José Gilmar Nunes de Carvalho Filho

Multi-Robot Exploration with Constrained Communication

Florianópolis, Brasil
28 de Junho de 2016

José Gilmar Nunes de Carvalho Filho

Multi-Robot Exploration with Constrained Communication

Tese submetida ao Programa de Pós-Graduação em Engenharia de Automação e Sistemas para a obtenção do Grau de Doutor.

Universidade Federal de Santa Catarina - UFSC

Orientador Jean-Marie Alexandre Farines, Dr.
Coorientador José Eduardo Ribeiro Cury, Dr.

Florianópolis, Brasil
28 de Junho de 2016

,
Multi-Robot Exploration with Constrained Communication : / José Gilmar Nunes de Carvalho Filho; orientador, Jean-Marie Alexandre Farines, Dr. ; coorientador, José Eduardo Ribeiro Cury, Dr.. - Florianópolis, Brasil 2016.
150 p.

- Universidade Federal de Santa Catarina - UFSC, Centro Tecnológico - CTC. Programa de Pós - Graduação em Engenharia de Automação e Sistemas - PPGEAS.

Inclui Referências

1.Multi-Robot exploration. 2. Constrained communication. 3. Map sharing. 4. Coordination. I. , . II. , . III. Universidade Federal de Santa Catarina - UFSC. Programa de Pós - Graduação em Engenharia de Automação e Sistemas - PPGEAS. III. Multi-Robot Exploration with Constrained Communication.

Esta Tese foi julgada adequada como **TESE DE DOUTORADO** no Curso de Doutorado em Engenharia de Automação e Sistemas e aprovada em sua forma final pela Banca Examinadora designada.

Dr. Jean-Marie Farines
Orientador

Dr. José Eduardo Ribeiro Cury
Coorientador

Dr. Daniel Ferreira Coutinho
Coordenador do PPGEAS

Banca Examinadora:

Dr. Jean-Marie Farines
Orientador

Dr. Luiz Chaimowicz

Dr. André Bittencourt Leal

Dr. Ubirajara Moreno

Dr. Fabio Baldissera

Dr. Edson Roberto de Pieri

Dr. Claude Martinez

“When we are not sure, we are alive.”
(Graham Greene)

À mulher da minha vida, Tamires, *pelo apoio incondicional em todos os momentos, principalmente naqueles onde as dúvidas, o cansaço e o estresse ofuscavam a linha de chegada.*

Sem você nenhuma conquista valeria a pena.

ACKNOWLEDGEMENTS

Primeiramente a Deus, que sempre iluminou a minha caminhada. Sem Ele, nada somos, nada podemos.

À minha esposa Tamires, pela dedicação, amor e inexaurível paciência.

À minha mãe, fonte inesgotável de amor, fé, caridade, temperança e ética. Ao meu pai (*in memorian*), pelo carinho e amor que sempre me teve.

À "Nin", minha segunda mãe, por todos os anos de cuidados, atenção, carinho e amor incondicional.

Às minhas irmãs, Livia e Letícia, pela amizade incondicional e pelo apoio que nunca me foi negado.

Aos meus sobrinhos Ana Flávia, Ana Sofia, Ruan e Yan, pelo afeto.

Aos meus avós, Vovô Dezinho (*in memorian*), que proporcionou meus primeiros contatos com o mundo da eletrônica e a Vovó Castália pelo carinho e amor. A "Vô Tonho" e "Vó Gilza" (*in memorian*), pelo modelo de simplicidade, honestidade, carinho e amor à família.

Agradeço também aos amigos do DAS, em especial ao Luciano, "Faixa", Lange, Odilson, Renan, Rattus, Thiago e Toscano pela amizade, compartilhamento e aprimoramento das ideias.

Aos meus Orientadores, Jean-Marie e Cury, sem a ajuda dos quais, esse trabalho não haveria de ser realizado.

Aos meus Orientadores estrangeiros, Domenc e Miguel Angel, pelo acolhimento e atencão enquanto estive na Espanha e após minha volta ao Brasil.

Por fim agradeço ao CNPq e a CAPES pelo apoio financeiro que tornou possível a realização deste trabalho.

ABSTRACT

Over the last two decades, several methods for exploration with Multi-Robot Systems (MRS) have been proposed, most of them based on the allocation of *frontiers* (exploration targets) and typically applying local optimization policies. However, communication issues have usually been neglected. This thesis investigates multi-robot exploration by considering that robots have limited communication radius. Two methods, one based on a flat network architecture (DSM) and another based on a hierarchical architecture (HSM), were proposed to share map information. While DSM considers a propagation scheme to share information and synchronize the map of robots, HSM organizes robots in a hierarchical architecture where some robots act as leaders (*clusterheads*) and are responsible for synchronizing the maps of the robots in the network. Formal proof that both methods guarantee the synchronization of the map of all robots in a network is presented. In addition, experiments were conducted by considering systems with different number of robots, network topologies and different map's sizes. The results show that both methods are able to synchronize the map of the robots when they can lose communication links, but HKM usually presents smaller convergence time, number of exchanged messages and amount of transmitted data. We also propose Hierarchical K-Means (HKME), a method for multi-robot coordination in exploration tasks that handles communication problems, such as link losses. To handle communication among robots, HKME arranges them into clusters and elects leaders for each. Clusters evolve dynamically as robots lose or establish communication with their peers. HKME uses HSM to guarantee that the map of the robots are synchronized and also uses the hierarchical organization of the robots to coordinate them in order to minimize the variance of the time at which they reach all regions of the workspace, while balancing their workload and decreasing the exploration time. Experiments were conducted by considering different types of workspace and communication radius. The results show that HKME behaves like a centralized algorithm when communication is granted, while being able to withstand severe degradation in communication radius.

Keywords: Multi-Robot exploration, Constrained communication, Map sharing, Coordination

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

Many mobile robotics applications require that multiple robots cooperate to perform specific tasks in an unknown workspace. However, the execution of those tasks depends in general of information about the environment where the robots are placed. When the workspace is initially unknown, the robots must perform an exploration task to generate a suitable model of the environment (VISSER et al., 2013).

The aforementioned exploration task requires that robots of a Multi-Robot System (MRS) move over the entire workspace in a coordinated way, using their sensors to collect data about static (walls, obstacles, etc.) and dynamic (people, vehicles, etc) elements (BUTZKE; LIKHACHEV, 2011). The key question in multi-robot exploration is what place each robot must explore in order to minimize aspects such as the time necessary to generate a complete map of the workspace.

In general, it is natural to expect that multiple robots could execute a task, such as exploration, faster than a single robot. However, some problems usually arise when multiple robots are used in exploration tasks. First, depending on the number of robots and the workspace in which they are deployed, they can be forced to move together. As a consequence, they can interfere with the motion of other robots or even collide. Second, some robots may move towards the same non-explored area or one already explored (PUIG; GARCÍA; WU, 2011). Third, exploration can be executed in an unbalanced way, with some robots doing most of the work while others being underused. To deal with these problems, the robots in a MRS need to be properly coordinated in order to execute exploration tasks in a balanced and efficient way.

To coordinate MRS in exploration tasks, most methods identify exploration targets, usually points in the frontier between already explored areas and non-explored ones, and assign them to the robots minimizing aspects such as the sum of the distances that all robots have to travel or the time necessary for the robots reach their targets

(BAUTIN; SIMONIN; CHARPILLET, 2012; BURGARD et al., 2005; MATIGNON; JEANPIERRE; MOUADDIB, 2012). To assign the best targets for each robot, the coordination mechanism, which can be ran by a single robot (centralized) or several ones (distributed), needs as much information as possible about the environment, so it can identify which areas are not explored yet and define and assign new exploration targets to all robots correctly.

Most methods for MRS coordinated exploration consider perfect communication, with all robots always having a direct communication link with the others and being always able to send or receive messages instantaneously. These methods usually propose centralized schemes where all robots send all information that they collect about the workspace to a leader or an operational base. Then, the leader (or the base) synchronizes all information in a single and most complete map of the workspace, detects new exploration targets and assigns them to the robots (SIMMONS et al., 2000; BURGARD et al., 2005; STACHNISS; MOZOS; BURGARD, 2008; SENTHILKUMAR; BHARADWAJ, 2012).

However, there are many situations where perfect communication among robots cannot be assumed, such as outdoor applications and exploration of large and unstructured workspaces. In that case, robots rely on others to forward messages to the final destination and the MRS can be viewed as a *mobile ad hoc* network. In addition, robots may lose communication and be separated in several unconnected networks. In this context, most centralized exploration methods are unable to coordinate the robots. Even straightforward extensions where leaders execute the centralized scheme independently in each network do not guarantee neither balance nor global efficiency. Moreover, as the exchange of messages among robots become harder to accomplish, they need a mechanism to share map information that ensures that all robots in the same network have the same map during the allocation of exploration targets.

This thesis investigates multi-robot exploration by considering that robots have limited communication radius. Thus, the existence

of a link between two robots depends on the distance between them. Communication problems such as message losses, delay in message exchanging, and losses of a message's packages are not addressed in this work.

Thus, the objective is to develop a method that avoids exploration of already explored areas, reduces the time necessary to explore the entire workspace (*exploration time*), disperses robots over the workspace quickly and balances the exploration workload among the robots. Next, the thesis objectives are described.

1.1 OBJECTIVE

During the PhD, we investigate multi-robots exploration and our goal is to develop an efficient method for exploration that handles the possibility of link losses. Efficiency of the method is defined by the following aspects:

Reduction of exploration time: Robots must complete the workspace map as soon as possible.

Avoidance of redundant exploration: The method must avoid that robots explore areas already explored by others.

Balancing of the workload of robots: Workload of exploration must be divided fairly among robots.

Quick spreading of robots: Robots must disperse through the workspace in order to reach all regions of it as soon as possible. (PUIG; GARCÍA; WU, 2011).

1.1.1 Specific Objectives

To handle all efficiency aspects, we define two specific objectives: developing an efficient method for map sharing and developing a method for multi-robots coordination in exploration tasks.

Development of an efficient method for map sharing

The first objective of the PhD is to propose a method for map sharing that handles constrained communication. The method must guarantee that all robots in the same network have the same map. By knowing all explored areas, robots can calculate better paths from its position to the next exploration target. Also, they can avoid exploring already explored areas.

Other important feature is that the method must avoid the exchanging of unnecessary information, which could increase significantly the amount of data transmitted in the network. As the amount of transmitted data increases, robots need better communication channels, with a higher bandwidth (rate of data transfer), to share their maps. Additionally, the proposed solutions must be able to share maps avoiding *information inconsistency* problems. In this work, an information inconsistency problem occurs when robots in the same network have different maps of the workspace.

In this thesis, we consider solutions based on *ad hoc* networks with flat and hierarchical architectures. Also, we investigate the use of *raw maps* (SHENG et al., 2005) to represent the map of robots.

Simulated experiments were performed to evaluate the proposed methods regarding the time, number of messages and amount of data necessary for sharing maps in different conditions. Experiments with Sheng's method were also performed and the results used to compare the performance of the methods.

Development of a method for multi-robots coordination

The second objective of the PhD is to propose a method to coordinate multiple robots in exploration tasks, considering robots with limited communication radius. The method must assign exploration targets to robots in order to minimize the time necessary to explore the entire workspace. Beside this, robots must be coordinated in order to quickly spread over the workspace and balancing the exploration workload fairly among them.

Specifically, robots must be assigned to explore areas with similar sizes. Also, the method must avoid that some robots travel much longer than the others. Regarding the objective of quickly spreading the robots, some strategies in which robots explore completely closer areas before moving to farther ones might generate maps quicker. However, when robots quickly spread over the workspace, they can find specific objects faster. In the context of search and rescue applications, for instance, potential victims in one region will not have to wait for assistance much longer than victims in regions close to the initial position of robots.

We investigate the partitioning of workspace in regions and the assignment of regions to robots as a strategy to quickly disperse robots through workspace and fairly distribute the workload among them. A hierarchical organization of robots is considered to handle communication problems and define leaders to coordinate groups of robots.

We assume the following premises: 1) the workspace boundaries are known; 2) robots communication system has a limited radius; 3) messages sent to robots within this radius are always received; 4) robots do not fail.

Several experiments are performed using the robot simulator Player/Stage (GERKEY; VAUGHAN; HOWARD, 2003) to verify if robots always complete (and how long it takes) a map of the workspace, how long it takes for each region to be reached by a robot and how fairly the workload was divided among robots. Instead of a comparison with other methods, we evaluate how much the performance of the method degrades when communication radius decreases. A baseline, corresponding to the execution of the method proposed in (PUIG; GARCÍA; WU, 2011) when communication is granted, is used to evaluate the performance of the method proposed in this thesis.

1.2 THESIS ORGANIZATION

This thesis is organized as follows. Chapter 2 introduces the multi-robots exploration problem and presents the main works on map

sharing and coordination of multi-robots systems in exploration tasks. Chapter 3 presents the methods we propose for map sharing. In chapter 4, we present experiments with the proposed methods for map sharing and discuss the results. Chapter 5 presents the method we propose for coordination in multi-robots exploration and chapter 6 the experiments and their results. Finally, chapter 7 presents the conclusions and perspectives of the work.

2 MULTI-ROBOTS EXPLORATION

In robotics, the exploration problem arises when robots have to construct a model of the environment in which they are placed. To do so, robots must move efficiently through the entire workspace in order to create a complete map of it. In next section, we describe the exploration problem, focusing in how robots can exchanged the workspace information they detected using their sensors and how they can be coordinated in order to execute exploration tasks efficiently.

Sections 2.3 and 2.4 present several works on the problems of map sharing and multi-robots coordination. The methods proposed in this thesis extend the works proposed by Sheng *et al.* (SHENG et al., 2006) and Puig *et al.* (PUIG; GARCÍA; WU, 2011) and we present these works in sections 2.3.1 and 2.4.3, respectively.

2.1 EXPLORATION PROBLEM

Exploration problem requires that robots move systematically through the workspace in order to create suitable environmental models or maps from the data collected by their sensors. Even when a single robots is used, exploration is a very complex problem, involving aspects as robot's localization and world features' detection through noisy sensor data (SIM; ROY, 2005).

It is natural to expect that several robots can explore an environment faster than a single robot. However, when multi-robots system are used in exploration, other aspects as robot's coordination and information sharing make the problem more complex (CARVALHO et al., 2013; FLOCCHINI et al., 2013).

The key question in multi-robots exploration is which place each robot must explore in order to minimize aspects as the time necessary to generate a complete map of the workspace. In this work, we define this problem as the *Coordination problem* (HAUMANN; WILLERT; LISTMANN, 2013).

Different centralized and distributed methods have been pro-

posed to coordinate robots in exploration tasks (BURGARD et al., 2000; BURGARD et al., 2005). In centralized methods, a central unit defines which places (named *exploration targets*) each robot will explore. In distributed approaches, robots interact with the others to choose the places they will explore.

In both approaches, coordination schemes consider several aspects, such as which areas of the workspace were already explored and the current position of robots, to select new targets for robots. To do so, robots must share the information they detect while exploring the workspace. Otherwise, robots can be assigned to explore targets in already explored areas, which can decrease significantly the efficiency of exploration.

In this thesis, we focus on the map sharing and coordination problems, so, exploration can be viewed as presented in figure 1.

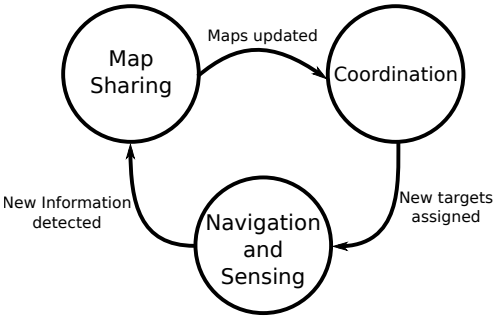


Figure 1 – Exploration diagram.

In figure 1, we represent that robots execute a cycle based on three states: *map sharing*, *coordination* and *navigation and sensing*. After robots detect new areas of the workspace, they share the new information with the other robots (state *map sharing*). Then, they are coordinated in order to define which place each robot is going to explore (state *coordination*). Next, robots start to move through the workspace, using their sensors to detect new areas of the workspace (state *navigation and sensing*).

Navigation and sensing involves aspects, such as robot's lo-

calization, navigation and fusion of sensor data and is commonly referred to as *Simultaneously Localization and Mapping* (SLAM) (MONTEMERLO; THRUN, 2003). There is a number of systems that can be reliably used for SLAM (DISSANAYAKE et al., 2001; FEI et al., 2013; SU, 2008; THRUN; LEONARD, 2008; KUO et al., 2011) and we do not address them in this thesis.

2.1.1 Map representation

There are several forms to represent the knowledge that robots have about the workspace, such as occupancy grid maps (ELFES, 1989), variable grid maps (KAPLOW; ATRASH; PINEAU, 2010), and road maps (HSU; LATOMBE; KURNIAWATI, 2006; KAVRAKI; LATOMBE, 1998). Occupancy grid is one of the most common approaches to represent the workspace and we present it in this thesis.

In a grid map, workspace is represented by a two-dimensional (or even three-dimensional) array of cells, as shown in figure 2. Cells correspond to discrete positions in the workspace and can be classified as: *occupied*, *free* or *unknown*.

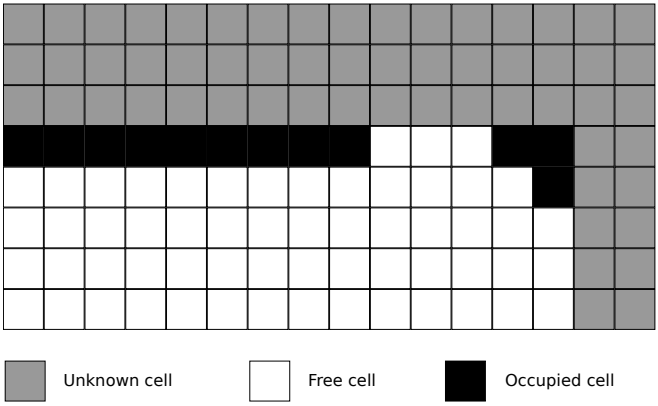


Figure 2 – Example of occupancy grid map.

To define the state of cells, robots employ an approach based on probabilistic models of their sensors (ELFES, 1989). Considering

models of sensors and the data collected by their own sensors or sent by other robots in the system, robots calculate the probability of each cell is occupied by an obstacle, *occupancy probability*. If the probability is higher than a threshold *OCC*, the cell is considered *occupied*. If it is smaller than a threshold *FREE*, the cell is considered *free*. Otherwise, the state of the cell is considered *unknown*.

How robots can perform data fusion in order to create a map of the workspace, reducing errors caused by sensor uncertainties, is defined the *mapping data fusion* problem. We do not address this problem and refer to (AHMED; SAMPLE; CAMPBELL, 2013; ELFES, 1989; KUBELKA et al., 2015; LUO; LAI, 2012; RODGER, 2012; SHALAL et al., 2015) and other works for more information about mapping data fusion.

2.2 COMMUNICATION IN DISTRIBUTED SOLUTIONS

In many situations, robots must perform an exploration task in a workspace where there is not a pre-existing network infrastructure (routers, access points, etc). Since the exchange of messages among robots is necessary to share map information and also for coordination, they have to act as routers and relay messages to the final destination. So, regarding the communication, MRS can be view as a mobile *ad hoc* network.

In mobile *ad hoc* networks, communication between two nodes (robots in our application) is performed by direct connection or through multiple hop relays, when there is not a direct link between them. If all nodes play the same role in the network, we say that the network has a *flat network architecture*.

In a flat network, when the number of nodes increases, routing schemes do not scale well in terms of performance. Also, problems related to message losses are harder to solve in flat networks (DHURANDHER; SINGH, 2005). Efficient solutions based on grouping nodes into clusters have been widely proposed by the research community to handle the scalability problem in networks

(ABBASI; YOUNIS, 2007). In these solutions, leaders (named clusterheads) are elected for each cluster and the networks have a *hierarchical architecture*. Due to the existence of leaders, problems as message losses become easier to detect and handle in hierarchical networks.

Several schemes have been proposed to group the robots in clusters and elect the clusterheads, such as the *Lowest ID* (EPHREMIDES; WIESELTHIER; BAKER, 1987), the *Highest Degree* (GERLA; TSAI, 1995) and the *Weighted Clustering* (CHATTERJEE; DAS; TURGUT, 2002).

In most methods, members of a cluster have a direct link with their clusterhead and can exchange messages only with it. The clusterheads handle the communication inside the cluster and with other clusters. Thus, networks can be represented just in the level of the clusters. Also, most of the changes in the network topology can be handled internally by the clusterheads. However, some application might use multi-hop communication inside the cluster due to specific requirements (ABBASI; YOUNIS, 2007).

In figure 3, we present an example of flat and hierarchical networks.

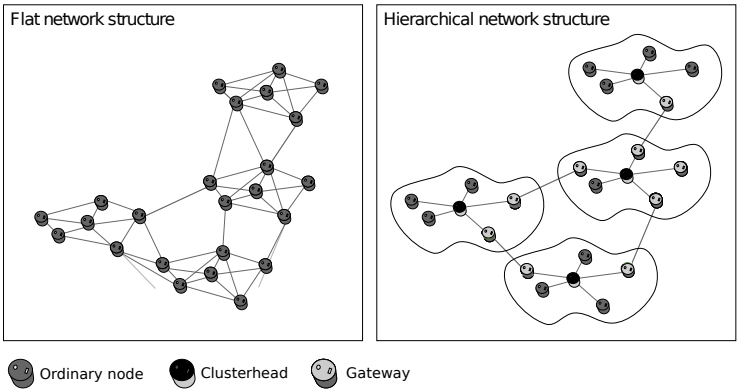


Figure 3 – Flat and hierarchical networks.

The light gray nodes in figure 3 represent the *gateways*, members of cluster that are used by their leaders to send and receive messages

from other clusters. Next, we present the simplest algorithm for cluster formation and maintenance: the lowest ID.

Lowest ID

In (EPHREMIDES; WIESELTHIER; BAKER, 1987), the authors propose a simple algorithm for cluster formation and maintenance based on the robots ID, a unique integer identifier that is associated with each robot. The cluster formation is performed as described below.

- Each node (robot in this work) broadcasts a message with its ID to its neighbors[1];
- If all messages that a robot receives are from robots with a higher ID, it becomes a clusterhead. As a clusterhead, it will add to its cluster all neighbors which request membership to its cluster;
- Otherwise, it identifies the neighbor with the lowest ID and tries to join its cluster. If the neighbor is not a clusterhead, the robot tries to join the cluster of its neighbor with the next smallest ID, and so on.

In the maintenance phase, which starts to be executed after the cluster formation phase, each node periodically broadcasts a message with its ID to its neighbors and verifies if it lost any link. When a node loses the link with its clusterhead it tries to join another cluster. If the node does not have a link with other clusterhead the cluster formation phase is executed again(CAMBRUZZI; FARINES; JUNIOR, 2009).

In the following sections, we present the main approaches for map sharing and multi-robots coordination in exploration tasks.

2.3 MAP SHARING

The simplest approach for map sharing, which is used by most multi-robots exploration methods, is described next. All robots can

[1] The neighbors of a node are the ones with what they have a direct communication link

communicate directly with a central unit (a robot leader or an operational base) and, every time they reach exploration targets, robots send the collected data to the central unit. After receives new information about the workspace, the central unit updates the map of the system and sends it to all robots.

In exploration methods as the ones proposed in (BURGARD et al., 2000; SIMMONS et al., 2000), for instance, robots rely on a central unit to generate the workspace's map and coordinate them. Every time a robot reaches an exploration target, it sends the information collected to the central unit, which updates the map, identifies new exploration targets and assigns them to the robots.

Other methods, such as (LAGOUDAKIS et al., 2004; YAMAUCHI, 1998), use a distributed scheme to share map information. In these methods, each robot has its own map of workspace and uses an *auxiliary map* to store the data it collects while moving to an exploration target. When the robot reaches the target, it updates its workspace map with the information in the *auxiliary map*. Then, it sends the *auxiliary map* to the other robots in the system. Next, the robot chooses another target and resets the *auxiliary map* before starting to move toward it.

These approaches (centralized and distributed) allow robots to share the information they detect about the workspace, avoiding the exchanging of unnecessary information (robots share only new information). However, they do not handle communication problems as link losses.

As robots share an information only once (when they detect it), if the central unit (in centralized approaches) or some robots (in distributed approaches) do not have a link with the robot that is sharing collected data, they will never get this piece of information, even if communication between them is reestablished. When robots have different maps, we have an *information inconsistency* problem.

To handle this problem, other authors use simple schemes in which robots share their entire maps. In (BURGARD et al., 2005;

STACHNISS; MOZOS; BURGARD, 2008), for instance, the authors propose a scheme where all robots that can communicate form a group and define a leader. Every time a robot reaches a target (named *frontiers*), it sends its entire map to the current leader, which updates its own map and share it with the other robots in the group. By doing so, robots guarantee that any information obtained before they had established communication with the current leader will be shared. Similarly, leaders share their entire maps to guarantee that robots get any information detected before they joined the group.

In distributed approaches as the ones proposed in (SARIEL; BALCH, 2005; SARIEL; BALCH, 2006; ZLOT et al., 2002), robots broadcast their entire map to all robots with what they can communicate. After receiving maps from another robot, robots update their own maps.

Although these approaches, where robots exchange their entire maps, can handle information inconsistency problems, they force robots to exchange a large amount on unnecessary data.

Other methods (FRANCHI et al., 2007; FRANCHI et al., 2009), propose schemes where, every time two robots establish a communication link, they exchange their entire maps. Then, while the robots can communicate, they exchange only the new information they detect about the workspace. Although this scheme reduces the amount of unnecessary information exchanged, when robots establish a link, they still send information that the others already have.

Sheng *et al.* (SHENG et al., 2005; SHENG et al., 2006) propose an efficient method for map sharing, defining a map representation that allows robots to easily identify which information others do not have yet and send only this information. In the following subsection we present Sheng's method.

2.3.1 Sheng's Method

In (SHENG et al., 2005; SHENG et al., 2006), Sheng *et al.* propose a method for multi-robots exploration, whose main contribution

is a scheme to synchronize the maps of robots. Authors do not assume any pre-existing network infrastructure (routers, access points, etc) in the workspace and robots can act as routers and relay messages to the final destination. So, regarding the communication, the MRS can be view as a mobile *ad hoc* network.

In Sheng's method, robots in the same network have the same map and, when a robot reaches its target and detects new information about the workspace, it shares this information with the others through a propagation scheme. Specifically, the robot broadcasts the new information to robots with what it has a direct communication link, its neighbors. Next, the neighbors update their maps and broadcast the information to their own neighbors, and so on. At the end, all robots in the network get the information and converge to the same map.

The main contribution in Sheng's method is a scheme to synchronize maps when robots in different networks establish a link and the networks merge in a single one.

Let's consider that robots in the system are separated in two unconnected networks $\mathcal{N}_1$ and $\mathcal{N}_2$. If two robots $R_a \in \mathcal{N}_1$ and $R_u \in \mathcal{N}_2$ establish a link, they interact to identify which information the other does not have yet. Next, the robots send only the information the other still lacks. After R_a and R_u update their own maps, they broadcast the information they got to their neighbors. Next, their neighbors broadcast the information to their own neighbors, and so on. At the end of the synchronization scheme, all robots in the new network converge to the same and most up to date map.

To allow robots to synchronize their maps without exchanging unnecessary information, Sheng *et al.* propose the concept of *raw map*. Every time a robot R_k reaches a frontier (the exploration target in Sheng's method), it defines a set ΔM_{kq} with the state of all cells it detected while moving toward the frontier and adds ΔM_{kq} to its *raw map*. The index q indicates that ΔM_{kq} corresponds to the set of cells detected by R_k while it was exploring its q-th frontier.

Each robot has its own *raw map*, which has ΔM_{kq} sets detected by itself and shared by other robots. Table 1 shows an example of *raw*

map for a robot R_k.

Table 1 – Example of the raw map of a robot R_k.

Raw map of a robot R_k					
R_1	R_2	$\cdots$	R_k	$\cdots$	R_n
ΔM_{11}	ΔM_{21}	$\cdots$	ΔM_{k1}	$\cdots$	ΔM_{n1}
ΔM_{12}	ΔM_{22}	$\cdots$	ΔM_{k2}	$\cdots$	ΔM_{n2}
ΔM_{13}	ΔM_{23}	$\cdots$	ΔM_{k3}	$\cdots$	ΔM_{n3}
.	.	$\cdots$	.	$\cdots$	.
.	ΔM_{2p}	$\cdots$	.	$\cdots$	.
ΔM_{1m}		$\cdots$	.	$\cdots$	.
		$\cdots$	.	$\cdots$	ΔM_{nu}
		$\cdots$	ΔM_{ks}	$\cdots$	

A *raw map* has n columns, each of them associated with a robot of the system. For example, in table 1, the column associated with robot R_2 has the sets $\Delta M_{21} \cdots \Delta M_{2p}$ generated by robot R_2 and shared with R_k.

To synchronize their *raw maps*, two robots send an array with the last index of each column in their *raw maps*. Based on the received array, each robot identifies what ΔM_{kq} sets (of all columns) the other does not have yet and sends only these information.

Generating Occupancy Grid Maps

Using *raw maps* to represent the workspace, robots can efficiently share map information. However, *raw maps* are not so useful when robots need to calculate paths to specific positions in the workspace.

To handle this problem, robots can convert their *raw maps* in occupancy grid maps, for which there are several methods that can be applied to calculate paths, such as $A^\star$ (HART; NILSSON; RAPHAEL, 1968) and *Value Iteration* (HOWARD, 1960; BURGARD et al., 2005). To do so, robots can create a two-dimensional array to represent the grid map. Then, based on the data in ΔM_{kq} sets from their *raw maps*, robots calculate the state of cells in the grid map.

2.4 COORDINATION IN MULTI-ROBOTS EXPLORATION

Over the last decades, a wide variety of methods to coordinate exploration tasks with multiple robots have been proposed. The simplest approach for multi-robot exploration was proposed by Yamauchi (YAMAUCHI, 1998), which defines *frontier cells* as the exploration targets. Frontiers are cells free of obstacles that have at least one non-explored (or *unknown*) adjacent cell. Each robot chooses its closest frontier cell regardless of the actions of the other robots. After exploring its frontier, the robot shares the detected cells with the other robots and chooses another frontier cell. This method implements a greedy strategy with no coordination among robots. It also assumes direct communication among all robots.

Because of the lack of coordination among robots, they can explore the same frontier simultaneously, which can significantly decrease the efficiency of exploration. Furthermore, this method does not consider any dispersion of robots. Thus, they may concentrate their exploration in some parts of the workspace. In that case, robots have to move close to each other, constantly recalculating their paths to avoid collision. In addition, when robots concentrate exploration in some region, they tend to take longer to generate a complete map of the whole workspace.

Several approaches have been proposed to handle these problems. They can be classified based on their optimization policies as: robot assignment policy, task assignment policy and region assignment policy.

Methods that consider the robot assignment policy allocate an exploration target (usually frontier cells) to each robot in order to avoid that robots explore the same target or too close targets. The objective is to assign a task (target to explore) to each robot. When there are more targets than robots, some targets can be left without an assigned robot.

In methods that consider task assignment policies, the goal is to allocate all exploration targets among the robots, minimizing the time

or distance necessary to explore all targets. In these methods, robots can be assigned to more than one target when there are more targets than robots.

At the end of this section, the method proposed by Puig *et al.* (PUIG; GARCÍA; WU, 2011), which considers a global exploration policy based on the assignment of regions to the robots, is summarized. In (PUIG; GARCÍA; WU, 2011), the workspace is partitioned in regions, and the regions and frontiers are allocated in order to disperse the robots over the entire workspace, exploring it efficiently and balancing the exploration workload among all robots. Next, we present the main methods for multi-robot exploration, organizing them according to their optimization policies.

2.4.1 Robot Assignment Policies

The method proposed by Burgard *et al.* (BURGARD et al., 2000; BURGARD et al., 2005) coordinates the robots of an MRS in order to explore the entire workspace as fast as possible. This is done through a centralized scheme that considers frontiers as targets and associates costs and utilities to them. Frontiers are assigned by a central unit that sequentially chooses targets for robots with the best trade-off between utility and cost, and updates the utility values based on the proximity to assigned frontiers.

The authors also describe how the proposed scheme can be used when robots have a limited communication radius. In that case, each group of connected robots executes the algorithm independently. Nevertheless, the authors do not discuss the loss of efficiency caused by the local execution of the centralized algorithm or other issues related to the execution in the context of several *ad-hoc* unconnected networks.

In (STACHNISS; MOZOS; BURGARD, 2006; STACHNISS; MOZOS; BURGARD, 2008), the authors propose an extension of the previous method that classifies the areas of the workspace into rooms or corridors, and prioritizes the exploration of frontiers in corridors through the assignment of higher utility values.

In (SIMMONS et al., 2000), the authors propose a method the improves the one presented in (BURGARD et al., 2000; BURGARD et al., 2005) in two ways: considers estimated the amount of data that will be detected exploring a frontier cell to assign frontiers; and uses a bidding scheme to execute the algorithm in a distributed way, decreasing the computational cost for the central unit.

Instead of defining the initial utility as 1, the method proposed by Simmons *et al.* (SIMMONS et al., 2000) defines it based on the amount of cells with "unknown" state that will be detected exploring the frontier. The robots calculate the cost to reach the frontiers and the utilities of frontiers to define the "bids" and send the bids to a central unit responsible to allocate the frontiers. Iteratively, the central unit identifies the highest bid and allocate the frontier to the robot. Then, it reduces the utilities of frontiers based on the overlap with the area of the last assigned frontier.

As happens in (BURGARD et al., 2000; BURGARD et al., 2005), the proposed method implements a greedy strategy instead of a scheme that optimizes the overall assignment performance, although tends to disperse, locally, the robots over the environment. The paper does not address any communication problems.

Fox *et al.* (FOX et al., 2006) propose a method that approaches the exploration problem considering that the robots do not know the other robots relative position. To be able to merge the information collected by the robots in one map, the robots exchange the collected data and define a hypothesis about the other robot position. Robots verify the hypothesis using a rendezvous technique and, if it is true, they form a *group* with the robot with smaller ID as the leader. Otherwise, they keep exchange information to refine the hypothesis.

In (FOX et al., 2006), the authors define two types of targets: frontiers cells and position hypothesis. A decision-theoretic scheme that assigns cost and utility for the targets, similar to the one proposed in (BURGARD et al., 2000; BURGARD et al., 2005). The costs are length of the smallest path between the robot and the frontier or the rendezvous position. The utility is defined as the frontiers associated

area and, in the case of the hypothesis targets, the amount of data in the other robot map. A linear program solver is used to find the optimal assignment, which maximizes the targets' trade off. However, it does not force the robots dispersion over the environment.

As occurs in (BURGARD et al., 2000; BURGARD et al., 2005), the proposed method can be executed in centralized way by the leaders of each robots *connected group* to handle limited communication.

In turn, Franchi *et al.* (FRANCHI et al., 2007; FRANCHI et al., 2009) propose a method for multi-robot exploration in which each robot explores the workspace almost regardless of the actions of the other robots. Each robot has a map, represented as a *Sensor-based Random Graph* (SRG), whose nodes correspond to the positions of explored frontiers and whose root is the initial position of the robot. Every time the robot explores a new frontier, it creates a new node and defines an edge connecting that node with the one from where it started exploring that frontier.

Robots select their exploration targets (frontiers) using a biased random policy that assigns higher probabilities to targets with larger areas and closer to the robot's position. Robots also cooperate by exchanging the information they detect about the environment every time they can communicate with each other. When some robots are within a predefined distance, they form a group and select a leader to coordinate their actions in order to avoid collisions. Moreover, to decrease the distance among nodes, the SRG map is enriched with new edges, named *bridges*.

Targets are selected using a biased random scheme with no explicit mechanism to force the robots to disperse through the environment. However, there is a local mechanism that coordinates the robots to avoid collision when they are too close to each other. Another important feature is that the method can handle communication problems caused by limited communication radius, such as link losses.

In (WURM; STACHNISS; BURGARD, 2008), Wurm *et al.* propose a method where the workspace map is segmented and represented as a Voronoi diagram. Figure 4 presents an example of Voronoi diagram

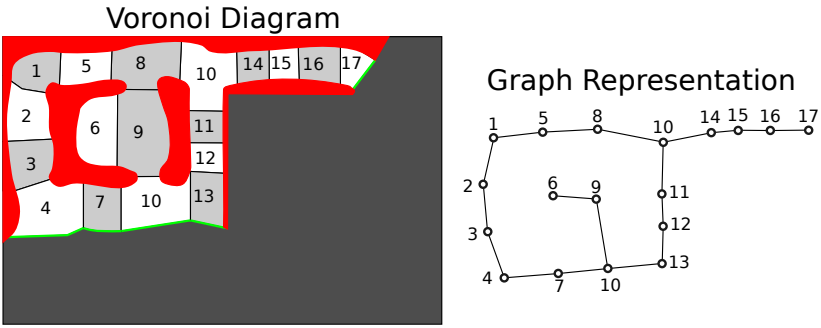


Figure 4 – Voronoi diagram and graph representation.

for a partially explored workspace and its corresponding graph-based representation. Dark gray indicates the non-explored areas, whereas obstacles are shown in red. The green lines represent frontier cells. For the example presented in figure 4, segments 4, 7, 10, 13 and 17 are the only ones that have frontier cells and correspond to the exploration targets.

The method proposed in (WURM; STACHNISS; BURGARD, 2008) coordinates the robots to avoid that they explore the same (or close) frontiers simultaneously, thus reducing the risk of collision and the amount of redundant explored area. Particularly, a central unit calculates the costs C_s^i of exploring a segment s with a robot i, for all robots and segments with frontier cells. C_s^i is the length of the smallest path between robot i and the closest frontier cell in segment s. The central unit then uses the Hungarian Method (KUHN, 1955) to assign a robot to each segment. In (WURM; STACHNISS; BURGARD, 2008), several robots can be assigned to the same segment when there are more robots than segments with frontier cells. As the authors define a central unit to allocate exploration targets to robots, that method cannot handle limited communication, which could prevent the communication between robots and the central unit depending on the distance among them.

In general, methods that consider robot assignment policies allocate a single exploration target (frontier, segment, etc.) to each robot

in a way that avoids the exploration of a target by several robots simultaneously, which could decrease the exploration efficiency.

Despite robots tend to disperse over the workspace, since the aforementioned methods only consider the identified exploration targets, the degree of dispersion is local. Another problem is that those methods consider the assignment of a single target to each robot and cannot minimize the exploration of all identified targets.

2.4.2 Target Assignment Policies

To improve exploration efficiency, some methods consider the assignment of all targets to the robots. By doing so, they avoid problems caused by the greedy behavior of methods that only consider the assignment of one target per robot.

Zlot *et al.* (ZLOT et al., 2002) propose a distributed scheme for MRS exploration based on a market architecture. Instead of using frontiers cells, in the method proposed in (ZLOT et al., 2002), the exploration targets are cells with unknown state that can be chosen by three different schemes: random; greedy exploration, which chooses a point centered in the closest non-explored region (of a fixed size); space division by quadtree, in which the unknown cells are represented using a quadtree and targets are located at the center of the quadtree leafs. Also, the method tries to improve globally the efficiency of the exploration by considering future targets in the assignment.

Each robot detects its own targets and tries to sell all of them sequentially (one auction at a time) to the robots it can communicate. The owner calculates the minimum price that it will accept for each target based on the distance to reach the target (cost) and the information gain expected (revenue), similar to (SIMMONS et al., 2000). The other robots also calculate their bids based on the cost and the revenue and send to the auctioneer (the target owner). If the auctioneer receive a bid greater than the minimum price, it "sells" the target to the robot with the higher bid.

After the robot tries to sell all targets, it inserts all remain targets

in its tour (sequence of targets to be visited by the robot) greedily minimizing the distance between the last target in the tour and the remaining targets. Then, it starts to move to the first target in the tour. When the robot reach a target, it detects new targets, adds them in its targets set and start a new auctioning process.

In (BERTSEKAS, 1990), Bertsekas showed that his auction algorithm can effectively find an optimal solution that maximizes the total benefit (utility-cost). Although the market economy method is based on the above described optimization of total benefit, it has two problems. There is no mechanism that forces the robots to disperse through the workspace. Second, some robots will obtain more targets since they are closer to them (their costs are lower than those of the robots which are farther away from their goals). Consequently, the workload is not balanced among the robots.

Also, the auctions consider only the distance between the targets and the robots, not the entire tour, and the size of the targets. Thus, this approach will behave like the greedy strategy proposed by Yamauchi (YAMAUCHI, 1998), in which robots always choose the closest targets.

In (FAIGL; KULICH; PŘEUČIL, 2012; FAIGL; KULICH, 2015), Faigl *et al.* proposes a centralized method for multi-robot exploration based on *goals clustering*. In the proposed method, a central unit synchronizes the maps of all robots and identifies the exploration targets (similar to frontiers). Then, instead of assign a target to each robot, it aggregates the goals in K clusters, where K is the number of robots in the system. To do so, a K-Means based algorithm is used (ASGHARBEYGI; MALEKI, 2008).

Instead of clustering the frontiers in K groups and, then, assigning robots to each of them, the method proposed in (FAIGL; KULICH; PŘEUČIL, 2012; FAIGL; KULICH, 2015) defines the position of the robots as the seeds[2] of the clusters, thus forming clusters already in the vicinity of the robots that will explore them.

[2] The seed of the frontier cluster is its initial center of mass.

Then, each robot identifies the closest target in its cluster, calculates a path to it and starts to move toward it.

Although this method minimizes the distance traveled by robots to reach their clusters of goals, it uses a simple greedy strategy to explore the goals in the clusters. Thus, exploration can become inefficient. In addition, it can find solutions where some robots do not get any target, leading to an unbalanced exploration. In that case, the method states that the robots must move toward the closest non-explored area, which can lead to a robot "following" the one that was assign to goals in that area. The method also assumes that all robots can communicate with the central unit to allocate targets and cannot handle link losses.

In (CARVALHO et al., 2013), Carvalho *et al.* proposes a method for multi-robot exploration that considers a partially explored workspace where there is a number of targets. Some of the targets are known while others are in the non-explored areas of the workspace. Thus, robots have to perform two types of tasks: exploring known targets and *wandering* through the workspace in order to detect the remaining targets.

The method proposes a scheme where all known targets must be assigned to robots first. Then, the remaining robots of the system wander through the workspace in order to find the targets in non-explored areas. To assign robots to targets, robots execute a distributed scheme based on *graph coloring* (KUBALE et al., 2002), where targets are defined as nodes of a graph and robots as colors. Robots execute the distributed graph coloring algorithm proposed in (KUBALE et al., 2002) to paint all nodes of the graph, defining which nodes each of them will explore. The algorithm minimizes the number of colors necessary to efficiently paint the entire graph, in order to leave as many robots as possible to wander through the workspace looking for targets in non-explored areas.

The method proposed in (CARVALHO et al., 2013) minimizes the time necessary to explore all known targets in a partially explored workspace, also minimizing the number of robots. However, Carvalho *et al.* do not present a contribution for the problem of exploring the

unknown workspace, defining a random mechanism to explore it using the remaining robots.

In (BERHAULT et al., 2003), the authors propose a distributed scheme based in market economy to explore a number of given targets in a partially unknown environment. Instead of considering single-item auctions, Berhault *et al.* propose a scheme where the robots bid on *bundles* of targets and an auctioneer (a virtual agent) determines the winners.

Different schemes are proposed to group the targets in the bundles. The results presented in the paper show that the Graph-Cut strategy is the one with the closest to optimal traveling cost. The Graph-Cut strategy defines a initial bundle that contains all targets and, recursively, create additional bundles using a maximum cut algorithm to split the graph.

Since the environment on which the experiments are carried out was only a partially unknown terrain and the number of robots was very small (only 3 robots), some conclusions in the paper need to be further examined. The method considers only the exploration of predefined targets and robots dispersion over the environment is not addressed. Also, the method centralizes the decisions in the *auctioneer* and the authors do not address problems related to communication link losses.

In (CAVALCANTE; NORONHA; CHAIMOWICZ, 2013), Cavalcante *et al.* improved the scheme proposed in (BERHAULT et al., 2003) by defining three strategies that improve combinatorial auctions for multi-robots task. Two strategies are based on packages-tree and a heuristic for the *Traveling Salesman Problem* (TSP). The third is based on a sorting scheme. However, the problems in (BERHAULT et al., 2003) are still present in (CAVALCANTE; NORONHA; CHAIMOWICZ, 2013).

Lagoudakis *et al.* (LAGOUDAKIS et al., 2004), for instance, propose a method to allocate a number of exploration tasks (or targets) to a multi-robot system. This method is based on the concept of *Minimum Spanning Tree* (MST) and its goal is to allocate the targets to the MRS minimizing the total traveled distance.

The exploration problem is represented by a weighted undirected graph G, whose vertices correspond to the location of targets and robots. Every edge keeps the cost of moving from one place to the other. A subgraph of G that contains all vertices and whose edges have the minimum cost is an MST.

An MST is defined for each robot through the *Prim Allocation* algorithm. The MST's root keeps the location of its corresponding robot. At each iteration, the robots bid on the non-assigned vertices that are closest to their MSTs. A central auctioneer decides the winning bids.

Although this method minimizes the total cost of edges, it does not consider the cost to jump from one branch of the MST to another. Thus, exploration can become inefficient. In addition, it can find solutions where some robots do not get any target, leading to an unbalanced exploration. The method also assumes that all robots can communicate with the central unit to allocate targets. Thus, this method cannot handle limited communication radius.

Sariel and Balch (SARIEL; BALCH, 2005; SARIEL; BALCH, 2006) also propose a method to allocate a number of exploration targets based on the concepts of MST and *Minimum Spanning Forest*[3] (MSF). However, instead of a combinatorial market-based scheme, single-item auctions are used to allocate targets. To avoid greedy behaviors, the authors propose schemes to calculate bids based on near optimal solutions for the TSP. However, this method is not efficient enough and the target sets can be very unbalanced, with some robots owning many targets while others just a few (PUIG; GARCÍA; WU, 2011). An important feature of this method is that it defines some mechanisms to handle communication link losses.

In (ROGERS; NIETO-GRANDA; CHRISTENSEN, 2013; NIETO-GRANDA; ROGERS; CHRISTENSEN, 2014), Nieto-Granda *et al.* define the exploration targets as clusters of frontier cells and propose a method for multi-robot exploration that allocates teams

[3] A Minimum Spanning Forest is a set of MSTs

of robots to explore each target. To allocate targets to teams, the method implements a greedy exploration strategy similar to the one proposed in (BURGARD et al., 2000), where the pairs ⟨team-target⟩ with the shortest distance are assigned first. The main contribution of the paper is the set of three coordination strategies for team formation: *reserve*, *divide and conquer* and *buddy system*.

In the first strategy, unassigned robots are called from *reserve* every time a new branch of the workspace (a new room, for instance) is discovered. In the *divide and conquer* strategy, the exploration starts with all robots belonging to a single team and, every time a new branch is discovered, the team splits. Finally, in the *buddy system* strategy, teams of two robots are assigned to explore the targets, splitting whenever a new branch of the workspace is detected. When a team has a single robot, it recruits a new team of two robots from the reserve to explore new branches. All targets are allocated among the teams and, when a team does not have enough robots to explore all targets that it owns, it sequentially explores the closest ones.

Regardless of the strategy being used, the method exhibits a greedy behavior and runs similarly to the one proposed in (BURGARD et al., 2000), assigning the robots to the closest targets. In addition, the coordination strategies only improve the exploration by decreasing the risk of collisions (*reserve* and *buddy system* strategies) or the exploration time by preventing idle robots (*divide and conquer* strategy). Moreover, this method does not address communication issues.

Methods that consider target assignment policies can explore the workspace quicker than the ones that only consider robot assignment policies. This occurs because, when methods assign the targets considering the exploration of all them, robots can calculate the smallest paths that take them through all targets assigned to them. However, these methods still consider just local optimization. Since robots exploring the workspace have only a partial map of it, even when the methods minimize the exploration of all identified targets, they cannot guarantee the optimization of the exploration as a whole.

2.4.3 Global Optimization: K-Means for multi-robot exploration

In (PUIG; GARCÍA; WU, 2011), Puig *et al.* propose a multi-robot exploration method based on centralized K-means (KME), which implements a policy that optimizes the exploration at a global level. The method improves the exploration efficiency by minimizing three aspects: the sum of traveled distances, the variance of the length of paths and the variance of the arrival times at all regions of the workspace. In the context of search and rescue applications, for instance, if all regions can be explored with similar arrival times, potential victims in one region will not have to wait for assistance much longer than victims in other regions (PUIG; GARCÍA; WU, 2011).

To minimize these three aspects and improve the exploration, the authors propose a centralized scheme with two stages: workspace partitioning and assignment, and frontier allocation. The workspace partitioning and assignment stage is periodically run by a central unit. First, K-means is applied to segment the non-explored areas of the workspace into K regions, where K is the number of robots. Then, KME assigns a region to each robot in a way that minimizes the distance that all robots have to travel to reach their assigned regions.

The central unit executes the frontier allocation stage every time a robot reaches its last assigned frontier. The new frontiers are selected in such a way that robots tend to get closer to their corresponding regions. Thus, regions can be viewed as global exploration targets and KME uses them to guide the exploration process. By assigning frontiers to robots considering these global targets, KME avoids the greedy behavior of previous methods, which only consider the identified frontiers or other local aspects.

KME implements a global optimization strategy to avoid problems of unbalanced exploration that occur in previous works, such as (SIMMONS et al., 2000; BERHAULT et al., 2003; LAGOUDAKIS et al., 2004; BURGARD et al., 2005; SARIEL; BALCH, 2006; STACHNISS; MOZOS; BURGARD, 2008; VISSER; SLAMET, 2008). However, it assumes that robots can

Table 2 – Comparison of different exploration approaches.

Optimization Policy	Approach	Target	Coordination	Communication	Obj.
none	Yamauchi	frontier cells	none	guaranteed	*a*
Robot Assignment	Burgard *et al.*	frontier cells	centralized	limited	*a*
	Simmons *et al.*	frontier cells	centralized	guaranteed	*a*
	Fox *et al.*	frontier cells and rendezvous points	centralized	guaranteed	*a*
	Franchi *et al.*	frontiers	centralized	limited	*a*
	Wurm *et al.*	segments	centralized	guaranteed	*b*
Target Assignment	Zlot *et al.*	points in the unknown space	distributed	limited	*a*
	Faigl *et al.*	clusters of goals	centralized	guaranteed	*b*
	Carvalho *et al.*	targets	distributed	guaranteed	*a*
	Berhault *et al.*	predefined points	distributed	guaranteed	*a*
	Cavalcante *et al.*	predefined points	distributed	guaranteed	*a*
	Lagoudakis *et al.*	predefined targets	distributed	guaranteed	*b*
	Sariel and Balch	predefined targets	distributed	limited	*b*
	Nieto-Granda *et al.*	exploration targets	centralized	guaranteed	*a*
Global Optimization	Puig *et al.*	frontier cells	centralized	guaranteed	*c*
	HKME (proposed method)	frontier cells	distributed	limited	*c*

always communicate directly with the central unit. Thus, KME cannot handle limited communication.

Table 2 summarizes the main features of all methods reviewed in this section. The different exploration approaches are compared with respect to four features: exploration target, coordination scheme, communication and objectives. The communication feature indicates if the method handles limited communication or assumes guaranteed communication among robots. The objectives are: (a) minimize completion time, (b) minimize the total path lengths traveled by a team of robots and (c) minimize the variance of regional waiting time.

Most methods only consider local aspects to coordinate the robots, such as frontiers in the partially known map. By doing so, robots tend to explore the workspace greedily, which can decrease efficiency and lead to problems of unbalanced exploration. On other hand, KME can balance the workload among the robots and perform exploration efficiently, but cannot handle communication link losses.

2.5 CONCLUSIONS

This thesis approaches the problem of multi-robots exploration, focusing on how robots can share the information they detect and how they can be coordinated in order to explore the workspace efficiently.

Regarding the map sharing problem, several authors propose schemes (BURGARD et al., 2000; SIMMONS et al., 2000; STACHNISS; MOZOS; BURGARD, 2008) where robots broadcast their entire maps, exchanging a large amount of unnecessary data. Other authors (YAMAUCHI, 1998; LAGOUDAKIS et al., 2004; FRANCHI et al., 2009) propose schemes in which robots send only new information, but these schemes rely on perfect communication channels.

Sheng *et al.* (SHENG et al., 2005; SHENG et al., 2006) propose an efficient method for map sharing, which allows robots to synchronize their maps without exchanging unnecessary information. However, the method uses a propagation scheme to share information with other robots in the network, where robots broadcast any new information to all neighbors. As robots can have links with several robots in the network, it can receive the same information many times, increasing the amount of unnecessary exchanged data.

In chapter 3, we propose two methods for map sharing based on Sheng's *raw map* that avoids robots exchanging unnecessary map information.

Regarding the coordination problem, most methods consider only local aspects to coordinate the robots, such as frontiers in the partially known map. By doing so, robots tend to explore the workspace greedily, which can decrease efficiency and lead to problems of unbalanced exploration.

The KME (PUIG; GARCÍA; WU, 2011) implements a global optimization strategy to avoid problems of unbalanced exploration that occur in previous works, such as (SIMMONS et al., 2000; BERHAULT et al., 2003; LAGOUDAKIS et al., 2004; BURGARD et al., 2005; SARIEL; BALCH, 2006; STACHNISS; MOZOS; BURGARD, 2008; VISSER; SLAMET,

2008). However, in (PUIG; GARCÍA; WU, 2011), the authors assume that robots can always communicate directly with the central unit and cannot handle problems associated with link losses.

In chapter 5, we propose the Hierarchical K-Means (HKME) method for multi-robots exploration, an extension of the KME that handles situations where, due to the limited communication radius, the existence of a communication link between two robots depends on the distance between them.

3 PROPOSED METHODS FOR MAP SHARING

3.1 PROBLEM DESCRIPTION

Map sharing is an important part of multi-robots exploration. If robots do not share the information they collect while exploring the workspace, other robots might go to places already explored. Moreover, when robots have more information about the workspace, they can calculate better paths to reach their exploration targets, avoiding obstacles already detected by other robots.

The work presented in this thesis does not assume any pre-existing network infrastructure in the workspace. In addition, we consider that robots have a limited communication radius, so, two robots only have a direct communication link if the distance between them is smaller than the radius.

Depending on how the robots are scattered over the workspace, they can even be separated in several unconnected network. So, how can the robots share the information they collected in order to guarantee that at least robots in the same network have the same map?

In this chapter, two map sharing methods for multi-robots exploration under limited communication are proposed: Distributed Synchronization Method (DSM) and Hierarchical Synchronization Method (HSM). Both methods are based on the *raw map* concept proposed in (SHENG et al., 2006) and can be used to share maps efficiently, in terms of time, number of exchanged messages and transmitted data. In the following sections, DSM and HSM are presented. Chapter 4 presents the result of experiments with DSM, HSM and Sheng's method.

3.2 DISTRIBUTED SYNCHRONIZATION METHOD

In order to allow robots to exchange messages and share map information (state of cells detected while exploring a frontier) in a scalable way, we propose a method based on a *flat network architecture*, the Distributed Synchronization Method. In mobile *ad hoc* networks, com-

munication between two nodes (robots in our application) is performed by a direct connection or through multiple hops relays, when there is not a direct link between them. If all nodes play the same role in the network, we say that the network has a flat network architecture.

DSM does not assume that robots know the network topology and is based on three aspects: *Local Application View* (LAV); *Sharing Processes*; and *Processes Coordinator Algorithm* (PCA). The LAV is a structure that robots use to store their maps, which include a representation of the static environment and information about the positions and frontiers assigned to each robot in the system.

Sharing processes are the mechanisms that robots use to share LAV information with their neighbors. Because robots can participate of several sharing processes at the same time, we propose PCA to coordinate their execution. Next, an example is presented to illustrate an execution of DSM. Then, each part of DSM is described.

3.2.1 DSM Illustrative Example

To illustrate how DSM works, we present an example where 14 robots are separated in two networks $\mathcal{N}_1$ and $\mathcal{N}_2$, shown in figure 5. In the example, we consider that robot R_7 finished exploring a frontier.

While explores a frontier, a robot (R_7 in this example) detects the state of adjacent cells and, after reaches the frontier's position, the robot has to share the information it collected about the environment.

To do so, R_7 creates a sharing process (sharing processes are defined in subsection 3.2.4) of which its neighbors participate (figure 5.2). At the end of R_7's sharing process, all neighbors obtain the information and create their own sharing processes to share the new information with their neighbors (figure 5.3), and so on. Thus, information propagates over the entire network and, after all robots in the network get the new information (figures 5.4), they do not create sharing processes anymore and the execution of DSM ends.

Using this propagation scheme, robots do not need to know the network topology, only their own neighbors. By doing so, DSM

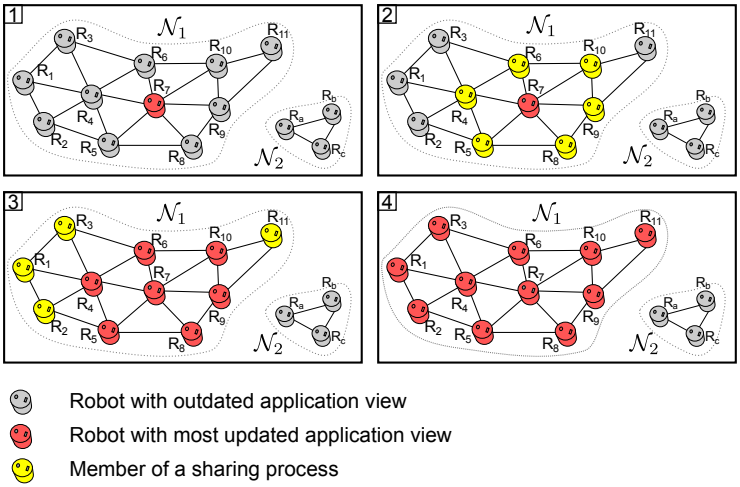


Figure 5 – Example of a DSM's execution.

does not have to handle problems such as network formation and management and message routing, which can become complex in robotics applications, where the connections can change quickly as the robots move over the workspace (CHATTERJEE; DAS; TURGUT, 2002; DHURANDHER; SINGH, 2005).

As illustrated in figure 5, R_7 cannot communicate with robots in $\mathcal{N}_2$ and they will not get the new information that R_7 is sharing. However, if in the future, a robot in $\mathcal{N}_2$ establishes a link with a robot in $\mathcal{N}_1$, the networks merge and, in the next DSM's execution, the map of all robots will converge.

The following sections describe the structure that robots use to represent their knowledge about the workspace (the Local Application View - LAV), how a sharing process is executed and the Processes Coordinator Algorithm (PCA).

3.2.2 Local Application View Concept

Local Application View is the structure that robots use to represent their knowledge about the workspace. A robot's LAV has three types of information: *raw map*, which represents the static workspace;

the last frontier assigned to each robot; and the last known position of each robot.

Both DSM and HSM use the raw map concept proposed by Sheng *et al.* (SHENG et al., 2005; SHENG et al., 2006) to decrease the amount of data exchanged to share maps, guaranteeing information consistency. We highlight that the raw map can be easily converted into an occupancy grid map (further details are presented in section 2.3). Table 3 shows an example of raw map for a robot R_k.

Table 3 – Example of the raw map of a robot R_k.

Raw map of a robot R_k					
R_1	R_2	$\cdots$	R_k	$\cdots$	R_n
ΔM_{11}	ΔM_{21}	$\cdots$	ΔM_{k1}	$\cdots$	ΔM_{n1}
ΔM_{12}	ΔM_{22}	$\cdots$	ΔM_{k2}	$\cdots$	ΔM_{n2}
ΔM_{13}	ΔM_{23}	$\cdots$	ΔM_{k3}	$\cdots$	ΔM_{n3}
.	.	$\cdots$	.	$\cdots$	.
.	ΔM_{2p}	$\cdots$	.	$\cdots$	.
ΔM_{1m}		$\cdots$	.	$\cdots$	.
		$\cdots$	.	$\cdots$	ΔM_{nu}
		$\cdots$	ΔM_{ks}	$\cdots$	

The raw map of each robot has n columns, each of them associated with a particular robot of the system. For instance, in table 3, the column associated with robot R_j has all sets of cells (ΔM_{jx}) detected by R_j and shared with R_k.

Definition 1. *A set ΔM_{jx} is defined as a set containing the information (position and occupancy probability) about all cells detected by R_j while it was exploring its x-th frontier.*

In addition to Sheng's raw map, the LAV of a robot has the sets of assigned frontiers and position of robots and a *tag*. Table 4 presents the elements of a robot R_k's LAV, except the raw map (already shown in the table 3).

Line *frontiers*, in table 4, represents R_k's knowledge about the frontier that each robot in the system is exploring. Similarly, line *positions* represents R_k's knowledge about the position of all robots. When

Table 4 – Example of the local application view of a robot R_k

LAV of a robot R_k						
	R_1	R_2	$\cdots$	R_k	$\cdots$	R_n
Frontiers	f_{R_1}	f_{R_2}	$\cdots$	f_{R_k}	$\cdots$	f_{R_n}
Positions	(x_{R_1}, y_{R_1})	(x_{R_2}, y_{R_2})	$\cdots$	(x_{R_k}, y_{R_k})	$\cdots$	(x_{R_n}, y_{R_n})
Tag	q_{R_1}	q_{R_2}	$\cdots$	q_{R_k}	$\cdots$	q_{R_n}

robots are separated in several networks, the information that R_k has about robots in other networks can be outdated. Specifically, robots can have ΔM_{jx} sets that R_k does not have and can be in other positions and exploring other frontiers.

The last line in table 4 defines the LAV tag, which is based on the last indexes of the raw map columns (table 3). The tag indicates how much the information that R_k has about other robots is up to date. Considering the raw map presented in table 3, R_k's tag has $q_{R_1} = m$, $q_{R_2} = p$ and so on.

3.2.3 Relations and Operations on LAVs

In this work, some relations and operations on LAVs are defined and used in both DSM and HSM to synchronize the LAVs of the robots. The relations are: *equivalence*($\equiv$) and *contains* ($\sqsupseteq$). The operations are: *difference* ($\ominus$) and *addition* ($\oplus$). In addition, we define attribute $\mathbf{LAV}(R_i)$ to represent the LAV of a robot R_i.

Equivalence ($\equiv$)

If the LAVs of two robots R_i and R_j have the same information, we say that they are *equivalent*, $\mathbf{LAV}(R_i) \equiv \mathbf{LAV}(R_j)$. To verify if the LAV of two robots are equivalent, one just need to compare their LAV tags. If they are equal, the LAVs are equivalent $(\mathbf{LAV}(R_i) \equiv \mathbf{LAV}(R_j))$. Otherwise, the LAVs of the robots are different $(\mathbf{LAV}(R_i) \neq \mathbf{LAV}(R_j))$.

Difference ($\ominus$)

The difference between the LAVs of two robots is the set with the information that the first robot has and the other does not have.

Let $\mathbf{LAV}(R_a)$ be the LAV of a robot R_a and $\mathbf{LAV}(R_b)$ be the LAV of R_b. The operation $\mathbf{LAV}(R_a) \ominus \mathbf{LAV}(R_b)$ results in the set of information that R_a has and R_b does not have (or has an outdated version). For simplicity, let's define the difference between the LAVs of a robot R_a and another robot R_b as $R_a \ominus R_b$.

Let $\langle q_{a_1}, q_{a_2}, \cdots, q_{a_n} \rangle$ be the tag of R_a's LAV and $\langle q_{b_1}, q_{b_2}, \cdots, q_{b_n} \rangle$ the tag of R_b. The difference $R_a \ominus R_b$ is defined as:

$$R_a \ominus R_b = \left\{ \begin{array}{lcl} \Delta M_{ij} \in \mathbf{LAV}(R_a) & | & (\forall R_i \in S)(\forall j \leq q_{a_i}),\ (j > q_{b_i}) \\ f_{R_i} \in \mathbf{LAV}(R_a) & | & (\forall R_i \in S),\ (q_{a_i} > q_{b_i}) \\ P_{R_i} \in \mathbf{LAV}(R_a) & | & (\forall R_i \in S),\ (q_{a_i} > q_{b_i}) \end{array} \right\} \tag{3.1}$$

where S is the set of all robots of the system and $P_{R_i} = (x_{R_i}, y_{R_i})$.

Notice, in equation 3.1, that the difference $R_a \ominus R_b$ results in the set of ΔM_{ij} such that $\Delta M_{ij} \in \mathbf{LAV}(R_a)$ and $\Delta M_{ij} \notin \mathbf{LAV}(R_b)$. $R_a \ominus R_b$ also has the set of assigned frontiers (f_{R_i}) and positions (P_{R_i}) from $\mathbf{LAV}(R_a)$, such that $q_{a_i} > q_{b_i}$. That is, the frontiers and positions that are more up to date in R_a's LAV than in R_b's. If R_b already has all information that R_a has, then $R_a \ominus R_b = \varnothing$.

Addition ($\oplus$)

The addition between the LAVs of two robots R_a and R_b corresponds to a LAV $R_a \oplus R_b$ with all information that R_a or R_b have. Let $\langle q_{a_1}, q_{a_2}, \cdots, q_{a_n} \rangle$ be the tag of R_a's LAV and $\langle q_{b_1}, q_{b_2}, \cdots, q_{b_n} \rangle$ be the tag of R_b. The addition $R_a \oplus R_b$ is defined as:

$$R_a \oplus R_b = \left\{ \begin{array}{lcl} \Delta M_{ij} & | & (\forall R_i \in S)\big(\forall j \leq \mathbf{max}(q_{a_i}, q_{b_i})\big),\ \Delta M_{ij} \in \mathbf{LAV}(R_a) \\ & & \quad or\ \Delta M_{ij} \in \mathbf{LAV}(R_b) \\ f_{R_i} & | & (\forall R_i \in S),\ f_{R_i} \in \mathbf{LAV}(R_a)\ \textbf{if}\ q_{a_i} \geq q_{b_i}\ \textbf{and} \\ & & \quad f_{R_i} \in \mathbf{LAV}(R_b)\ \textbf{if}\ q_{a_i} < q_{b_i} \\ P_{R_i} & | & (\forall R_i \in S),\ P_{R_i} \in \mathbf{LAV}(R_a)\ \textbf{if}\ q_{a_i} \geq q_{b_i}\ \textbf{and} \\ & & \quad P_{R_i} \in \mathbf{LAV}(R_b)\ \textbf{if}\ q_{a_i} < q_{b_i} \end{array} \right\} \tag{3.2}$$

where $\mathbf{max}(x, y)$ is an operation that return the biggest integer between x and y.

Notice that, in equation 3.2, the addition $R_a \oplus R_b$ results in a LAV whose raw map have all ΔM_{ij} sets, such that $\Delta M_{ij} \in \mathbf{LAV}(R_a)$ or $\Delta M_{ij} \in \mathbf{LAV}(R_b)$. Also, in $R_a \oplus R_b$, for each robot R_i of the system, if the index $q_{a_i} \geq q_{b_i}$, then the information about the last frontier assigned to R_i (f_{R_i}) and its known position (P_{R_i}) come from R_a's LAV. Otherwise, f_{R_i} and P_{R_i} come from R_b's LAV.

Because $R_b \ominus R_a$ has all information that R_b has and R_a does not have (or have an outdated version), $R_a \oplus (R_b \ominus R_a) \equiv R_a \oplus R_b$.

The addition operation also satisfies the following properties:

$$\begin{aligned}
\text{Commutative: } & R_a \oplus R_b \equiv R_b \oplus R_a \\
\text{Associative: } & R_a \oplus (R_b \oplus R_c) \equiv R_b \oplus (R_a \oplus R_c) \equiv R_c \oplus (R_a \oplus R_b) \\
\text{Identity: } & R_a \oplus \varnothing \equiv R_a \\
\text{Idempotent: } & R_a \oplus R_a \equiv R_a
\end{aligned}$$

Contains ($\sqsupseteq$)

Relation *contains*, $R_a \sqsupseteq R_b$, indicates that the LAV of a robot R_a has all the information that the LAV of another robot R_b has. If R_a has all the information that R_b has, then the difference $R_b \ominus R_a = \varnothing$ and the addiction $R_a \oplus R_b \equiv R_a$. So, the relation $R_a \sqsupseteq R_b$ is true if and only if $\mathbf{LAV}(R_a) \equiv \mathbf{LAV}(R_a \oplus R_b)$ (or $R_b \ominus R_a = \varnothing$).

Lemma 1. *Let R_a, R_b and R_c be three robots. If the LAV of R_a contains all information that the LAV of R_c has, $R_a \sqsupseteq R_c$, then*

$$R_a \oplus (R_b \ominus R_c) \equiv R_a \oplus R_b$$

Proof. *$R_a \oplus (R_b \ominus R_c) \equiv R_a \oplus R_b$ if and only if $R_a \oplus R_b \sqsupseteq R_a \oplus (R_b \ominus R_c)$ and $R_a \oplus (R_b \ominus R_c) \sqsupseteq R_a \oplus R_b$. $R_a \oplus (R_b \ominus R_c)$ has all information that R_a has and some information from the LAV of R_b. Thus, $R_a \oplus R_b \sqsupseteq R_a \oplus (R_b \ominus R_c)$.*

Since $R_a \sqsupseteq R_c$, the LAV of R_a already have any information from the LAV of R_c that is not in the set $R_b \ominus R_c$. Thus, $R_a \oplus (R_b \ominus R_c) \sqsupseteq R_a \oplus R_b$ and, as a consequence, $R_a \oplus (R_b \ominus R_c) \equiv R_a \oplus R_b$. □

3.2.4 Sharing Process

Every time a robot gets new LAV information (detected by itself or shared by another robot), it creates a sharing process to share the new information with its neighbors. The robot that creates the process also manages it, becoming the sharing process's *manager*, and its neighbors become the *members* of the process.

A sharing process has four states: *creation*, *setup*, *synchronization* and *end*. In figure 6, we describe the dynamics of a sharing process.

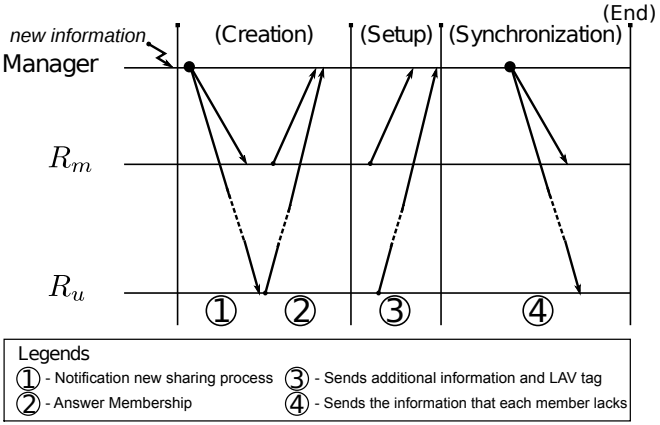


Figure 6 – Sharing process execution.

When a robot R_i creates a sharing process sh_k, it starts in creation state. In this state, the manager sends a broadcast notifying its neighbors about the new process sh_k and waits until its neighbors reply the broadcast. In notification messages (messages of type 1, in figure 6), the manager also informs its LAV tag, allowing neighbors to verify if the manager has any information they do not have yet.

Neighbors can answer the manager (messages of type 2) either acknowledging their participation in sh_k or "declining the invitation", when the manager does not have any additional information.

After all neighbors answer R_i's notification, the sharing process goes to setup state, where sh_k stays until all members (neighbors that acknowledge R_i's notification) send a message informing that they are

"ready" to execute sh_k (messages of type 3). Each member R_u also sends any additional information it has, $R_u \ominus R_i$. Whenever the manager, R_i, receives additional information from a member R_u, it updates its LAV to $R_i \oplus R_u$.

After setup state, sh_k goes to state synchronization, where the manager sends a synchronization message (messages of type 4, in figure 6) to the members of its cluster. For each member R_j, R_i sends a message with the information that R_j does not have yet, $R_i \ominus R_j$. Then, the sharing process goes to end state and all members (and the manager) become free again to synchronize their LAVs in another sharing process.

In algorithm 1, we describe the sharing process algorithm executed by its manager.

Algorithm 1: Sharing Process manager algorithm.

Data: sh_k is the process
Data: R_i is the robot that created sh_k
Data: *LAV* is manager's local application view
1 **inform_newprocess**(sh_k);
2 *members* $\leftarrow$ **identify_members**();
3 **Until** $\forall R_j \in members |\ R_j$ *is ready* **wait**
4 *LAV* $\leftarrow$ update_LAV(*members*);
5 **for** *each* $R_j \in members$ **do**
6 new_info $\leftarrow R_i \ominus R_j$;
7 **send**(new_info, R_j);

When a robot R_i creates a sharing process sh_k, it sends a broadcast notifying its neighbors about the new process (line 1). After the neighbors reply the notification, R_i define sh_k members as the neighbors that acknowledge their participation (line 2). Then, the manager waits until all members are ready to execute sh_k (line 3).

In order to avoid information inconsistency problems, DSM defines that robots can synchronize their LAVs only in one sharing process at a time. So, even if a robot R_j is member of n sharing processes, it can send a message "ready" to only one sharing process. After this shar-

ing process ends (reaches end state), R_j can send a message "ready" to another sharing process, and so on. To select the next sharing process for what the robot will be "ready", each robot executes the PCA (we describe the PCA in the following section).

After receiving "ready" from all members of sh_k, R_i updates its LAV (line 4) and sends to each member the information that it does not have yet (lines 5-7). Then, sh_k ends and all robots become free to execute other processes. Moreover, after getting new information in a sharing process, robots create new sharing process with themselves as managers.

3.2.5 Processes Coordinator Algorithm

During an execution of DSM, robots can be member of many sharing processes at the same time. To avoid information inconsistency problems, DSM defines that robots can synchronize their LAVs with a single manager at a time. In addition, a sharing process cannot reach synchronization state until all members are "ready" to execute it.

Thus, in DSM context, the "ready" confirmation can be viewed as a *token*. Each robot has a single token and, to leave setup state and go to synchronization, the manager of a sharing process needs the tokens of all members (including its own token). After the sharing process finishes, all robots get their tokens back.

If robots do not consider any policy to select the next manager that they will give their tokens, situations where a sharing process waits indefinitely for members executing other sharing processes can happen. In this work, we refer to these situations as an *unfairness problem*.

A worse situation occurs when two (or more) sharing processes have some members in common and some of them give their tokens to one process while the other robots give their tokens to the other sharing process. In this case, both processes will block, waiting that the other finishes and free its members. This situation is defined as *deadlock*.

To avoid unfairness problems and deadlocks, each robot runs a simple *Processes Coordinator Algorithm* (PCA) that coordinates its

participation in all sharing processes of what it is member. PCA is presented in algorithm 2.

Algorithm 2: Processes Coordinator Algorithm.

```
   Data: R_i robot's ID
   Data: proc_list list of process
 1 if new data detected then
 2     if ∃sh_k ∈ proc_list | R_i is sh_k.manager then
 3         cancel();
 4     new_process ←create_newprocess();
 5     proc_list.add(new_process);
 6 else if process sh_i informed then
 7     proc_list.add(sh_i);
 8     sort(proc_list);
 9 else if process sh_i finished then
10     proc_list.remove(sh_i);
11     start(proc_list.first_element);
```

Each robot of the system has its own PCA and executes it whenever it gets new information (detected by its own sensors or received from other robots) or a manager informs the robot about a new sharing process. Robots also have a list of sharing processes of which they are members or the manager.

When a robot R_i gets new information (lines 1-5), PCA verifies if it is already the manager of a sharing process sh_k in the processes list, *proc_list* (line 2). If yes, it cancels sh_k before creating a new process (line 3). Then, the PCA creates a new sharing process that has R_i as its manager (line 4) and adds this process to the end of the list (line 5).

When a manager informs R_i about a new sharing process sh_i (lines 6-8), the algorithm adds sh_i to the processes list (line 7) and sorts the list (line 8). The goal of the sorting mechanism is to define a total order relation (the proof is presented next) among the sharing processes, such that the following properties are satisfied.

1. Let R_i and R_j be two robots in a network and sh_a and sh_b two sharing processes. If $sh_a \prec sh_b$ in R_i's list, then $sh_a \prec sh_b$ in R_j's list.

2. Let R_i, R_j and R_k be three robots in a network and sh_a, sh_b and sh_c three sharing processes. If $sh_a \prec sh_b$ in R_i's list, $sh_b \prec sh_c$ in R_j's list and R_k' list has sh_a, sh_b and sh_c, then $sh_a \prec sh_b \prec sh_c$ in R_k's list.

where the $sh_a \prec sh_b$ means that the sharing process sh_a precedes sh_b in the list.

To do so, the sorting mechanism considers the time at which the managers create the processes and, in case of "draws"[1], the *ID* of the managers.

Robots also execute their PCA when the process that it was executing ends or is canceled by the manager (lines 9-11). In this case, the algorithm removes the finished (or canceled) process from the robot's list (line 10) and starts the next process scheduled (line 11).

3.3 LAV CONVERGENCE ON AN EXECUTION OF DSM

The goal of a DSM's execution is that, at the end of it, all robots in the same network converge to the same LAV. To guarantee that the LAVs will always converge, DSM was designed to satisfy three properties:

Property 1. *A sharing process always ends.*

Property 2. *At the end of a sharing process, all members converge to a LAV that contains all information known by all members of the sharing process.*

Property 3. *If a robot R_i gets new information and there is a route that connects R_i to a robot R_j, then R_j eventually gets the new information.*

Property 1 assures us that DSM will not be in deadlock because some robots are blocked executing a sharing process that never ends.

[1] A draw occurs when, due to discretization of time, two process are considered as created at the same time.

Property 2 guarantees that if some robots in the network, despite they had not initiated the DSM's execution, have additional information (e.g robots that were not in the network during the last DSM's execution), this information will be shared with other robots in the network.

Finally, since there is at least one route that connects each robot to the others in the same network and because of property 3, DSM guarantees that if a robot R_i detects new information, eventually all robots in the network converge to the most up to date LAV version.

Next, we show the proofs, step by step, that these three properties are valid. Then, those three properties are used to prove that the LAVs of the robots always converge at the end of a DSM's execution.

First Property

A sharing process always ends.

Property 1 can be described by temporal logic formula 3.4, which means that: if a sharing process sh_k is in the creation state, eventually it reaches end state.

$$\Box\Big(\textbf{Creation}(sh_k) \rightarrow \Diamond\, \textbf{End}(sh_k)\Big) \tag{3.4}$$

where the attribute $\textbf{Creation}(sh_k)$ indicates that the sharing process sh_k is in creation state and $\textbf{End}(sh_k)$ indicates that sh_k is in end state.

To prove that equation 3.4 is true, and the property 1 is valid, equation 3.4 is rewrote in three parts:

$$\Box\Big(\textbf{Creation}(sh_k) \rightarrow \Diamond\textbf{Setup}(sh_k)\Big) \tag{3.5a}$$

$$\Box\Big(\textbf{Setup}(sh_k) \rightarrow \Diamond\textbf{Synchronization}(sh_k)\Big) \tag{3.5b}$$

$$\Box\Big(\textbf{Synchronization}(sh_k) \rightarrow \Diamond\textbf{Ended}(sh_k)\Big) \tag{3.5c}$$

Equation 3.5a defines that if a process is created (it is in creation state), it eventually goes to setup state. Likewise, equation 3.5b means that, if a process is in setup state, it eventually goes to synchronization

state. Finally, equation 3.5c states that, if a process is in synchronization state, it eventually goes to end state and the process finishes. Next, we prove that the properties described in these equations are true.

Theorem 1. *If a sharing process is created, it eventually goes to setup state.*

$$\textbf{Creation}(sh_k) \rightarrow \Diamond\, \textbf{Setup}(sh_k)$$

Proof. *After a sharing process sh_k is created, its manager (R_i) sends a broadcast informing all neighbors about the new process. After receiving the broadcast, the neighbors send a message to R_i acknowledging or declining their participation in sh_k.*

This work assumes that robots do not fail and that, if two robots have a communication link, messages sent by one are always received by the other. So, if R_i broadcasts a message to notify its neighbors about a new process, all members will get the message and reply it. Likewise, R_i will receive all replies. Thus, if a process sh_k is created, it eventually goes to setup state and property 3.5a is true. □

Theorem 2. *If a sharing process is in synchronization state, it eventually goes to state end.*

$$\textbf{Synchronization}(sh_k) \rightarrow \Diamond\, \textbf{End}(sh_k)$$

Proof. *In synchronization state, the manager R_i updates its LAV with any additional information sent by the members. Next, it sends to each member R_j only the information that it does not have yet, $R_i \ominus R_j$, and sh_k ends (goes to end state). Thus, if a sharing process is in synchronization state, it eventually ends and property 3.5c is true.* □

The proof that a sharing process eventually goes from setup to synchronization state is based on the fact that the Processes Coordinator Algorithm (PCA) defines a *total order relation Ct* on the set of active sharing processes, SP_{active}.

Let $\textbf{time}(sh_i) \in \mathbb{R}$ be the time at which the sharing process sh_i was created, Ct is defined as:

$$Ct = \{(sh_i, sh_j) | \textbf{time}(sh_i) \leq \textbf{time}(sh_j)\} \tag{3.6}$$

It is clear that Ct is a *total order relation on* SP_{active} and satisfies the *reflexive*, *antisymmetric* and *transitive* properties. In ad-

dition, all elements in SP_{active} are, in pairs, comparable by Ct (ROSEN; KRITHIVASAN, 1999).

Theorem 3. *If a sharing process sh_k is in setup state, it eventually goes to synchronization state.*

$$\textbf{Setup}(sh_k) \rightarrow \Diamond\ \textbf{Synchronization}(sh_k)$$

Proof. Each robot runs a PCA to sort the processes of which it is a member. For a robot R_i that is currently member of n sharing processes, the list of sharing processes of which it is a member is defined as $list_i = \{sh_{i_1}, sh_{i_2}, \cdots sh_{i_n}\}$. Also, every time the robot finishes executing a process (the process reaches state end), PCA removes it from the robot's list and selects the next process to be executed.

Since Ct is a total order relation on SP_{active}, it satisfies the following properties:

$$\begin{aligned}
&\textit{(Reflexivity)} && \text{if } a \in SP_{active}, \text{ then } (a,a) \in Ct \\
&\textit{(Antisymmetry)} && \text{if } (a,b) \in Ct \text{ and } a \neq b, \text{ then } (b,a) \notin Ct \\
&\textit{(Transitivity)} && \text{if } (a,b) \in Ct \text{ and } (b,c) \in Ct \text{ , then } (a,c) \in Ct \\
&\textit{(Total Comparability)} && \text{if } a,b \in SP_{active}, \text{ then } (a,b) \in Ct \text{ or } (b,a) \in Ct
\end{aligned}$$

Let $a,b \in SP_{active}$ be two active sharing processes. If $(a,b) \in Ct$, a precedes b or $a \preceq b$. As a result of the transitivity property, if $a,b,c \in SP_{active}$ and $a \preceq b$ and $b \preceq c$, then $a \preceq c$. In addition, as all elements in SP_{active} are comparable by Ct, there is a minimum element $e_{min} \in SP_{active}$ and a maximum element $e_{max} \in SP_{active}$. So, a sorted list with all sharing processes can be defined as follows:

$$e_{min} \preceq a \preceq b \preceq \cdots \preceq e_{max} \tag{3.7}$$

With no loss of generality, let's redefine the list 3.7 as:

$$sh_1 \preceq sh_2 \preceq sh_3 \preceq \cdots \preceq sh_x \tag{3.8}$$

As sh_1 is the minimum element, there is no active process that precedes it ($\nexists sh_i \in SP_{active} | sh_i \neq sh_1$ and $sh_i \preceq sh_1$) and the members of sh_1 ($M_{sh_1} = \{R_j | sh_1 \in list_j\}$) cannot be busy synchronizing their LAV in another process. Thus, sh_1 goes to synchronization state.

From theorem 2, if a sharing process is in the state synchronization, it eventually ends. So, sh_1 will reach end state and all members will be free to execute another active sharing process.

When sh_1 ends, its former members remove it from their lists and sh_2 becomes the new minimum element in SP_{active}. Then, regardless of which robots are the members of sh_2 (M_{sh_2}), they are ready to synchronize their LAVs in sh_2 and this process goes to synchronization state. Again from theorem 2, if sh_2 goes to synchronization state, it eventually ends. Then the sharing process sh_3 becomes the new minimum element in SP_{active} and goes to synchronization state, and so on. Let sh_k be a process in SP_{active} and $sh_1, \cdots, sh_{k-1} \in SP_{active}$ all processes that precede sh_k. If all sharing processes sh_j, such that $j < k$, ends, then sh_k goes to synchronization state.

By mathematical induction (sh_2 base case; and sh_k the induction step), if a robot R_i creates a sharing process sh_m, eventually all members of sh_m will be ready to synchronize their LAV in it and sh_m goes to synchronization state. So the property described in equation 3.5b is true.

As the properties described 3.5a, 3.5b and 3.5c are true, property 1 is valid. □

Second Property

At the end of a sharing process, all members converge to a LAV that contains all information known by all members of the sharing process.

When a robot R_i gets new information and creates a sharing process, at the end of this process, the LAVs of the manager and of all members converge to the most up to date version, in context of the sharing process. In other words, the new version has all information that R_i has and any additional information that the members shared with R_i, which consist in all information known by the group (members and manager).

Let LAV_k be the most up to date LAV version considering all robots participating of a sharing process sh_k. LAV_k can be defined as:

$$LAV_k = \mathbf{LAV}(R_i) \oplus \mathbf{LAV}(R_m) \oplus \cdots \oplus \mathbf{LAV}(R_u) \tag{3.9}$$

where R_i is the manager of sh_k and $M_{sh_k} = \{R_m, \cdots, R_u\}$ is the set members of sh_k.

Theorem 4. *At the end of a sharing process sh_k, all members of sh_k converge to LAV_k.*

$$\forall R_j \in \left(M_{sh_k} \cup \{R_i\}\right) \textbf{End}(sh_k) \rightarrow \textbf{LAV}(R_j) \equiv LAV_k \tag{3.10}$$

Proof. Let R_i be the manager of a sharing process sh_k. When a member R_j of sh_k becomes ready to execute sh_k, it sends a "ready to go" message (message of type 3 in figure 6) to R_i with any additional information. The addition information from R_j to R_i is defined by $R_j \ominus R_i$.

Since all members send any additional information to R_i, we have that:

$$\forall R_j \in M_{sh_k},\ R_i \textbf{ gets } R_j \ominus R_i$$

After receiving the "ready to go" messages from all members, the manager updates its own LAV with any information sent by them. Let LAV'_{R_i} be the LAV of R_i after the update, LAV'_{R_i} can be described as:

$$LAV'_{R_i} = R_i \oplus (R_m \ominus R_i) \oplus \cdots \oplus (R_u \ominus R_i) \tag{3.11}$$

where $R_m, \cdots, R_u \in M_{sh_k}$ are the members of sh_k.

Let R_a, R_b and R_c be three robots. Section 3.2.3 shows that $R_a \oplus (R_b \ominus R_a) \equiv R_a \oplus R_b$. Moreover, $R_a \oplus R_b \oplus R_c \equiv R_b \oplus R_a \oplus R_c \equiv R_c \oplus R_a \oplus R_b$. Thus, equation 3.11 can be simplified to:

$$LAV'_{R_i} = R_i \oplus R_m \oplus \cdots \oplus R_u \tag{3.12}$$

Thus, from equations 3.9 and 3.12, $LAV'_{R_i} \equiv LAV_k$.

After updating its LAV, R_i identifies which information from its new LAV each member R_j still lacks, $R_i \ominus R_j$, and sends it to R_j. Next, the members update their LAVs with the information sent by R_i. Let R_j be a member of sh_k, its updated LAV will defined by:

$$LAV'_{R_j} = R_j \oplus (R_i \ominus R_j) \equiv R_j \oplus R_i \tag{3.13}$$

As $LAV_{R_i} \equiv LAV_k$, R_j's LAV is defined as:

$$LAV'_{R_j} = R_j \oplus LAV_k \tag{3.14}$$

Since LAV_k contains all information that the LAV of R_j has ($LAV_k \sqsupseteq R_j$), $LAV'_{R_j} \equiv LAV_k$. Thus,

$$\forall R_j \in \left(M_{sh_k} \cup \{R_i\}\right),\ LAV'_{R_j} \equiv LAV_k$$

and equation 3.10 is true. Thus, property 2 is true.

□

Third Property

If a robot R_i gets new information and if there is a route that connects R_i to a robot R_j, then R_j eventually gets the new information

Let $\mathcal{N}$ be a network represented by the graph $\mathcal{N} = G(V,E)$, where the nodes are the robots in the set $V \subseteq S$ (S is the set of all robots in the system) and the edges E represents direct links between the robots in V. If a robot $R_i \in V$ detects new information and there is a route that connects a robot $R_j \in V$ to R_i (and that route is kept during the execution of DSM), R_j eventually gets the new information.

Let **info**(R_i) be an attribute that indicates that the robot R_i has new information to share. Also, let **route**$(a,b) : \langle e_{a-v_1}, e_{v_1-v_2}, \cdots, e_{v_n-b} \rangle$ be a route that connects the nodes (robots in the context of this work) a and b, where e_{i-j} represents an edge between nodes i and j.

Theorem 5. *If a robot R_i detects new information and there is a route connecting R_i to a robot R_j, R_j eventually gets all information R_i has.*

$$(\forall R_i, R_j \in V)\ \textbf{info}(R_i)\ \textit{and}\ \textbf{route}(R_i,R_j) \rightarrow \Diamond\, R_j \sqsupseteq R_i \tag{3.15}$$

where $R_j \sqsupseteq R_i$ indicates that the LAV of R_j contains all information that the LAV of R_i has.

Proof. *Let* **route**$(R_i,R_j) : \langle e_{i-v_1}, e_{v_1-v_2}, \cdots, e_{v_n-j} \rangle$ *be a route that connects the robots R_i and R_j. So, there is a robot R_{v_1} that has a direct link with R_i and a robot R_{v_2} that has a link with R_{v_1} and so on. The route ends in a robot R_{v_n} that has a link with R_j.*

Assuming that the connections between the robots are kept during the execution of DSM, if R_i gets new information, it creates a sharing process sh_0 of which R_{v_1} will be a member. From the properties 1 and 2, DSM guarantees that sh_0 ends and R_{v_1} gets any information that R_i and others members have.

Let's consider now that two robots R_{v_k} and $R_{v_{k+1}}$ have a direct link between them and are in the middle of the route between R_i and R_j. If R_{v_k} gets the information detected by R_i, it creates a sharing process sh_k of which $R_{v_{k+1}}$ will be a member. Again from properties 1 and 2, DSM guarantees that sharing process sh_k ends and $R_{v_{k+1}}$ gets any information that R_{v_k} and others members of sh_k have.

Thus, by mathematical induction (base case: if R_i gets new information, then R_{v_1} will get the information; induction step: if R_{v_k} gets the infor-

mation, then $R_{v_{k+1}}$ will get the information), we have that if R_i detects (or receives from another robot) new information, a sequence of sharing processes $\langle sh_0, sh_1, \cdots, sh_n \rangle$ initiating in R_i and reaching R_j will be executed and R_j eventually gets any information that R_i has.

Since property 2 guarantees that all members of a sharing process sh_u converge to LAV_u, a local application view that contains all information known by the manager and the members of sh_u. After sharing process sh_0 ends, the LAV of R_{v_1} contains any information that the LAV of R_i has, $R_{v_1} \sqsupseteq R_i$. Likewise, after a sharing process sh_k that has R_{v_k} as manager and $R_{v_{k+1}}$ as member ends, the LAV of $R_{v_{k+1}}$ contains any information that the LAV of R_{v_k} has, $R_{v_{k+1}} \sqsupseteq R_{v_k}$.

By mathematical induction (base case: after sh_0 ends, $R_{v_1} \sqsupseteq R_i$; induction step: after sh_k ends, $R_{v_{k+1}} \sqsupseteq R_k$), we have that after sharing process sh_n, which has R_{v_n} as manager and R_j as member, ends, $R_j \sqsupseteq R_{v_n} \sqsupseteq \cdots \sqsupseteq R_{v_2} \sqsupseteq R_{v_1} \sqsupseteq R_i$. Thus, the LAV of R_j contains all information that R_i has and property 3 is true. □

3.3.1 DSM Convergence: Single Robot Detecting new Information

Let LAV_{net} be the application view that has all information known by all robots in the network. LAV_{net} can be defined by equation 3.16.

$$LAV_{net} = R_1 \oplus R_2 \oplus \cdots \oplus R_N \tag{3.16}$$

where $R_1, R_2, \cdots, R_N \in V$ are the robots of network $\mathcal{N} = G(V,E)$.

Because DSM is a distributed scheme, the system (and even the network) does not have a leader (or an operational base) that synchronizes the LAV of all robots. However, whenever a robot gets new information, it starts a new execution of DSM and, at the end of it, all robots in the network converge to the same and most up to date LAV version, which corresponds to LAV_{net}.

Theorem 6. *Let $\mathcal{N} = G(V,E)$ be a network and $R_i \in V$ a robot of this network that detected new information. If $\mathcal{N}$ does not lose connectivity during a DSM's execution, all robots will converge to the most up to date LAV version, LAV_{net}.*

$$[\exists R_i \in V | \mathbf{info}(R_i)] \rightarrow \Diamond\, [\forall R_j \in V | \mathbf{LAV}(R_j) = LAV_{net}] \tag{3.17}$$

Proof. Since R_i is a robot of network $\mathcal{N}$, there is at least a route that connects R_i to the other robots in V. From property 3, if there is a route between two robots R_a and R_b and R_a gets new information, R_b eventually gets this information and any additional information that R_a has, so $\mathbf{LAV}(R_b) \sqsupseteq \mathbf{LAV}(R_a)$.

If the network does not lose connectivity during a DSM's execution, R_i always has a route to the other robots. Thus, if R_i detects new information, each robot $R_j \in V$ eventually converges to a local application view $\mathbf{LAV}(R_j)$, such that $\mathbf{LAV}(R_j) \sqsupseteq \mathbf{LAV}(R_i)$.

Since R_i detected the information, initially, there is no robot in the network that already has this information. So, when each robot gets this information (in a sharing process managed by other robot), it will create a sharing process. If some robots of the network have a LAV with additional information that R_i does not have yet (e.g. a robot that was not part of the network during the last DSM's execution), when they create a sharing process, they will share the additional information with their neighbors. Likewise, the neighbors will share this information with their own neighbors, and so on. Thus, at the end of a DSM's execution, if a robot R_k has additional information, all robots in the network eventually get this information, including R_i.

Let R_k be a robot that has additional information that R_i does not have yet. As R_k and R_i are in the same network, there is at least one route $r\langle: e_{i-a}, e_{a-b}, \cdots, e_{t-k}\rangle$ connecting both robots. So, despite the fact that R_k was not the robot that initiated the DSM's execution, R_k eventually gets the information detected by R_i and creates a sharing process. Since there is at least one route that connects R_k with the other robots in the network, property 3 guarantees that any additional information that R_k has will be shared with the other robots in the network. Thus, at the end of the DSM's execution, $(\forall R_j \in V)\mathbf{LAV}(R_j) \sqsupseteq \mathbf{LAV}(R_k)$.

Let $V_{\neq R_i} \subset V$ be the set of robots whose LAV have additional information that the R_i does not have. The set $V_{\neq R_i}$ can be described as:

$$V_{\neq R_i} = \left\{R_j \in V | (\forall R_j \in V),\ R_j \ominus R_i \neq \varnothing\right\} \tag{3.18}$$

Equation 3.18 defines $V_{\neq R_i}$ as the set of robots $R_j \in V$, such that the difference between the LAVs of R_j and R_i is not empty.

From property 3, we have that all robots in V gets the information that R_i detected:

$$(\forall R_j \in V),\ \mathbf{LAV}(R_j) \sqsupseteq \mathbf{LAV}(R_i) \tag{3.19}$$

As each robot creates a sharing process every time it gets new information, also from property 3, we have that all robots in V gets the information

that any robot $R_k \in V_{\neq R_i}$ have:

$$(\forall R_j \in V)(\forall R_k \in V_{\neq R_i}),\ \mathbf{LAV}(R_j) \sqsupseteq \mathbf{LAV}(R_k) \tag{3.20}$$

where R_i is the robot that detected new information and started the DSM's execution.

Based on equations 3.19 and 3.21, the LAV of any robot $R_j \in V$ satisfy the following property after the DSM's execution:

$$(\forall R_j \in V),\ R_j \sqsupseteq R_i \oplus R_k \oplus \cdots \oplus R_p \tag{3.21}$$

where $V_{\neq R_i} = \{R_k, \cdots, R_p\}$.

Let's consider that $L = R_i \oplus R_k \oplus \cdots \oplus R_p$ is not equivalent to LAV_{net}, $L \neq LAV_{net}$. Since LAV_{net} contains all information from the LAVs of all robots in the network (see equation 3.16), $LAV_{net} \sqsupseteq L$. If $L \neq LAV_{net}$ and $LAV_{net} \sqsupseteq L$, then $L \not\sqsupseteq LAV_{net}$. So, there must exist a robot $R_m \in V$ such that, $LAV_{net} \sqsupseteq \mathbf{LAV}(R_m)$ and $L \not\sqsupseteq \mathbf{LAV}(R_m)$.

If such robot R_m exists, it has information that neither R_i nor the robots in $V_{\neq R_i}$ have. So, $R_m \ominus R_i \neq \varnothing$. However, if $R_m \ominus R_i \neq \varnothing$, then $R_m \in V_{\neq R_i}$ and $L \sqsupseteq \mathbf{LAV}(R_m)$. Thus, it is absurd that exists a robot $R_m \in V$ such that, $LAV_{net} \sqsupseteq \mathbf{LAV}(R_m)$ and $L \not\sqsupseteq \mathbf{LAV}(R_m)$. Therefore, $L \sqsupseteq LAV_{net}$ and, as $LAV_{net} \sqsupseteq L$, $L \equiv LAV_{net}$.

Thus, from equations 3.17 and 3.21, we have that the robots in the network always converge to the most up to date LAV version, LAV_{net}, at the end of a DSM's execution. □

3.3.2 DSM Convergence: Multiple Robots Detecting Information

Let's consider now the case where several robots detects new information and creates sharing processes simultaneously. As soon as the first robot creates a sharing process to share its new information, the DSM's execution begins and lasts until all robots in the network get all information known by all robots in the network, which includes the information detected by the other robots.

To prove that, at the end of the DSM's execution, the LAVs of all robots in the network converge to most up to date version (LAV_{net}, defined in equation 3.16), we show that DSM also satisfies properties 1, 2 and 3 when several robots detect new information simultaneously.

First Property

Property 1, *a sharing process always ends*, can be separated in three parts: if a sharing process is created, it eventually goes to setup state; if a sharing process is in setup state it eventually goes to synchronization state; and if a sharing process is in synchronization state, it eventually ends.

A sharing process goes from creation to setup state when the neighbors of the manager answer its notification about the new process. As this work assume that robots do not fail and that messages sent between robots with a communication link are always received, a sharing process eventually goes from creation to synchronization state. Similarly, a process in synchronization state eventually ends.

Finally, the sorting mechanism defined in the PCA algorithm enforces that a sharing process in setup state eventually goes to synchronization. As the sorting mechanism does not consider the origin of the information that will be shared in a sharing process, only the time at which the sharing process was created, relation Ct is total order relation on SP_{active} even when several robots detects information simultaneously. Thus, property 1 is also valid for situations where multiple robots detects new information simultaneously.

Second Property

Property 2 states that *at the end of a sharing process, all members converge to a LAV that contains all information known by all members of the sharing process*.

Theorem 4 states that, at the end of a sharing process sh_k, the LAVs of the manager and the members converge to LAV_k, a LAV that contains all information known by the manager and the members of sh_k. Thus, even when several robots detects new information simultaneously, which could lead to situations where every member of a sharing process has additional information, the LAVs of the manager and members converge to a LAV version that contains all information known by the manager and all members of the sharing process. There-

fore, property 2 is also true for situations with simultaneous information detection.

Third Property

Property 3 states that *if a robot R_i gets new information and if there is a route that connects R_i to a robot R_j, then R_j eventually gets the new information*.

This property is based on the existence of a communication route between two robots and properties 1 and 2 (both valid in cases where several robots detect new information). As the detection of new information by several robots simultaneously does not change the network topology, property 3 is still valid.

Convergence

Likewise the case where a single robot detects new information, property 1 guarantees that the sharing processes always end and DSM will not be in deadlock because some robots are blocked executing a sharing process that never ends. Moreover, property 2 guarantees that even when some members of a sharing process have LAVs with additional information, all members of the sharing process converge to a LAV version that contains all information known by all members and the manager of the sharing process.

Finally, since there is at least one route that connects each robot to the others in the same network, property 3 assures us that if a set of robots $\{R_a, \cdots, R_u\}$ detects new information simultaneously, all robots in the network eventually converge to the most up to date LAV version (LAV_{net}).

3.3.3 On the parallel execution of several sharing process

An important aspect regarding DSM is that two (or more) sharing process can be in the synchronization state simultaneously. However, they must satisfy the following requirements.

Let $sh_a, sh_b \in SP$ be two different sharing processes such that $sh_a \preceq sh_b$. DSM allows sh_a and sh_b to be in synchronization simultaneously if and only if there is no member of sh_b that is also a member of a process sh_j, such that $sh_a \preceq sh_j$ and $sh_j \preceq sh_b$. This requirement is described in equation 3.22.

$$\forall R_i \in M_{sh_b}, \nexists sh_j \in SP_{active} | sh_j \in list_i \textbf{ and } sh_a \preceq sh_j \textbf{ and } sh_j \preceq sh_b \quad (3.22)$$

If this requirement is not satisfied and there is a robot R_i such that, $\exists sh_j \in list_i | sh_a \preceq sh_j$ **and** $sh_j \prec sh_b$, R_i will not be ready to execute sh_b while sh_j is active. Thus, sh_b cannot go to synchronization state.

3.4 HIERARCHICAL SYNCHRONIZATION METHOD

This work proposes a second method for map sharing, the Hierarchical Synchronization Method (HSM) that is also based on the LAV concept. Instead of using sharing processes and PCA, HSM considers a hierarchical architecture for the *ad hoc* network and exploits the advantages of this kind of organization to synchronize the LAV of robots.

In hierarchical networks, robots are grouped in *clusters* and leaders (named *clusterheads*) are elected for each cluster. Due to the existence of leaders, problems as message losses become easier to detect and handle. Moreover, routing schemes for multi-hop communication scales well in hierarchical networks (ABBASI; YOUNIS, 2007).

In HSM, there are two situations where robots need to share their maps: after a robot explores a frontier; and after two networks merge. In the first case, the robot that finished exploring its frontier needs to share the new information it detected with the other robots in the network. In the second case, robots in one network may have information that robots in the other do not have. So, these robots need to identify which LAV information the others still lack and send it to them. In DSM, robots do not know the network topology and, thus,

cannot identify when two networks merge and use this event to trigger map sharing.

In next sections, we present HSM dynamics and define the schemes HSM uses to handle these two situations.

3.4.1 HSM Dynamics

The main goal of HSM is to keep the LAV of all robots in a network synchronized. In other words, HSM's goal is to share the information in the LAVs of robots in order to guarantee that, in a network, all robots have the same and most up to date LAV version.

Let's consider that the LAV of all robots in a network $\mathcal{N}$ is $\mathbf{LAV_{net}}$ and property 4 is true.

Property 4. *All robots in a network $\mathcal{N}$ have the same LAV,* $\mathbf{LAV_{net}}$.

$$\forall R_i \in \mathcal{N} \ : \ \mathbf{LAV}(R_i) \equiv \mathbf{LAV_{net}}$$

When a robot R_u finishes exploring a frontier and updates its LAV, R_u needs to share the information it detected with the other robots in the network. To do so, it sends the new information, which corresponds to $R_u \ominus R$, to its clusterhead.

After a clusterhead receives an update from a member of its cluster, it shares the information with the other members and with the other clusterheads in the network. If another clusterhead sent the LAV update, the clusterhead shares the information only with its cluster members. We name this scheme *HSM simple sharing*.

When two (or more) networks merge, clusterheads of the new network exchange their LAV tags. Then, they send to each clusterhead the information it still lacks. We name this second sharing scheme *HSM LAV synchronization*.

Algorithm 3, executed by clusterheads, summarizes an execution of HSM.

Because of property 4, the LAVs of all robots in the network are the same ($\mathbf{LAV_{net}}$), until a robot detects new information. After R_i (clusterhead of a cluster cl_i) receives an update from a member of

Algorithm 3: HSM execution.

```
   Data: R_i robot's ID
   Data: LAV_net LAV of all robots in the network
1  if R_u ∈ 𝒩 sends R_u ⊖ LAV_net then
2      simple sharing(R_u ⊖ LAV_net);
3  else if networks merge then
4      LAV synchronization(𝒩);
```

its cluster or from another clusterhead (line 1), R_i executes the *simple sharing* scheme. After two networks merge (line 3), R_i executes the *LAV synchronization* scheme (line 4).

Next, in sections 3.4.2 and 3.4.3, we define the *simple sharing* and *LAV synchronization* schemes, respectively, and how these scheme are used to enforces property 4. In each section, we also present an example of the scheme execution.

3.4.2 Simple Sharing Scheme

In HSM, whenever a robot R_u explores a frontier, it shares the information detected with its clusterhead. Next, the clusterhead shares the information with the other members of the cluster and with the other clusterheads in the same network using the simple sharing scheme. At the end, all robots in the network get the new information. Algorithm 4 summarizes the simple sharing execution. Next, an illustrative example is presented in order to help the reader understanding the simple sharing scheme.

Let $\mathcal{N}$ be a network such that $\forall R_k \in \mathcal{N}, \mathbf{LAV}(R_k) \equiv \mathbf{LAV_{net}}$. After a robot $R_u \in \mathcal{N}$ explores a frontier, it updates its LAV and $\mathbf{LAV}(R_u)$ becomes the most up to date in the network, $\mathbf{LAV}(R_u) \sqsupseteq \mathbf{LAV_{net}}$. The additional information that R_u has corresponds to $R_u \ominus \mathbf{LAV_{net}}$.

In the simple sharing scheme, after receiving an update from R_u's LAV ($R_u \ominus \mathbf{LAV_{net}}$), a clusterhead R_i shares the information that it got with the members of its cluster (lines 1-2). Then, if the robot that detected the information (R_u) is a member of R_i's cluster, it also

Algorithm 4: Simple Sharing Scheme.

```
   Data: R_i robot's ID
   Data: LAV_net LAV of all robots in the network
   Data: R_u ⊖ LAV_net Information sent
1  for all R_j ∈ cl_i do
2      send(R_u ⊖ LAV_net, R_j)
3  if R_u ∈ cl_i then
4      for all cl_v ∈ 𝒩 do
5          send(R_u ⊖ LAV_net, R_v)
```

sends the information to the other clusterheads in the network using multi-hop communication (lines 3-5). In this work, we name clusters based on their clusterheads ID, so, a cluster that has R_x as clusterhead is named cl_x.

At the end of a simple sharing execution, all robots in the network get the information in $R_u \ominus \mathbf{LAV_{net}}$ and their LAV converge to the most up to date version. Thus, simple sharing scheme keeps all robots with the same LAV and enforces property 4 whenever a robot detects new information.

Theorem 7. *At the end of a simple sharing scheme, started to share information detected by a robot R_u, the LAV of all robots in the network converge to* $\mathbf{LAV}(R_u)$.

Proof. *Let $\mathbf{LAV_{net}}$ be the LAV of all robots in the network before R_u finishes exploring its frontier. After update its LAV, R_u sends $R_u \ominus \mathbf{LAV_{net}}$ to its clusterhead. Next, its clusterhead shares the information with the other members and with the other clusterheads. Finally, each clusterhead will send the information to its own cluster's members. Thus, at the end of the simple sharing scheme all robots get the new information.*

$$\forall R_k \in \mathcal{N},\ R_k \textbf{ gets } R_u \ominus \mathbf{LAV_{net}}$$

After a robot R_k gets the new information, it updates its LAV to:

$$\mathbf{LAV}(R_k) \equiv R_k \oplus (R_u \ominus \mathbf{LAV_{net}})$$

From property 4, the LAV of all robot in the network (except R_u) are equal to $\mathbf{LAV_{net}}$*. So, R_k's new LAV is given by equation 3.23.*

$$\mathbf{LAV}(R_k) \equiv \mathbf{LAV_{net}} \oplus (R_u \ominus \mathbf{LAV_{net}}) \tag{3.23}$$

From operations $\ominus$ and $\oplus$ definitions (section 3.2.3), we have that:

$$\mathbf{LAV}(R_k) \equiv \mathbf{LAV_{net}} \oplus R_u \tag{3.24}$$

Since $\mathbf{LAV}(R_u) \sqsupseteq \mathbf{LAV_{net}}$*), from relation contains definition, we have that:*

$$\mathbf{LAV}(R_k) \equiv \mathbf{LAV}(R_u) \tag{3.25}$$

Thus, as all robots in the network get the information detected by R_u, $R_u \ominus \mathbf{LAV_{net}}$, the LAVs of all robots converge to $\mathbf{LAV}(R_u)$*, which becomes the new* $\mathbf{LAV_{net}}$*.* □

Illustrative Example

Figure 7 presents an example of simple sharing execution where 14 robots are separated in two networks, $\mathcal{N}_1$ and $\mathcal{N}_2$. $\mathcal{N}_1$ has 11 robots and is organized in two clusters, cl_4 and cl_9, and $\mathcal{N}_2$ has a single cluster with 3 robots, cl_a. The example considers that robot R_7, in $\mathcal{N}_1$, finished exploring its frontier. Figure 7 presents the organization of robots and, to illustrate the steps of the simple sharing scheme, figure 8 presents a sequence diagram summarizing the execution.

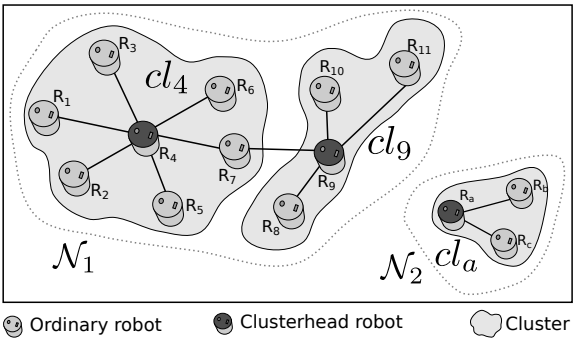


Figure 7 – Network topology.

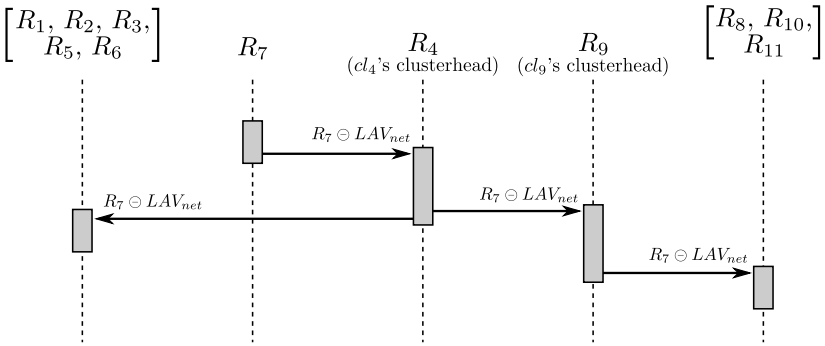


Figure 8 – Sequence diagram of a simple sharing execution.

Let LAV_{net} be the local application view of all robots in $\mathcal{N}$ before R_7 detects new information. After R_7 explores a frontier, it sends only the new information ($R_7 \ominus LAV_{net}$) to its clusterhead, R_4, which is indicated in figure 8 by the arrow from R_7's column to R_4's. Next, R_4 sends the information to its cluster members (R_1, R_2, R_3, R_5 and R_6) and to R_9 (using multi-hop communication), the other clusterhead in the network. Then, R_9 shares the information with its cluster members, R_8, R_{10} and R_{11}. Thus, at the end of HSM simple sharing scheme, all robots in $\mathcal{N}_1$ converge to the same LAV. As robots in $\mathcal{N}_2$ cannot communicate with robots in $\mathcal{N}_1$, they do not get the new information.

Simultaneous Detection

If several robots finish exploring their frontiers at the same time, each detection triggers a simple sharing execution and all robots converge to the most up to date LAV version.

Let $R^{detInfo} = \{R_1, R_2, \cdots, R_n\}$ be the set of robots that finished exploring their frontiers at the same time. Each robot in $R^{detInfo}$ sends the information that it detected to its clusterhead, which will share the information with the other members of the cluster and with the other clusterheads in the network. Hence, at the end of the simple sharing scheme, all robots get all information.

Let $\mathbf{LAV_{net}}$ be the LAV of all robots in the network before the

robots in $R^{detInfo}$ finished exploring their frontiers. At the end of the simple sharing, all robots get the information detected by all robots in $R^{detInfo}$ and their LAVs are updated to the one presented in equation 3.26.

$$\mathbf{LAV}'_{\mathbf{net}} \equiv R_1 \oplus R_2 \oplus \cdots \oplus R_n \tag{3.26}$$

Theorem 8. *At the end of the simple sharing execution with simultaneous information detection, the LAV of all robots converge to* $\mathbf{LAV}'_{\mathbf{net}}$.

Proof. Let R_k be a robot in network $\mathcal{N}$ and R_1 a robot in $R^{detInfo}$. After R_k receives the information detected by R_1, R_k's LAV is updated to the one given by equation 3.27.

$$\mathbf{LAV}(R_k) \equiv \mathbf{LAV}_{\mathbf{net}} \oplus (R_1 \ominus \mathbf{LAV}_{\mathbf{net}}) \tag{3.27}$$

As $\mathbf{LAV}(R_1) \sqsupseteq \mathbf{LAV}_{\mathbf{net}}$, the properties of operations $\ominus$ and $\oplus$ allows us to simplify equation 3.27 as:

$$\mathbf{LAV}(R_k) \equiv R_1 \tag{3.28}$$

With no loss of generality, after receiving information detected by another robot $R_2 \in R^{detInfo}$, $\mathbf{LAV}(R_k)$ is updated to:

$$\mathbf{LAV}(R_k) \equiv R_1 \oplus (R_2 \ominus \mathbf{LAV}_{\mathbf{net}}) \tag{3.29}$$

As $\mathbf{LAV}(R_2) \sqsupseteq \mathbf{LAV}_{\mathbf{net}}$, equation 3.29 can be simplified to:

$$\mathbf{LAV}(R_k) \equiv R_1 \oplus R_2 \tag{3.30}$$

Considering that R_k got information from m robots in $R^{detInfo}$, its LAV can be defined by:

$$\mathbf{LAV}(R_k) \equiv R_1 \oplus R_2 \oplus \cdots \oplus R_m \tag{3.31}$$

After gets the information detected by $R_{m+1} \in R^{detInfo}$, R_k's LAV is updated to:

$$\mathbf{LAV}(R_k) \equiv R_1 \oplus R_2 \oplus \cdots \oplus R_m \oplus (R_{m+1} \ominus \mathbf{LAV}_{\mathbf{net}}) \tag{3.32}$$

As $\mathbf{LAV}(R_{m+1}) \sqsupseteq \mathbf{LAV}_{\mathbf{net}}$, equation 3.32 can be simplified to:

$$\mathbf{LAV}(R_k) \equiv R_1 \oplus R_2 \oplus \cdots \oplus R_m \oplus R_{m+1} \tag{3.33}$$

From the base case (equation 3.28) and the iteration step (equation 3.33), we have that, after receiving information from all robots in $R^{detInfo}$, R_k's LAV is given by:

$$\mathbf{LAV}(R_k) \equiv R_1 \oplus R_2 \oplus \cdots \oplus R_n \equiv \mathbf{LAV'_{net}} \tag{3.34}$$

Thus, as all robots in $\mathcal{N}$ get the information detected by all robots in $R^{detInfo}$, the LAV of all robots converge to $\mathbf{LAV'_{net}}$. □

Considering again the network topology presented in figure 7, an illustrative example in which two robots, R_7 and R_{11}, finish exploring their frontiers simultaneously is presented. Figure 9 shows a sequence diagram summarizing the execution when multiple robots detects new information.

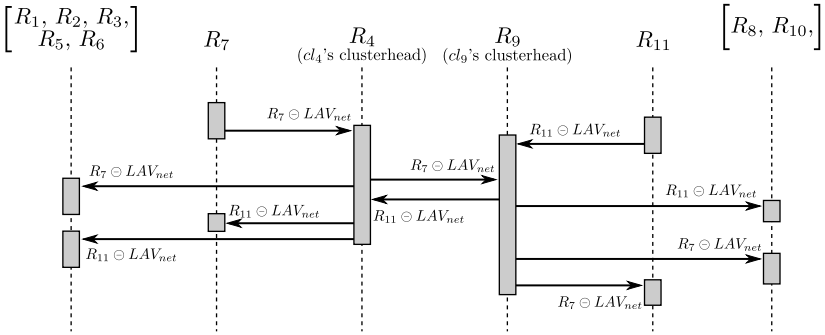


Figure 9 – Sequence diagram of a simple sharing execution with simultaneous detection.

Let LAV_{net} be the local application view of all robots in $\mathcal{N}$ before R_7 and R_{11} detect new information. R_7 sends $R_7 \ominus LAV_{net}$ to its clusterhead, R_4, which is indicated in figure 9 by the arrow from R_7's column to R_4's. Similarly, R_{11} sends $R_{11} \ominus LAV_{net}$ to its clusterhead, R_9. Next, each clusterhead sends the information that they got to the other members of their cluster and to the other clusterhead. Then, they share the information received from the other clusterhead with their members. Thus, at the end of HSM simple sharing scheme, all robots in $\mathcal{N}_1$ get the information detected by R_7 ($R_7 \ominus LAV_{net}$) and by R_{11} ($R_{11} \ominus LAV_{net}$) and converge to the most up to date LAV, $\mathbf{LAV'_{net}} \equiv R_7 \oplus R_{11}$.

3.4.3 LAV Synchronization Scheme

Clusterheads execute HSM *LAV synchronization* scheme to synchronize the LAVs after two or more networks merge. Algorithm 5 presents the algorithm ran by clusterheads to execute the LAV synchronization.

Algorithm 5: LAV synchronization scheme.

Data: R_i clusterhead's ID
1 **for all** $R_j \in cl_i$ **do**
2 **send(LAV**(R_i).tag, R_j**)**;
3 **Until** $\forall R_j \in cl_i |\ R_j$ *sent* $R_j \ominus R_i$ **wait**
4 **for all** $R_j \in cl_i$ **do**
5 **LAV**$(R_i) = R_i \oplus (R_j \ominus R_i)$;
6 **for all** $cl_a \in \mathcal{N}$ **do**
7 **send(LAV**(R_i).tag, R_a**)**;
8 **Until** $\forall cl_a \in \mathcal{N} |\ R_a$ *sent* **LAV**$(R_a).tag$ **wait**
9 **for all** $cl_a \in \mathcal{N}$ **do**
10 **send(**$R_i \ominus R_a$, R_a**)**;
11 **Until** $\forall cl_a \in \mathcal{N} |\ R_a$ *sent* $R_a \ominus R_i$ **wait**
12 **for all** $cl_a \in \mathcal{N}$ **do**
13 **LAV**$(R_i) = R_i \oplus (R_a \ominus R_i)$;
14 **for all** $R_j \in cl_i$ **do**
15 **send(**$R_i \ominus R_j$, R_j**)**;

Before synchronizing its LAV with other clusterheads, each clusterhead needs to get any additional information that its cluster members might have. In lines 1-2, a clusterhead R_i sends to members of its cluster a message with its LAV tag. Then, the members send to R_i any additional information. Let R_j be a member of cluster cl_i, which has R_i as clusterhead. R_j sends to R_i the set $R_j \ominus R_i$.

After all members answer R_i (line 3), it updates its LAV with the information sent by the members (line 4-5). Let $R_1, R_2, \cdots, R_n$ be the members of a cluster cl_i. R_i's updated LAV to $\mathbf{LAV}(R_i')$, defined in

equation 3.35.

$$\mathbf{LAV}(R_i') = R_i \oplus R_1 \oplus R_2 \oplus \cdots \oplus R_n \tag{3.35}$$

Next, R_i shares its new LAV tag with the other clusterheads (lines 6-7). After R_i receives the tags of all clusterheads in the network (line 8), it sends to each clusterhead any additional information it has (lines 9-10). Let cl_a be the clusters in network $\mathcal{N}$, R_i sends the information $R_i \ominus R_a$ to R_a.

After receiving the sets of information from all clusterheads in the network (line 11), R_i updates its LAV (lines 12-13). Let $cl_a, cl_b, \cdots, cl_i, \cdots, cl_u$ be the clusters in network $\mathcal{N}$, R_i's updated LAV is defined by:

$$\mathbf{LAV}(R_i'') = R_i \oplus R_a \oplus R_b \oplus \cdots \oplus R_u \tag{3.36}$$

Since the LAV of the clusterheads contains all information known by all members of the cluster, $\mathbf{LAV}(R_i'')$ contains the information known by all robots in the network, which corresponds to the most up to date LAV version.

Finally, R_i sends to its members the information that each of them still lacks (lines 14-5). So, all robots in the networks converge to the most up to date LAV version. Thus, the LAV synchronization scheme enforces property 4 whenever two (or more) networks merge.

Illustrative Example

Figure 10 presents an example where robots R_9 and R_a (from the previous example presented in figure 7) establish a link. Figure 10.1 shows the new link and figure 10.2 presents the new network, $\mathcal{N}$, formed from robots in $\mathcal{N}_1$ and $\mathcal{N}_2$. Network $\mathcal{N}$ has 14 robots and is organized in three clusters, cl_4, cl_9 and cl_b. As in the first example, a sequence diagram is presented in figure 11 to illustrate the execution of HSM LAV synchronization scheme.

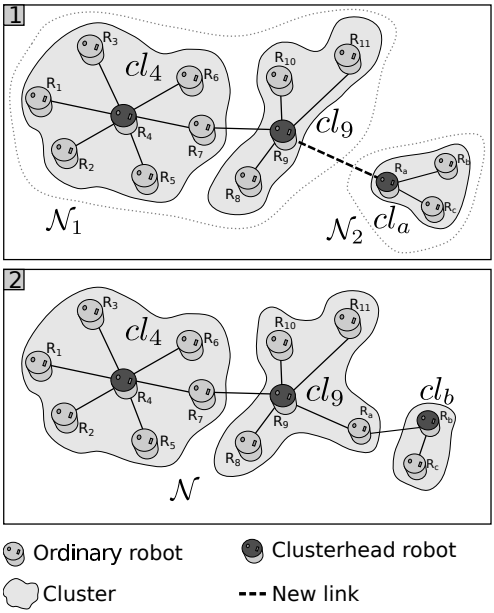


Figure 10 – Network topology.

In the sequence diagram presented in figure 11, we illustrate the execution of LAV synchronization considering the viewpoint of R_9. The execution by the other clusterheads is similar.

At the first step of HSM LAV synchronization scheme, clusterheads send their LAV tags to their members (R_9's LAV tag message in figure 11). Then, each member R_j sends any additional information to its clusterhead ($R_j \ominus R_9$). Next, clusterheads update their LAVs and, through multi-hop communication, exchange their new LAV tags (cl_i's LAV tag messages among clusterheads). Then, each clusterhead sends to the others any information they still lack ($cl_i \ominus cl_j$ messages). Finally, each clusterhead sends to members of its cluster the information that they need to update their LAVs ($cl_9 \ominus R_j$ messages). At the end of HSM LAV synchronization scheme, all robots in $\mathcal{N}$ converge to the same and most up to date LAV version.

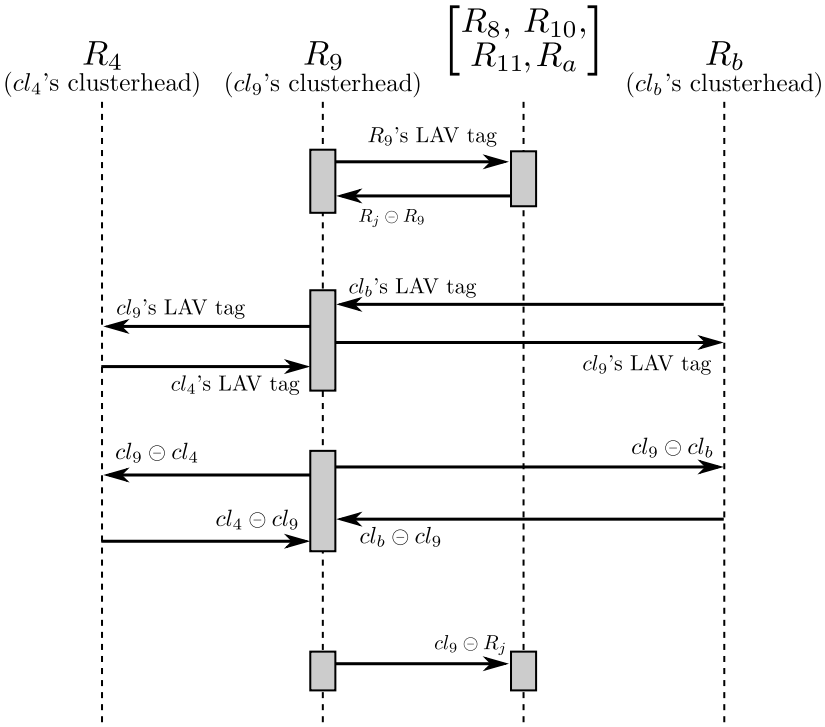


Figure 11 – Sequence diagram of a LAV synchronization.

3.5 CONCLUSIONS

In this chapter, we propose two methods for map sharing, one considering a flat *ad hoc* network (DSM) and a second method (HSM) that considers a hierarchical network architecture. Both methods are based on the concept of raw maps and can be used to share maps efficiently, in terms of time, number of exchanged messages and transmitted data. In addition, DSM and HSM can handle limited communication, guaranteeing that all robots in the same network will always converge to the most up to map of the workspace.

In DSM, robots are organized in a flat network and, whenever they get new information, they use sharing processes to share it with their neighbors, propagating the information over the network. In ad-

dition, they use processes coordinators (PCAs) to avoid deadlock and fairness problems. However, in DSM, robots cannot notice when new robots join the network and synchronize their maps in these situations. Moreover, the propagation scheme considered in DSM can increase significantly the number of messages exchanged among the robots.

On the other hand, HSM organizes robots in a hierarchical architecture and the leaders (clusterheads) become responsible for synchronize the LAVs. When robots of two network establish a link, these two networks merge in one and, next, the clusterheads synchronize the LAVs of all robots in the new network. In addition, by centralizing information exchanging in the leaders, HSM avoids the exchange of unnecessary messages.

Next, in chapter 4, we present the result of experiments with DSM and HSM. Experiments were also performed with Sheng's method. At the end of chapter 4 we discuss the results.

4 EVALUATION OF MAP SHARING METHODS

Both Distributed Synchronization Method and Hierarchical Synchronization Method have been extensively tested in simulation. The goal of experiments is to validate and evaluate the performance of DSM and HSM. Results of these experiments are also compared with the method proposed by Sheng *et al.* (SHENG et al., 2005; SHENG et al., 2006).

We run the experiments by considering systems with different number of robots (*Experiment A*) and network topologies (*Experiment B*). Another parameter that we consider in the experiments is the amount of data in the maps (size of maps) of robots (*Experiment C*).

Moreover, each experiment considers three situations: a robot finishes exploring its frontier and detects new information; two networks merge; and several networks merge simultaneously. In the first situation, we consider a network $\mathcal{N}$ with n robots, where all robots have the same map (LAV in DSM and HSM) until a single robot detects new information. In the second situation, two networks $\mathcal{N}_1$ and $\mathcal{N}_2$ merge, forming a network $\mathcal{N}$ with n robots. Finally, in the third situation, we consider that there are n networks, each of them with a single robot, and they merge in a single network $\mathcal{N}$ with n robots. This is an extreme case of multiple networks merging simultaneously and we consider it to evaluate the worst case for DSM and HSM execution. As Sheng's method does not handle situations where several networks merge simultaneously, we do not present results for such experiments.

In multi-robots exploration based on frontiers allocation, the main goal of map sharing is to guarantee that all robots have the same information about the workspace, avoiding information inconsistency problems. So, in experiments, we verify if all robots in the network get the information and converge to the same map. Also, we measure three parameters: the convergence time (T_C), number of exchanged messages (N_{msg}) and amount of transmitted data (D_T).

This chapter is organized as follows. Section 4.1 presents the

experimental setup. Sections 4.4, 4.2 and 4.3 present the result of experiments. Finally, section 4.5 discusses the results.

4.1 EXPERIMENTAL SETUP

Experiments were performed using implementations in java of DSM, HSM and Sheng's method. A simple communication channel simulator was also implemented to allow the experiments. Based on the robots' positions and their communication radiuses, the simulator determines which robots can exchange messages. The experiment parameters, number of robots, network topology and amount of data in the maps of robots (size of maps), are also defined in the simulator. Varying these parameters, we can artificially simulate different conditions of robots exploration.

The size of a map, in DSM, HSM and Sheng's method, is defined by the number of frontiers explored by each robot and the number of cells detected in each exploration. The number of frontiers explored by a robot R_i corresponds to the number of sets of detected cells associated with R_i, ΔM_{iq}. From the number of cells detected by the robots, we can estimate the average size (number of cells) of ΔM_{iq} sets. Based on these information, we can generate maps for robots in order to simulate the maps of robots at different instants of exploration.

From experiments with the multi-robots exploration method proposed in this thesis (presented in chapter 6) performed in a $100 \times 50m^2$ workspace, we estimate the number of frontiers explored by each robot in the end of exploration as $NF_{max} = 150 \pm 30$, where 150 corresponds to the average value and 30 indicates the deviation in the number of frontiers explored by each robot. The average size of ΔM_{iq} sets was estimated as 20 cells.

In the following sections, we present three experiments, which evaluate how the number of robots in the system, network topology and size of maps can influence the performance of the map sharing schemes.

4.2 EXPERIMENT A: INFLUENCE OF THE NUMBER OF ROBOTS

In *experiment A*, we evaluate how increasing the number of robots influences the performance of the methods for map sharing. To do so, we run experiments considering systems ranging from 5 to 30 robots. Moreover, three situations are considered in experiment A: a robot finishes exploring its frontier and detects new information; two networks merge; and several networks merge simultaneously. Figure 12 shows the network topologies considered in the experiments.

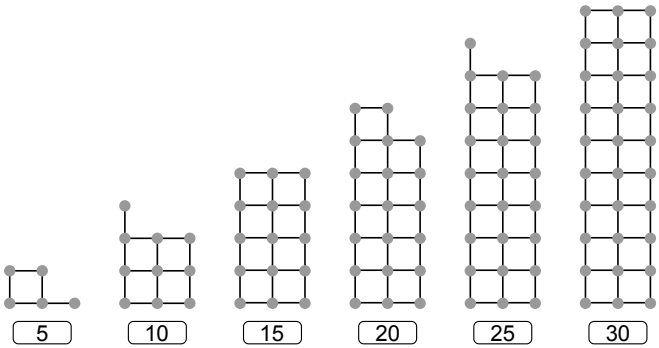


Figure 12 – Networks topologies in experiment A.

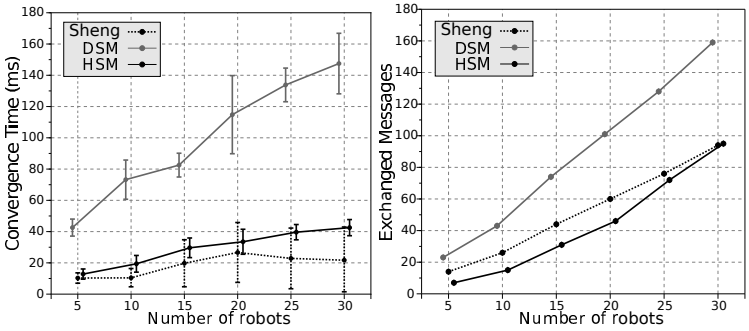
Figure 12 presents robots as network nodes (gray dots) and their communication links with other robots (black lines).

Regarding the size of maps, we consider that robots explored 75 ± 15 ($0.5NF_{max}$) frontiers, which corresponds to situations where 50% of the workspace was already explored by robots. Let n be the number of robots in the system, this means that the *raw maps* of robots have n columns, each of them with a number of ΔM_{ik} sets ranging from 60 to 90 (75 ± 15). In experiments, we used a simple generator of random numbers to define the number, within the range $60 - 90$, of ΔM_{iq} sets for each robot. The size of ΔM_{ik} sets is 20 cells.

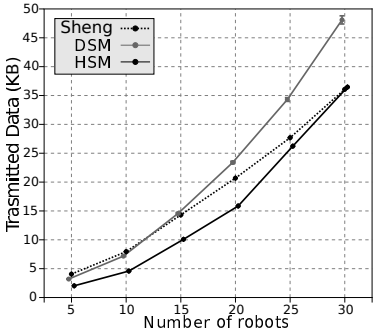
For each configuration, we ran 20 trials and calculate the average values and standard deviation of each metric. Figures 13, 14 and 15 present the results.

In figure 13, we present the convergence times (T_C, in millisec-

onds), number of exchanged messages (N_{msg}) and transmitted data (D_T, in *Kilobytes*) when a single robot detects new information.



(a) Convergence times. (b) Exchanged messages.



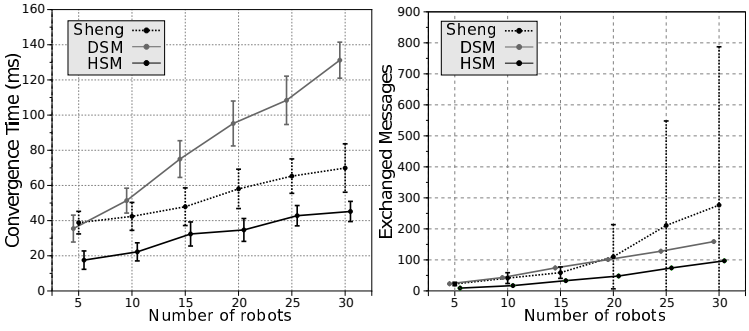
(c) Transmitted data.

Figure 13 – Results obtained in experiment A considering that a robot detects new information.

Figure 13a shows that, regarding the convergence time (T_C), both HSM and Sheng's method scale well with the number of robots, while T_C tends to increase faster in experiments with DSM. On the other hand, the number of exchanged messages and the amount of transmitted data increases significantly with the number of robots in all methods, with HSM and Sheng's method having the best performance (figures 13b and 13c).

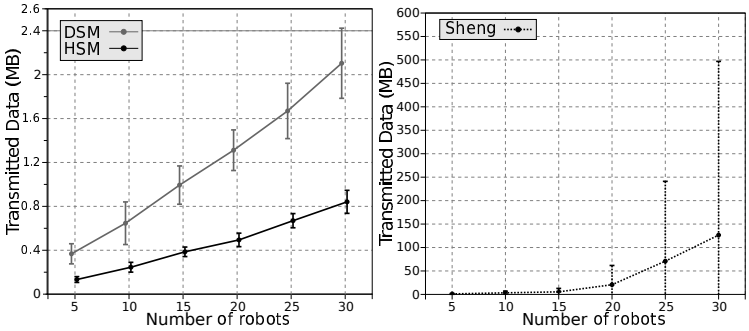
In figure 14, we present the result of experiments that considers two networks merging. Because the amount of transmitted data is much higher in Sheng's method than in DSM and HSM, we present it

separately in figure 14d.



(a) Convergence times.

(b) Exchanged messages.



(c) Transmitted data.

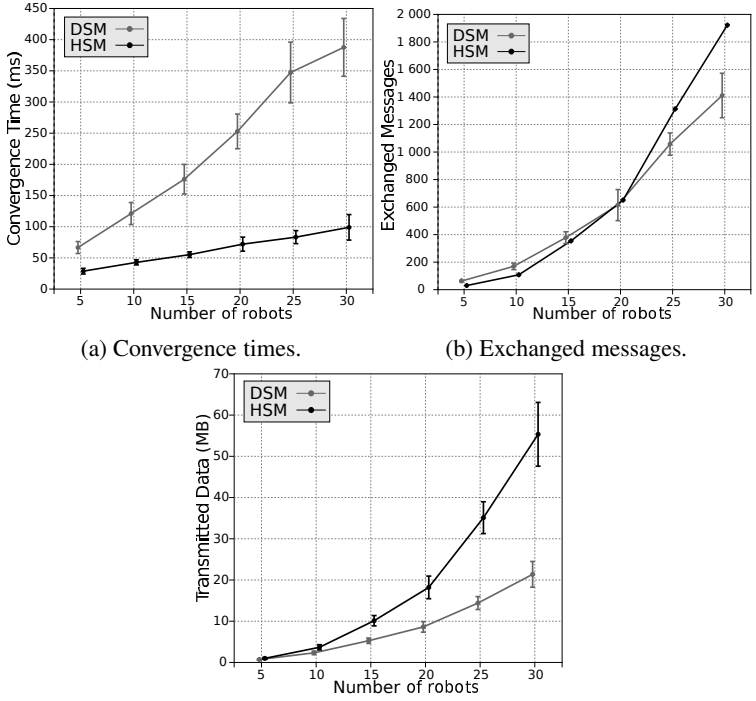
(d) Transmitted data in Sheng's method.

Figure 14 – Results obtained in experiment A considering that two networks merge.

Figure 14 shows that HSM have the best performance when two networks merge, considering the topology and amount of data in experiment A. Sheng's method have the worst performance in number of exchanged messages and transmitted data. Both N_{msg} and D_T increase faster with the number of robots in Sheng's method. In addition, the number of exchanged messages and, as a consequence, the amount of transmitted data can variate significantly in the trials of experiments with Sheng's method, as shown in figures 14b and 14d.

In figure 15, we present the result of experiments that considers several networks merging. In this experiment, we considered the

extreme case where each network has a single robot. Because Sheng's method cannot handle situations where several networks merge, we present only results of experiments with DSM and HSM.



(a) Convergence times.

(b) Exchanged messages.

(c) Transmitted data.

Figure 15 – Results obtained in experiment A considering that several networks merge.

Regarding the experiment where all networks merge, figure 15 shows that, despite HSM has had a better performance in terms of convergence time, the number of messages and transmitted data can increase faster with the number of robots in HSM than in DSM (figures 15b and 15c).

4.3 EXPERIMENT B: INFLUENCE OF NETWORK TOPOLOGY

In *experiment B*, we evaluate how the topology of networks influences the performance of the methods for sharing maps. Specifically, the performance of DSM, HSM and Sheng's method are evaluated by considering network with different distances, in communication hops, among robots. To do so, we perform experiments considering a system with 25 robots, ΔM_{ik} sets with 20 cells and that each robot explored 75 ± 15 frontiers. Figure 16 presents the topology of networks considered in the experiments.

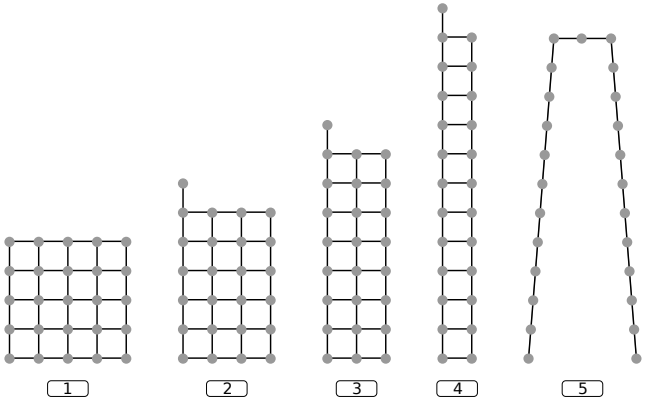


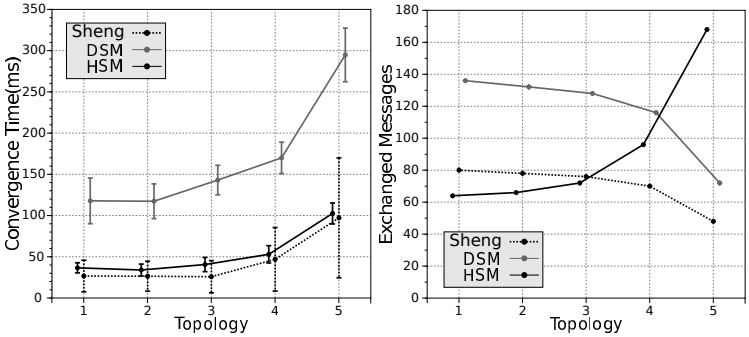
Figure 16 – Network topologies in experiment B.

In topology 1, the average distance among the robots is $\mathbf{d_{comm_1}} = 4.16$ hops while the maximum distance is $d^{max}_{comm_1} = 8$ hops. For topologies 2, 3, 4 and 5, the average and maximum distances are $\mathbf{d_{comm_2}} = 4.29$, $\mathbf{d_{comm_3}} = 4.62$, $\mathbf{d_{comm_4}} = 5.62$ and $\mathbf{d_{comm_5}} = 9.28$ hops and $d^{max}_{comm_2} = 9$, $d^{max}_{comm_3} = 10$, $d^{max}_{comm_4} = 13$ and $d^{max}_{comm_5} = 24$ hops, respectively.

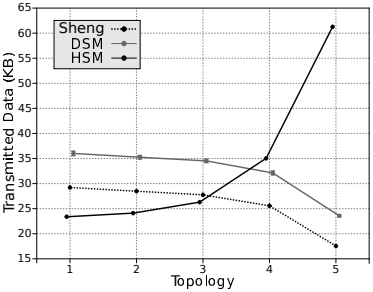
For each configuration, we ran 20 trials and figures 17, 18 and 19 present the average values and standard deviation of each metric.

In figure 17, we present the convergence times, number of exchanged messages and transmitted data (in *Kilobytes*) when a single robot detects new information.

Figure 17a shows that the convergence time increases signifi-



(a) Convergence times. (b) Exchanged messages.



(c) Transmitted data.

Figure 17 – Results obtained in experiment B considering that a robot detected new information.

cantly with the distance among robots. The performance of HSM and Sheng's methods is similar for this metric and better than DSM's performance. On the other hand, both the number of exchanged messages and transmitted data decreases when the distance increases in experiments with DSM and Sheng's method, while it increases for HSM.

This occurs because DSM and Sheng's method are based on propagation schemes, instead of relying on multi-hop communication. Thus, the amount of unnecessary messages can increase when robots have more neighbors. As a consequence, robots can share their maps efficiently using DSM and Sheng's method in topologies as the one shown in figure 16.5, where there is a single possible flow for information sharing.

When robots have multiple links, there are several possible flows of information and robots can receive the same information from different sources. In that case, the number of unnecessary exchanged messages and the amount of transmitted data increases in experiments with DSM and Sheng's method. As DSM uses the PCAs to coordinate the execution of *sharing processes*, the amount of transmitted data do not increase as much as the number of messages. In Sheng's method, where robots propagate all new information they get, both the number of messages and the amount of transmitted data can increase significantly (figures 17b and 17c).

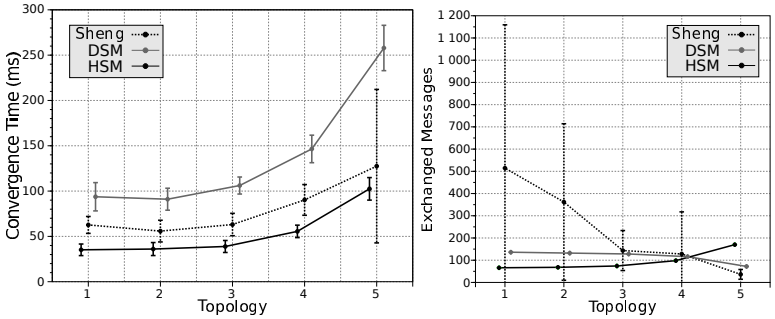
Regarding HSM, the method considers a hierarchical network structure and multi-hop communication. So, robots have to relay messages to allow the communication between clusterheads in different sides of the network. Thus, in situations where the networks topology is similar to a chain (as the one presented in figure 16.5), the number of relayed messages can increase significantly (figure 17b).

In figure 18, we present the result of experiments that considers two networks merging. Because the values of the amount of transmitted data can be very high in Sheng's method, we present them in figure 18d.

Figure 18 shows that HSM have the best performance in terms of convergence time, number of exchanged messages and transmitted data when two networks merge. Sheng's method have the worst performance in number of exchanged messages and transmitted data. Both N_{msg} and D_T can be very large in Sheng's method when robots have many neighbors. In addition, the number of exchanged messages and, as a consequence, the amount of transmitted data can variate significantly in the trials of experiments with Sheng's method, as shown in figures 18b and 18d.

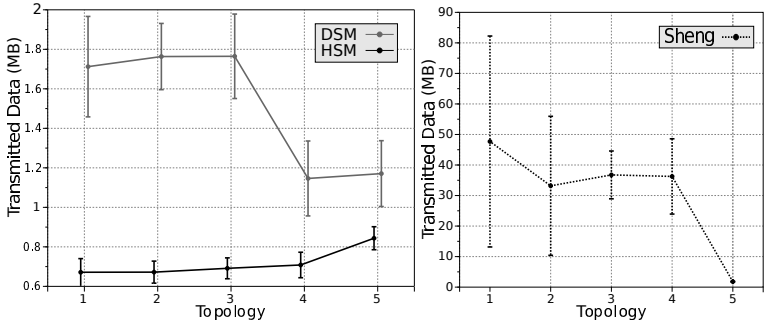
In figure 19, we present the result of experiments that considers several networks merging simultaneously. As in experiment A, we present only the results of experiments with the DSM and HSM.

Regarding the experiment where all networks merge, figure 19 shows that, despite HSM has had a better performance in terms of



(a) Convergence times.

(b) Exchanged messages.



(c) Transmitted data.

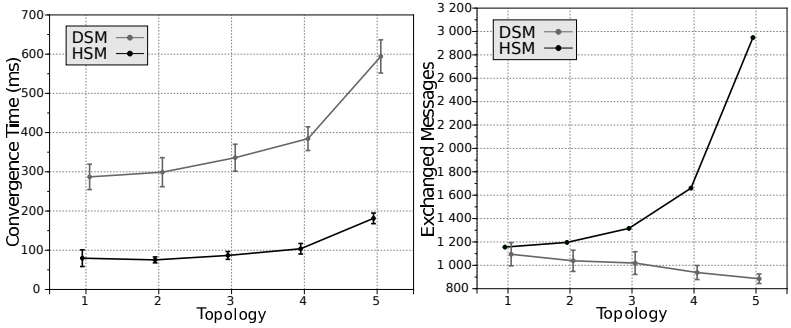
(d) Transmitted data in Sheng's method.

Figure 18 – Results obtained in experiment B considering that two networks merge.

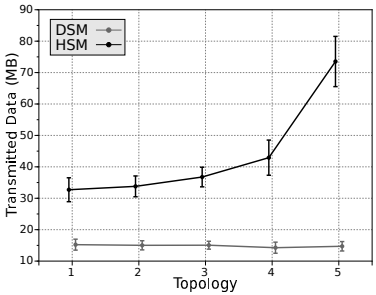
convergence time, the number of messages and transmitted data can increase exponentially with the distance among robots in experiments HSM (figures 15b and 15c). On the other hand, the number of exchanged messages and transmitted data is not significantly influenced by the network topology in experiments with DSM.

4.4 EXPERIMENT C: INFLUENCE OF THE SIZE OF MAPS

In *experiment C*, we evaluate how the size of maps influences the methods for map sharing. To do so, we consider that sets ΔM_{iq} have 20 cells and the number of frontiers explored by each robot ranges from 1% of NF_{max} (rounded to 2 ± 1) to 100% of NF_{max} (150 ± 30). I addition, we



(a) Convergence times. (b) Exchanged messages.



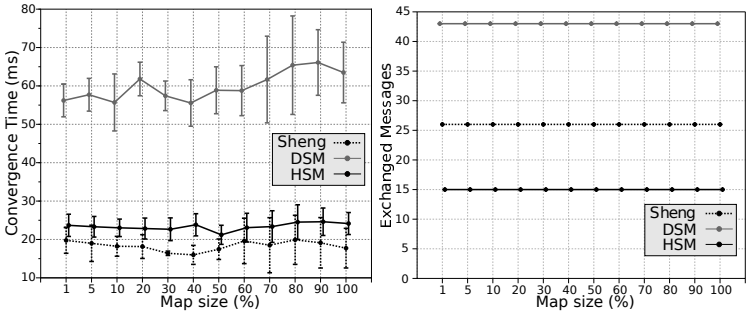
(c) Transmitted data.

Figure 19 – Results obtained in experiment B considering that several networks merge.

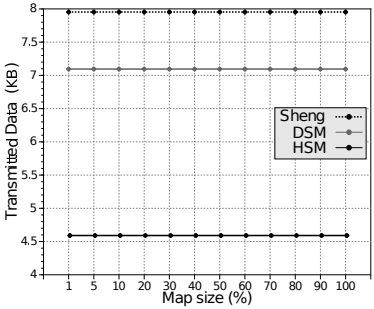
consider a system with 10 robots. The network topology is the same presented in figure 12 (experiment A) when the number of robots is 10. For each configuration, we ran 20 trials and figures 20, 21 and 22 present the average values and standard deviation of each metric.

In figure 20, we present the convergence times, number of exchanged messages and transmitted data (in *Kilobytes*) when a single robot detects new information.

Figure 20 shows that HSM has the best performance in terms of exchanged messages and transmitted data, while having convergence times similar to Sheng's. As the robot that detected information is fixed in experiment C, the number of exchanged messages is always the same for all methods. Since the amount of data detect is the same in all trials (a set ΔM_{ij} with 20 cells), the transmitted data is not influenced



(a) Convergence times. (b) Exchanged messages.



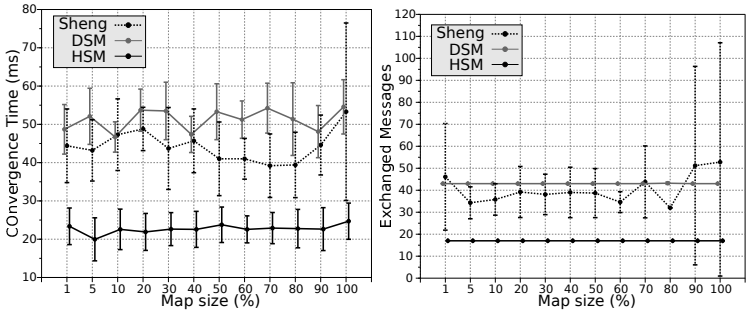
(c) Transmitted data.

Figure 20 – Results obtained in experiment C considering that a robot detects new information.

by the amount of data previously detected either.

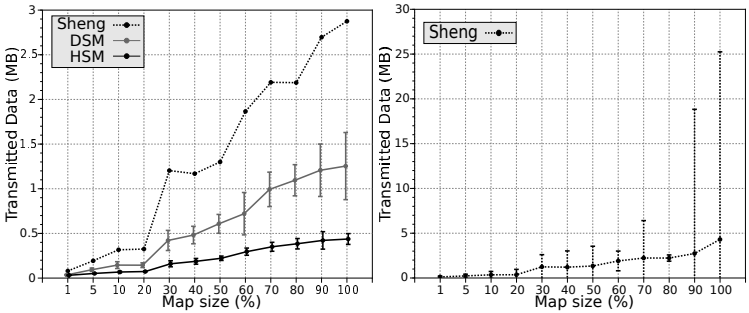
In figure 21, we present the result of experiments that considers two networks merging simultaneously. As in experiment A, figure 21 presents only the average values of D_T in Sheng's method. Figure 21d shows the average values and standard deviation of the amount of transmitted data in Sheng's method.

Figure 21 shows that HSM has the best performance in terms of convergence time, number of exchanged messages and transmitted data when two networks merge. As in experiment A and B, Sheng's method have the worst performance in number of exchanged messages and transmitted data. Both the number of exchanged messages and, as a consequence, the amount of transmitted data can variate significantly



(a) Convergence times.

(b) Exchanged messages.



(c) Transmitted data.

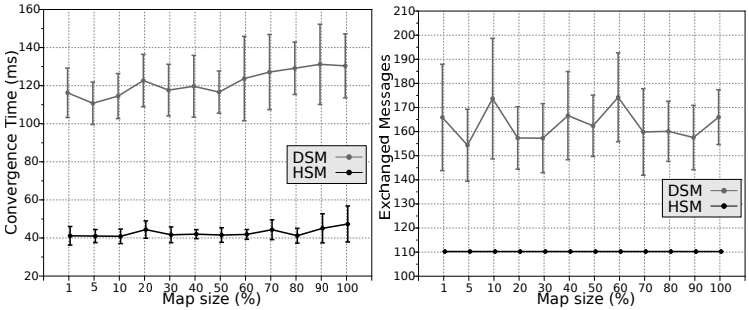
(d) Transmitted data in Sheng's method.

Figure 21 – Results obtained in experiment C considering that two networks merge.

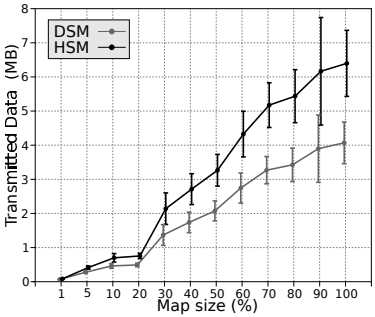
in the trials of experiments with Sheng's method, as shown in figures 21b and 21d.

In figure 22, we present the convergence times, number of exchanged messages and transmitted data when several networks merge simultaneously. As in experiments A and B, we present only the results of experiments with the DSM and HSM.

Regarding the experiment where all networks merge, figure 19 shows that HSM has a better performance in terms of convergence time and the number of messages than DSM (figures 22a and 22b). However, the amount of transmitted data can increase faster in experiments with HSM (figure 22c).



(a) Convergence times. (b) Exchanged messages.



(c) Transmitted data.

Figure 22 – Results obtained in experiment C considering that several networks merge.

4.5 DISCUSSIONS

The results obtained from experiments allowed us to verify that both the DSM and HSM are able to share map information efficiently, keeping the LAVs of robots in the same network synchronized.

In situations where a robot finishes exploring its frontier, only the new information is shared and HSM and Sheng's method have a similar performance (except in experiment B). On the other hand, DSM usually presents the worst convergence time, number of exchanged messages and transmitted data.

When two networks merge, the amount of map information that needs to be exchanged depends on the size of maps. In these situations, the efficiency of Sheng's method decreases significantly when the

number of robots or the size of maps increases. DSM and HSM have similar performances in these situations, with HSM usually having better results than DSM. However, in cases where several networks merge, DSM can surpass HSM.

Sheng *et al.* (SHENG et al., 2006) propose the *raw map* concept and a synchronization scheme that allows two robots to exchange map information efficiently when their networks merge. However, as robots use a propagation scheme to share the information with other robots in the network, they exchange a large number of unnecessary messages with map information. Thus, the amount of data transmitted by robots executing Sheng's method usually is much higher than in DSM and HSM, as shown in figures 14c, 14d, 18c, 18d, 21c and 21d.

Regarding the DSM, this method is based on the execution of several *sharing processes*, which need to be coordinated (by the Processes Coordinator Algorithm) in order to avoid *deadlocks* and information inconsistency. Thus, DSM can take longer to converge and exchange more messages than Sheng's method and HSM.

However, most messages exchanged by robots executing DSM have only LAV tag information, an array of integers with size n (number of robots). So, even when robots executing DSM exchange a large number of messages, the amount of transmitted data is usually smaller than in Sheng's method. In extreme situations, where several networks (with a single robot each) merge, the amount of transmitted data in DSM can be even smaller than in HSM (figures 15c, 19c and 22c).

In HSM, leaders coordinate the map sharing schemes. So, because HSM does not use propagation schemes or *sharing processes* that need to be coordinated, the LAVs of robots converge quickly and usually with the smallest amount of exchange messages and transmitted data.

When several networks with a single robot each merge (extreme situation evaluated in the experiments, whose results are presented in figures 15c, 19c and 22c), HSM's performance can be worse than the performance of DSM. This occurs because, in HSM, clusterheads rely on multi-hop communication to share maps with other clusterheads.

Thus, depending on the network topology and number of networks merging, robots might need to relay a large number of messages with map information and HSM can be less efficient than DSM.

However, we highlight that, the most common situations found in multi-robots exploration are robots sharing information after explores a frontier and two networks merging. In these situations, HSM is more efficient than both DSM and Sheng's method.

5 PROPOSAL OF A HIERARCHICAL METHOD FOR MULTI-ROBOT EXPLORATION

This chapter proposes a method to coordinate multiple robots in exploration tasks considering robots with limited communication radius, Hierarchical *K*-Means (HKME) method for multi-robots exploration, an extension of KME (PUIG; GARCÍA; WU, 2011) that handles link losses due to limited communication radius. HKME considers that robots in a network have the same LAV version and uses HSM method for map sharing to guarantee that.

Section 5.1 presents the problem description. Next, both a general description of HKME (section 5.2) and a detailed description of its phases (section 5.3) are presented.

5.1 PROBLEM DESCRIPTION

Coordination of robots is the core of exploration tasks, defining strategies that robots will use to explore the workspace. Coordination, which can be achieved in centralized or distributed ways, involves the identification and assignment of exploration targets to robots, fulfilling specific aspects of the application.

The main objective of this work is to develop an exploration method that minimizes exploration time, avoids redundant exploration, balances the workload of robots and disperse the robots quickly through the workspace. In addition, we consider that robots have limited communication radius and we do not assume any communication infrastructure in the workspace. So, robots can be separated in several unconnected *ad hoc* networks.

Due to the possibility of link losses, robots can be separated into different unconnected networks. Several works based on centralized schemes, such as (BURGARD et al., 2000; BURGARD et al., 2005; STACHNISS; MOZOS; BURGARD, 2008; WURM; STACHNISS; BURGARD, 2008), handle this problem by executing their centralized methods independently in each network. Specifically, they define a leader for each network and use routing

schemes to allow the exchange of messages between robots and leaders. However, routing schemes do not scale well and problems related to message losses are difficult to handle in flat networks. Therefore, these methods can be very costly in terms of both communication and computational power (DHURANDHER; SINGH, 2005; ABBASI; YOUNIS, 2007).

Other methods, such as (YAMAUCHI, 1998; SARIEL; BALCH, 2006; FRANCHI et al., 2009), use fully distributed schemes to coordinate the robots instead of relying on central units. However, most authors do not address problems related to multi-hop communication in *ad hoc* networks. Moreover, these methods can be costly in terms of number of messages and amount of data transmitted.

In (PUIG; GARCÍA; WU, 2011), Puig *et al.* propose a multi-robot exploration method based on centralized *K*-means (KME), which implements a policy that optimizes the exploration at a global level. The method improves the exploration efficiency by minimizing three aspects: the sum of traveled distances, the variance of the length of paths and the variance of the arrival times at all regions of the workspace. Despite KME can balance the workload among the robots and perform exploration efficiently, it cannot handle communication link losses.

This work proposes Hierarchical *K*-Means (HKME), a method for multi-robot exploration that extends KME (PUIG; GARCÍA; WU, 2011) in order to handle communication losses due to limited communication radius. No pre-existing communication infrastructure (routers, access points, etc.) is assumed in the workspace to allow the exchange of messages among robots. Instead, communication has to be achieved using the robots themselves to relay messages, thus acting as nodes of a mobile *ad hoc* network.

In HKME, cluster formation and maintenance algorithms group robots into clusters, defining a hierarchical network architecture. By doing so, we define a scalable mechanism to handle robots communication. This hierarchical organization of the robots also helps improve efficiency, making clusterheads responsible for executing workspace partitioning and for assigning regions and frontiers to the members of their

clusters.

Since robots in different networks cannot exchange messages, HKME runs independently in each network. However, HKME guarantees that, even when robots are separated in several networks, regions assigned to different robots do not overlap, which could decrease efficiency by assigning robots to the same areas.
In this work, the following premises are assumed:

- The workspace boundaries are known. Despite robots do not known the workspace, they known its limits.
- Robots communication system has a limited radius. If the distance between two robots is smaller than the communication radius, they a direct communication link.
- Messages sent to robots within this radius are always received.
- Robots do not fail.

5.2 HIERARCHICAL K-MEANS FOR MULTI-ROBOTS EXPLORATION

Similarly to KME, the basic idea of HKME is to partition the workspace into regions and allocate them to robots. Then, robots are assigned to explore frontiers that take them closer to their regions. This scheme coordinates the robots in order to reach all regions of the workspace as soon as possible. The phase diagram presented in figure 23 summarizes the execution of HKME.

HKME partitions the workspace into two levels: *cluster sectors* and *regions*. As in KME, a region is the area assigned to a robot. A cluster sector is the union of all regions assigned to the members of a cluster (cluster sectors are described in subsection 5.3.2).

In the *global partitioning* phase, clusterheads execute an iterative scheme to partition the workspace into cluster sectors. Then, HKME enters the *local partitioning* phase, where each clusterhead partitions its corresponding sector into as many regions as robots in the cluster and assigns each region to a single robot. The workspace partitioning and

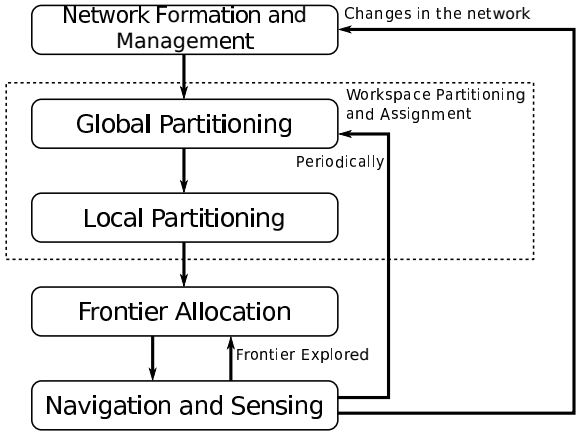


Figure 23 – Description of the HKME.

assignment task is executed periodically, reshaping the regions assigned to the robots as they explore the workspace.

After all regions have been assigned, clusterheads assign new frontiers to members of their clusters in the *frontier allocation* phase. Next, robots calculate a path to their frontiers and start to move toward them (*navigation and sensing* phase). When a robot reaches its assigned frontier, it scans its surroundings, updates its local map and removes the detected cells from its corresponding region. Then, HKME enters the *frontier allocation* phase and the clusterhead assigns a new frontier to the robot.

As the robots move through the workspace, communication links can be lost or established, hence changing the network. In that case, the robots in the network go to the *network formation and management* phase, in which common algorithms for cluster formation and maintenance are used to redefine the network. After handling the changes in the network, HKME enters the global partitioning phase.

Since HKME assumes that robots always have an assigned region, it executes the centralized K-means algorithm at the beginning of the exploration to define an initial partitioning and assignment of the workspace. This *setup* phase is run off-line.

Figure 25 illustrates the execution of HKME, showing how the clusterheads partition the workspace, assign regions and allocate frontiers to the robots. In this work, clusters and sectors are named based on their clusterhead identifiers. In figure 25, for instance, cs_8 is the sector associated with cluster cl_8, whose clusterhead is R_8. The example assumes that two robots, R_2 and R_3, lose communication when the robots of the system have reached the configuration shown in figure 24a. Due to this link loss, robots are separated in two networks: $\mathcal{N}_1$ and $\mathcal{N}_2$. When the network changes, HKME enters the network formation and management phase and the robots update the network. Figure 24c shows the previous organization of the robots (network $\mathcal{N}$) and the result after R_2 and R_3 lose communication (networks $\mathcal{N}_1$ and $\mathcal{N}_2$).

After updating the network, HKME enters the global partitioning phase, where clusterheads redefine the sectors of clusters. Figures 24b and 24d show those sectors before and after the global partitioning of the workspace, respectively. Afterwards, HKME enters the local partitioning phase, where each clusterhead partitions its sector into regions and assigns them to its cluster members, yielding the workspace partitioning and assignment shown in figure 24e.

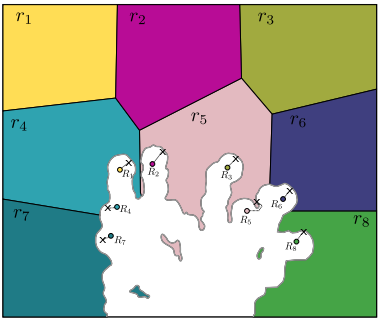
Next, HKME enters the frontier allocation phase, were clusterheads assign new frontiers to the members of their clusters. Figure 24f presents the frontiers identified in the partially explored workspace (light green dots) and the frontiers allocated to each robot (symbol **x**). Then, robots enter the navigation and sensing phase, where they move through the workspace until they reach the new assigned frontier.

The different phases of HKME are fully described in the next section.

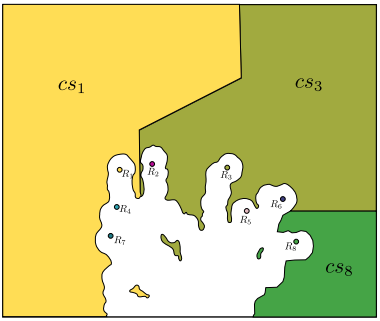
5.3 PHASES OF HIERARCHICAL K-MEANS

5.3.1 Network Formation and Management

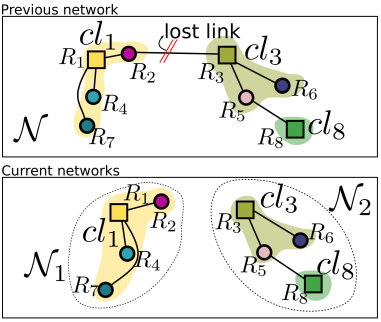
As robots move through the workspace, they can lose or establish new communication links, thus changing the net-



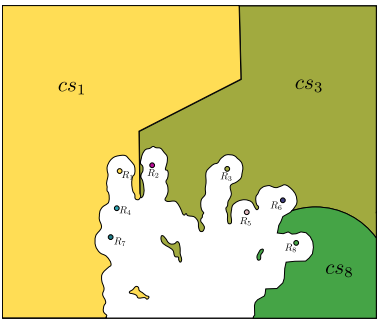
(a) Workspace partitioning.



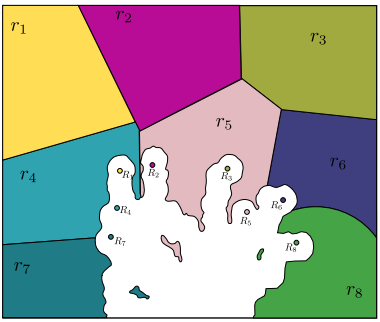
(b) Previous *cluster sectors*.



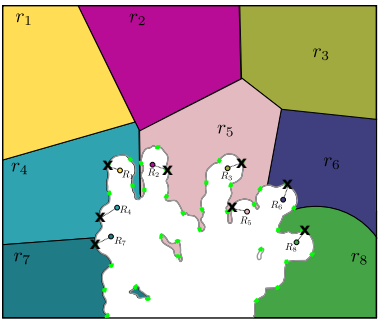
(c) Network formation and management.



(d) New global partitioning.



(e) New local partitioning.



(f) New frontier allocation.

Figure 24 – Example of HKME execution.

work configuration. In that case, the robots run a cluster formation and maintenance algorithm to reorganize the network. Any method that organizes the MRS in a hierarchical topology, such as *Lowest ID* (EPHREMIDES; WIESELTHIER; BAKER, 1987), *Highest Degree* (GERLA; TSAI, 1995) and *Weighted Clustering* (CHATTERJEE; DAS; TURGUT, 2002), can be applied to execute the network formation and management phase in HKME.

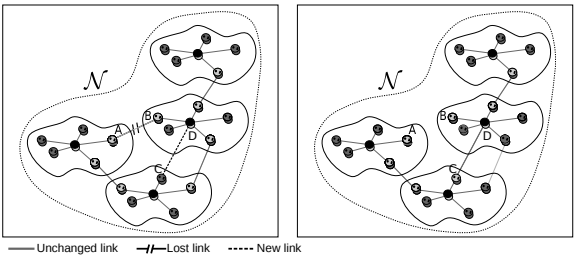
In this work, changes in the network are classified as *internal* or *connective*. An internal change occurs when the establishment or loss of links do not add or remove robots from the network. Only the internal structure of the network changes. Connective changes occur when the establishment or loss of links results in two networks being merged into one, or in a network being split in two unconnected ones. Figure 25 illustrates the two types of network changes.

In figure 25a, we represent the loss of the link between nodes A and B and the establishment of a new link between nodes C and D of network $\mathcal{N}$. Despite the changes, the network remains connected and with the same robots. In figure 25b, the loss of the link between nodes A' and B' partitioned network $\mathcal{N}$ in two unconnected ones, $\mathcal{N}_a$ and $\mathcal{N}_b$. The first network contains the robots in clusters cl_a, cl_b and cl_c, and the second one is formed by the robots in cl_d. The example presented in figure 25c shows the case in which two robots (nodes A'' and B'') from different networks ($\mathcal{N}_a$ and $\mathcal{N}_b$) establish a link. In that case, $\mathcal{N}_a$ and $\mathcal{N}_b$ will merge in a single network, $\mathcal{N}$.

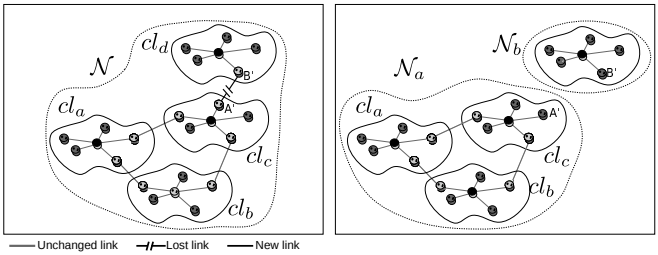
Subsections 5.3.2, 5.3.4 and 5.3.3 point out how the different types of network changes influence the workspace partitioning, region assignment and frontier allocation phases.

5.3.2 Global Partitioning

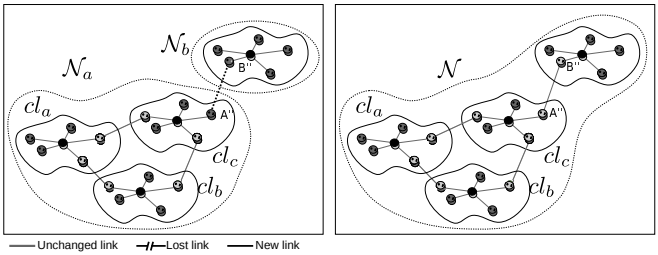
In the global partitioning phase, clusterheads execute a distributed algorithm to partition the workspace into cluster sectors, whose size is proportional to the number of members in every cluster. The following steps are executed:



(a) Internal changes.



(b) Connective changes: link loss.



(c) Connective changes: new link.

Figure 25 – Network changes.

1. Each clusterhead defines its cluster sector;

2. The clusterheads share the weight ω_i (presented next) and centroid coordinates with the other clusterheads in the network;

3. Based on the weights and centroids of all sectors, each clusterhead redefines the ownership of its cells, giving away the ones that are closer to other sectors;

4. Each clusterhead updates its sector, calculates the new centroid

as the sector's center of mass and shares the new centroids;

5. If the exit condition is satisfied, end the global partitioning phase, otherwise, iterate from step 3.

Cluster Sector Definition

A cluster sector cs_j associated with cluster cl_j is defined by the tuple $\langle \mathbf{Cs}_j, ct_j, \omega_j \rangle$, where $\mathbf{Cs}_j$ is the set of cells belonging to the regions of all robots in cl_j, ct_j is the sector's centroid and ω_j its weight. $\mathbf{Cs}_j$ and ω_j are defined as:

$$\mathbf{Cs}_j = \bigcup_{R_i \in cl_j} \mathbf{C}_i \tag{5.1}$$

$$\omega_j = \sqrt{|cl_j|}\ , \tag{5.2}$$

where $\mathbf{C}_i$ is the set of cells in the region r_i assigned to robot R_i and $|cl_j|$ is the number of robots in cluster cl_j at the instant in which the global partitioning is executed.

At the beginning of the global partitioning phase, each clusterhead defines its cluster sector cs_j, with $\mathbf{Cs}_j$ and ω_j defined as (5.1) and (5.2). The initial coordinates of the sector's centroid (ct_j), referred to as *seed*, are defined as the mean value of the positions of the robots belonging to the cluster (the cluster centroid). The coordinates of the cluster centroid are calculated as:

$$ct_j = \left(\frac{1}{|cl_j|} \sum_{R_i \in cl_j} x_{R_i}, \frac{1}{|cl_j|} \sum_{R_i \in cl_j} y_{R_i} \right)\ , \tag{5.3}$$

where x_{R_i} and y_{R_1} are the 2D coordinates of robot R_i.

Redefining ownership of cells

The basis of HKME global partitioning is the redefinition of the ownership of cells, which allows clusterheads to partition the workspace into sectors proportional to the number of robots in their clusters, while

avoiding that sectors overlap when robots are separated in several networks.

In HKME, each sector cs_i has a weight ω_i, defined in (5.2), associated with the number of robots in the cluster. To redefine the ownership of cells, each clusterhead verifies if there are cells in its sector that should be transferred to another cluster sector. In that case, it sends them to the respective clusterhead. Let cs_j be the cluster sector associated with cluster cl_j and c a cell in cs_j ($c \in \mathbf{Cs}_j$). The new owner of c will be the sector cs'_j that fulfills:

$$\forall cl_k \in \mathscr{N} \; : \; \frac{\text{dist}(c, ct'_j)}{\omega'_j} \leq \frac{\text{dist}(c, ct_k)}{\omega_k} \; , \tag{5.4}$$

where $dist(c, ct_k)$ is the Euclidean distance between cell c and the centroid of cs_k.

By using (5.4) to redefine the ownership of cells, clusterheads minimize function D, defined as:

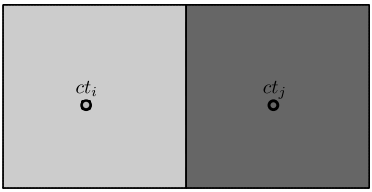
$$D = \sum_{cl_i \in \mathscr{N}} \sum_{j \in \mathscr{C}} a_{i,j} \frac{\text{dist}(j, ct_i)}{\omega_i} \tag{5.5}$$

$$\mathscr{C} = \bigcup_{cl_k \in \mathscr{N}} \mathbf{Cs}_k \; , \tag{5.6}$$

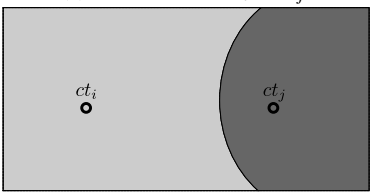
where cl_i is a cluster from network $\mathscr{N}$ and $\mathscr{C}$ is the set of cells owned by all clusters in $\mathscr{N}$. Coefficient $a_{i,j}$ is one if cell j belongs to sector cs_i and zero otherwise.

By minimizing D, HKME minimizes the sum of the weighted distances between cells and centroids. Moreover, since HKME considers weighted distances, it generates larger sectors for clusters with more robots. Figure 26 illustrates the influence of the weighting parameter ω_i in the global partition.

Let ct_i and ct_j respectively be the centroids of sectors cs_i and cs_j, and ω_i and ω_j their weights. If $\omega_i = \omega_j$, cs_i and cs_j will have similar areas, figure 26a. However, if $\omega_i > \omega_j$, some cells that are closer to ct_j will be transferred to cs_i, which will end up having a larger area than cs_j, figure 26b. A particularity of the weighted partitioning scheme



(a) Sectors when $\omega_i = \omega_j$.



(b) Sectors when $\omega_i > \omega_j$.

Figure 26 – Example of sectors partitioning.

proposed in this work is the rounded shape of the resulting sectors. In KME, the workspace partitions (regions) are always polygons, usually with four or five sides.

Exit Condition

Let $D(T)$ be the value of D, defined in (5.5), at iteration T of the global partitioning phase. The exit condition is satisfied when one of the following conditions is satisfied:

1. The maximum number of iterations M is reached;

2. The variation of $D(T)$ is smaller than a threshold $\Delta > 0$:

$$D(T) - D(T-1) \leq \Delta$$

In centralized approaches, the above conditions are verified by a central unit. In HKME, maximum consensus is used to verify the first condition, while average consensus is used to verify the second. To do so, after step 4, clusterheads exchange the number of iterations

that have already been executed and their $D(T)$ partial values, $D_j(T)$, defined as:

$$D_j(T) = \sum_{c \in \mathbf{Cs}_j} \frac{\text{dist}(j, ct_i)}{\omega_i} \tag{5.7}$$

Based on this information, each clusterhead can verify if at least one of the exit conditions is satisfied. The first condition is true if, for all clusterheads in the network, the number of executed iterations is greater than or equal to M. To verify the second condition, clusterheads use the $D_j(T)$ values received from the other clusterheads to calculate the value of $D(T)$ as:

$$D(T) = \sum_{cl_j \in \mathscr{N}} D_j(T) \tag{5.8}$$

Then, if the average value of the variation of $D(T)$, according to (5.9), is equal to or smaller than Δ, the second condition is valid:

$$\frac{D(T-1) - D(T)}{|cl(\mathscr{N})|} \leq \Delta \ , \tag{5.9}$$

where $|cl(\mathscr{N})|$ is the number of clusters in the network.

5.3.3 Local Partitioning

After partitioning the workspace into sectors, each clusterhead partitions its corresponding sector into regions and assigns them to its cluster members. Let R_j be the clusterhead of a cluster cl_j, and cs_j its associated sector. R_j executes the centralized K-means algorithm proposed in (PUIG; GARCÍA; WU, 2011) within the cluster scope, thus partitioning cs_j into $|cl_j|$ regions and assigning them to its cluster members using the Hungarian algorithm.

In (PUIG; GARCÍA; WU, 2011), a central unit executes the iterative K-means algorithm to partition the workspace into K regions, where K is the number of robots. The workspace is represented by an occupancy grid map (ELFES, 1989), whose cells with *unknown* state are clustered into regions according to the following iterative process:

1. Randomly choose K unknown cells c_i, $1 \leq i \leq K$, as region centroids;

2. For every unknown cell in the workspace, calculate its Euclidean distance to the K centroids, identify the closest centroid ct_i and define the cell as part of region r_i;

3. Calculate the center of mass cm_i of each region r_i;

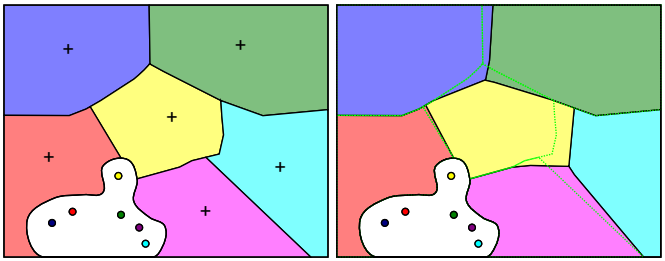
4. If $\forall i \in \{1..K\}$, $ct_i = cm_i$ (convergence condition), the process ends. Otherwise, substitute every ct_i for its corresponding cm_i and proceed from step 2.

When the above process terminates, all unknown cells are partitioned into K stable disjoint regions. In KME, a region r_i can be defined by the tuple $\langle \mathbf{C}_i, ct_i, \mathbf{cc}_i \rangle$, where $\mathbf{C}_i$ is the set of cells of the region, ct_i is the centroid and $\mathbf{cc}_i$ is the set of contour cells[1]. Figure 27 shows an example of the K-means algorithm iteration.

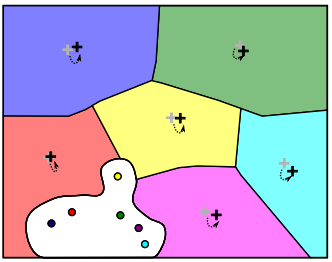
Figure 27a presents the regions and centroids corresponding to iteration $T-1$ of the algorithm. The colored circles represent the positions of the robots. In iteration T, the partition of the workspace changes to the one showed in figure 27b (green lines indicate the previous partition) after the execution of step 2. Figure 27c shows the centroids ct_i (symbol +) and the centers of mass (calculated in step 3) cm_i (black circles) of the regions. The arrows in figure 27c indicate that the centers of mass will become the new centroids at the end of iteration T. In figures 27a, 27b and 27c, the gray lines separating the regions correspond to their contour cells.

In (PUIG; GARCÍA; WU, 2011), the assignment of regions to robots is formulated as a Linear Programming (LP) problem. An LP solver is applied to obtain the region assignment that minimizes the distances between robots and regions. Let $\mathrm{d}(r_i, R_j)$ be the distance between robot R_j and region r_i, which corresponds to the minimum

[1] Contour cells are the cells that at least have one neighbor of another region or of the free space.



(a) Previous regions. (b) K-means iteration.



(c) New centroids.

Figure 27 – Example of local partitioning iteration.

distance between R_j and any cell in $\mathbf{cc}_i$ (contour of r_i). The assignment problem is described as:

$$\begin{aligned} \min \quad & \sum_{i}^{K}\sum_{j}^{K} a_{ij}\mathrm{d}(r_i, R_j) \\ s.t. \quad & \\ & \sum_{i}^{K} a_{ij} = 1 \\ & \sum_{j}^{K} a_{ij} = 1 \ , \end{aligned} \tag{5.10}$$

where K is the number of robots and regions. Coefficient a_{ij} is one if region r_i is assigned to robot R_j and zero otherwise.

5.3.4 Frontier Allocation

In HKME, frontiers are allocated by considering an on-demand scheme: every time a robot completes the exploration of a frontier, its clusterhead allocates a new frontier to it, such that the robot tends to

get closer to its assigned region. To do so, the clusterhead calculates the distance isbetween the robot and all frontiers and assigns the closest one. Let F be the set of identified frontiers. The clusterhead will assign the frontier f_l to a robot R_i iff

$$\forall f \in F \ : \ \mathrm{d}(R_i, f_l) \leq \mathrm{d}(R_i, f) \ , \tag{5.11}$$

where $\mathrm{d}(R_i, f_l)$ is the distance between R_i and f_j, described below:

$$\mathrm{d}(R_i, f_j) = \delta(R_i, f_j) + g(f_j, cc_{R_i}) + \sigma_1(f_j, cc_{R_i}) + \sigma_2(f_j) \ , \tag{5.12}$$

where $\delta(R_i, f_j)$ is the distance between robot R_i and frontier f_j, considering a path calculated using the A^* algorithm (or a similar path planning algorithm). $g(f_j, cc_{R_i})$ is the Euclidean distance between frontier f_j and cc_{R_i}, the cell from the contour of r_i that is closest to R_i, $\sigma_1(f_j, cc_{R_i})$ is a penalization equal to the workspace diagonal L if there is an obstacle between f_j and cc_{r_i} or zero otherwise, and $\sigma_2(f_j)$ is a punishment equal to a constant v (defined by the designer of the system) if f_j is already assigned or zero otherwise.

Notice that the frontier allocation phase is executed in an on-demand scheme, which considers the allocation to one robot at a time. If several robots finish exploring their frontiers at the same time, the clusterhead will allocate new frontiers sequentially, based on the time it was informed by each robot.

Since a frontier can be allocated to more than one robot, depending on the number of robots and frontiers, the state space of the frontier allocation problem can grow exponentially. Therefore, the clusterhead allocates the closest frontier to each robot instead of using an optimal method such as the Hungarian algorithm, which would have a prohibitive computational cost in this context.

5.3.5 Allocating Frontiers to Robots with Explored Regions

An important aspect of HKME is what happens after a robot explores all cells from its associated region. Let $r_i : \langle \mathbf{C}_i, ct_i, \mathbf{cc}_i \rangle$ be the

region assigned to robot R_i. After R_i explores its region, $\mathbf{C}_i = \varnothing$, $\mathbf{cc}_i = \varnothing$ and ct_i becomes the position of R_i. If other robots in the cluster still have non-empty regions ($\mathbf{C}_j \neq \varnothing$), the clusterhead can repartition the cluster sector and assign a new region to R_i. Otherwise, the robot remains with a region r_i in which $\mathbf{C}_i = \varnothing$ and $\mathbf{cc}_i = \varnothing$.

For robots with explored (or empty) regions, the values of $g(f_j, cc_{R_i})$ and $\sigma_1(f_j, cc_{R_i})$ in (5.12) are always zero and the computation of the distance between R_i and a frontier f_j can be simplified to:

$$\mathrm{d}(R_i, f_j) = \delta(R_i, f_j) + \sigma_2(f_j) \tag{5.13}$$

Thus, after a robot R_i explores its region, its clusterhead will assign frontiers to it only by considering the length of the smallest path between R_i and the frontiers, $\delta(R_i, f_j)$, and which frontiers were already assigned to other robots, $\sigma_2(f_j)$.

From this exploration with empty regions, two important aspects of HKME emerge. First, robots do not stay idle after exploring their regions. Instead, they start exploring the workspace greedily, helping other robots to explore their regions. This behavior can also take robots with already explored regions closer to clusters that still have areas pending to be explored. After a robot joins such a cluster, a new region can be assigned to it.

The second aspect is that, even if a robot fails, the cells from its region will be explored. In HKME, the workspace partitioning is based on the exchange of cells. Therefore, if a robot fails, its region's cells will not be assigned to other robots. However, after robots explore their regions, they will start to greedily explore the unknown cells owned by other robots. Thus, eventually, robots that did not fail will explore the entire workspace.

5.4 CONCLUSIONS

In this chapter, we propose an extension of KME (PUIG; GARCÍA; WU, 2011), the Hierarchical K-Means method

for multi-robots exploration. The main difference between HKME and KME is that the first defines a hierarchical scheme to handle the possibility of link losses.

In KME, a central unit partitions the workspace in regions and assigns regions and frontiers to robots. In HKME, robots in the same network are grouped in clusters and clusterheads interact to partition the areas they own in *sectors*. Then, each clusterhead partitions its sector in regions and assigns regions and frontiers to members of its cluster.

Depending on the distance among robots and the communication radius, all robots can have a direct link with the others. In situations like those, all robots are grouped in a single cluster and, as there is a single clusterhead, its sector corresponds to the entire non-explored workspace. So, the clusterhead acts as KME central unit, partitioning the workspace in regions and assigning regions and frontiers to all robots in the system.

Thus, KME can be view as a particular case of HKME, where the communication radius is large enough to guarantee that robots always have a direct link with the others.

In the next chapter, we present experiments with HKME, where the method's efficiency is evaluated when the communication deteriorates. Specifically, we define the case where all robots always have a direct link with the others as the baseline. Then, experiments are performed with different communication radius and the results compared with the ones of baseline.

6 HKME EVALUATION

HKME has extensively been tested in simulation by considering different types of workspace and communication radius. Sections 6.1 and 6.2 present the experimental setup and measures used to evaluate the proposed method, respectively. In turn, sections 6.3, 6.4 and 6.5 present the results of the experiments for each type of workspace. Finally, section 6.6 discusses the results.

6.1 EXPERIMENTAL SETUP

HKME was implemented in Java and run using the Player/Stage simulator (GERKEY; VAUGHAN; HOWARD, 2003). Experiments were conducted by considering an MRS with 10 simulated *Pioneer2DX* robots equipped with a 360° (a sample per 2°) laser sensor with a range of 3 meters. The robots' velocity bounds are $1.6^{m}/_{s}$ (translation) and $2.5^{rad}/_{s}$ (rotation), but the implemented controller limits velocities to $1^{m}/_{s}$ and $2^{rad}/_{s}$. Since the goal is multi-robot coordination, no localization method was implemented. Instead, the controller simply obtains the position of the robots from the simulator.

Three different types of workspace were considered in the experiments: empty ($100 \times 50m^2$), scattered obstacles ($100 \times 50m^2$) and office-like ($63 \times 50m^2$). Figure 28 depicts them. For each workspace, experiments were performed by considering communication systems with different radius: from L to $0.05L$ (or 100% to 5%), where L is the workspace's diagonal. For each combination of workspace and communication radius, 10 trials were ran. The results are presented in sections 6.3, 6.4 and 6.5. The performance criteria used to analyze HKME are presented below.

6.2 CRITERIA FOR HKME EVALUATION

During each trial, the path of each robot ($path_i$) and the amount of explored area[1] (A_i) was recorded, as well as the relative size of the

[1] The area explored by a robot is the number of unknown cells it sensed

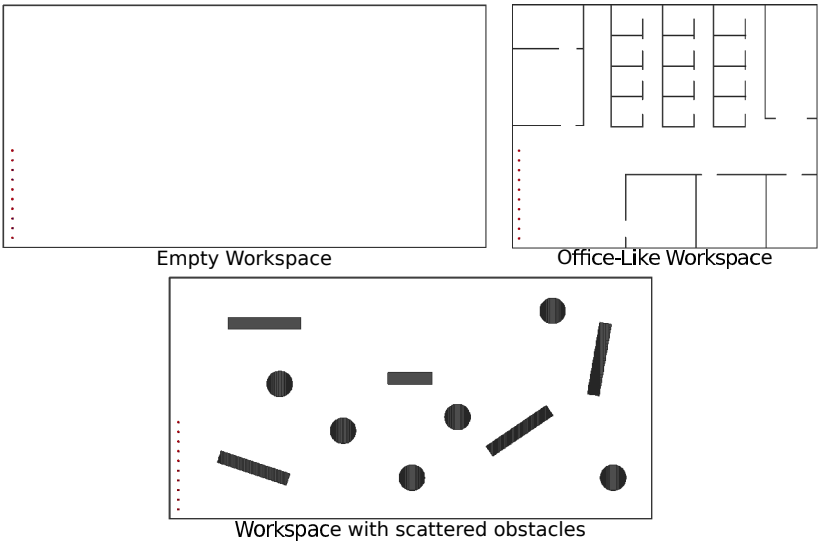


Figure 28 – Workspaces considered in the experiments.

regions assigned to each robot during the exploration, $rs_i(t)$, and the percentage of explored workspace over time, $w(t)$. The relative size of a region corresponds to the number of unexplored cells in the region divided by the total number of cells in the workspace.

To evaluate HKME, the following measures were considered: exploration and arrival times, traveled distance, explored area, size of regions and exploration quality. These measures are described below.

Exploration and arrival times

An important aspect about multi-robot exploration is that, at its last stages, robots may have to travel long distances to explore small areas scattered over the workspace. Thus, the time necessary to completely explore the workspace can be much longer than the time necessary to explore most of it.

In this work, three exploration times were measured: $expT^{90}$, $expT^{95}$ and $expT^{100}$. They represent the instants at which 90% ($expT^{90}$), 95% ($expT^{95}$) and 100% ($expT^{100}$) of the workspace was explored.

In addition, the workspace was partitioned into K (number of robots) areas and we verify how long the robots take to disperse and reach them. The arrival time for a region r_j is the time elapsed until a robot reaches r_j. $T^{max}_{arrival}$ is the arrival time of the last region to be reached in the trial and $\mathbf{T_{arrival}}$ is the average value of the arrival times for all regions. Let $T^{r_j}_{arrival}$ be the instant at which region r_j was reached by a robot. $T^{max}_{arrival}$ and $\mathbf{T_{arrival}}$ can be calculated as:

$$T^{max}_{arrival} = \max_{j=\{1..K\}}(T^{r_j}_{arrival}) \tag{6.1}$$

$$\mathbf{T_{arrival}} = \frac{1}{K}\sum_{j=1}^{K} T^{r_j}_{arrival} \tag{6.2}$$

Traveled distance

Regarding the traveled distances, the average and standard deviation of the distances traveled by the robots were analyzed. As with the exploration time, the average traveled distance (length of the paths) was considered at three instants: when 90% ($\mathbf{td^{90}}$) , 95% ($\mathbf{td^{95}}$) and 100% ($\mathbf{td^{100}}$) of the workspace was explored. Let td^X_i be the length of the path traveled by R_i until instant $expT^X$. The values $\mathbf{td^{90}}$, $\mathbf{td^{95}}$ and $\mathbf{td^{100}}$ can be calculated as:

$$\mathbf{td^X} = \frac{1}{K}\sum_{i=1}^{K} td^X_i \tag{6.3}$$

The standard deviation of the distance traveled by all robots, σ_{td}, is defined as:

$$\sigma_{td} = \sqrt{\frac{1}{K}\sum_{i=1}^{K}(\mathbf{td^{100}} - td^{100}_i)^2} \tag{6.4}$$

The relative deviation of the traveled distance (σ^{rel}_{td}), defined in (6.5), is also used to evaluate the performance of the MRS:

$$\sigma^{rel}_{td} = \frac{\sigma_{td}}{\mathbf{td^{100}}} \tag{6.5}$$

Explored Area

When robots move to two close frontiers, they can sense the same area simultaneously. Moreover, due to communication losses, robots can explore areas already explored by others. In both cases, the exploration is redundant and the sum of the areas explored by each robot will be higher than the area of the workspace. Let A_T be the workspace area and A_i the area explored by robot R_i. The percentage of redundant explored area, ra, is defined as:

$$ra = \left[\frac{1}{A_T} \sum_{i=1}^{K} A_i \right] - 1 \tag{6.6}$$

Let $A_r^i = \frac{A_i}{A_T}$ be the percentage of workspace explored by R_i. The average percentage of workspace explored by the robots, $\mathbf{A_r}$, and its standard deviation, σ_{Ar}, are defined as:

$$\mathbf{A_r} = \frac{1}{K} \sum_{i=1}^{K} A_r^i \tag{6.7}$$

$$\sigma_{Ar} = \sqrt{\frac{1}{K} \sum_{i=1}^{K} (\mathbf{A_r} - A_r^i)^2} \tag{6.8}$$

Size of regions

Another indicator of the exploration's balance (or fairness) is the standard deviation of the size of regions assigned to the robots. Let $rs_i(t)$ be the relative size of the region assigned to R_i at instant t. The average region size over time, $\mathbf{rs}(t)$, and its standard deviation, $\sigma_{rs}(t)$, are defined as:

$$\mathbf{rs}(t) = \frac{1}{K} \sum_{i=1}^{K} rs_i(t) \tag{6.9}$$

$$\sigma_{rs}(t) = \sqrt{\frac{1}{K} \sum_{i=1}^{K} \Big(\mathbf{rs}(t) - rs_i(t) \Big)^2} \tag{6.10}$$

Exploration quality

In (ZLOT et al., 2002), Zlot *et al.* propose a measure based on the workspace area and the sum of distances traveled by all roots to calculate the quality of exploration, $\mathcal{Q}$:

$$\mathcal{Q} = \frac{A_T}{\sum_{i=1}^{K} td_i} \tag{6.11}$$

The maximum exploration quality, $\mathcal{Q}_{max}$, is defined as the area that can be explored by a robot when it moves one meter, considering the range of its sensors. In this work, robots are assumed to have a laser sensor with a 3 meters range. Thus, the maximum exploration quality is $\mathcal{Q}_{max} = 6m^2/m$. To evaluate the performance of HKME, the *relative exploration quality*, $\mathcal{Q}_r$, is considered instead of $\mathcal{Q}$. The relative exploration quality is defined as:

$$\mathcal{Q}_r = \frac{\mathcal{Q}}{\mathcal{Q}_{max}} \tag{6.12}$$

6.3 EXPERIMENT A: EMPTY WORKSPACE

Several experiments were performed over a $100 \times 50m^2$ workspace with no obstacles. Six different communication radiuses were considered: 100%, 50%, 30%, 20%, 10% and 5% of the workspace diagonal ($L = 111.8m$). Figure 29 shows examples of robot paths for different communication radiuses.

Figure 30 shows that the percentage of explored area over time, $w(t)$, increases when the radius decreases. For communication radiuses greater than 50%, the exploration does not improve significantly. Moreover, for all communication radiuses, the time necessary for the robots to explore 100% ($expT^{100}$) of the workspace is much greater than the time to explore 90% ($expT^{90}$). For instance, when the communication is 30% of L, $expT^{90}$ is close to 200s while $expT^{100}$ is almost 280s.

Likewise, figure 31 shows that the standard deviation of the size of regions over time, $\sigma_{rs}(t)$, also increases when the communication radius decreases. For simplicity, only three curves are presented. The

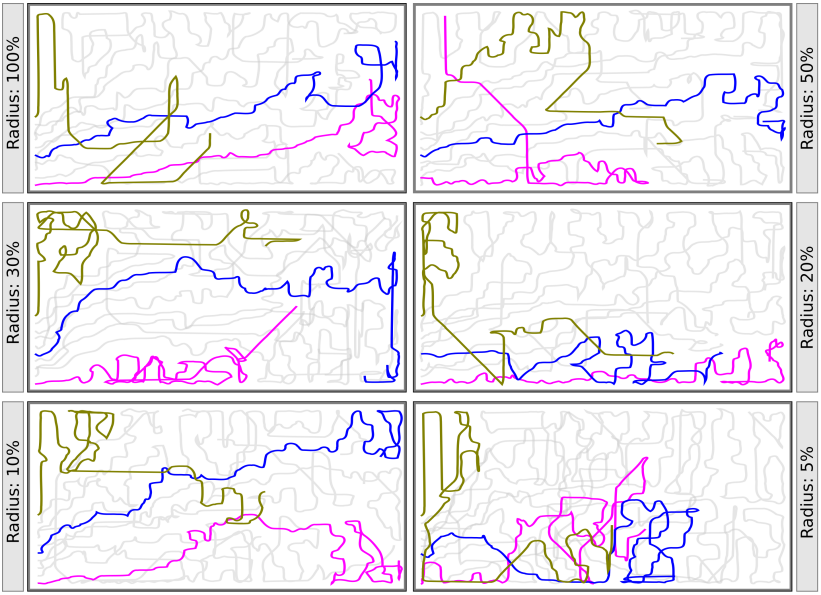


Figure 29 – Paths of robots in empty workspace for different communication radiuses.

other measures, presented in figure 32, are: exploration times $expT^{90}$, $expT^{95}$ and $expT^{100}$, figure 32a; maximum and average arrival times, $T^{max}_{arrival}$ and $\mathbf{T_{arrival}}$, figure 32c; traveled distances $\mathbf{td^{90}}$, $\mathbf{td^{95}}$ and $\mathbf{td^{1000}}$, figure 32b; relative standard deviation of traveled distance, σ^{rel}_{td}, figure 32d; redundant explored area, ra, figure 32e; and relative exploration quality, $\mathscr{Q}_r$, figure 32f.

In figure 32, bars represent the standard deviation of the measures for all the trials, summed and subtracted from the average value.

Figures 32a and 32c show that both the exploration and arrival times increase when the communication radius decreases. Likewise, the traveled distance, figure 32b, and its standard deviation, figure 32d, also increase when the communication radius decreases.

Figure 32e shows that the percentage of redundant explored area is close to 4% for communication radiuses greater than 20% of L, being almost 80% when the radius is 5%. Regarding the exploration quality,

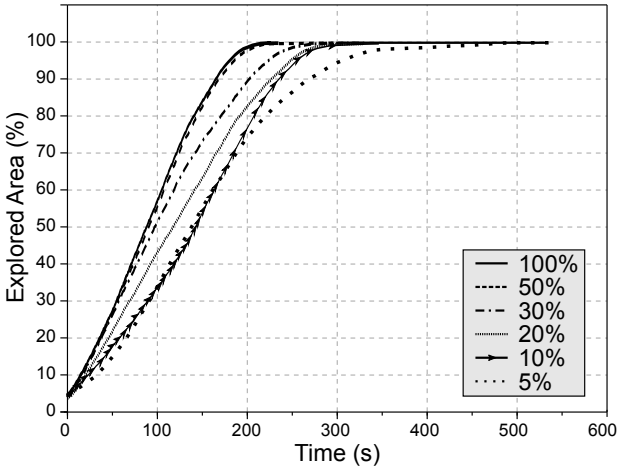


Figure 30 – Explored area in empty workspace for different communication radiuses.

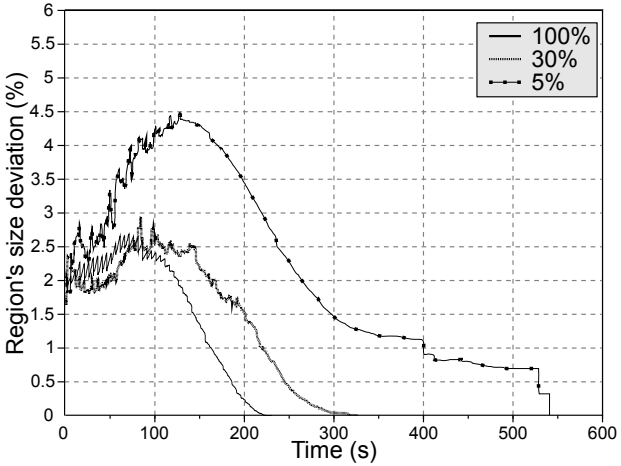
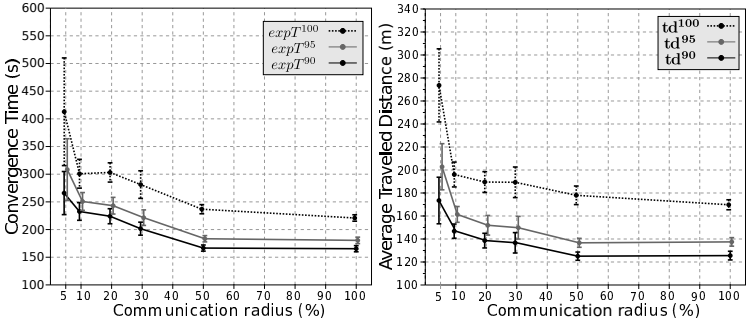


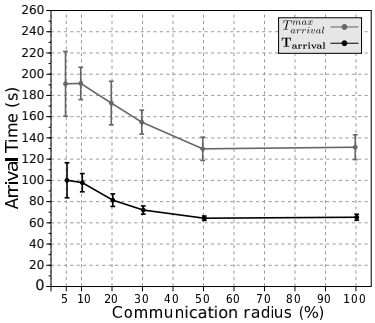
Figure 31 – Deviation of size of regions over time in empty workspace for different communication radiuses.

figure 32f shows that $\mathscr{Q}_r$ decreases with the communication radius.

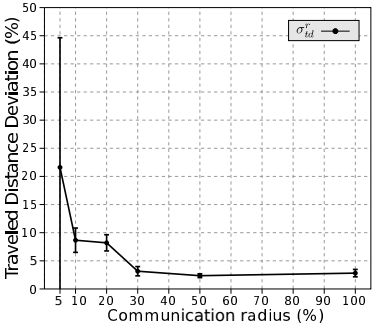


(a) Exploration times.

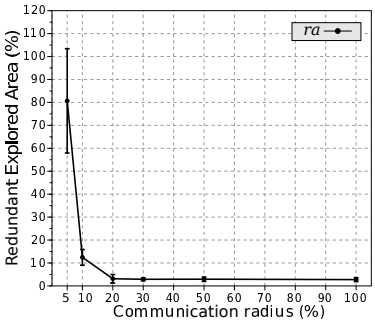
(b) Traveled distances.



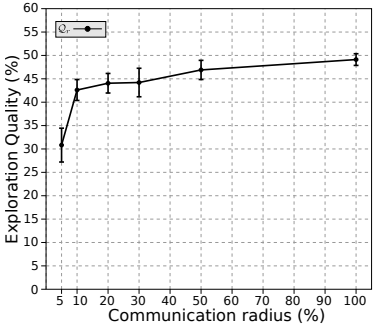
(c) Arrival times.



(d) Standard deviation of the distance.



(e) Redundant explored area.



(f) Relative exploration quality.

Figure 32 – Results obtained in the experiment with an empty workspace.

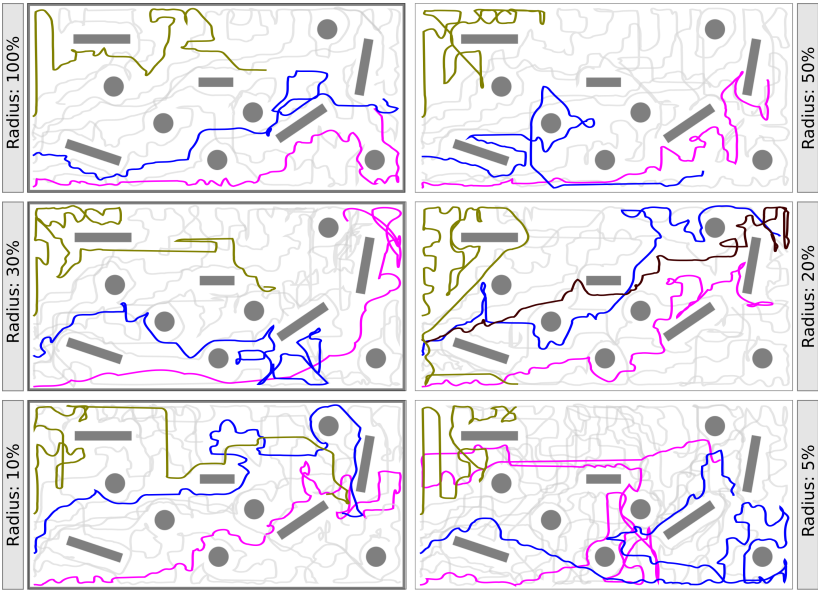


Figure 33 – Paths of robots in a workspace with cluttered obstacles for different communication radiuses.

6.4 EXPERIMENT B: WORKSPACE WITH SCATTERED OBSTACLES

Experiments were also performed over a $100 \times 50m^2$ workspace with scattered obstacles. As in experiment A, six different communication radiuses were considered: 100%, 50%, 30%, 20%, 10% and 5% of the workspace diagonal ($L = 111.8m$). Figure 33 illustrates the trajectories executed by the robots for different communication radiuses.

Figures 34 and 35 present the percentage of explored area over time, $w(t)$, and the standard deviation of the size of regions over time, $\sigma_{rs}(t)$, respectively. As in experiment A, both the exploration time and the deviation in the size of regions increase when the communication radius increases. For communication radiuses greater than 50%, the time necessary to complete the exploration does not significantly decrease.

Figure 36 presents the exploration times, maximum and average arrival times, traveled distances, standard deviation of traveled

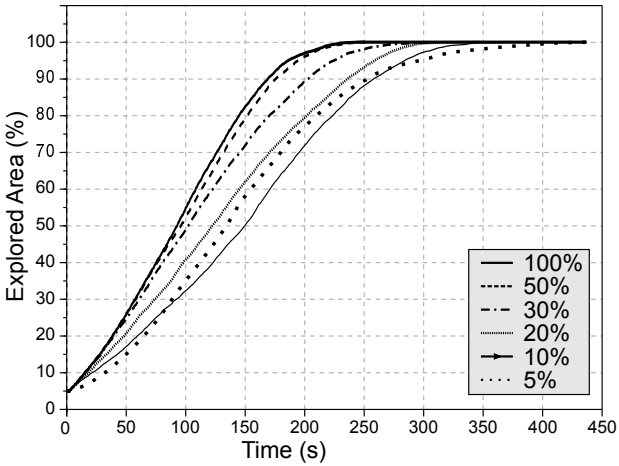


Figure 34 – Explored area in a workspace with cluttered obstacles for different communication radiuses.

distance, redundant explored area and relative exploration quality for that workspace.

The performance of HKME in experiment B is similar to the one presented in experiment A. Figure 36 show that the exploration and arrival times, the traveled distance and its standard deviation increase when the communication radius decreases. As in experiment A, the redundant explored area is close to 5% for communication radiuses greater than 20% of L, approaching to 80% when the radius is 5%, figure 36e. Moreover, the exploration quality also decreases with the communication radius.

6.5 EXPERIMENT C: OFFICE-LIKE WORKSPACE

Experiments were also performed over a $63 \times 50m^2$ office-like workspace. As in experiments A and B, six different communication radiuses were considered: 100%, 50%, 30%, 20%, 10% and 5% of the workspace diagonal ($L = 80.4$m). Figure 37 illustrates the trajectories executed by the robots for different communication radiuses.

The exploration over time, $w(t)$, and the standard deviation in

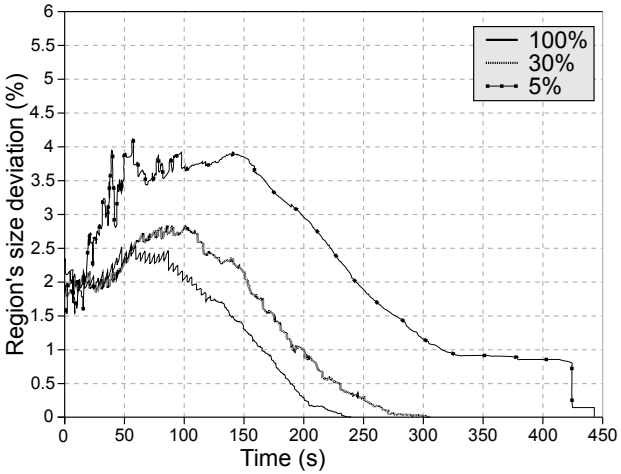


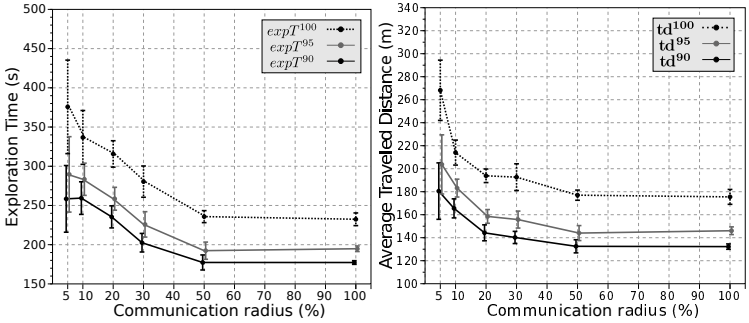
Figure 35 – Deviation of the size of region in a workspace with cluttered obstacles for different communication radiuses.

the size of regions over time, $\sigma_{rs}(t)$, for the office-like workspace are respectively presented in figures 38 and 39. Figure 40 present the other measures.

As in experiments A and B, both the exploration time and the standard deviation in the size of regions increase when the communication radius increases. However, figures 38 and 39 show that the explored area over time, $w(t)$, and the deviation in the size of regions, $\sigma_{rs}(t)$, do not significantly differ for communication radiuses between 10% and 30% of L.

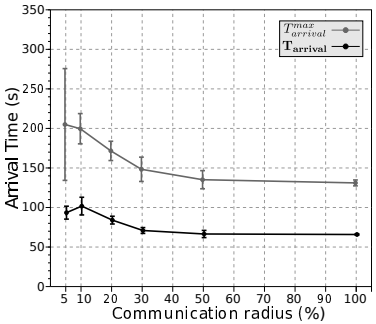
Regarding the exploration and arrival times, figures 40a and 40c, they also increase when the communication radius decreases. However, those figures show that the exploration and arrival times do not significantly differ for communication radiuses smaller then 20%.

Other significant differences in experiment C are shown in figures 40e and 40f. The redundant explored area is much larger in experiment C than in A and B. Moreover, the exploration quality is worse in experiment C.

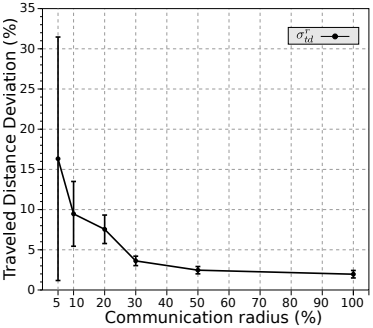


(a) Exploration times.

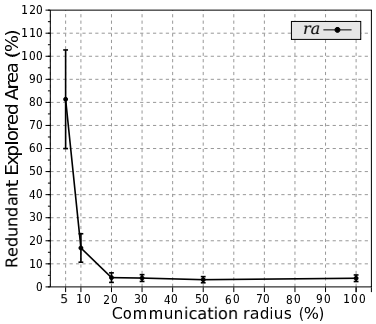
(b) Traveled distances.



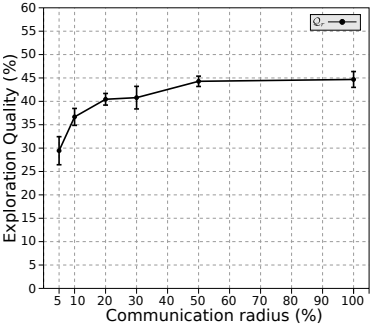
(c) Arrival times.



(d) Standard deviation of the distance.



(e) Redundancy in the exploration.



(f) Relative exploration quality

Figure 36 – Results obtained in the experiment with a workspace with cluttered obstacles.

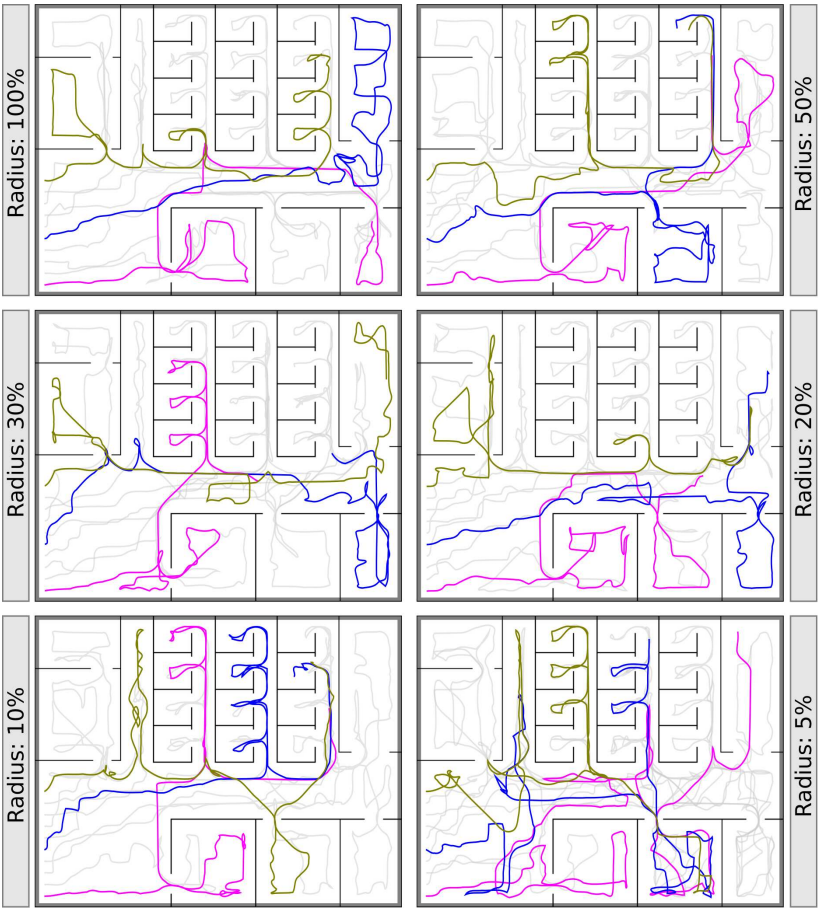


Figure 37 – Paths of robots in office-like workspace for different communication radiuses.

6.6 DISCUSSIONS

The results obtained from the previous experiments indicate that, even when robots have a limited communication radius, HKME is able to coordinate them to explore unknown workspaces and build a complete map in an online fashion. In general, the exploration efficiency decreases with the communication radius as expected. Specifically, when the radius decreases, the exploration times, arrival times

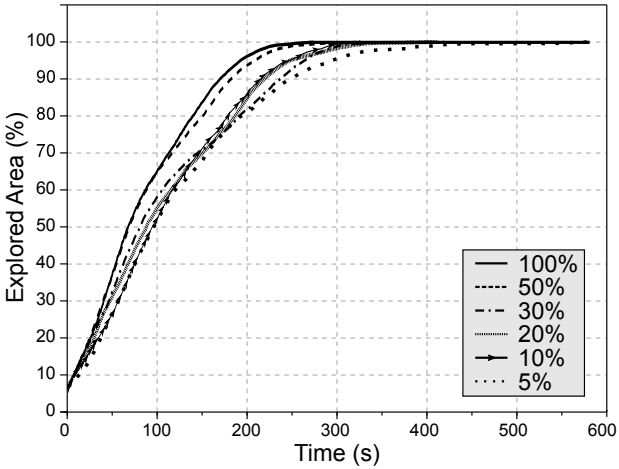


Figure 38 – Explored area in office-like workspace for different communication radiuses.

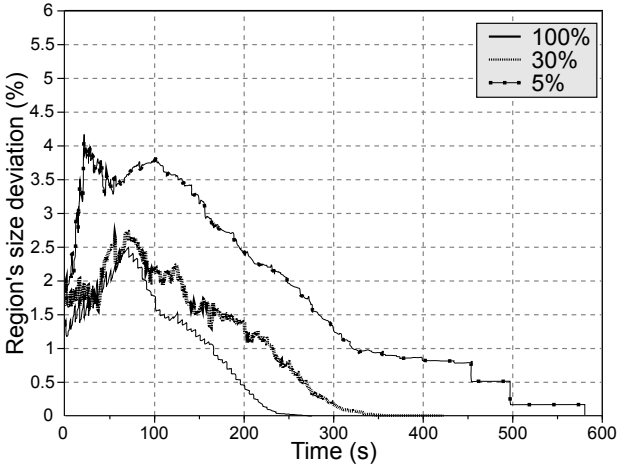
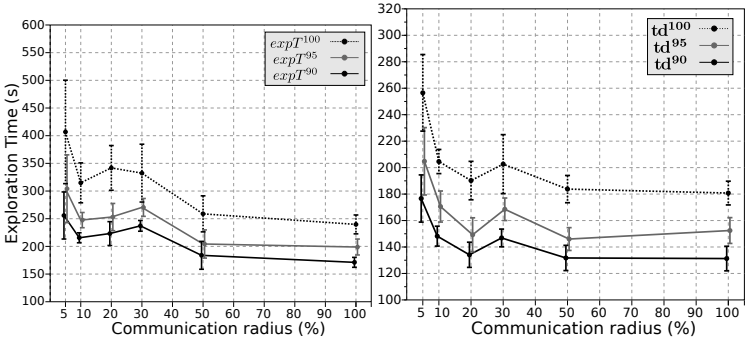


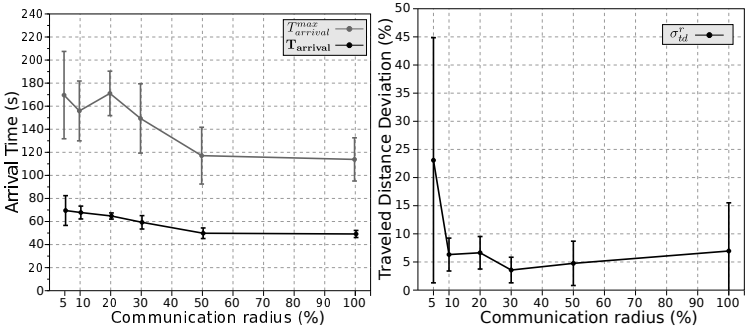
Figure 39 – Deviation of the size of regions in office-like workspace for different communication radiuses.

and traveled distances increase, whereas the exploration quality decreases.

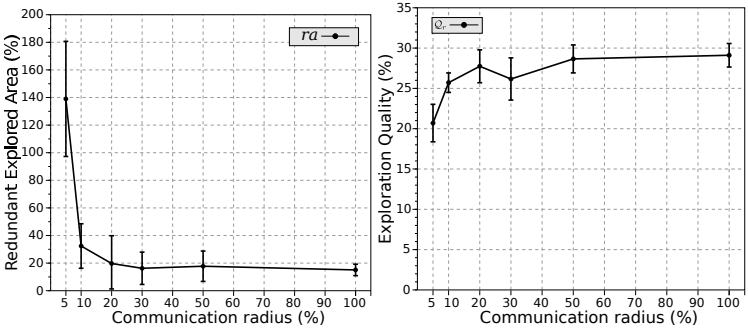
Measures such as the exploration times, traveled distances and redundant area decay exponentially when the communication radius



(a) Exploration times. (b) Traveled distances.



(c) Arrival times. (d) Standard deviation of the distance.



(e) Redundancy in the exploration. (f) Relative exploration quality.

Figure 40 – Results obtained in the experiment with an office-like workspace.

decreases and can be fitted by a function $f(x) = \alpha e^{-\beta x} + \gamma$. In turn, the exploration quality, $\mathcal{Q}_r$, can be approximated to the function $f(x) = \gamma - \alpha e^{-\beta x}$. Table 5 shows the parameters of the exponential functions $f(x)$ that approximate the measures considered in our experiments, and the average and maximum errors between these fitting functions and the real data.

Table 5 – Fitting functions.

Measure	Exp.	α	β ($\times 10^{-2}$)	γ	**e**	e_{max}
$expT^{90}$	A	132.6	5.05	164.4	3.3%	4.8%
	B	122.7	4.87	176.3	3.4%	5.5%
	C	99.1	3.22	170.2	5.1%	12.3%
$expT^{95}$	A	145.3	5.31	179.5	4.3%	5.6%
	B	147.11	5.10	191.4	3.4%	5.5%
	C	112.2	3.45	197.8	6.1%	12.1%
$expT^{100}$	A	205.0	6.10	219.8	6.9%	10.1%
	B	203.0	5.27	231.4	3.1%	4.3%
	C	169.7	3.90	238.7	6.9%	12.3%
$\mathbf{td}^{90}$	A	59.1	7.22	124.6	2.4%	4.4%
	B	67.2	7.42	131.3	1.1%	1.6%
	C	41.3	5.10	130.2	4.7%	8.2%
$\mathbf{td}^{95}$	A	81.8	8.96	138.1	3.1%	6.2%
	B	81.1	7.89	145.0	1.5%	2.0%
	C	61.5	5.32	145.0	6.0%	9.4%
$\mathbf{td}^{100}$	A	118.5	1.00	173.1	5.1%	10.5%
	B	125.8	9.10	174.5	2.9%	5.2%
	C	70.9	51.6	180.2	5.0%	8.6%
ra	A	516.3	37.8	1.8	1.1%	1.1%
	B	385.5	31.9	2.4	1.1%	1.4%
	C	796.7	40.7	17.0	8.8%	12.9%
$\mathcal{Q}_r$	A	25.7	9.58	41.1	5.1%	7.8%
	B	20.4	7.53	44.7	2.3%	4.3%
	C	10.0	7.10	29	3.9%	6.3%

The coefficients α, β and γ associated with the measures in experiments A and B are similar, but significantly differ from the ones associated with experiment C. The office-like workspace considered in experiment C is quite different from the ones considered in A and B, suggesting that there is a relationship between the values of the coefficients of the fitting functions and the workspace characteristics, such as its size and how much cluttered it is. However, the data gathered in this work are not sufficient to draw further conclusions about the impact of the workspace on the values of those coefficients. Thus, new

experiments must be conducted to analyze and define the relationship between the values of α, β and γ and different characteristics of the workspace (size, percentage of area occupied by obstacles, average distance among obstacles, etc.).

For most fitting functions whose coefficients are presented in table 5, the average error is usually smaller than 4%, with the maximum error usually smaller than 6%. However, for some measures in experiment A and most of them in experiment C, the error is much larger, almost reaching 13% in the worst case. The larger error in those cases is associated with the larger deviation in the value of the measures for the different trials, indicated by triangles ($\triangle$ and $\triangledown$) in the figures where these measures are presented.

Regarding the exploration efficiency, the main reason for its exponential decay when the communication radius is reduced is that the robots tend to be separated in several unconnected networks. Since robots in different networks cannot exchange messages, they cannot share information or coordinate themselves. Furthermore, the workspace partitioning becomes less efficient.

In HKME, instead of running a centralized scheme to partition the entire unexplored workspace, clusterheads of a network exchange cells from their own sectors to reshape the workspace global partitioning. Then, they partition their sectors in regions and assign the regions to the members of their clusters.

If all robots always have a communication link with each other, HKME behaves like KME. When the robots are in a network with several clusters, the efficiency of HKME may degrade. This occurs because, after the global partitioning phase, clusterheads partition their sectors into regions locally. Clusterheads also assign regions and allocate frontiers locally, considering only members of their clusters. Finally, if robots are separated in several unconnected networks, HKME is executed independently in each network. Therefore, clusterheads generate sectors during the global partitioning phase, only considering the areas owned by the robots in their network, which can significantly decrease the efficiency of the workspace partitioning.

As the partitioning efficiency decreases, regions can be generated in a way that their cells spread over large areas. In those situations, robots may need to move long distances through already explored areas to reach all parts of their regions. Figures 29, 33 and 37 show that, as the communication radius decreases, the robots' paths become longer and with a higher degree of intersections[2] with both the paths of other robots and their own paths.

Despite the aforementioned partitioning efficiency problems and the difficulty to share information, figures 32e, 36e and 40e show that the percentage of redundant explored area is usually small and almost constant. This occurs because HKME disperses the robots through the workspace quickly ($\mathbf{T_{arrival}}$ is usually close to 25% of the exploration time and $T^{max}_{arrival}$ close to 50%), even when the communication radius decreases. Redundancy significantly increases only for small values of radius (10% and 5% of L), which can prevent even nearby robots from exchanging information and coordinating the exploration together.

Regarding the topology of the workspace, robots take more time (and travel longer distances) to explore more cluttered workspaces. This occurs because in some workspaces, such as the office-like, robots can be forced to move close to each other and go back through already explored areas when they reach a dead-end. On the other hand, in empty workspaces, robots can move freely, decreasing both the exploration time and traveled distance.

In the experiments, the exploration efficiency in the office-like workspace (experiment C) is worse than in the experiments with other workspaces. Despite the size of the office-like workspace ($63 \times 50m^2$) is smaller than the others ($100 \times 50m^2$), the exploration and arrival times and the traveled distances are the worst in experiment C. Furthermore, the relative exploration quality in experiment C is significantly smaller than in the others.

The office-like workspace is made of rooms and corridors, hence

[2] The paths of two robots intersect if they move through the same position (at different instants of the simulation) or through positions that are so close that the areas sensed from them are highly overlapped.

forcing some robots to move close to each other in several moments, which can be noticed in the robots' paths (figure 37). As a result, the redundant explored area is also very high. Figures 32e and 36e show that the redundant explored area in experiments A and B is close to 5% for most communication radiuses, reaching 80% in the worst case. On the other hand, in the office-like workspace (figure 40e), the redundancy is close to 20% for communication radiuses above 20% of L, approaching 140% in the worst case.

Another parameter that is affected by the structure of the workspace is the relative standard deviation of traveled distances (σ_{td}^{rel}). In experiments A and B, σ_{td}^{rel} increases when the radius decreases, figures 32d and 36d. In experiment C, however, figure 40d shows that σ_{td}^{rel} does not follow a monotonic rising curve. Instead, all values of σ_{td}^{rel} are close to 6% (except for a communication radius of 5%). This occurs because the disposition of obstacles (walls) in the workspace can unbalance the distance traveled by the robots more than the problems associated with the communication radius itself.

7 CONCLUSIONS

This thesis approaches the problem of multi-robots exploration, focusing on how robots can share the information they detect and how they can be coordinated in order to explore the workspace efficiently. Specifically, we approach multi-robots exploration considering that robots have a limited communication radius and can be separated in several unconnected networks.

The multi-robots exploration was divided in two problems: map sharing, which allows robots to share the information they detect about the environment and synchronize their maps as the explore the workspace; and allocation of exploration targets, which allows robots to coordinated themselves in order to fairly distribute the workload among them and quickly spread over the workspace. Thus, during the PhD, two complementary goals were pursued: to develop an efficient method for map sharing and to develop a method for multi-robots coordination in exploration tasks. Next, we present a summary of the contributions on each problem.

MAP SHARING

Two methods, one based on a flat network architecture (DSM) and another based in a hierarchical architecture (HSM), were proposed to share map information when robots have limited communication.

In Distributed Synchronization Method (DSM), robots use a propagation scheme to share information and there is no need for network formation and maintenance, making the method easier to implement. The Hierarchical Synchronization Method (HSM) organizes robots in a hierarchical architecture, which can make robots lose time executing cluster formation algorithms. However, the HSM surpass DSM performance in most situations and also makes easier for robots to handle problems as message losses.

The methods were also compared with Sheng's method. Only in situations where a robot finishes exploring a frontier, the HSM and

Sheng's method have a similar performance, while the DSM usually presents the worst performance, with higher convergence times, number of exchanged messages and transmitted data. When two networks merge, the efficiency of Sheng's method decreases significantly when the number of robots or the size of maps increases. Also, Sheng's method do not guarantee convergence when several networks merge.

On the other hand, both the DSM and HSM scale well when two (or more) networks merge, with the HSM performing better than the DSM in most scenarios.

COORDINATION IN ROBOTS EXPLORATION

This thesis also proposes Hierarchical K-Means (HKME), a new distributed method for multi-robot exploration under constrained communication. The basis of HKME is the workspace partitioning scheme and the assignment of regions and frontiers in a way that minimizes aspects such as the variance of arrival time and the traveled distance. In HKME, robots are grouped in clusters, and clusterheads are responsible for managing the communication in the network, partitioning the workspace and assigning regions and frontiers to the members of their clusters. In addition, HKME considers that robots in a network have the same LAV version and uses HSM method for map sharing to guarantee that.

Experiments with HKME by considering different types of workspace and communication radiuses have been conducted, showing that HKME is able to explore the entire workspace efficiently. The smaller the communication radius, the less efficient HKME becomes, takinglonger to complete the exploration. In these situations, the traveled distance, arrival times and deviation in the size of regions also become larger.

The main influence of the communication radius in HKME is on the quality of the workspace partition and region assignment, which are sub-optimal when the robots are separated in several unconnected networks. When robots have a small communication radius, the detected

information takes longer to be shared within the system, leading to redundant exploration.

The topology of the workspace has also a big impact on the efficiency of HKME. In office-like workspaces, for instance, a region can have areas in two or more rooms, which will force its assigned robot to travel a long distance to fully explore that region. This problem is reduced by the periodic partitioning and assignment of regions.

Another important benefit of HKME is that, after exploring their regions, robots do not remain idle and start exploring regions of other robots. Besides improving the exploration efficiency, this feature makes HKME robust to robot failures, thus relaxing the initial assumption that robots cannot fail.

7.1 FUTURE WORK

Regarding the workspace boundaries

Future work can explore issues related to the workspace boundaries, region exploration and robot failures. With respect to the first issue, we plan to extend HKME to handle unbounded workspaces or situations where the boundaries are unknown.

Regarding regions exploration

Regarding regions exploration, we will investigate the use of coverage methods by robots that already reached their assigned regions. Currently, we use a greedy strategy to assign frontiers for robots that already reached their regions, assigning the closest one. Using a coverage method to calculate the shortest path that takes the robot through all areas of its region, we can improve exploration quality and decrease exploration time. Another aspect that will be investigated is the possibility of assigning more than one robot per region, which might improve the exploration by given more freedom to robots explore the workspace.

Regarding robot failures

Regarding robot failures, the mechanisms the HKM uses to assign frontiers to robots make it robust to robot failures. However, in this thesis, we do not investigate the influence of robot failures in the exploration efficiency. In future work, we will evaluate the HKM's in scenarios with robot failures.

Regarding the workspace characteristics

Finally, we plan to further investigate the influence of the workspace in the performance of HKME. To do so, new experiments will be conducted in order to analyze the impact of the workspace characteristics on the exploration efficiency and the relationship between them and the values of the coefficients α, β and γ of the fitting functions presented in chapter 6.

BIBLIOGRAPHY

ABBASI, A. A.; YOUNIS, M. A survey on clustering algorithms for wireless sensor networks. *Computer communications*, Elsevier, v. 30, n. 14, p. 2826–2841, 2007. 28, 29, 72, 102

AHMED, N. R.; SAMPLE, E. M.; CAMPBELL, M. Bayesian multicategorical soft data fusion for human–robot collaboration. *IEEE Transactions on Robotics*, IEEE, v. 29, n. 1, p. 189–206, 2013. 28

ASGHARBEYGI, N.; MALEKI, A. Geodesic k-means clustering. In: IEEE. *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*. [S.l.], 2008. p. 1–4. 40

BAUTIN, A.; SIMONIN, O.; CHARPILLET, F. Minpos: A novel frontier allocation algorithm for multi-robot exploration. In: SPRINGER. *International Conference on Intelligent Robotics and Applications*. [S.l.], 2012. p. 496–508. 19

BERHAULT, M. et al. Robot exploration with combinatorial auctions. In: IEEE. *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*. [S.l.], 2003. v. 2, p. 1957–1962. 42, 45, 47

BERTSEKAS, D. P. The auction algorithm for assignment and other network flow problems: A tutorial. *Interfaces*, INFORMS, v. 20, n. 4, p. 133–149, 1990. 40

BURGARD, W. et al. Collaborative multi-robot exploration. In: IEEE. *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*. [S.l.], 2000. v. 1, p. 476–481. 25, 30, 35, 36, 37, 43, 44, 46, 101

BURGARD, W. et al. Coordinated multi-robot exploration. *Robotics, IEEE Transactions on*, IEEE, v. 21, n. 3, p. 376–386, 2005. 19, 20, 25, 31, 34, 35, 36, 37, 45, 47, 101

BUTZKE, J.; LIKHACHEV, M. Planning for multi-robot exploration with multiple objective utility functions. In: IEEE. *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. [S.l.], 2011. p. 3254–3259. 19

CAMBRUZZI, W. E.; FARINES, J.-M.; JUNIOR, W. K. Um algoritmo baseado em peso para formaçao e manutençao de agrupamentos em redes veiculares ad hoc. *Anais do 27ö Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos*, 2009. 30

CARVALHO, F. F. et al. A multi-robot exploration approach based on distributed graph coloring. In: IEEE. *Robotics Symposium and Competition (LARS/LARC), 2013 Latin American*. [S.l.], 2013. p. 142–147. 25, 41

CAVALCANTE, R. C.; NORONHA, T. F.; CHAIMOWICZ, L. Improving combinatorial auctions for multi-robot exploration. In: IEEE. *Advanced Robotics (ICAR), 2013 16th International Conference on*. [S.l.], 2013. p. 1–6. 42

CHATTERJEE, M.; DAS, S. K.; TURGUT, D. Wca: A weighted clustering algorithm for mobile ad hoc networks. *Cluster Computing*, Springer, v. 5, n. 2, p. 193–204, 2002. 28, 51, 105

DHURANDHER, S. K.; SINGH, G. Weight based adaptive clustering in wireless ad hoc networks. In: IEEE. *Personal Wireless Communications, 2005. ICPWC 2005. 2005 IEEE International Conference on*. [S.l.], 2005. p. 95–100. 28, 51, 102

DISSANAYAKE, M. G. et al. A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on robotics and automation*, IEEE, v. 17, n. 3, p. 229–241, 2001. 26

ELFES, A. Using occupancy grids for mobile robot perception and navigation. *Computer*, IEEE, v. 22, n. 6, p. 46–57, 1989. 27, 28, 112

EPHREMIDES, A.; WIESELTHIER, J. E.; BAKER, D. J. A design concept for reliable mobile radio networks with frequency hopping signaling. *Proceedings of the IEEE*, IEEE, v. 75, n. 1, p. 56–73, 1987. 28, 29, 105

FAIGL, J.; KULICH, M. On benchmarking of frontier-based multi-robot exploration strategies. In: IEEE. *Mobile Robots (ECMR), 2015 European Conference on*. [S.l.], 2015. p. 1–8. 40

FAIGL, J.; KULICH, M.; PŘEUČIL, L. Goal assignment using distance cost in multi-robot exploration. In: IEEE. *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. [S.l.], 2012. p. 3741–3746. 40

FEI, W. et al. A comprehensive uav indoor navigation system based on vision optical flow and laser fastslam. *Acta Automatica Sinica*, Elsevier, v. 39, n. 11, p. 1889–1899, 2013. 26

FLOCCHINI, P. et al. Computing without communicating: Ring exploration by asynchronous oblivious robots. *Algorithmica*, Springer, v. 65, n. 3, p. 562–583, 2013. 25

FOX, D. et al. Distributed multirobot exploration and mapping. *Proceedings of the IEEE*, IEEE, v. 94, n. 7, p. 1325–1339, 2006. 36, 37

FRANCHI, A. et al. A randomized strategy for cooperative robot exploration. In: IEEE. *Robotics and Automation, 2007 IEEE International Conference on*. [S.l.], 2007. p. 768–774. 32, 37

FRANCHI, A. et al. The sensor-based random graph method for cooperative robot exploration. *Mechatronics, IEEE/ASME Transactions on*, IEEE, v. 14, n. 2, p. 163–175, 2009. 32, 37, 46, 102

GERKEY, B.; VAUGHAN, R. T.; HOWARD, A. The player/stage project: Tools for multi-robot and distributed sensor systems. In: *Proceedings of the 11th international conference on advanced robotics*. [S.l.: s.n.], 2003. v. 1, p. 317–323. 23, 119

GERLA, M.; TSAI, J. T.-C. Multicluster, mobile, multimedia radio network. *Wireless networks*, Springer, v. 1, n. 3, p. 255–265, 1995. 28, 105

HART, P. E.; NILSSON, N. J.; RAPHAEL, B. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, IEEE, v. 4, n. 2, p. 100–107, 1968. 34

HAUMANN, D.; WILLERT, V.; LISTMANN, K. D. Discoverage: From coverage to distributed multi-robot exploration. *IFAC Proceedings Volumes*, Elsevier, v. 46, n. 27, p. 328–335, 2013. 25

HOWARD, R. A. Dynamic programming and markov processes. 1960. 34

HSU, D.; LATOMBE, J.-C.; KURNIAWATI, H. On the probabilistic foundations of probabilistic roadmap planning. *The International Journal of Robotics Research*, SAGE Publications, v. 25, n. 7, p. 627–643, 2006. 27

KAPLOW, R.; ATRASH, A.; PINEAU, J. Variable resolution decomposition for robotic navigation under a pomdp framework. In: IEEE. *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. [S.l.], 2010. p. 369–376. 27

KAVRAKI, L. E.; LATOMBE, J.-C. Probabilistic roadmaps for robot path planning. Citeseer, 1998. 27

KUBALE, M. et al. A better practical algorithm for distributed graph coloring. In: IEEE. *null*. [S.l.], 2002. p. 72. 41

KUBELKA, V. et al. Robust data fusion of multimodal sensory information for mobile robots. *Journal of Field Robotics*, Wiley Online Library, v. 32, n. 4, p. 447–473, 2015. 28

KUHN, H. W. The hungarian method for the assignment problem. *Naval research logistics quarterly*, Wiley Online Library, v. 2, n. 1-2, p. 83–97, 1955. 38

KUO, B.-W. et al. A light-and-fast slam algorithm for robots in indoor environments using line segment map. *Journal of Robotics*, Hindawi Publishing Corporation, v. 2011, 2011. 26

LAGOUDAKIS, M. G. et al. Simple auctions with performance guarantees for multi-robot task allocation. In: IEEE. *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*. [S.l.], 2004. v. 1, p. 698–705. 31, 42, 45, 46, 47

LUO, R. C.; LAI, C. C. Enriched indoor map construction based on multisensor fusion approach for intelligent service robot. *IEEE Transactions on Industrial Electronics*, IEEE, v. 59, n. 8, p. 3135–3145, 2012. 28

MATIGNON, L.; JEANPIERRE, L.; MOUADDIB, A.-I. Coordinated multi-robot exploration under communication constraints using decentralized markov decision processes. In: *AAAI 2012*. [S.l.: s.n.], 2012. p. p2017–2023. 19

MONTEMERLO, M.; THRUN, S. Simultaneous localization and mapping with unknown data association using fastslam. In: IEEE. *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*. [S.l.], 2003. v. 2, p. 1985–1991. 26

NIETO-GRANDA, C.; ROGERS, J. G.; CHRISTENSEN, H. I. Coordination strategies for multi-robot exploration and mapping. *The International Journal of Robotics Research*, SAGE Publications, p. 0278364913515309, 2014. 43

PUIG, D.; GARCÍA, M. A.; WU, L. A new global optimization strategy for coordinated multi-robot exploration: Development and comparative evaluation. *Robotics and Autonomous Systems*, Elsevier, v. 59, n. 9, p. 635–653, 2011. 19, 21, 23, 25, 35, 43, 44, 45, 47, 101, 102, 112, 114, 116

RODGER, J. A. Toward reducing failure risk in an integrated vehicle health maintenance system: A fuzzy multi-sensor data fusion kalman filter approach for ivhms. *Expert Systems with Applications*, Elsevier, v. 39, n. 10, p. 9821–9836, 2012. 28

ROGERS, J. G.; NIETO-GRANDA, C.; CHRISTENSEN, H. I. Coordination strategies for multi-robot exploration and mapping. In: SPRINGER. *Experimental Robotics*. [S.l.], 2013. p. 231–243. 43

ROSEN, K. H.; KRITHIVASAN, K. *Discrete mathematics and its applications*. [S.l.]: McGraw-Hill New York, 1999. 62

SARIEL, S.; BALCH, T. Real time auction based allocation of tasks for multi-robot exploration problem in dynamic environments. In: *Proceedings of the AAAI-05 Workshop on Integrating Planning into Scheduling*. [S.l.: s.n.], 2005. p. 27–33. 31, 43

SARIEL, S.; BALCH, T. R. Efficient bids on task allocation for multi-robot exploration. In: *FLAIRS Conference*. [S.l.: s.n.], 2006. p. 116–121. 31, 43, 45, 47, 102

SENTHILKUMAR, K.; BHARADWAJ, K. Multi-robot exploration and terrain coverage in an unknown environment. *Robotics and Autonomous Systems*, Elsevier, v. 60, n. 1, p. 123–132, 2012. 20

SHALAL, N. et al. Orchard mapping and mobile robot localisation using on-board camera and laser scanner data fusion–part a: Tree detection. *Computers and Electronics in Agriculture*, Elsevier, v. 119, p. 254–266, 2015. 28

SHENG, W. et al. Distributed multi-robot coordination in area exploration. *Robotics and Autonomous Systems*, Elsevier, v. 54, n. 12, p. 945–955, 2006. 25, 32, 46, 49, 52, 85, 99

SHENG, W. et al. Efficient map synchronization in ad hoc mobile robot networks for environment exploration. In: IEEE. *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*. [S.l.], 2005. p. 2297–2302. 22, 32, 46, 52, 85

SIM, R.; ROY, N. Global a-optimal robot exploration in slam. In: IEEE. *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. [S.l.], 2005. p. 661–666. 25

SIMMONS, R. et al. Coordination for multi-robot exploration and mapping. In: *AAAI/IAAI*. [S.l.: s.n.], 2000. p. 852–858. 20, 30, 36, 39, 45, 46, 47

STACHNISS, C.; MOZOS, O. M.; BURGARD, W. Speeding-up multi-robot exploration by considering semantic place information. In: IEEE. *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. [S.l.], 2006. p. 1692–1697. 36

STACHNISS, C.; MOZOS, Ó. M.; BURGARD, W. Efficient exploration of unknown indoor environments using a team of mobile robots. *Annals of Mathematics and Artificial Intelligence*, Springer, v. 52, n. 2-4, p. 205–227, 2008. 20, 31, 36, 45, 46, 47, 101

SU, H. *FastSLAM An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping: A Survey*. [S.l.], 2008. 26

THRUN, S.; LEONARD, J. J. Simultaneous localization and mapping. In: *Springer handbook of robotics*. [S.l.]: Springer, 2008. p. 871–889. 26

VISSER, A. et al. Discussion of multi-robot exploration in communication-limited environments. Max Planck Institute for biological Cybernetics, 2013. 19

VISSER, A.; SLAMET, B. A. Balancing the information gain against the movement cost for multi-robot frontier exploration. In: SPRINGER. *European Robotics Symposium 2008*. [S.l.], 2008. p. 43–52. 45, 47

WURM, K. M.; STACHNISS, C.; BURGARD, W. Coordinated multi-robot exploration using a segmentation of the environment. In: IEEE. *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. [S.l.], 2008. p. 1160–1165. 38, 101

YAMAUCHI, B. Frontier-based exploration using multiple robots. In: ACM. *Proceedings of the second international conference on Autonomous agents*. [S.l.], 1998. p. 47–53. 31, 34, 40, 46, 102

ZLOT, R. et al. Multi-robot exploration controlled by a market economy. In: IEEE. *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*. [S.l.], 2002. v. 3, p. 3016–3023. 31, 39, 123