as it does not require the assumption of the 2 data populations having similar variance. From the t-value, a corresponding p-value is computed. If the p-value is less than a predefined value α, then the difference between is said to be significant. The value of α is usually set as 0.05, which indicates that there is 95% chance that the means of the populations are different. Thus, in the tables given below whenever p-value ≤ 0.05 then the populations are said to have a statistically significant difference.

Team	PC Mean	PC CI	PC Std	PCP Mean	PCP CI	PCP Std	t_val	p_val	Sig?
1	611	39	140	613	36	128	-0.1	0.92	No
2	378	15	54	382	18	63	-0.4	0.69	No
3	283	15	52	282	14	50	0.02	0.98	No
4	222	10	37	225	10	34	-0.36	0.72	No
5	189	8	30	191	15	52	-0.19	0.85	No
6	166	7	23	157	6	20	2.1	0.04	Yes
7	146	7	27	149	10	36	-0.45	0.66	No
8	129	5	18	134	5	19	-1.23	0.22	No
9	120	6	20	114	5	18	1.46	0.15	No
10	112	5	16	107	6	20	1.29	0.2	No

Table 7.1: Statistical Significance Check - Cvg v/s TeamSize 20x20

Team	PC Mean	PC CI	PC Std	PCP Mean	PCP CI	PCP Std	t_val	p_val	Sig?
1	3328	292	1044	3515	421	1503	-0.72	0.47	No
2	1994	156	556	2062	208	743	-0.53	0.6	No
3	1436	82	294	1455	104	372	-0.29	0.77	No
4	1145	59	211	1161	59	210	-0.36	0.72	No
5	987	76	272	915	33	119	1.73	0.09	No
6	826	50	180	789	33	120	1.19	0.24	No
7	742	49	174	688	28	100	1.89	0.06	No
8	620	31	112	629	27	95	-0.43	0.67	No
9	595	52	187	583	39	138	0.36	0.72	No
10	511	18	63	514	14	50	-0.25	0.81	No

Table 7.2: Statistical Significance Check Team - Cvg v/s TeamSize 40x40

Comm	PC Mean	PC CI	PC Std	PCP Mean	PCP CI	PCP Std	t_val	p_val	Sig?
1	293	16	59	283	19	67	0.8	0.43	No
2	253	12	44	259	16	57	-0.58	0.56	No
3	228	8	27	232	10	35	-0.7	0.49	No
4	233	26	94	215	6	23	1.29	0.2	No
5	214	10	35	207	6	23	1.18	0.24	No

Table 7.3: Statistical Significance Check Team - Cvg v/s Comm 20x20, 3 agents

Comm	PC Mean	PC CI	PC Std	PCP Mean	PCP CI	PCP Std	t_val	p_val	Sig?
1	1431	39	402	1433	37	381	-0.08	0.93	No
2	1408	52	532	1334	42	433	2.16	0.03	Yes
3	1312	44	446	1241	30	302	2.62	0.01	Yes
4	1241	38	392	1205	37	376	1.32	0.19	No
5	1217	41	415	1187	41	418	1.01	0.31	No

Table 7.4: Statistical Significance Check Comm - Cvg v/s Comm 40x40, 3 agents

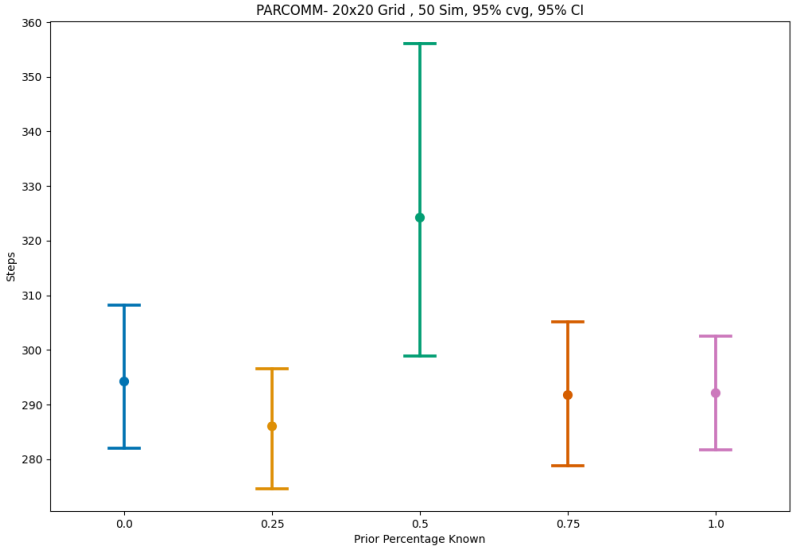
Comm	PC Mean	PC CI	PC Std	PCP Mean	PCP CI	PCP Std	t_val	p_val	Sig?
1	787	32	114	792	30	106	-0.24	0.81	No
2	709	38	137	697	33	117	0.46	0.65	No
3	640	40	143	595	26	93	1.85	0.07	No
4	576	31	109	578	43	152	-0.08	0.94	No
5	579	46	165	543	45	161	1.12	0.26	No

Table 7.5: Statistical Significance Check Comm - Cvg v/s Comm 40x40, 6 agents

Comm	PC Mean	PC CI	PC Std	PCP Mean	PCP CI	PCP Std	t_val	p_val	Sig?
1	565	24	87	567	30	108	-0.09	0.93	No
2	491	31	111	522	45	162	-1.11	0.27	No
3	413	15	54	421	26	93	-0.49	0.63	No
4	382	31	112	380	21	73	0.06	0.95	No
5	365	26	93	355	18	66	0.62	0.53	No

Table 7.6: Statistical Significance Check Comm - Cvg v/s Comm 20x20, 9 agents

7.2. Appendix B - Experiment Additional Graphs

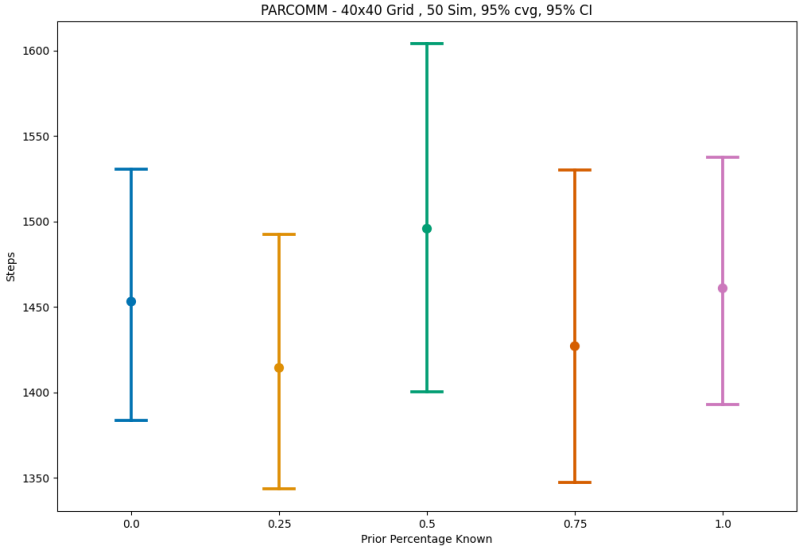


(a) 20x20 Map

Prior	Steps mean	Steps CI
0	294	13
0.25	286	11
0.5	324	28
0.75	292	14
1	292	11

(b) 20x20 Data

Figure 7.1: Coverage steps vs Prior Information (PARCOMM) - 20x20



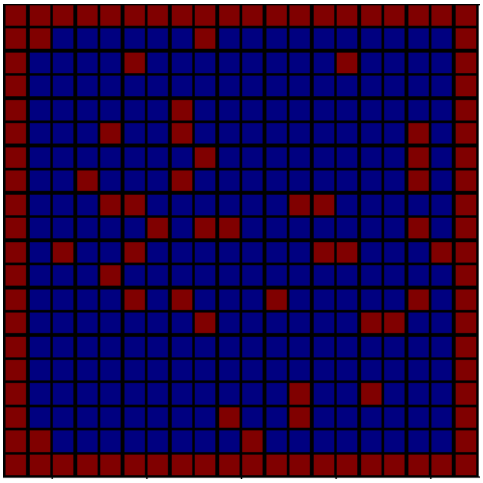
(a) 40x40 Map

Prior	Step mean	Step CI
0	1453	76
0.25	1414	81
0.5	1496	115
0.75	1427	91
1	1461	74

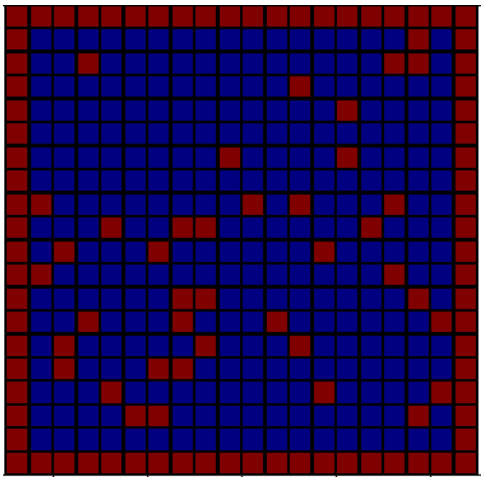
(b) 40x40 Data

Figure 7.2: Coverage steps versus Prior Information (PARCOMM) - 40x40

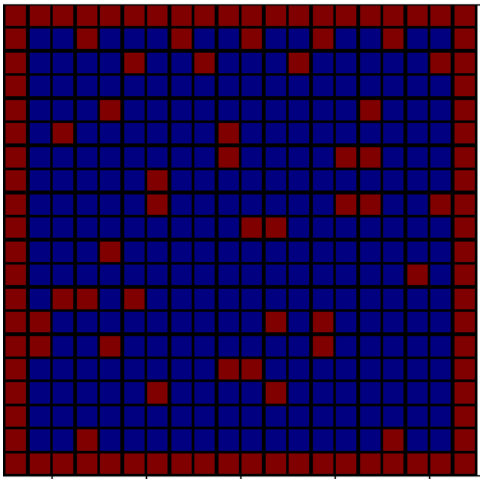
7.3. Appendix C - Grid-Map Collection



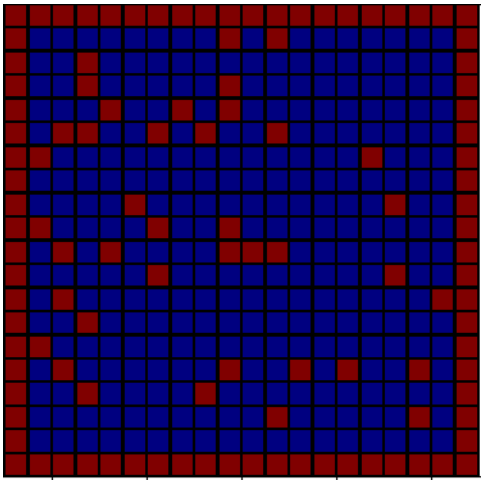
(a) Map 1



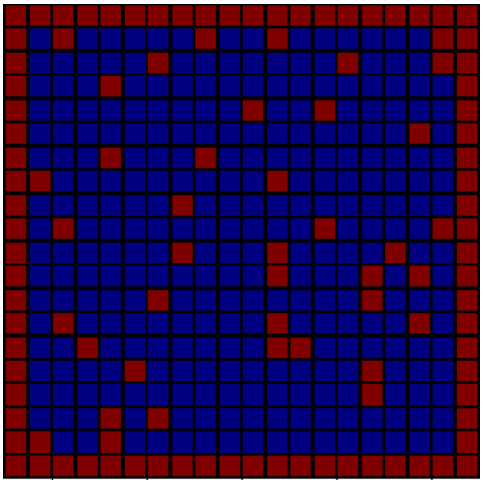
(b) Map 2



(c) Map 3

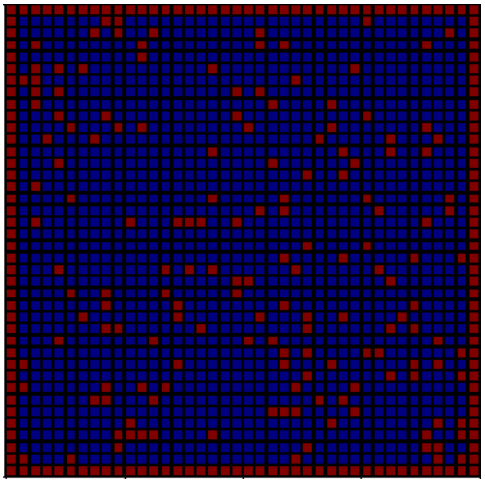


(d) Map 4

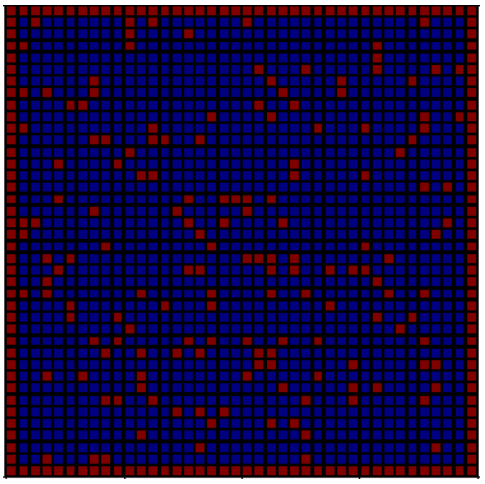


(e) Map 5

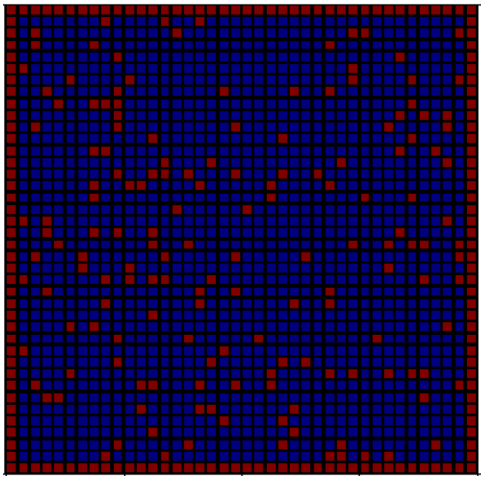
Figure 7.3: 20x20 Maps



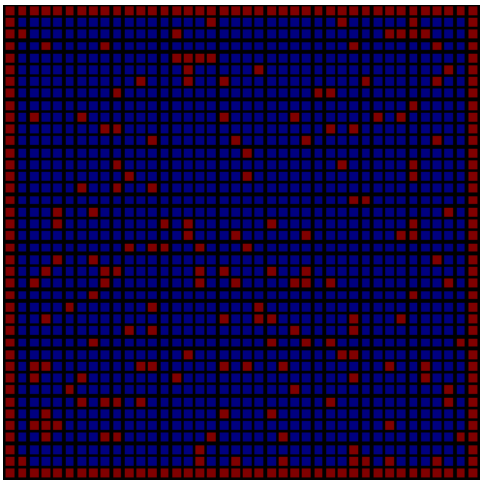
(a) Map 1



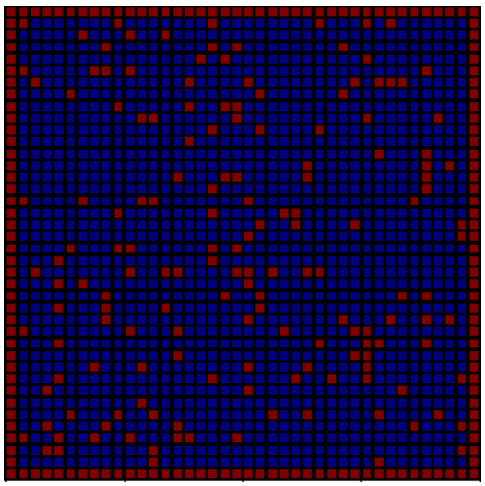
(b) Map 2



(c) Map 3



(d) Map 4



(e) Map 5

Figure 7.4: 40x40 Maps

8

Scientific Article

Multi Robot Exploration and Planning in Limited Communication Environments

Abstract:- Distributed cooperative robots can be highly beneficial in mapping disaster environments and assisting with search and rescue operations. In most situations such environments only allow for only limited communication between robots. This paper reports on simulation experiments conducted to test the impact of having only partial communication capabilities between cooperative agents on area exploration strategies. The Monte Carlo Tree Search (MCTS) planning algorithm has been utilised by multi-robot teams to cooperate and explore an environment effectively. On top of this base case, other communication scenarios are applied: No communication at all, and near-neighbor communication at various ranges. In addition to these communication strategies, robots are also given the ability to predict the paths of peers. From extensive simulation tests, it is shown that partial communication can recover a significant amount of performance in a limited communication environment. Giving agents a peer prediction ability is not shown to have a significant positive efficiency effect, except in very specific situations. It is also shown that providing prior information of the environment obstacle locations to agents is not useful. Instead, increasing the number of chances of agents sharing information positively effects the exploration performance.

Keywords *Artificial Intelligence, Multi-Agent Systems, Robotic Exploration, Planning*

I. Introduction

Multi robot exploration is an area coverage strategy where a team of robots work together to explore and discover the contents of an environment. This is particularly useful in applications like region surveillance, space exploration and disaster management missions. In the case of disaster management or search and rescue missions, a team of robots usually encounter a situation where the area to be covered is completely unknown. Further, being a disaster zone there is high possibility that the area has restricted communication. An environment with restricted communication makes it difficult for robots to share information with one another. When robots cannot share information then this hampers the cooperation between robots to achieve the overall exploration goal. This leads to an increase in the overall exploration time, with the same sub-region of the area being explored multiple times by many robots. This is referred to as redundant coverage. Figure 1 gives an example of 4 robots spawned in a grid-cell arena with obstacle grid-cells indicated in red, unexplored ones in blue and already covered regions in green. To avoid redundant coverage, the 4 robots would need to share the regions they have already covered. Redundant coverage is particularly problematic in search and rescue operations, as an increase in the time for complete exploration would delay any further rescue work. A restricted communication environment also leads to robots having no information about the global state of the environment and robots can only observe the contents of the sub-regions of the environment that are within its sensing range (local observibility) . There is also no possibility to have reliance on any centralised unit of the environment and thus distributed approaches of exploration are required. Thus, when deployed into a disaster environment, robots of a team are required to be distributed and deal with local observibility. The environment itself has restricted communication and the location of obstacles is unknown to robots.

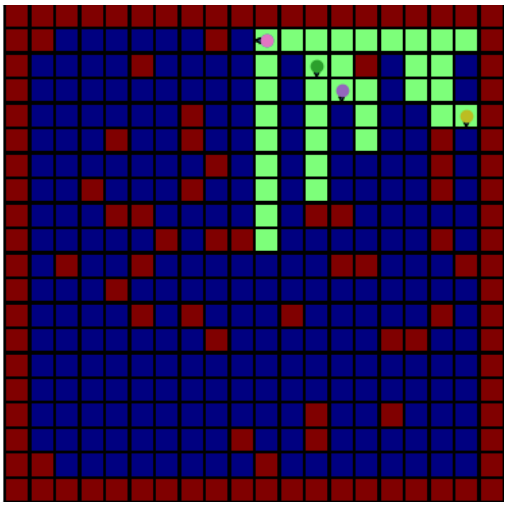


Fig. 1: Multi-Robot Exploration using 4 robots

From existing literature on the topic, most previous distributed cooperative multi-robot exploration methods considered only one amongst the two environment constraints of an unknown region and limited communication. One algorithm that incorporated both these constraints is the work on minPos algorithm by Bautin et al [1] that was further adapted by Benavides et al [2] to perform Frontier based Exploration. Apart from this another approach of the Monte Carlo Tree Search (MCTS) algorithm was used by Hyatt et al [3] specifically for multi-robot exploration. The work however assumed an environment having full communication. MCTS however has the ability to plan paths by looking ahead into the future for a predefined number of steps and the simulation based approach also makes it possible for an agent to simulate the paths of peer agents. Due to this, MCTS was the algorithm selected as

the baseline for this study.

The approach taken was to first implement the existing work of Hyatt et al [3] as a baseline. Following this, a scenario was created where the communication dependency was completely removed from the replicated work. This led to a situation called No Communication. In order to bring in some form of cooperation between robots, while taking into account the environment physical limitations, robots were given the capability to share information with peers within a predefined communication range. This scenario is referred to as partial communication. With the aim of moving the performance of robots even closer to the full communication case, robots were then given the ability to predict the plans of peer agents that were beyond the communication range. This type of prediction is based on the approach that was used in the work of Claes et al [4]. By following this approach, this paper aims to focus on designing a distributed cooperative planning strategy for multi-robot exploration systems that can be deployed in disaster management situations, by considering the lack of prior environment information and restricted communication that exists in such time crucial missions. From this research focus, 3 research questions are to be answered:

1) **RQ1** - What distributed strategies can cooperative agents utilise to explore unknown environments with limited communication, in minimal number of steps?
2) **RQ2** - What is the effect of limited communication on the cooperative exploration strategies?
3) **RQ3** - What strategies could help in replicating the performance of agents as in an arena of full communication?

The contributions made by the work are to extend the study on Coverage Path Planning by Hyatt et al [3] to an environment of limited communication, to extend the heuristics dealt with in SPATAPS [4] towards applications where the global state of the environment is unknown and to compare the performance of look-ahead path planning methods with one step methods like Bautin et al [1] & Benavides et al [2].

The paper begins by explaining the problem and the direction of research in this section. The next section II on related work explains the relevant work that has been used previously to perform multi-robot surveillance and exploration. Section III defines the specific problem being tackled and the important assumptions made in the approach taken in this paper. Section IV on theoretical framework goes into explaining important ideas behind MCTS that will be needed to understand the approach taken. Section V explains the approach taken by the thesis. Following this section VI presents the experimental results that are used in section VII to answer the research questions mentioned previously. Finally, section VIII draws key conclusions and also provides direction for future work.

II. Related Work

One of the earliest techniques taken to approach multi-robot exploration is to have robots perform exploration individually. The robots are then deployed into an unknown environment where they spread out. In the method presented by Howard et al in [5], this was done by considering an artificial potential field algorithm where each robot exerts a virtual force that causes robots to move away from each other. Steven et al in [6] run two algorithms namely the Clique-Intensity Algorithm and the backbone dispersion algorithm for exploration, where robots give high priority to maintaining communication with one another. In some other methods, robots focus on varying their length of movement based on the neighbourhood density. This was the approach taken by Bao Pang et al in [7] where each robot takes smaller steps when the neighborhood density is high, else it takes larger step size.

Some approaches follow the strategy of taking into account the contents of the environment and direct robots to move towards points that are beneficial to the overall exploration. A popular idea that has been used by Umari et al [8], Nair et al in [9], Dadvar et al [10], Antoine et al [1] and Benavides et al in [2] is for each robot to identify frontier points in the environment. Frontier points are those parts of the environment that lie on the boundary of known and unknown regions. Figure 2 shows an example of frontier points that can be seen in yellow. Once robots

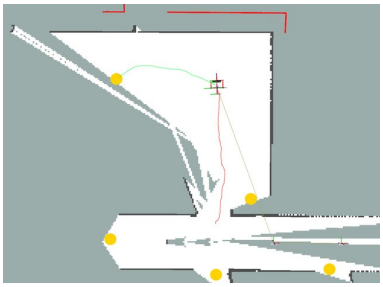


Fig. 2: A robot detecting frontier points [8]

identify frontier locations, a task location algorithm is run such that robots get assigned a frontier point. This task location algorithm is usually based on the distance of the robots to the task. Benavides et al in [2] also considers how well connected a robot is at a frontier point with peer robots.

Methods like Lopez et el [11] and Upma et al [12] follow the strategy of dividing the arena into subareas that each robot is assigned to explore. This technique is referred to as scene partitioning and aims at reducing redundant coverage by multiple robots.

As multi-robot exploration involves multiple agents cooperating with one another to achieve a common goal, certain works have used ideas from the field of game theory in exploration. One of the earliest techniques involving game theory is the work by Meng et al [13]. In this work, agents share map information of sub regions of the exploration area and allocate sub-regions based on the computation of the Nash Equilibrium for a predefined utility measure. In another method by Ni et al [14], the authors extend the same idea and use methods that achieve Nash equilibrium faster. The authors also make sure that sub-region allocations that provided no benefit (zero utility) are avoided.

In some methods, authors have used sequential decision planners in which the utility or benefit of a particular action is based on the reward of subsequent future actions that an agent performs. Hyatt et al [3] used Monte Carlo Tree Search (MCTS) for the cooperation of multiple robots

exploring an unknown environment. Another method by Claes et al [4] uses MCTS for a warehouse commissioning task where robots gather and deliver items in an efficient manner. The method also introduces the idea of a single robot predicting the behaviour of peer robots. In another work by Best et al [15], a limited communication setting was considered for the multi-robot scenario of generalised team orienteering and active object recognition. The algorithm also uses the MCTS algorithm and agents share highly compressed data with one another in a probabilistic manner. Minglong Li et al in [16] combine the works of Claes et al [4] and Best et al [15], to create a system where agents predict the actions of peers in a computationally cheap manner, reduce the communication pressure when sharing information and get better coordination.

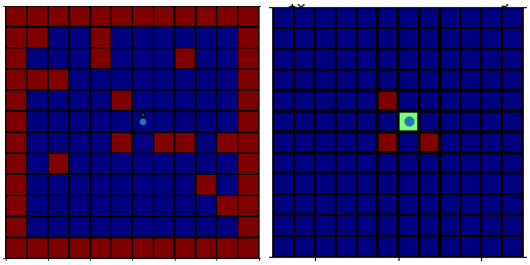
Amongst the various techniques presented in section II, the method on Coverage Path Planning by Hyatt et al [3] using MCTS could deal with completely unknown maps, work in environments of limited communication, possess only local observibility and support be used in a distributed manner. The method was also used by the authors to tackle a search and rescue exploration scenario. Further the ability of an agent to select an action by looking ahead with a certain number of steps could help agents avoid moving towards cluttered regions where it could get stuck. Robots could also share information with one another. Being a simulation based method, each agent had the ability to consider the plans shared by the other robots in the computation of its own plans, which brought in the aspect of cooperation. The only drawback however was that the method assumed unrestricted communication. However due to the other mentioned advantages, this algorithm was used as the baseline for this research.

III. Problem Statement

The specific application considered in this paper is an exploration scenario consisting of a fleet of aerial robots navigating and mapping an unknown environment [17]. The unknown environment is cluttered, radio-hampered, GPS-denied and thus facilitates limited communication between peer agents. Each agent of the team can only observe nearby surroundings. This is referred to as local observibility. Local observibility can be understood by considering figure 3 where the agent's position in the environment can be seen in sub-figure 3a, while what the agent perceives of the environment can be seen in sub=figure 3b. The colours $Green, Blue, Red$ correspond to the states of $\{Covered, Unexplored, Obstacle\}$ respectively. Keeping these constraints in mind, the following assumptions are made by each agent:

1) Each agent only knows the size of the region of exploration and the origin as a frame of reference.
2) Each agent divides the region of exploration into grid-cells that can take one of the values of $\{Covered, Unexplored, Obstacle\}$.
3) Each agent knows the total number of agents that are present in the exploration team.
4) An agent considers a grid-cell to be covered when it is directly over it.

Each agent has the option to execute one of the actions $\{UP, RIGHT, LEFT\}$ and can take up positions in the grid-world which is defined by the tuple (x, y, yaw). The position (x, y) refer to the x and y coordinates in the grid-world, while yaw refers to the heading angle of the agent with respect to the horizontal axis. The heading angle indicates the orientation of the agent and can take the values $\{0, 90, 180, 270\}$ corresponding to pointing right, up, left and down respectively. From the action list, the action UP moves an agent 1 grid-cell ahead based on the yaw value. $RIGHT$ and $LEFT$ actions turn the agent Right and Left only, and does not actually shift the agent by any grid-cell.



(a) Agent position in Environment (b) Agent's Environment Belief

Fig. 3: Map Discovery by a Single Agent

IV. Theoretical Framework

A Markov Decision Process (MDP) is a mathematical framework that is used to model the decision of an agent in a fully observable, stochastic environment with a Markovian transition model and additive rewards. It is mathematically defined by a set of states (with initial value s_0), a set of possible actions, a transition model $P(s^{'}|s, a)$ and a reward function $R(s)$ [18]. In an MDP, the sequence of actions can be grouped together into a list. This list of actions is usually referred to as a policy (π). Amongst the possible sequences of actions or policies, there exists an optimal policy (denoted by π^*) which directs the agent to a desired behaviour/goal [19]. The goal can be to maximise a total reward, minimise a cost or reach a particular state. If we consider that the agent interacts with the environment for H number of time-steps, then an example of a policy π is the list of actions $\pi = [a_1, a_2...a_H]$. The search over the entire space of policies to obtain the optimal policy is called **Planning**.

Monte Carlo Tree Search (MCTS) is a planning algorithm that is used to find an approximate value of the optimal policy π^*. The main intuition behind MCTS is that by using Monte Carlo simulations to quickly sample thousands of possible trajectories, we can achieve good approximations of the best value among possible actions from the root node (the node from which the search started) [4]. The tree nodes are the environment states and the branches are the actions that can be taken. There are 4 major steps that are repeated in every search iteration, which have been described below [20]:

1) Selection - This step starts at the root node and keeps selecting nodes until an expandable node is reached. An expandable node is a non terminal state that still has some actions that are not yet executed.

2) Expansion - Based on the available actions, more child nodes are added to the tree in this step.
3) Simulation - In this phase, a rollout of usually random actions is run for a predefined number of steps following which a reward value is computed.
4) Back Propagation - The simulation result is backed up through the tree to update the result of the path upto the root-node.

There are 2 important rules/policies as mentioned above that must be followed. These are:

1) Tree Policy: Used in the selection or expansion step to select or create an expandable node from the nodes already present in the search tree.
2) Default Policy: Used in the simulation step to play out actions for a predefined number of times or until a non-terminal state is reached.

$$\text{TreePolicy} = \bar{X}_j + 2C_p\sqrt{\frac{2\ln n}{n_j}} \tag{1}$$

The Tree policy is given by equation 1. In this equation, $\bar{X}_j$ refers to the average reward obtained by taking the action j, n_j refers to the number of times that action j / child j has been selected and n refers to the number of times the current parent node has been visited. From the equation it can be seen that for a particular action j, n_j will be equal to 0 initially which leads to a UCT value of $\inf$. This leads to every child being visited at least once, which is essential given the random nature of play-outs. The default policy used in the above tree is usually a sequence of random actions.

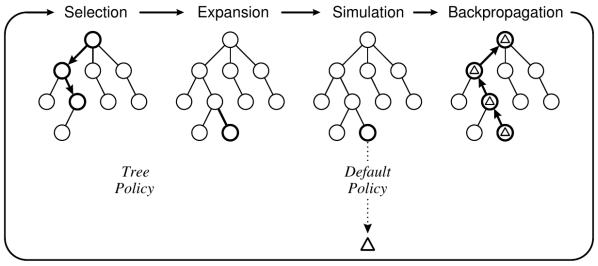


Fig. 4: MCTS Basic [20]

One iteration of the MCTS process has been shown in figure 4. The first stage involves the tree policy step where starting at the root node, child nodes are recursively selected until a node is reached that is either a terminal state or is not fully expanded. In the expansion stage, at such a node an action that has not been executed yet is selected from the action list. This action is executed, which leads to a new child node that is added to the tree. Following this the simulation step starts where simulations are run from the newly created node according to the default policy. After a predefined number of times, an objective function is evaluated which produces a reward value Δ. The reward is then propagated up the sequence of nodes in order to update each node's visit count and reward value. These steps continue to occur for many iterations, until the computation budget is met or the search is interrupted. Following this, the best action corresponding to the root node is selected and executed. Some ways of selecting the best possible action are:

1) Max child: Selects the child with highest reward
2) Robust child: Selects the child that is most visited
3) Max-Robust child: Selects the child with highest visit count and highest reward

V. Approach

As mentioned in section I there were 4 different scenarios created in this study based on the level of communication between agents in the environment. The first scenario implemented is referred to as the Full Communication scenario (subsection V-A). This scenario is the baseline method and is the method on coverage path planning by Hyatt et al [3] in an environment where unrestricted communication exists between agents. Once tested to be working the communication dependency of the algorithm was dropped in order to have no communication between agents (subsection V-B). In this situation agents could not share any information with one another, but could only sense nearby agents for the purpose of collision avoidance. In order to relax the constraint of communication while keeping physical limitations in mind, agents were then allowed to communicate important information only when within a predefined communication range from one another. This is referred to as partial communication (subsection V-C). In order to bring the behaviour of agents having partial communication closer to the full communication case, agents were then given the capability to predict the strategies of peers. Agents performed this by using the most recently shared information from peers as well as through the use of computationally cheap heuristics (subsection V-D).

A. Full Communication

The approach that each agent uses in this scenario can be divided into 4 stages, which are Environment Sensing, Data Sharing, MCTS Planning and Action Execution. Environment Sensing is used by the agent to obtain the status of nearby grid-cells. This is done using the technique that was explained earlier in figure 3. During the data Sharing stage each agent obtains the map belief of all peer agents, the position information and the best plans made by the peers in the previous time-step. The agent updates it own map belief of the environment using this information from peer agents. Further, the agent simulates the plans made by the peers in the MCTS planning stage. This helps bring in the aspect cooperation in the plans made by the agent.

An MCTS planner is used in planning the paths of agents. The planning begins from the root node. At the root node, the agent utilises the environment state, its own position and the position & plans of the known peer agents. Using this information the agent needs to choose the most promising action from its action set $\{UP, RIGHT, LEFT\}$. Equation 1 is the tree policy used and is referred to as the Upper confidence bound for trees (UCT1). The Default Policy selects actions that directs agent planning towards $Unexplored$ grid-cells. There are 3 cases that arise here:

1) If grid-cell in front is $Unexplored$, move UP.

2) If grid-cell in front is not $Unexplored$, check $RIGHT$ and $LEFT$ grid-cells. Move to the grid-cell which is $Unexplored$.
3) If none of the surrounding cells are $Unexplored$, move in the direction of one of the grid-cells that is $Covered$.

As mentioned in the previous section after the completion of the simulation/rollout stage, a reward value Δ is computed. The Δ computed in this case is referred to as the $TotalReward$ and is given,

$$TotalReward = w_{LR} LocalReward + w_{GR} GlobalReward \tag{2}$$

w_{GR} and w_{LR} are the weights given to the global and local rewards respectively. In the above equation $GlobalReward$ is given by the relation:

$$GlobalReward = -\frac{N(UnexploredCells)}{N(TotalCells)} \tag{3}$$

In the above equation $N(UnexploredCells)$ refers to the number of unexplored grid-cells in the gridmap and $N(TotalCells)$ refers to the total number of grid-cells present in the gridmap. The $LocalReward$ is given by equation 4,

$$LocalReward = \sum_{k=1}^{T} \left[\frac{1}{(t_k+1)^2} (C_{cov} - C_{hit}) \right] \tag{4}$$

In the above equation $C_{cov} = positivenumber$ if the robot lands up on a newly discovered grid cell at time step t_k and "covers" it, else $C_{cov} = 0$. $C_{hit} = positivenumber$ if the robot encounters an obstacle or another robot at time step t_k, else it remains 0. The normalisation using the $(t_k + 1)^2$ term is used to decay the reward overtime. The decay overtime is done to give a higher weight to rewards obtained at a nearer time-step t_k. This has been done as the state of the environment is still being discovered and is less certain further into the future. The weighted value at each time-step t_k is computed upto a predefined time horizon T.

In the simulation/rollout stage of MCTS planning, the agent first simulates the plans of all peer agents. Once computed, the agent begins to make its own plans by following the default policy using the 3 cases mentioned previously. During this time, the agent also computes the $LocalReward$. After time horizon T, the agent computes the $GlobalReward$ using equation 3. The relation of the $GlobalReward$ is such that, a higher score is given to plans that result in lower number of $Unexplored$ grid-cells. The aim of doing this is to include a reward for team behaviour and bias the agent to choose actions that do not lead it to redundant coverage. After the computation of the $TotalReward$ or Δ, the value computed obtained is back-propagated until the root of the tree.

Once the MCTS planning stage is complete, the most suitable action that an agent needs to execute is obtained and the agent moves as per this action in the action execution stage.

B. No Communication

In this case of No Communication, each agent follows 3 stages which are Environment Sensing, MCTS Planning and Action Execution. In this scenario, agents have no capability to communicate any information to peer agents. While no robot can communicate with one another, agents can still sense the presence of peers that are within the sensing range. This is done for the purpose of collision avoidance. When planning the next action, the MCTS planner of the agent only simulates its own actions in the simulation step. The $TotalReward$ that each agent computes is obtained using equation 2. Once the agent computes a suitable action, it executes it in the action execution stage.

C. Partial Communication

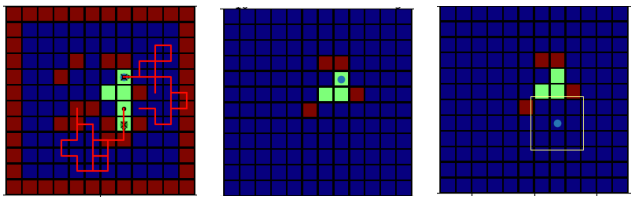
In the partial communication stage, agents that are present within a predefined communication range are said to be neighbours. The 4 high level stages followed in order are Environment Sensing, Neighbour Data Sharing, MCTS Planning and Action Execution. Environment Sensing and Action Execution are similar to the previous 2 scenarios. In the Neighbour data sharing stage however, an agent only obtains information from neighbouring agents. The data shared is the same as the full communication scenario. An agent uses this shared data to update its own map of the environment. During the simulation/rollout stage, an agent only simulates the path of known peer agents and then computes the reward using equation 2. Once a suitable action is obtained from the planning stage, the agent executes it in the action execution stage.

D. Partial Communication with prediction

There are 5 stages present in this scenario. They are in the order of Environment Sensing, Neighbour Data Sharing, Non-Neighbour Prediction, MCTS Planning and Action Execution. This scenario builds over the previous scenario by predicting the paths of agents that are not within communication range (Non-Neighbours). As each agent knows the total number of team members, obtaining the Non-Neighbours from the known Neighbours is possible. To make predictions, an agent uses the most recent plans that it obtains from Non-Neighbours. The plans obtained is a list of actions that a peer agent executes. This list is finite in number. When this list gets exhausted, agents predict the path of the peer agents using a heuristic computation. The result of the heuristic prediction is an action that the current agent believes the peer non-neighbour agent will execute.

The heuristic computed is called the coverage-by-step heuristic. In this, the current agent predicts the next action of the peer agent by computing the reward for the 8 grid-cells around the predicted/last seen position of the peer agent. This can be understood by considering figure 5a. In the scenario shown, there are 2 agents (agent0 and agent1) which are present outside communication range. Sub-figure 5b shows agent0's position and belief of the environment at the current time step while sub-figure 5c shows the estimated position of its peer agent, agent 1 at the current time step. It is important to note that this sub-figure shows the position of agent 1, as predicted by

agent 0. The yellow box shows the 8 zones/grid-cells that agent0 uses to predict the next position and next action of agent1, using the heuristic reward.



(a) Environment (b) Agent Position (c) Peer Belief

Fig. 5: Heuristic h1 computation

The heuristic reward ($Reward_{heur}$) is computed using the reward value for each zone. This reward is computed based on equation 5 where R_{cov} denotes the coverage reward of the particular zone and N_{min_steps} gives the minimum number of steps that the belief agent is required to move from the current position to the zone position.

$$Reward_{heur} = \frac{R_{cov}}{N_{minsteps}} \tag{5}$$

The value of R_{cov} is given by the equation

$$R_{cov} = \begin{cases} +2, & \text{if the zone is unknown} \\ +1, & \text{if the zone is covered} \\ 0, & \text{if the zone is an obstacle} \end{cases} \tag{6}$$

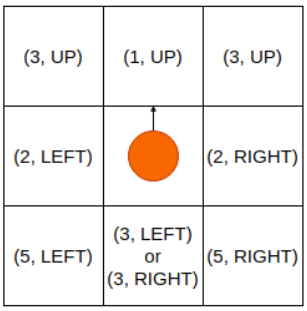


Fig. 6: Step Computation, yaw = 90

The value of N_{min_steps} can be understood by considering figure 6. The figure shows the orientation of the agent (indicated in orange) in pointing upwards (yaw=90) and denotes the corresponding values N_{min_steps} and the first step in that list as a tuple format (N_{min_steps},First step). As an example if the case of the grid-cell just below the agents location is considered, then there are minimum 3 steps that would be required to move to the grid square. These steps are [RIGHT,RIGHT,UP] or [LEFT,LEFT,UP]. Thus the tuple is given by (3,LEFT) or (3,RIGHT). The values for (N_{min_steps},First step) for other yaw angles (0,180 and 270) are the same but only rotated by 90 degrees. Overall using equation 5, the agent uses $Reward_{h1}$ to obtain the first action that moves it to the zone with the highest value of $Reward_{h1}$. In most cases a higher reward indicates an uncovered zone that can be reached with minimum number of steps.

In the Non-Neighbour prediction stage of this scenario, the actions of the peers are predicted for a predefined number of steps T. This is also the size of the best selected paths list that peers share with one another and which is executed in the simulation/rollout phase of the MCTS planning. An important aspect to consider is that for each agent, the environment is itself unknown and is being discovered by each agent as it moves around the area. Due to this, agents would not be able to make perfect predictions about the movement of peers at all times. The number of possibilities for accurate predictions decreases with increase in time since the last meeting between agents. When the predictions are right, this potentially leads to the coverage of more unknown grid-cells in the map by the team. However when the predictions are go wrong, the overall number of steps for coverage can increase. This is because, in a case when the agent assumes that its mispredicted peer agent would cover a particular region (say $region_r$), then it would move elsewhere. However as the peer agent position is mispredicted, none of the agents move towards $region_r$ and the grid-cells in this region remain unexplored.

One way to solve this is inspired from the do-it-yourself (DIY) reward by Claes et al in [4]. The idea is for an agent to have certain bias to perform tasks on its own. This idea is incorporated into the default policy of the simulation/rollout step of the MCTS planner. In the MCTS planner, the current agent only simulates the actions of the Neighbour agents in its own map. For the Non-Neighours, the current agent simulates the plans only after it has planned its own path. By doing this the current agent is able to override the plans of peers for which it predicts paths and has less information about, whereas it considers the plans of agents within communication range in its plans. It is important to note that the reward computation remains the same and Non-Neighbour peers only effect the computation of the $GlobalReward$. This ensures that the current agent still takes into account the plans of Non-Neighbour peers while making its own plans.

E. Frontier Communication

In order to compare the various scenarios having MCTS with an existing work in literature, the frontier communication algorithm as implemented in the work of Antoine et al [1] and adapted by Benavides et al [2] was implemented. This specific work was selected as it also worked with the constraints of limited communication, a distributed setup and an unknown environment. In the approach a team of multiple agents is spawned into the environment. Agents have the capability of sharing information with one another, when within a communication range. The information shared is the map coverage status of an agent and their environment position. Each agent then executes an algorithm that consists of 3 main phases - task identification, task allocation and execution. A task is defined to be a frontier grid-cell. Each agent senses tasks in the task identification phase. In the task allocation phase, agents select frontier zones that are nearest to itself while predicting the allocation made by nearby peer agents. This is done by using the min-Pos algorithm whose working can be found in [1]. Once allocated, each agent executes the computed action following which the process is repeated again.

VI. Experiments

The experiments used to answer the main questions of this paper (in section I) will be explained in this section. The implementation of all the experiments was done on a python based simulator. In this section, each scenario is given an key word and abbreviation based of communication. These are FULLCOMM (FC), NOCOMM (NC), PARCOMM (PC) for full, no and partial communication respectively. Partial Communication with Prediction is given the abbreviation of PARCOMM_PRED (PCPD) and the scenario of limited communication where agents identify frontiers is given by FRONTCOMM. The experiments involved spawning a team of multiple robots into a 20x20 and 40x40 gridmap and recording the number of steps for coverage. The specific size of a 20x20 gridmap with an obstacle density of 10% was chosen, as it is the same size that the authors of the baseline method, Hyatt et al [3] chose in their experiments. A 40x40 gridmap was also selected to test the working of the algorithms in larger gridmaps. In each gridmap dimension case, all experiments were conducted on at least 5 different gridmaps for 10 times each. Therefore each population consists of at least 50 simulations. Further, all explorations were run with a team of agents having a 95% coverage goal. Initially a 100% exploration was given to the team of agents. However in many cases consisting of smaller number of agents (upto 3), there was high variance of results. This was due to situations in which agents would cover a large portion of the arena but leave out exploring far off lying single gridcells. Due to finite computation, agents would take more number of steps to make plans that reach such single unexplored gridcells. A 95% exploration goal on a 20x20 map translates to exploring all the 400 grid-cells but leaving 3 grid-cells unexplored, while a 95% exploration goal on a 40x40 map leaves out 12 grid-cells from the 1600 gridcells present. By default, the sensor range and communication range of each agent were set as 1. With a sensor range as 1, agents could sense the status of nearby 8 grid-cells as explained in subsection III. A communication range of 1 implied that agents could share information with agents when present in the surrounding 8 grid-cells. Further, termination of experiments/simulations in each case took place based on the coverage of the actual environment with what the team of agents covered and not based on the status of coverage of what an agent perceived. The hyper-parameters that were utilised is listed in table I

Sl No	Hyperparameter	Description	Value
1	C_cov	Coverage Reward	5
2	C_hit	Collision Penalty	2
3	lw	Local Reward Weight	1
4	gw	Global Reward Weight	1
5	T_horizon	Rollout steps	30
7	C_p	UCT1 Exploration	$\frac{1}{\sqrt{2}}$

TABLE I: Hyperparameter Values

As mentioned in section V, FULLCOMM is the scenario that was implemented first. After dropping the communication dependency from FULLCOMM, this led to the NOCOMM scenario. As there is no communication, agents in a NOCOMM environment usually takes more time for coverage than FULLCOMM. By adding partial communication to the system, some of the lost performance can be recovered. In most of the experiments, this recovery of lost performance by PARCOMM and even PARCOMM_PRED is measured using a percentage recovery metric. The metric can be understood by considering figure 7. The figure shows an example of the general trend

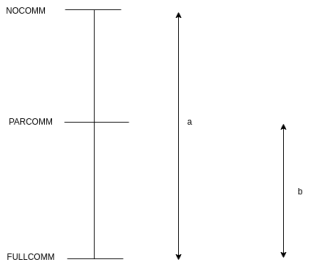


Fig. 7: Recovery Computation

between NOCOMM, PARCOMM and FULLCOMM. The percentage recovery that PARCOMM gives can be measured by using equation 7, where a is the lost performance when moving from FULLCOMM to NOCOMM and b is the lost performance when moving from FULLCOMM to PARCOMM. The same equation can also used to compute the recovery for PARCOMM_PRED.

$$Recovery = \frac{a - b}{a} \tag{7}$$

When the percentage recovery provided is larger, it indicates a smaller value of b. When b is smaller, PARCOMM is closer to FULLCOMM. Thus a larger value of recovery indicates higher amount of performance regained by the PARCOMM (or PARCOMM_PRED) scenario.

A. Exploration Performance v/s Team Size

In this experiment, the number of exploration steps required for coverage is studied for increasing team size, for each scenario. This was done on a 20x20 and 40x40 gridmap.

1) 20x20

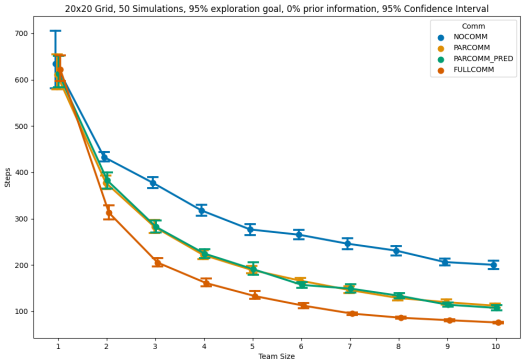


Fig. 8: 20x20 Map

From figure 8, it can be seen that the performance of PARCOMM and PARCOMM_PRED lies in between NOCOMM and FULLCOMM. The performance of PARCOMM and PARCOMM_PRED is almost similiar and

from table II in the Appendix, it can be seen the recovery increases with the increase in number of agents in both scenarios. The recovery increases from 46% for 2 agents upto 71% for 10 agents in PARCOMM and from 42% upto 75%, in the case of PARCOMM_PRED. From statistical significance tests, it is found that there is no significant difference between the exploration steps of PARCOMM and PARCOMM_PRED in a 20x20 map.

2) 40x40

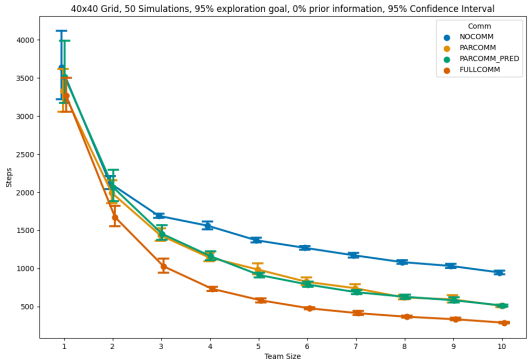


Fig. 9: 40x40 Map

From figure 9, it can be seen that the trend of PARCOMM & PARCOMM_PRED lying between NOCOMM and FULLCOMM is true for even a 40x40 graph. The recovery provided also increases with the increase in number of agents. The recovery increases upto 66% with 10 agents, for both scenarios. While table III has some points where PARCOMM_PRED gives higher recovery than PARCOMM, this difference is not found to be significant.

B. Exploration Performance v/s Prior Information

In this experiment, 3 agents were spawned into an arena under the PARCOMM_PRED scenario. The number of steps were measured for various values of prior information percentage. Prior information percentage refers to the percentage of obstacle locations in the arena that each spawned agent knows about. Thus 0% corresponds to the agent knowing nothing about the obstacle positions while 100% implies that the agent knows the location of all obstacles in the arena.

From both figures it can be seen that irrespective of the gridmap size, having prior information does not effect the exploration performance. Figures 10 and 11 also show no trend of the results. In the case of a 20x20 map, from table IV it can be see that irrespective of the prior percentage the team takes roughly around 300 steps for coverage. For the 40x40 grid case of table V the mean steps roughly lie between 1300 until 1500. This behaviour is also seen for each gridmap in the PARCOMM scenario from tables VI and VII.

C. Exploration Performance v/s Communication Range

1) 20x20

In this experiment, a team of 3 agents with the no information of the environment are spawned into an 20x20

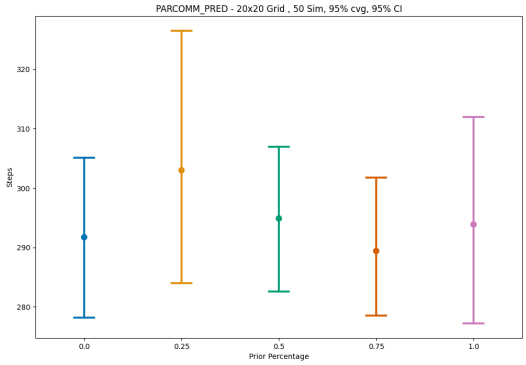


Fig. 10: 20x20 Map

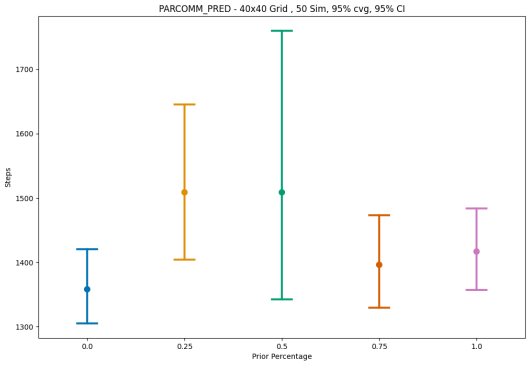


Fig. 11: 40x40 Map

arena. The communication range is varied to take one of the values in the set $\{1, 2, 3, 4, 5\}$. A communication range of N indicates that an agent can communicate N grid-cells around it in all directions. Each of the scenarios of NOCOMM, FULLCOMM, PARCOMM and PARCOMM_PRED have been simulated and the recovery provided by the PARCOMM and PARCOMM_PRED has been measured using equation 7.

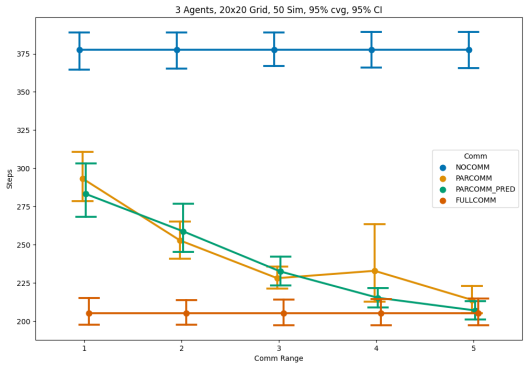


Fig. 12: 20x20 Map

It can be seen from figure 12 that with an increase in communication range, both PARCOMM and PARCOMM_PRED approach the behaviour of FULLCOMM. At a communication range of 5, the recovery provided is

as high as 99% for PARCOMM_PRED.

2) 40x40

In this subsection, a team of 3 agents with the no information of the environment was spawned into a 40x40 arena. Due to the high variance in the 50 simulation case, more number of simulations were added into the study thereby leading to 400 simulations for each population.

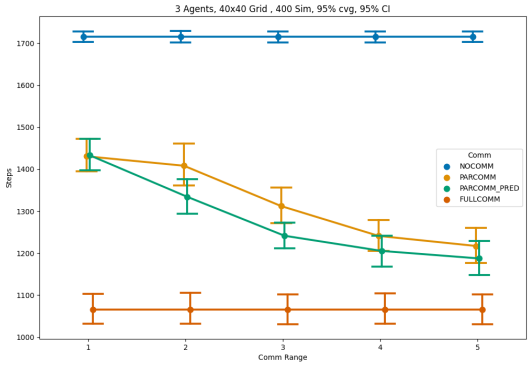


Fig. 13: 40x40 Grid, 3 agents

In the case of 3 agents spawned into the arena, the performance of PARCOMM and PARCOMM_PRED approach the coverage steps of FULLCOMM with an increase in the communication range. It is also found that there is significant difference between PARCOMM and PARCOMM_PRED and that PARCOMM_PRED is able to provide additional benefit when the communication range is set as 2 or 3. This is indicated in table XI. At communication range 2, the additional benefit provided by PARCOMM_PRED is about 12%, while it is 9% in the case of a communication range of 3. Apart from this, the working with a team of 9 agents in a 40x40 map was also studied.

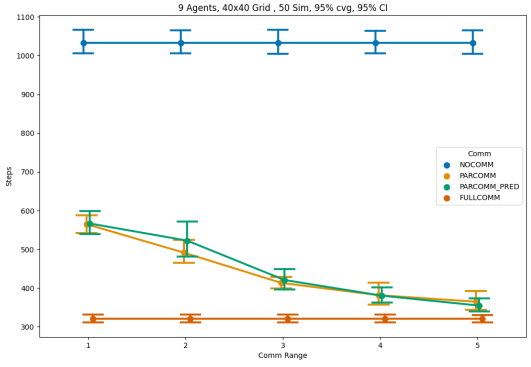
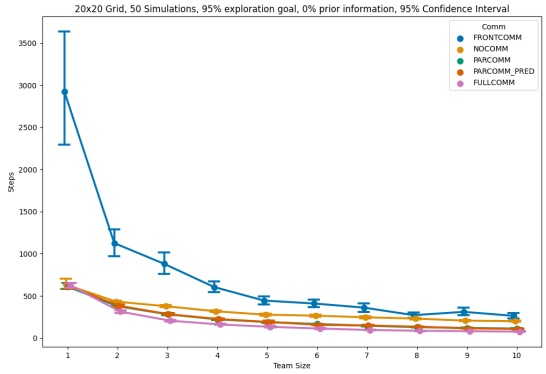


Fig. 14: 40x40 Grid, 9 agents

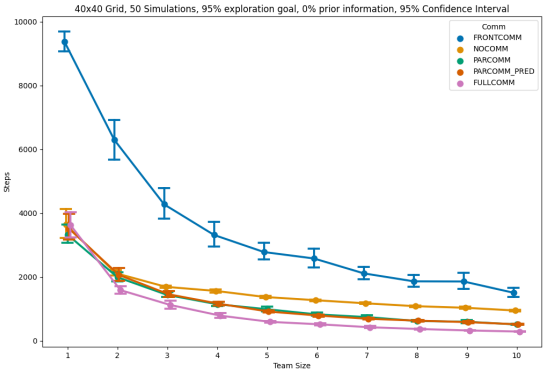
From figure 14, it can be seen that there is a lot of overlap in the behaviour of PARCOMM and PARCOMM_PRED. This is due to the increased number of chances where agents can meet one another and exchange information.

D. Exploration Performance - FRONTCOMM Comparison

In this section each MCTS scenario is compared with the FRONTCOMM algorithm. For comparison, the number of steps required for each scenarios has been compared for 1 until 10 agents. The results have been shown in figure 15.



(a) 20x20 Map



(b) 40x40 Map

Fig. 15: Scenarios v/s FRONTCOMM

It can be observed from the figure, that the MCTS methods perform area exploration much faster than the frontier communication method. This also includes the NOCOMM method that has no possibility for communication. This can attributed to the benefit provided by look ahead planning in the MCTS algorithms.

VII. Discussion

In this section, the results from various experiments conducted are summarised. The MCTS scenarios were compared for their exploration steps as a function of the team size in section VI-A. In this case, the key observations made were that as per expectation the order of number of coverage steps in decreasing order was NO_COMM, PARCOMM & PARCOMM_PRED and then FULL_COMM. It was found that the recovery of performance provided in the 10 agent case by PARCOMM & PARCOMM_PRED was greater than 65% for both 20x20 and 40x40 maps. It was also noticed that there was no significant benefit being provided by PARCOMM_PRED to PARCOMM in this experiment. Therefore, the exact

scenario where predictions could provide additional benefit was to be tracked. The first speculation was that giving prior knowledge could help the predictions. However from the study in section VI-B, it was found that giving prior information did not have much effect. Instead, the next speculation of increasing the number of meetups & information sharing instance by increasing the communication range was tested in section VI-C. It was found that with increase in communication range, the performance of both PARCOMM as well as PARCOMM_PRED comes closer to FULLCOMM and the performance recovery provided by each was greater than 90% for both 20x20 and 40x40 maps in the 5 grid-cell communication range case. Further, it was found that for 3 agents in a 40x40 gridmap, increasing the communication range can provide an additional benefit of prediction over the PARCOMM by 12%. This difference was also found to be statistically significant. The additional benefit however decreased with an increase in the communication range. The experiments ended with section VI-D where MCTS based algorithms were found to explore faster than the frontier communication method.

We turn back to the research questions mentioned in section I and answer them using the observations made.

1) ***RQ1*** *- What distributed strategies can cooperative agents utilise to explore unknown environments with limited communication, in minimal number of steps?*

The strategy utilised by each agent was that of MCTS. From subsection VI-D of the previous chapter it has been observed that this strategy helps a team of agents perform faster than the existing frontier communication method by effectively avoiding cluttered regions. In a limited communication environment, including partial communication helps decrease the exploration steps and the number of exploration steps decreases with the increase in the number of agents and communication range.

2) ***RQ2*** *- What is the effect of limited communication on the individual strategies?*

Limited Communication, especially of useful information like map coverage and best plan information increases the number of steps required by the team for area coverage. If the case of completely removing communication is considered then the No Communication scenario takes the maximum number of steps. This can be seen in the graphs of chapter VI. Partial communication improves this behaviour and reduces the number of steps for exploration. The number of steps can be further reduced with an increase in the number of agents or communication range.

3) ***RQ3*** *- What strategies would help in replicating the performance of agents as in an arena of full communication?*

As seen from the graphs in section VI, full communication behaviour can be replicated upto a significant extent by introducing partial communication. In addition to this, giving agents the possibility to predict the behaviour of peer agents can help further reduce the steps of coverage in very specefic situations. It has also been experimentally found that either increasing the number of agents in the team or the communication range can help replicate the behaviour full communication. This is because of the increased number of chances for agents to share important map coverage information and also correct the mispredictions of their paths made by peer agents, in the case of partial communication with prediction. Further, it has also been found that giving prior information of the arena in terms of the locations of obstacles does not have much of an effect on improving exploration, for the scenarios of partial communication and partial communication with prediction.

VIII. Conclusion

In this thesis, strategies that enable multi-robot exploration in limited communication environments have been studied. The first phase was to perform an in-depth literature study on existing methods. Monte Carlo Tree Search (MCTS) algorithm was selected from this study, due to its advantages of being a look ahead planner and its ability to simulate the actions of peer agents which could help with a cooperative strategy. An existing work that used MCTS by Hyatt et al [3] was implemented first. The communication dependency of this work was completely removed and strategies to recover the lost efficiency were studied. As seen from chapter VI the strategy of providing robots with the ability communicate when within a communication range could regain the lost efficiency by a percentage as high as 75%. By increasing the communication range this percentage could be increased upto 99%. It was also found for larger maps, that increasing the communication range of the agents up-to a limit could enable peer prediction to bring an additional benefit to the exploration for a small team size.

Some recommendations for future work include finding techniques on how to ensure 100% area coverage and enabling distributed termination such that exploration terminate only when each agent discovers that the area is terminated. Another direction is to consider the working with dynamic obstacles, that are very likely to be present in disaster environments. Further, the working of MCTS in multi-agent exploration is yet to be performed in a 3-D case, where the action set would increase from 3 $\{UP, RIGHT, LEFT\}$ to 5 $\{UP, RIGHT, LEFT, TOP, BOTTOM\}$.

References

[1] A. Bautin, O. Simonin, and F. Charpillet, "Minpos : A novel frontier allocation algorithm for multi-robot exploration," in *Intelligent Robotics and Applications*, C.-Y. Su, S. Rakheja, and H. Liu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 496–508.

[2] F. Benavides, P. Monzón, C. P. Carvalho Chanel, and E. Grampín, "Multi-robot cooperative systems for exploration: Advances in dealing with constrained communication environments," in *2016 XIII Latin American Robotics Symposium and IV Brazilian Robotics Symposium (LARS/SBR)*, 2016, pp. 181–186.

[3] P. Hyatt, Z. Brock, and M. D. Killpack, "A versatile multi-robot monte carlo tree search planner for on-line coverage path planning," 2020.

[4] D. Claes, F. A. Oliehoek, H. Baier, and K. Tuyls, "Decentralised online planning for multi-robot warehouse commisioning," May 2017, pp. 492–500, **Best paper nominee.** [Online]. Available: http://www.ifaamas.org/Proceedings/aamas2017/pdfs/p492.pdf

[5] A. Howard, M. J. Matarić, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," in *Proceedings of the International Symposium on Distributed Autonomous Robotic Systems*, 2002, pp. 299–308.

[6] S. Damer, L. Ludwig, M. A. LaPoint, M. Gini, N. Papanikolopoulos, and J. Budenske, "Dispersion and exploration algorithms for robots in unknown environments," in *Unmanned Systems Technology VIII*, G. R. Gerhart, C. M. Shoemaker, and D. W. Gage, Eds., vol. 6230, International Society for Optics and Photonics. SPIE, 2006, pp. 251 – 260. [Online]. Available: https://doi.org/10.1117/12.668915

[7] B. Pang, Y. Song, C. Zhang, H. Wang, and R. Yang, "A swarm robotic exploration strategy based on an improved random walk method," *J. Robotics*, vol. 2019, pp. 6 914 212:1–6 914 212:9, 2019.

[8] H. Umari and S. Mukhopadhyay, "Autonomous robotic exploration based on multiple rapidly-exploring randomized trees," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 1396–1402.

[9] M. Nair and S. Givigi, "The impact of communications considerations in multi-robot systems," in *2019 International Conference on Communications, Signal Processing, and their Applications (ICCSPA)*, 2019, pp. 1–6.

[10] M. Dadvar, S. Moazami, H. R. Myler, and H. Zargarzadeh, "Exploration and coordination of complementary multi-robot teams in a hunter and gatherer scenario," 2020.

[11] J. Lopez-Perez, U. Hernandez-Belmonte, J.-P. Ramirez-Paredes, M. Contreras-Cruz, and V. Ayala, "Distributed multirobot exploration based on scene partitioning and frontier selection," *Mathematical Problems in Engineering*, vol. 2018, pp. 1–17, 06 2018.

[12] U. Jain, R. Tiwari, and W. Godfrey, "A hybrid evsa approach in clustered search space with ad-hoc partitioning for multi-robot searching," *Evolutionary Intelligence*, vol. 13, 12 2020.

[13] Y. Meng, "Multi-robot searching using game-theory based approach," *International Journal of Advanced Robotic Systems*, vol. 5, no. 4, p. 44, 2008. [Online]. Available: https://doi.org/10.5772/6232

[14] J. Ni, G. Tang, Z. Mo, W. Cao, and S. X. Yang, "An improved potential game theory based method for multi-uav cooperative search," *IEEE Access*, vol. 8, pp. 47 787–47 796, 2020.

[15] G. Best, O. M. Cliff, T. Patten, R. R. Mettu, and R. Fitch, "Dec-mcts: Decentralized planning for multi-robot active perception," *Int. J. Robotics Res.*, vol. 38, no. 2-3, 2019. [Online]. Available: https://doi.org/10.1177/0278364918755924

[16] M. Li, W. Yang, Z. Cai, S. Yang, and J. Wang, "Integrating decision sharing with prediction in decentralized planning for multi-agent coordination under uncertainty," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 7 2019, pp. 450–456. [Online]. Available: https://doi.org/10.24963/ijcai.2019/64

[17] R. Nouacer, M. Hussein, H. Espinoza, Y. Ouhammou, M. Ladeira, and R. Castiñeira, "Towards a framework of key technologies for drones," *Microprocessors and Microsystems*, vol. 77, p. 103142, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0141933120303094

[18] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Prentice Hall, 2010.

[19] F. A. Oliehoek and C. Amato, *A Concise Introduction to Decentralized POMDPs*, 1st ed. Springer Publishing Company, Incorporated, 2016.

[20] C. Browne, E. Powley, D. Whitehouse, S. Lucas, P. I. Cowling, S. Tavener, D. Perez, S. Samothrakis, S. Colton, and et al., "A survey of monte carlo tree search methods," *IEEE TRANSACTIONS ON COMPUTATIONAL INTELLIGENCE AND AI*, 2012.

APPENDIX A
EXPERIMENT NUMERIC RESULTS

In all the tables, the abbreviation and meaning are as follows:

1) TS - Team Size
2) CR - Communication Range
3) FC - Full Communication
4) NC - No Communication
5) PC - Partial Communication
6) PC - Partial Communication with Prediction

TS	FC	NC	PC Rec	PCPD Rec
1	622	634	197	175
2	313	433	46	42
3	205	378	55	55
4	161	318	61	59
5	133	277	61	60
6	112	265	65	71
7	95	246	66	64
8	86	231	70	67
9	81	206	69	73
10	76	201	71	75

TABLE II: Coverage steps versus Team Size Data - 20x20

TS	FC	NC	PC Rec	PCPD Rec
1	3265	3637	83	33
2	1674	2118	28	12
3	1028	1691	38	36
4	733	1562	50	48
5	582	1371	49	58
6	477	1271	56	61
7	414	1174	57	64
8	367	1084	65	63
9	334	1034	63	64
10	289	948	66	66

TABLE III: Coverage steps versus Team Size Data - 40x40

Prior	Steps mean	Steps CI
0	292	14
0.25	303	24
0.5	295	13
0.75	289	13
1	294	19

TABLE IV: PARCOMM_H3G Coverage steps versus Prior Information Data - 20x20

Prior	Step mean	Step CI
0	1358	62
0.25	1509	125
0.5	1509	234
0.75	1396	77
1	1417	69

TABLE V: PARCOMM_H3G Coverage steps versus Prior Information Data - 40x40

Prior	Steps mean	Steps CI
0	294	13
0.25	286	11
0.5	324	28
0.75	292	14
1	292	11

TABLE VI: PARCOMM Coverage steps versus Prior Information Data - 20x20

Prior	Step mean	Step CI
0	1453	76
0.25	1414	81
0.5	1496	115
0.75	1427	91
1	1461	74

TABLE VII: PARCOMM Coverage steps versus Prior Information Data - 40x40

CR	FC	NC	PC rec	PCPD rec
1	205	377	49	55
2	205	377	72	69
3	205	377	87	84
4	205	377	84	94
5	205	377	95	99

TABLE VIII: Coverage steps versus Communication Range Data - 20x20, 3 agents

CR	FC	NC	PC rec	PCPD rec
1	1066	1715	44	43
2	1066	1715	47	59
3	1066	1715	62	73
4	1066	1715	73	79
5	1066	1715	77	81

TABLE IX: Coverage steps versus Communication Range Data - 40x40, 3 agents

CR	FC	NC	PC rec	PCPD rec
1	321	1034	66	66
2	321	1034	76	72
3	321	1034	87	86
4	321	1034	92	92
5	321	1034	94	95

TABLE X: Coverage steps versus Communication Range Data - 40x40, 9 agents

Comm	PC Mean	PCP Mean	t_val	p_val	Sig?
1	1431	1433	-0.08	0.93	No
2	1408	1334	2.16	0.03	Yes
3	1312	1241	2.62	0.01	Yes
4	1241	1205	1.32	0.19	No
5	1217	1187	1.01	0.31	No

TABLE XI: Statistical Significance Check Comm - Cvg v/s Comm 40x40, 3 agents

Bibliography

[1] URL: https://people.eecs.berkeley.edu/~pabbeel/cs287-fa12/#syllabus.

[2] Stefano V. Albrecht and Peter Stone. “Autonomous agents modelling other agents: A comprehensive survey and open problems”. In: *Artificial Intelligence* 258 (May 2018), pp. 66–95. ISSN: 0004-3702. DOI: 10.1016/j.artint.2018.01.002. URL: http://dx.doi.org/10.1016/j.artint.2018.01.002.

[3] F. Amigoni, J. Banfi, and N. Basilico. “Multirobot Exploration of Communication-Restricted Environments: A Survey”. In: *IEEE Intelligent Systems* 32.6 (2017), pp. 48–57. DOI: 10.1109/MIS.2017.4531226.

[4] P. Baran. “On Distributed Communications Networks”. In: *IEEE Transactions on Communications Systems* 12.1 (1964), pp. 1–9. DOI: 10.1109/TCOM.1964.1088883.

[5] Antoine Bautin, Olivier Simonin, and François Charpillet. “MinPos : A Novel Frontier Allocation Algorithm for Multi-robot Exploration”. In: *Intelligent Robotics and Applications*. Ed. by Chun-Yi Su, Subhash Rakheja, and Honghai Liu. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 496–508. ISBN: 978-3-642-33515-0.

[6] F. Benavides et al. “Multi-robot Cooperative Systems for Exploration: Advances in Dealing with Constrained Communication Environments”. In: *2016 XIII Latin American Robotics Symposium and IV Brazilian Robotics Symposium (LARS/SBR)*. 2016, pp. 181–186. DOI: 10.1109/LARS-SBR.2016.37.

[7] Graeme Best et al. “Dec-MCTS: Decentralized planning for multi-robot active perception”. In: *Int. J. Robotics Res.* 38.2-3 (2019). DOI: 10.1177/0278364918755924. URL: https://doi.org/10.1177/0278364918755924.

[8] Manuele Brambilla et al. *Swarm robotics: a review from the swarm engineering perspective*. 2012.

[9] Cameron Browne et al. “A survey of Monte Carlo tree search methods”. In: *IEEE TRANSACTIONS ON COMPUTATIONAL INTELLIGENCE AND AI* (2012).

[10] Dominique Chabot. “Trends in drone research and applications as the Journal of Unmanned Vehicle Systems turns five”. In: *Journal of Unmanned Vehicle Systems* 6.1 (2018), pp. vi–xv. DOI: 10.1139/juvs-2018-0005. eprint: https://doi.org/10.1139/juvs-2018-0005. URL: https://doi.org/10.1139/juvs-2018-0005.

[11] Saad Chakkor et al. “Comparative Performance Analysis of Wireless Communication Protocols for Intelligent Sensors and Their Applications”. In: *CoRR* abs/1409.6884 (2014). arXiv: 1409.6884. URL: http://arxiv.org/abs/1409.6884.

[12] Daniel Claes et al. “Decentralised Online Planning for Multi-Robot Warehouse Commisioning”. In: **Best paper nominee.** May 2017, pp. 492–500. URL: http://www.ifaamas.org/Proceedings/aamas2017/pdfs/p492.pdf.

[13] Daniel Claes et al. “Effective Approximations for Multi-Robot Coordination in Spatially Distributed Tasks”. In: *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*. AAMAS ’15. Istanbul, Turkey: International Foundation for Autonomous Agents and Multiagent Systems, 2015, pp. 881–890. ISBN: 9781450334136.

[14] Aleksander Czechowski and Frans A. Oliehoek. “Decentralized MCTS via Learned Teammate Models”. In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence* (July 2020). DOI: 10.24963/ijcai.2020/12. URL: http://dx.doi.org/10.24963/ijcai.2020/12.

[15] Mehdi Dadvar et al. *Exploration and Coordination of Complementary Multi-Robot Teams in a Hunter and Gatherer Scenario*. 2020. arXiv: 1912.07521 [cs.MA].

[16] Steven Damer et al. “Dispersion and exploration algorithms for robots in unknown environments”. In: *Unmanned Systems Technology VIII*. Ed. by Grant R. Gerhart, Charles M. Shoemaker, and Douglas W. Gage. Vol. 6230. International Society for Optics and Photonics. SPIE, 2006, pp. 251–260. DOI: 10.1117/12.668915. URL: https://doi.org/10.1117/12.668915.

[17] Matthew Dunbabin and Lino Marques. “Robots for Environmental Monitoring: Significant Advancements and Applications”. In: *IEEE Robotics Automation Magazine* 19.1 (2012), pp. 24–39. DOI: 10.1109/MRA.2011.2181683.

[18] Alessandro Gasparetto et al. “Path Planning and Trajectory Planning Algorithms: A General Overview”. In: *Motion and Operation Planning of Robotic Systems: Background and Practical Approaches*. Ed. by Giuseppe Carbone and Fernando Gomez-Bravo. Cham: Springer International Publishing, 2015, pp. 3–27. ISBN: 978-3-319-14705-5. DOI: 10.1007/978-3-319-14705-5_1. URL: https://doi.org/10.1007/978-3-319-14705-5_1.

[19] Perry J. Hardin and Ryan R. Jensen. “Small-Scale Unmanned Aerial Vehicles in Environmental Remote Sensing: Challenges and Opportunities”. In: *GIScience & Remote Sensing* 48.1 (2011), pp. 99–111. DOI: 10.2747/1548-1603.48.1.99. eprint: https://doi.org/10.2747/1548-1603.48.1.99. URL: https://doi.org/10.2747/1548-1603.48.1.99.

[20] Andrew Howard, Maja J. Matarić, and Gaurav S. Sukhatme. “Mobile Sensor Network Deployment using Potential Fields: A Distributed, Scalable Solution to the Area Coverage Problem”. In: *Proceedings of the International Symposium on Distributed Autonomous Robotic Systems*. 2002, pp. 299–308.

[21] Phillip Hyatt, Zachary Brock, and Marc D. Killpack. *A Versatile Multi-Robot Monte Carlo Tree Search Planner for On-Line Coverage Path Planning*. 2020. arXiv: 2002.04517 [cs.MA].

[22] Upma Jain, Ritu Tiwari, and Wilfred Godfrey. “A hybrid EVSA approach in clustered search space with ad-hoc partitioning for multi-robot searching”. In: *Evolutionary Intelligence* 13 (Dec. 2020). DOI: 10.1007/s12065-020-00356-1.

[23] Minglong Li et al. “Integrating Decision Sharing with Prediction in Decentralized Planning for Multi-Agent Coordination under Uncertainty”. In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, July 2019, pp. 450–456. DOI: 10.24963/ijcai.2019/64. URL: https://doi.org/10.24963/ijcai.2019/64.

[24] Chun Fui Liew et al. *Recent Developments in Aerial Robotics: A Survey and Prototypes Overview*. 2017. arXiv: 1711.10085 [cs.RO].

[25] Jose Lopez-Perez et al. “Distributed Multirobot Exploration Based on Scene Partitioning and Frontier Selection”. In: *Mathematical Problems in Engineering* 2018 (June 2018), pp. 1–17. DOI: 10.1155/2018/2373642.

[26] Zhenqiu Lu and Ke-Hai Yuan. “Welch’s t test”. In: Jan. 2010, pp. 1620–1623. DOI: 10.13140/RG.2.1.3057.9607.

[27] Yan Meng. “Multi-Robot Searching using Game-Theory Based Approach”. In: *International Journal of Advanced Robotic Systems* 5.4 (2008), p. 44. DOI: 10.5772/6232. eprint: https://doi.org/10.5772/6232. URL: https://doi.org/10.5772/6232.

[28] M. E. Mkiramweni et al. “A Survey of Game Theory in Unmanned Aerial Vehicles Communications”. In: *IEEE Communications Surveys Tutorials* 21.4 (2019), pp. 3386–3416. DOI: 10.1109/COMST.2019.2919613.

[29] M. Nair and S. Givigi. “The Impact of Communications Considerations in Multi-Robot Systems”. In: *2019 International Conference on Communications, Signal Processing, and their Applications (ICCSPA)*. 2019, pp. 1–6. DOI: 10.1109/ICCSPA.2019.8713614.

[30] Iñaki Navarro and Fernando Matía. “An Introduction to Swarm Robotics”. In: *ISRN Robotics* 2013 (Sept. 2012). DOI: 10.5402/2013/608164.

[31] J. Ni et al. “An Improved Potential Game Theory Based Method for Multi-UAV Cooperative Search”. In: *IEEE Access* 8 (2020), pp. 47787–47796. DOI: 10.1109/ACCESS.2020.2978853.

[32] Réda Nouacer et al. “Towards a framework of key technologies for drones”. In: *Microprocessors and Microsystems* 77 (2020), p. 103142. ISSN: 0141-9331. DOI: https://doi.org/10.1016/j.micpro.2020.103142. URL: https://www.sciencedirect.com/science/article/pii/S0141933120303094.

[33] Frans A. Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. 1st. Springer Publishing Company, Incorporated, 2016. ISBN: 3319289276.

[34] Bao Pang et al. “A Swarm Robotic Exploration Strategy Based on an Improved Random Walk Method”. In: *J. Robotics* 2019 (2019), 6914212:1–6914212:9.

[35] Ankit Ravankar et al. “Autonomous Mapping and Exploration of UAV Using Low Cost Sensors”. In: vol. 4. Nov. 2018, p. 5753. DOI: 10.3390/ecsa-5-05753.

[36] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 3rd ed. Prentice Hall, 2010.

[37] Melanie Schranz et al. “Swarm Robotic Behaviors and Current Applications”. In: *Frontiers in Robotics and AI* 7 (2020), p. 36. ISSN: 2296-9144. DOI: 10.3389/frobt.2020.00036. URL: https://www.frontiersin.org/article/10.3389/frobt.2020.00036.

[38] Reid G. Simmons et al. “Coordination for Multi-Robot Exploration and Mapping”. In: *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*. AAAI Press, 2000, pp. 852–858. ISBN: 0262511126.

[39] D. K. Sutantyo et al. “Multi-robot searching algorithm using Lévy flight and artificial potential field”. In: *2010 IEEE Safety Security and Rescue Robotics*. July 2010, pp. 1–6. DOI: 10.1109/SSRR.2010.5981560.

[40] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018. ISBN: 0262039249.

[41] E. Ugur, A. E. Turgut, and E. Sahin. “Dispersion of a swarm of robots based on realistic wireless intensity signals”. In: *2007 22nd international symposium on computer and information sciences*. 2007, pp. 1–6. DOI: 10.1109/ISCIS.2007.4456899.

[42] H. Umari and S. Mukhopadhyay. “Autonomous robotic exploration based on multiple rapidly-exploring randomized trees”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 1396–1402. DOI: 10.1109/IROS.2017.8202319.

[43] Chunhua Zhang and John M Kovacs. “The application of small unmanned aerial systems for precision agriculture: a review”. In: *Precision agriculture* 13.6 (2012), pp. 693–712.