Exploration of Unknown Structured Environments with Multiple Robots

by

Flavio Cabrera Mora

A dissertation submitted to the Graduate Faculty in Electrical Engineering in partial fulfillment of the requirements for the degree of Doctor of Philosophy at The City University of New York

2012

This manuscript has been read and accepted for the Graduate Faculty in Engineering in satisfaction of the dissertation requirements for the degree of Doctor of Philosophy.

Prof. Jizhong Xiao

Date

Chair of Examining Committee

Prof. Mumtaz Kassir

Date

Executive Officer

Prof. Yi Sun, Dept. of Electrical Engineering

Prof. Kenneth M. Sobel, Dept. of Electrical Engineering

Prof. M. Ümit Uyar, Dept. of Electrical Engineering

Prof. Michael Conner, Dept. of Electrical Engineering

Supervisory Committee

The City University of New York

Abstract

EXPLORATION OF UNKNOWN STRUCTURED ENVIRONMENTS WITH MULTIPLE ROBOTS

by

Flavio Cabrera Mora

Adviser: Prof. Jizhong Xiao

Multi-robot systems are expected to perform faster than their single-robot counterpart in different areas of robotics such as exploration, localization and mapping. For exploration, a faster task completion is only one of the advantages of using multi-robot systems. Among other benefits are better reliability, increased robustness and improved efficiency. Coordination of the movement of the robots is required in order to attain those benefits.

As such, a fundamental problem in multi-robot exploration is to know how the robots should coordinate their movements inside the environment in order to perform the exploration process either faster or more efficiently. In this work, we propose two coordination algorithms that address two of the constraints imposed on multi-robot systems: exploration time and total traversed distance. We characterize their behavior mathematically and find out their performance in time and distance.

We consider the situation of multiple robots exploring a structured environment, modeled as a graph, from a single starting vertex. The graph is initially unknown; the existence of edges becomes known only when a robot sees one end of the edge from a

vertex, and the other end of the edge becomes known only when the robot actually follows that edge. This models an environment of sites with passages between them, where the passages are opaque: from either end it is not clear where the passage goes.

The mathematical analysis allows us to obtain the main properties of the algorithms and the bounds of the exploration time. In order to compare the efficiency of the algorithms in time and traversed distance, we derive three criteria of performance for multi-robot systems.

In the last part of this thesis we study the effects of the number of robots in the exploration process. Given the fact that the exploration time cannot be reduced indefinitely, even when the number of robots is increased infinitely, we perform an analysis in order to obtain the limit on the number of robots that produces the maximum reduction in the exploration time.

ACKNOWLEDGMENTS

I would like to express my appreciation towards the members of my doctoral committee, Prof. Jizhong Xiao, Prof. Yi Sun, Prof. Kenneth M. Sobel, Prof. M. Ümit Uyar, and Prof. Michael Conner for their time and valuable suggestions to improve this work. Specially to my adviser, Prof. Xiao, for its guidance and support throughout my doctoral studies, and for allowing me to become part of the great research group that comprises the CCNY Robotics Laboratory. His comments have allowed me to understand better the scientific world in which we live and to steer the direction of the research I should pursue.

I also would like to acknowledge all the fellow members of the CCNY Robotics Laboratory, in particular Mr. Rex Wong and Mr. Xiaochen Zhang, and alumni Dr. Ravi Kaushik and Dr. Xiaohi Li, for their help and comradeship.

I would like to acknowledge Prof. Peter Brass from the Department of Computer Science of CCNY, and Dr. Andrea Gasparri from the Department of Computer Science and Automation of the University of "Roma Tre", for their invaluable comments during the development of the MR-DFS algorithm.

My deepest gratitude goes to my family that have endured not having me around as frequently as they would like to during all these years. Specially, my wife Ivette and my daughter Sofia that have supported me and encouraged me in so many ways, and who have allowed me the time to go away from home for long periods of time in order to pursue this research. To my father and my mother for being the wonderful parents they are, and to my sister for always being there.

Last but not least, I would like to greatly acknowledge my dearest friend and colleague Dr. Clara Nieto-Wire, for her friendship and support, for her clear thinking when things were not, for her advice and for those "*tertulias*" that served as the

inspiration for many great ideas to come. The motivation of a good friend and the love of a good family make these years of research quite enjoyable.

A special mention to those who allowed me to have some financial support during my doctoral studies, specially the CUNY Graduate Center and its Chancellor's Scholarship program, The Office of Research and Sponsored Programs from the CUNY Graduate Center and its Doctoral Student Research Grant Program, and the Electrical Engineering Department of the City College of New York (CCNY) for the opportunity to be part of its teaching staff during these years of research.

Table of Contents

List of Figures

List of Tables

Nomenclature

α	minimum reduction in exploration time that is allowed to occur when then number of robots is increased by one
Λ_i	Total number of traversed edges of robot i
Λ_{total}	Total number of traversed edges of all the robots in the system
$\mu(m_i)$	Excess multiplicity of edge m_i
D_r	Diameter of the root
$diam(\mathcal{G})$	Diameter of the graph
k	Number of robots
k_0	Maximum number of robots that contribute in reducing the exploration time
m	Number of edges in a graph
n	Number of vertices in a graph
r	Root vertex of a rooted graph
t_c	Complete exploration time

t_e Finish exploration time

deg_v Degree of vertex v

Chapter 1

Introduction

1.1 Multi-Robot Exploration

The exploration of a completely unknown environment is one of the cornerstone applications on mobile robotics, along with localization and mapping, because the first task of any autonomous robot is to find its way around it. This holds whether the robot is a Mars Rover, a household cleaning appliance, or on a search-and-rescue mission in a collapsed building. The problem has been well-studied with many different models for a *single* robot exploring the environment, under line-of-sight or distance sensing constraints, in obstacle-dense or sparse environments, with various motion constraints and many other model variants. A particular characteristic of these approaches is the way in which the environment is modeled: as a geometric structure (Blum et al., 1991), (Papadimitriou & Yannakakis, 1991), (Deng et al., 1991); as a grid (Yamauchi, 1997), (Moravec, 1988); or as a graph (Awerbuch et al., 1995; Awerbuch & Kobourov, 1998; Panaite & Pelc, 2000; Dessmark & Pelc, 2002; Duncan et al., 2006).

The situation is much less clear for exploration by *multiple* robots. Multi-robot systems have been regarded as the natural progression for many areas of robotics. Multi-robot systems allow not only for a better reliability, but also provide benefits in terms of increased robustness and improved efficiency. These benefits are attainable only if there exist coordination among the robots (Zlot et al., 2002). For instance, negotiation-based coordination algorithms prevent the robots to repeat tasks that have already been performed by a different robot, reducing the exploration time and traveling distance of the robots, thus increasing the efficiency of the system (Simmons et al., 2000). Decentralized coordination algorithms, prevent the occurrence of the single-point-of-failure problem, increasing the robustness of the system (Sheng et al., 2006).

In this thesis, we consider the situation of multiple robots exploring a structured environment, modeled as a graph, from a single starting vertex. The graph is initially unknown; the existence of edges becomes known only when a robot sees one end of the edge from a vertex, and the other end of the edge becomes known only when the robot actually follows that edge. This models an environment of sites with passages between them, where the passages are opaque: from either end it is not clear where the passage goes. The model assumes that each edge can be traversed by a robot in one time step. This assumption is the same used by other authors in the literature (Awerbuch et al., 1995; Dessmark & Pelc, 2004; Duncan et al., 2006; Panaite & Pelc, 2000, 1999; Fraigniaud et al., 2006, 2008; Cormen et al., 2001).

A fundamental question in multi-robot exploration is how the robots should coordinate their movements inside the environment so that the exploration process is performed either faster or more efficiently. As such, we are interested in developing different algorithms for multi-robot exploration, characterize their behavior mathe-

matically and find out their performance in time and distance.

In particular, this work introduces two strategies that always explore completely a graph. The algorithms represent the complete opposite on the way in which they allow the robots to move in the environment: multi-robot depth first search produces the maximum parallelism on the exploration process by allowing all the robots available to perform the exploration and traverse the edges of the environment at will. On the other hand, the flooding algorithm constrains the number of robots allowed to traverse an edge on each step time to be only one.

The difference in the coordination models of the algorithms is intended to address two different constraints imposed on multi-robot systems. The first constraint and probably the one that has received more attention in the literature is time, since multi-robot systems are expected to perform faster than its single-robot counterpart. However, time is only one of the constraints that govern a multi-robot system. Another important constraint is the cost of exploration, which is related to a variety of different issues on exploration systems such as energy, communication, computation, monetary cost, or distance. In our work, cost of exploration refers only to the overall traversed distance of the robots, which is linked to other limiting factors such as energy consumption.

Time and distance are usually conflicting optimization objectives: the time it takes one robot to explore a given environment can be reduced by allowing a second robot to help in the exploration; however, this will increase the combined traverse distance of both robots. Generally speaking, adding more robots may reduce the exploration time but will definitely increase the overall traverse distance. Consequently, an optimal exploration strategy should *i*) explore the environment in the least amount of time, *ii*) make the robots traverse the overall shortest distance, and *iii*) coordinate the

movement of the robots so that deadlocks are minimized. Thus, the multi-robot depth first search algorithm is intended to address the time constraint while the flooding algorithm is intended to address the traversed distance constraint. The mathematical analysis we perform on the proposed algorithms aims to obtain the bounds for the exploration time. These bounds allow us to observe the behavior of the exploration time for either algorithm given only three parameters (the number of robot k, the number of edges of the graph m and the diameter of the root vertex D_r) and to compare their performance in time among them and with other algorithms (e.g., single-robot DFS). Note that obtaining closed-form expressions that *predict* the actual exploration time is an NP-hard problem (Fraigniaud et al., 2006) since multi-robot exploration algorithms can have different performance in different graphs, when they are not known in advance.

To further the above comparison of the algorithm, we derive three metrics of performance for multi-robot exploration algorithms that consider the time and the overall traversed distance of the robots. These metrics compare the performance of any parallel algorithm to the optimal case.

Another parameter that has not received much attention in the literature is the size of the multi-robot system. That is, multi-robot systems has been perceived as a collection of unbounded number of elements that need not to be constrained in order to allow the system to fulfill all its potentialities. In the area of exploration, although it has never been directly expressed, the notion is that the more robots are performing the exploration, the faster this exploration process is going to be. But a careful observation uncovers the fact that the exploration process cannot be reduced indefinitely, even if the number of robots is increased infinitely. There exists a limit upon which the exploration time cannot be improved and that is determined by the

structure of the environment being explored. A consequence of this, is that there should also exist a limit on the size of the multi-robot system. That is, we should be able to specify the maximum number of robots that provides the maximum reduction on the exploration time. Consequently, we dedicate a section of this dissertation to study the behavior of the average exploration time of the multi-robot depth first search algorithm, in order to obtain an expression in terms of the size of the environment (i.e., number of edges and diameter of the root), that can constrain the size of the multi-robot system.

1.2 Contributions and Outline of the Thesis

The contributions of this thesis are:

- We propose two algorithms for multi-robot exploration of unknown structured environments: *multi-robots depth first search* (MR-DFS) and *flooding* algorithm. Those algorithms are opposite to one another in terms of the level of parallelism allowed for the robots to perform the exploration.

- For MR-DFS, we prove mathematically that the algorithm is always better than single-robot DFS for both cases, general graphs and trees. For trees, we obtain an expression for the upper bound of the exploration time that improves a previous result presented on (Fraigniaud et al., 2006). For two robots on a tree, we prove mathematically that the algorithm is optimal.

- For the *flooding* algorithm on trees, we prove mathematically that the algorithm is never worse than classical single-robot DFS. For trees, we obtain analytically the upper and lower bounds on the exploration time that, according

with simulation results, are tight, improving the trivial upper bound given by the single-robot DFS approach.

- Based on the argument that the performance of multi-robot systems cannot be judged exclusively by the time it takes for them to perform an exploration process, we define three metrics of performance for multi-robot algorithms: the total distance overhead μ_{total}, the time efficiency E_t, and the distance efficiency E_d.

- Based on the argument that there should exists a limit on the number of robots (k_0) upon which the exploration time cannot be improved, and that this limit should be proportional to the size of the environment to be explored, we obtain analytically an expression of k_0 for the MR-DFS algorithm on trees.

This thesis is organized as follows. In chapter 2, background information of this work is presented. In chapter 3, we provide a description of the scenario and the main assumptions that make the analysis of the algorithms possible. Also in chapter 3, a set of definitions for the terms that are most widely used throughout this thesis is presented, so that it serves as a quick reference guide for a better understanding of our work. In chapter 4, the multi-robot depth first search algorithm is introduced and a mathematical analysis on graphs and trees is performed in order to obtain the main characteristics of the algorithm and the bounds (upper and lower) for the exploration time on trees. Simulations are presented that corroborate the analytical results. In chapter 5, the second strategy, flooding algorithm, is presented along with a mathematical analysis for the bounds of exploration time in trees. In chapter 6, we present three metrics of performance for multi-robot systems and perform simulations in order to compare the algorithms of this work. In the second part of chapter 6, we

present an analysis of the effect of the number of robots on the exploration time for the MR-DFS algorithm. The analysis produces two expressions: one for the model of the average exploration time, and the other one for the calculation of the limit of the number of robots that produces the maximum reduction on the exploration time. Simulations are performed to verify the accuracy of both models. Finally, in chapter 7, we state the main conclusions and present recommendations for a future extension of this work.

Chapter 2

Background and Related Work

2.1 Multi-robot Exploration

2.1.1 Exploration of Unknown Environments

The previous work on exploration can be roughly divided into the following classes, according to the underlying model where the environment can be:

1. a geometric structure represented as union of polygonal obstacles
2. a geometric structure represented as raster cells
3. a graph structure with uniquely identifiable vertices
4. a graph structure with anonymous vertices which need to be marked to be recognized
5. a directed graph structure

Each of these models has its motivation, and has been studied in numerous variants. The first model has been studied in (Papadimitriou & Yannakakis, 1991; Deng et al., 1998; Blum et al., 1997); it typically assumes that the robot knows everything within line-of-sight visibility, and is thus related to Art Gallery problems (O'Rourke, 1987), but differs from watchman tours (Chin & Ntafos, 1986; Ntafos & Gewali, 1994) in that the polygons are initially unknown. This model is popular in the computational geometry community, as an example we cite (Hoffmann et al., 2001), where a competitive algorithm for exploring the inside of a simple polygon is given, and (Fleischer et al., 2008), where the optimal competitive ratio is studied.

The second model is more popular in the robotics community: the environment is viewed as a grid in which some cells are open, others blocked, and still others unknown, or more complicated cell states as in evidence grids (Moravec, 1989). This model is more compatible with diverse types of sensing, like line-of sight, fixed radius, limited viewing angle, etc. In this model, even the exploration of a mostly empty plane might be nontrivial, solved in (Yamauchi, 1997), for instance, by maintaining and following the frontier of the unexplored terrain; but in an obstacle-dense environment, that frontier might decompose in many components.

The third model is the model assumed in this research: the environment is given as a graph, nodes corresponding to locations and edges to passages between the locations. Edges are assumed to be opaque: we know where an edge leads only when we have explored it. This is a natural model, both as an abstraction of obstacle-dense environments that we may divide into cells corresponding to the graph nodes, and as a model for state space exploration when the state transitions work in both directions. The assumption that the vertices are identifiable, and will be recognized when revisited, is reasonable in this context, and an essential model property. It has

long been known that depth-first search is an efficient method to explore any graph by a single robot in this model, at most by a factor two slower than the optimum exploration strategy. A number of papers studied the influence of further information, and decreased the factor-two gap for specific graph classes (Panaite & Pelc, 2000; Dessmark & Pelc, 2004), or simulated breadth-first search, where the robot always maintains a short return path to the start vertex (Awerbuch & Kobourov, 1998; Awerbuch et al., 1999; Duncan et al., 2006).

The fourth model, which differs from the third by the nodes being anonymous, and recognizable only by a marker placed on them, or by their degree or other abstract graph properties, comes from the labyrinth exploration setting. The question for the smallest capabilities, like how many "pebbles" or how many bits of memory, that allow an abstract robot to find its way out of a labyrinth, is a classic and much-studied question in the theory of computation (Budach, 1975; Hoffmann, 1981; Bender et al., 2002; H.Wang et al., 2008; Gasieniec et al., 2007). For real robots, the question seems irrelevant, since the robot can recognize its position by other means like odometry, GPS-coordinates, or a picture of the node environment.

The fifth model, exploring a directed graph, was studied in (Deng & Papadimitriou, 1999). The situation changes fundamentally from the undirected graph by the fact that you cannot go back an edge and as such depth-first search becomes impossible. This model is equivalent to exploring the state space of an unknown finite automaton; for any input, there happens some state transition, initially unknown to us. The states correspond to vertices and the transitions to directed edges, and we recognize states we have visited before. This has been proposed in (Deng & Papadimitriou, 1999) as model of learning: each action makes a change on the outside world. Initially, we do not know the effect of the actions, but by trying the actions and rec-

ognizing previous states we acquire knowledge about the possible actions. This model again has been studied in theoretical computing (Kwek, 1997; Albers & Henzinger, 2000; Fleischer & Trippen, 2005). (Das et al., 2007) extends the research to exploring a directed graph by multiple robots.

Of these different exploration models, only the second (grid) has received wider study in the context of multi-robot exploration. A major problem for the grid model is the fusion of the exploration maps of the individual robots. This problem does not occur with the graph model, even when starting from a continuous or a grid model. Thus, deriving a graph as a representation is a reasonable step (Dudek et al., 1991). The graph might even be made physical by dropping nodes in the explored region (Batalin & Sukhatme, 2004, 2007). The frontier approach is extended to multiple robots in (Yamauchi, 1998). For multi-robot undirected graph exploration, which is our underlying model, the most relevant paper is (Fraigniaud et al., 2006).

2.1.2 Multi-robot Exploration with Active Landmarks

The exploration of unknown environments using an arrangement of robots and active landmarks (tokens) has been a topic of research in recent studies where one major difference is the role played by the landmarks. Our proposed scenario considers that the tokens are deployed every time a robot reaches a junction or when the communication range of the previously deployed token is reached. Whenever another robot reaches that junction, it will be able to identify the landmark. The purpose of using landmarks is that they can provide some kind of information to the exploring robots, either by giving direct instructions to the robots or by suggesting the presence of another robot in the vicinity. This is a behavior that has been observed in nature on some ant colonies in which foragers in search of food sources leave chemical traces

indicating the best direction towards the source (Fourcassie & Deneubourg, 1994; Detrain et al., 1999; Fourcassie et al., 2010).

Landmarks have been considered before in (Vaughan et al., 2002), where a robot stores its own coordinates when an event has occurred and transmits this information to other robots that can observe the same event through wireless communication. The definition of an event is very loose (task-relevant occurrence) and in general the robots do not mark the environment. This approach diverges from ours by the fact that a landmark is not a physical element, so that it cannot provide any useful information to robots far from the event.

In (Batalin & Sukhatme, 2004), the dynamic coverage problem is approached by using a single robot that deploys nodes capable of communication and sensing, and that are equipped with 2-bit compasses letting the node to have the four cardinal directions. The robot deploys a node every time it cannot find a node within communication range. When required, each node can communicate locally with the robot and recommend a preferred direction for it to take. The criterion for the suggestion follows the *least recently visited* (LRV) direction policy. LRV uses a counter for each one of the four possible directions. Each time the robot travels towards a specific direction, the counter is increased by one. The suggested direction given by the node is the one with the smallest counter value. The robot combines the node suggestion with its own local sensing and makes a final decision about what direction to pursue. In (Batalin & Sukhatme, 2007), the authors study two properties of the LRV algorithm described above, namely completeness on graphs and optimality on trees. Our approach considers a different criterion for the deployment of the nodes: a node will be dropped not only when a connectivity criterion is met but also in the intersection where more than one path can be followed by the robot (junction). As stated

in chapter 3, we are assuming that each robot is capable to determine when it has reached a junction and to distinctively identify the direction of each unexplored path. Consequently, we are not restricted only to the four cardinal directions.

In (Yamauchi, 1997), a frontier based method is used where an occupancy grid map is filled up progressively as a single robot gets more information about the environment. A frontier between known and unknown space is created and the robot always moves toward the unknown space in order to increase its knowledge about the environment. A multi-robot version of this approach is presented in (Yamauchi, 1998) where each robot has two different evidence grids: one for its local information and the other, a global one, to save the information incoming from other robots. In this case, each robot maintains its own local and global maps and when it encounters another robot, both exchange their local maps, allowing the other robot to update their respective global maps. There are two clear drawbacks to this approach: firstly, all the robots must share the same coordinate system; and secondly, the exchange of information is only possible when the two robots actually meet. This means that it is possible for a robot to unknowingly explore a region that has already been explored for somebody else. These drawbacks prevent this approach to be optimally efficient.

Another approach for multi-robot exploration using the frontier-based approach is given in (Rooker & Birk, 2007) where a group of robots explore an environment maintaining constant communication among them. This approach relies in a utility function that weights the exploration of unknown area versus the preservation of communication among the robots. A problem with this approach is that the exploration is not being done by all the robots at a time since many of them are kept static to maintain connectivity.

In a sense, we are applying the frontier concept mentioned in (Yamauchi, 1997)

and (Yamauchi, 1998), where the frontier is kept not by the robots but by the nodes. Since we are considering a multi-robot scenario, whenever a robot finds a node, the latter will instruct the robot to go towards the unexplored direction. This means that the decision about the next direction of the robot is mainly taken by the nodes, creating a decentralized system, freeing robots resources for other tasks.

2.1.3 Exploration of Graphs

Exploration of graphs has been a well-studied problem in the mathematical and computer science communities, especially for the single-agent case, for both directed and undirected graphs, but the multi-robot scenario is still an open research topic. These studies have relied on several different algorithms that have been developed in order to accomplish graph exploration (graph traversal), such as those shown in table 2.1, all with different degrees of efficiency depending on the purpose of the exploration.

From these, depth-first search has been regarded as one of the most efficient algorithm to explore any graph by a single robot. As such, the algorithms we have developed on this work are based on the depth first search algorithm. Another peculiarity of our work is that we consider the use of low power transceivers as physical active landmarks that are deployed by the robots while they explore the environment and that can give some directions to other robots that encounter them along the way. Since the exploration of an unknown environment, specially an indoor one, can be assimilated to the problem of traversing a graph (visiting all vertices and/or traversing all edges) we are interested in the time it takes a set of robots to do the complete exploration (i.e., the exploration time). Thus, we consider the exploration time as a measure of efficiency. For us, this is the time until the entire graph has been explored, and the last robot has returned to the starting position. Let us consider

an arbitrary graph with m edges. In order to be explored, a robot must follow each edge at least once. Thus, for a single robot the exploration time is at least m (the exploration time will be equal to m steps in the special case that the graph is just a loop with m edges). If the graph is a tree the time is at least $2m$ (because the robot needs to return along each edge toward the starting vertex for the tree to be considered explored)

In (Dudek et al., 1991), it was proven that by letting a single robot to mark the environment with passive nodes the exploration time is dramatically improved. In (Bender et al., 2002), it was shown that a single robot can learn its way through a directed graph with n vertices in time polynomial in n, using only one pebble, if the upper bound on the total number of vertices n is known. If, on the other hand, this upper bound is not known, it is stated that $\Theta(\log \log n)$ pebbles are both necessary and sufficient to perform the exploration. However, it is not mentioned how much the exploration time will be improved when using more than one pebble.

Two main drawbacks with these approaches can be pointed out: Firstly, if we consider that the nodes are anonymous, and recognizable only by either a marker placed on them, by their degree, or by any other abstract property, the question to be solved is not how efficient an algorithm will make a robot to explore the graph, but how many "pebbles" will allow the robot to find its way out of the environment (labyrinth exploration problem). This is due to the reduced capabilities of the nodes and the fact that they cannot provide any useful information to the robot that contribute to the exploration process. This is a classic and much-studied question in the theory of computation (Budach, 1975; Hoffmann, 1981; Bender et al., 1998; H.Wang et al., 2008; Gasieniec et al., 2007). For real robots though, the question seems irrelevant, because in general a robot can recognize its position by other means like odometry,

GPS-coordinates, or a picture of the environment. Especially in indoor environments, where those tools for position estimation mention before are either unavailable (GPS), or are regarded as inaccurate (odometry, environment picture), other means have been studied. On the contrary, low power transceivers (LPTs) are active elements capable of storing and transmitting information at will, make them easily distinctive to the robots.

Secondly, exploring a directed graph has been studied in different works such as (Deng & Papadimitriou, 1990). The fact that you cannot go back an edge makes algorithms such as depth-first search impossible to implement. This model is equivalent to exploring the state space of an unknown finite automaton: for any input, there are some state transitions initially unknown to us. The states correspond to vertices and the transitions to directed edges, and it is possible to recognize states that have been visited before. This has been proposed in (Deng & Papadimitriou, 1990) as a model for learning: each action makes a change on the outside world. Initially, we do not know the effect of those actions, but by trying the actions and recognizing previous states we acquire knowledge about the possible outcome of the actions. This model again has been well-studied in theoretical computing (Kwek, 1997; Albers & Henzinger, 2000; Fleischer & Trippen, 2005). (Das et al., 2007) Extends the research to exploring a directed graph by multiple robots. This scenario might apply to the exploration of very restrictive environments such as collapsed buildings, where a robot can traverse in one way but, because of debris, might not be able to come back through the same edge. This can be considered as a special case of the more general and unsolved problem that deals with undirected graphs. In this regard, an undirected graph is a more realistic assumption because we can argue that if a robot can move in one direction, it is likely that it can do it in the opposite direction as

well.

A number of papers studied the influence of further information, and decreased the factor-two gap for specific graph classes (Panaite & Pelc, 2000; Dessmark & Pelc, 2002), or by using breadth-first search, where the robot always maintains a short return path to the start vertex (Awerbuch et al., 1995; Awerbuch & Kobourov, 1998; Duncan et al., 2006).

Exploration of undirected graphs was considered previously in (Awerbuch et al., 1999), where the goal is to achieve an efficient exploration under the constraint that the robot must return to the root for refueling periodically. They defined efficiency as the total traversed distance of the robot, and reported that with their strategy a robot explores a graph with n vertices and m edges by traversing at most $O(m + n^{1+O(1)})$ edges with overhead $O(log^2 n)$. In (Panaite & Pelc, 1999), the exploration of all vertices and all edges was achieved while minimizing the total number of edge traversals. They reported that the overhead in the number of traversals (*penalty*) of their exploration algorithm is $O(n)$ for every graph.

Multi-robot exploration of trees has been studied for example in (Dynia et al., 2006), where an exploration algorithm explores sparse trees with k robots, and it is shown that its competitive ratio is influenced only by the density and the diameter of the tree. In this approach, the communication exists exclusively among robots and it occurs only when robots meet in the same vertex. In (Fraigniaud et al., 2006), k robots are used to explore a tree with m edges and root diameter D_r, using a communication model that allows all robots to exchange information at every step of the process through the use of tokens. The authors proposed an exploration algorithm with a running time $O(m/\log k + D_r)$, which is $O(k/\log k)$ larger than the optimal exploration time when the tree is known a-priori.

2.2 Analysis of Multi-robot systems

Multi-robot systems performing the exploration of unknown environments is a problem that can be analyzed by considering other aspects different from time. For instance, the cost of performing the exploration. The cost can be related to many different factors such as how much the robots on the system must travel in order to complete the exploration, or how many robots should be used to do it.

Cost of exploration has been previously evaluated particularly in task allocation of multi-robot systems, where market economy-based approaches have gained relevance in recent years because they allow for the successful minimization of the time and traveling distance. It is noteworthy that cost of exploration relates to very different issues on exploration systems such as time, energy, communication, computation, monetary cost, or distance. In this thesis cost of exploration is related exclusively to traveling distance, and as such, we use both expressions interchangeably. We have restricted the cost of exploration to the traveling distance of the robots because how far a robot needs to move in an unknown environment will determine other parameters on the system, such as type of robots that need to be deployed, amount of energy that will be consumed during the exploration process, etc. The market economy-based approaches use the cost as a parameter of an auction process where robots present bids to an auctioneer (the element offering the task) that determines which robot wins the task. In (Zlot et al., 2002) a distributed multi-robot exploration approach aims at maximizing the area being explored while minimizing the cost of exploration, where cost is defined as the collective traverse distance of the robots. The goal of the exploration process is the construction of a map of the environment which is represented by an occupancy grid. The team of robots continuously negotiates with

each other by means of single-item auctions. Similar approaches are used in (Dias & Stentz, 2002), (Burgard et al., 2005) and (Gerkey & Mataric, 2002).

In (Berhault et al., 2003), for instance, robots negotiate using combinatorial auctions, where bundles of targets are presented to the auctioneer, which in return decides what target is assigned to each robot. The performance of the multi-robot system is measured in terms of the travel cost and the travel time where the former is defined as the sum of the travel distances of all robots, which determines the amount of energy consumed; and the latter is defined as the total time needed to complete the exploration task. As such, both criteria of performance used are similar to the definitions of cost and time used in our work. The market-based approach has also been studied using different world representations. In (Wu et al., 2007), for example, a polygonal representation of the world is used in order to increase the efficiency of exploration by dividing the available space using Voronoi diagrams. In (Elizondo-Leal et al., 2008), a distributed multi-robot coordination algorithm for exploration and mapping is presented, where each robot calculates the bids of the other robots and make a decision on its own. The calculated bids consider the cost of reaching a frontier between known and unknown area.

Our study is related to these market-based approaches in the sense that the robots inform the active landmark in a vertex their interest to move there in the next step (this can be seen as a bid placed by each robot attempting to reach that vertex). The token then collects all the bids and give them to each robot in the auction. Each robot acts as a local auctioneer that decides, using a strict set of rules, which robot is allowed to arrive to the vertex in the next step. The bids of the robots are not related to the cost of the movement itself but to the characteristics of such movement (returning to a parent vertex, exploring an unused edge, etc). Since all the robots are

knowledgeable of these set of rules given by the algorithm, all of them will draw the same conclusion about who is the winner of the auction.

Algorithm	**Brief description**
A*	It finds the least-cost path from a given starting vertex to a given goal. Incrementally searches all routes leading from the starting point until it finds the shortest path to a goal.
B*	Related to the A* algorithm since it finds the least-cost path from a given starting vertex to a given goal but it stores intervals for nodes of the tree as opposed to single point-valued estimates.
Best-first search	Explores a graph by expanding the most promising node chosen according to some predefine rule in order to reach a given goal. It may depend on the description of the goal, the information gathered by the search up to that point, and most important, on any extra knowledge about the problem domain.
Bidirectional search	Runs two simultaneous searches: one forward from the initial state, and one backward from the goal, and stopping when the two meet in the middle. It is a computationally expensive algorithm.
Breadth-first search	Begining at the root, it explores all the neighboring nodes first. Then for each of those nearest nodes, it explores their unexplored neighbor nodes, and so on, until it finds the goal
Depth-first search	Beginning at the root, it expands all its children by selecting one at a time and exploring it until further progress is not possible. Then, it backtracks to previous node. A blocking point is a node without children. The process is repeated until the goal is achieved. Works well when there are many solutions for the search and all of them are desirable
Hill climbing	Beginning at the root, it expands the node that offers the greatest promise of successfully reach the given goal. Then, it repeats the process in the subsequent node it arrives, but it does not stores any information about the parent node or other sibblings of it. It requires a highly informative evaluation function in order to avoid falling in local minima situations. It is an irrevocable strategy because it does not let the robot to shift the attention back to previously suspended alternatives.

Table 2.1: Summary of different algorithms for graph searching (Pearl, 1984)

Chapter 3

Definitions, Scenario and Assumptions

This chapter provides a description of the scenario used in our analysis and the most important assumptions that are considered throughout the rest of this document. It also includes a section with definitions, that it is intended as a reference guide for the reader to navigate quickly through the terminology used in this work.

3.1 General Considerations

The scenario of our research is a structured environment, formed by corridors and obstacles, predominantly present in indoor scenarios. We consider a multi-robot system formed by multiple identical robots that explore the environment following the same exploration algorithm. The robots enter the scenario from a single point and maintain their own coordinate system. The robots are assisted by *active landmarks/tokens* that are deployed by the robots while performing the exploration. Note that the terms

active landmark and *token* are used interchangeably throughout this work. The active landmarks help the robots on two areas: communication and navigation. The active landmarks are elements capable of storing and transmitting information at will. The type of information manipulated by these elements will be defined further down. These landmarks provide directions to the robots that find them along the way. The criteria for token deployment are *i*) the communication range, since we are interested in maintain connectivity among the tokens and between tokens and robots, so that in any event the robots can track back its route to exit the environment, and *ii*) the presence of a junction. Junctions are places in the environment where there are more than one path to follow and a robot is required to choose any of those paths to follow in the next step. We assume that each robot is capable of determining when it has reached a junction and distinctively identifying the direction of each unexplored path. The process of exploring and dropping tokens creates a trail for each robot in the scenario. Eventually, a robot may encounter either *i*) its own trail or *ii*) the trail left by other robots. In the second case, both trails might merge creating a bigger one. As such, the exploration process can be assimilated to a graph construction process, where junctions represent vertices and paths between junctions represent edges (see Fig. 3.1).

3.2 Definitions

In this section we present some terms that are widely used throughout this thesis and deserve to be properly defined for a better understanding of our work. The following definitions are terms commonly found in the graph theory literature:

Definition 3.1 (Connected Graph) *A connected graph is one in which there is a*

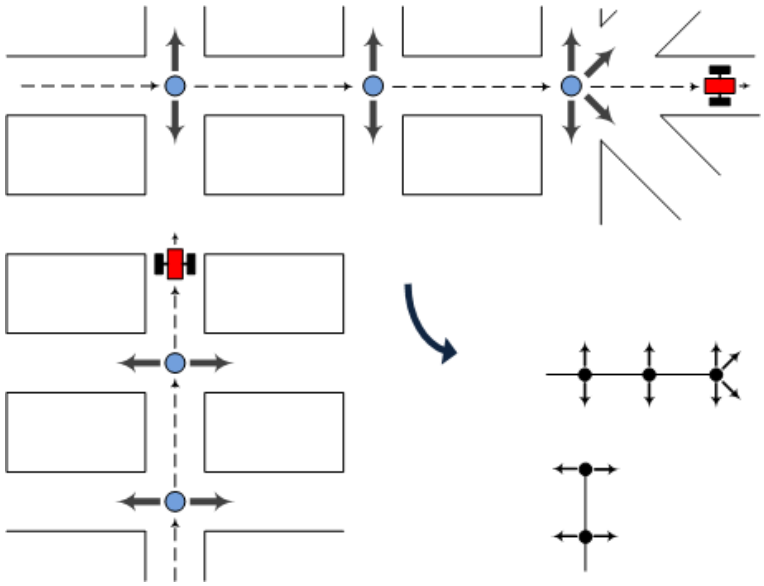


Figure 3.1: Robot exploration of an indoor environment as a graph creation process

path from any vertex to any other vertex in the graph. In this work we consider only finite connected graphs.

Definition 3.2 (Tree) *A tree $\mathcal{T}$ is a connected graph in which there exists only one path from one vertex to another vertex, and no loop paths exists.*

Definition 3.3 (Loop) *A path from a vertex to itself.*

Definition 3.4 (Rooted Graph/Tree) *A rooted graph/tree is one with a distinguished vertex called the root (r). The vertices in a rooted tree form a hierarchy, with the root at the highest level, and the level of every other vertex being determined by its distance from the root (Harris et al., 2008). A rooted graph is usually draw in such a way that the root is at the top, and the vertices at each level are horizontally aligned.*

Definition 3.5 (Root Vertex) *The root of a graph/tree is a special single vertex in the graph that allows for the graph to have a natural orientation toward or away from that vertex.*

Definition 3.6 (Parents, Children and Leaves) *A parent vertex is one with at least one edge leading to another vertex in a downward direction. The latter vertex is called a child vertex. A leaf is a vertex with no children.*

Definition 3.7 (Internal Vertex) *The vertices that are not leaves are said to be internal vertices. The root counts as an internal vertex.*

Definition 3.8 (Degree of a Vertex) *The degree of a vertex v (deg_v) is the total number of edges incident to the vertex.*

Definition 3.9 (Distance) *The distance $d(u, v)$ between two vertices u and v of a finite connected graph is the minimum length of the paths connecting them.*

Definition 3.10 (Eccentricity) *The eccentricity $\epsilon(v)$ of a vertex v is the maximum graph distance between v and any other vertex u of $\mathcal{G}$.*

Definition 3.11 (Diameter) *The graph diameter $diam(\mathcal{G})$ is the maximum eccentricity of any vertex in the graph.*

Definition 3.12 (Radius) *The graph radius $rad(\mathcal{G})$ is the minimum eccentricity of any vertex in the graph.*

The following definitions are from terms specially created for the work and analysis presented in this thesis.

Definition 3.13 (Diameter of the root) *The diameter of the root D_r is equivalent to the eccentricity of the root vertex $\epsilon(r)$*[1].

[1]Throughout this work, we use D_r instead of $\epsilon(r)$ because the term *diameter* conveys the message that our analysis considers the longest distance from r to any point in the graph. Do not confuse with the diameter of the graph, since in general $D_r \leq diam(\mathcal{G})$

Definition 3.14 (Back-pointer) *a list of all edges a robot has visited in order to be in its current position. This list can be used later on by the robot to back-trace its steps to the root r.*

Definition 3.15 (Finished tree) *A tree in which all edges have been traversed and all vertices have been visited at least once by any robot.*

Definition 3.16 (Completed tree) *A tree that is finished and in which all robots have exited the tree (i.e., all robots have returned to r).*

Definition 3.17 (Finish exploration time - t_e) *The time (in steps) needed for a group of k robots to finish a tree.*

Definition 3.18 (Complete exploration time - t_c) *The time it takes k robots to traverse every edge, visit every vertex of a tree, and return back to the root.*

Definition 3.19 (Total number of traversed edges - Λ_{total}) *It refers to the cumulative number of edges that all the robots that enter the tree (k_m) will traverse from the beginning of the exploration process (step 0) until it is completed (t_c). That is*

$$\Lambda_{total} = \sum_{i=1}^{k_m} \Lambda_i, \tag{3.1}$$

where Λ_i is the total number of edges traversed by robot i.

Definition 3.20 (Unused/Unexplored Edge) *An edge that has not been traversed by any robot.*

Definition 3.21 (Open edge) *An edge through which a robot left a vertex v, without returning back to it or sending a message that it has reached a leaf (i.e., the robot is still exploring downwards the subtree rooted at v).*

3.3 Scenario and Assumptions

In this section we formally define the scenario and establish the main assumptions that make the analysis of the algorithms possible.

We assume that our environment can be modeled as a unknown, finite, connected, undirected graph $\mathcal{G} = \{V, E\}$, formed by n vertices and m edges. We consider that k robots are initially located at the root vertex r of $\mathcal{G}$. We consider that the robots enter the environment by this single point, and that they return to this vertex once the exploration is complete. We assume that the robots can move freely back and forth between vertices. We consider that all vertices are identifiable, and will be recognized when revisited. This is a reasonable assumption and an essential property of our model.

Each robot is uniquely identified with an integer number from 1 to k. Robots explore the environment by moving along the edges that link a pair of vertices. They are helped by active landmarks (tokens) that the robots deploy while performing the exploration.

An active landmark is a bookkeeping device that has communication capabilities and is able to store at least the following information:

- The number of edges converging in a vertex, where each one of these edges is locally labeled with an integer number from 1 to j.
- The ID of the robots that have visited the vertex and the edge each robot has taken after visiting it.
- A list of available edges for robots to explore.

We assume that the robots are capable of:

- Carrying an unlimited number of tokens.
- Assessing whether they have reached a vertex.
- Deploying a token if none is present in a vertex.
- Distinctively identifying every edge on that vertex.

We assume that the active landmarks are identical and their communication ranges determine the length of each edge in the tree (i.e., the places where the active landmarks are deployed constitute the vertices of the tree). Note that although the communication range of each token may be different due to different conditions in the environment, we assume that the time (measured in steps) that takes a robot to move from one vertex to another is the same and equals one step.

We assume that the communication between a robots and a token in a vertex, and between tokens, is either instantaneous or it takes the same amount of time. In any case, we consider that the time spent performing the required communication is negligible.

Chapter 4

Multi-Robot Depth First Search Algorithm

The first algorithm we have developed considers the use of a fixed number of robots to explore the environment. The algorithm offers the maximum parallelism on the exploration an unknown environment because it allows the robots to enter the environment all at once, and does not restrict the number of robots that traverse an edge simultaneously or the number of robots that can arrive at a vertex in each step of the exploration process.

In this section we present in detail this algorithm, called from now on *Multi-Robot Depth First Search*, discuss its properties, perform a theoretical analysis concerning its performance on general graphs and on trees, prove that the algorithm is never worse than depth-first search with a single robot, prove that on trees the algorithm is optimal for two robots, and perform simulations in both tree and graph explorations to corroborate these mathematical results.

4.1 General Considerations

The algorithm, Multi-Robot Depth First Search (MR-DFS), is a natural adaptation of Depth First Search (DFS) to parallel search by multiple robots. The idea of the algorithm is simple: an edge is considered finished, if a robot, leaving a vertex v, follows an edge and later returns to v by that same edge. By this we assume that the robot has explored everything that can be reached by that edge. As long as there are unfinished edges, the robot selects one of them to explore; only if all edges have been finished, it returns by the edge by which it originally entered the vertex.

This natural strategy can be used in many settings; most relevant to real implementation would be a completely asynchronous movement of the robots. For our analysis, we assume the robots to move synchronously in time-steps, and we want to minimize the total number of time-steps before the robots return to the start vertex and declare the search completed. In each time-step, we assume that robots standing at the same vertex have an initial negotiation phase in which they decide which robot takes which edge. The robots at the same vertex announce one after another which edge they will follow. The decision of each robot is based on the edges that have been already taken. Since there is wireless communication between robots standing at the same vertex, we can assume it to be instantaneous, and does not contribute to the duration of the exploration.

Beyond this local communication, our algorithm requires only very weak communication between the robots: a robot arriving at a vertex must be able to see whether this vertex has been visited before. If that is the case, it should be able to know by which edges the robots have left the vertex, and by which edges they have returned. This communication is classically achieved for human explorers by leaving

chalk marks on the exits; for robots, the first robot to enter a vertex would drop a bookkeeping device, on which every robot, that visits this vertex afterwards, will register the sequence of its entering and leaving edges. Note that, at first, additional communication does not appear useful, since in our lower bound we allow complete shared information, and the algorithm almost reaches the lower bound even with this vertex-local information only. Furthermore, this is the same communication model used in (Fraigniaud et al., 2006).

4.2 MR-DFS Algorithm

Algorithm 4.1 provides a description of the MR-DFS for general graphs. On trees, the algorithm becomes simpler since all robots enter a vertex by the same edge for the first time, coming from the root, and it cannot happen that a robot reenters a vertex by a different edge than the one by which it left that vertex.

At each vertex, a bookkeeping device is dropped by the first robot to visit that vertex, and updated by all further robots on every visit. MR-DFS requires the following minimal set of information to be stored at each vertex:

- the number of edges converging in this vertex.
- the ID of the robots that have visited this vertex before.
- for each of these robots, the original entrance edge of the robot
- For each edge, the IDs of the robots entering and leaving through that edge

Thus, every edge that is followed by a robot will be recorded, including the direction, by the bookkeeping devices at either end. If we assume that each robot, entering a

Algorithm 4.1: Algorithm Multi-robot DFS - general graph version

```
1  Let rob_i be a robot arriving at a vertex v through edge e;
2  if rob_i has been at v before, and the edge e by which it returned is different
   from the edge by which it last time left v then
3  |  Mark e as finished edge, go back through edge e;
4  else
5  |  Either v is a new vertex for rob_i, or it returned to v after exploring the
   |  component to which edge e leads;
6  |  if rob_i has never been at v before then
7  |  |  Mark e as the original entrance edge of rob_i to v;
8  |  else
9  |  |  rob_i has been at v before, and returned by the same edge e by which
   |  |  last time it left v;
10 |  |  Mark e as finished edge;
11 |  end
12 |  if there is an edge leaving v that is neither finished, nor the original
   |  entrance edge of any robot to v then
13 |  |  Choose one of those edges, preferring edges that have been used by the
   |  |  least number of other robots before, and leave v by that edge;
14 |  else
15 |  |  Return from v by rob_i's original entrance edge;
16 |  end
17 end
```

Algorithm 4.2: Algorithm Multi-robot DFS - tree version

```
1  Let rob_i be a robot arriving at a vertex v through edge e;
2  Either v is a new vertex for rob_i, or it returned to v after exploring the subtree
   to which edge e leads;
3  if v is a new vertex, not visited by any robot before then
4  |  Mark e as the original entrance edge to v;
5  end
6  if rob_i has been at v before then
7  |  Mark e as finished edge;
8  end
9  if there is an edge leaving v that is neither finished, nor the original entrance
   edge to v then
10 |  Choose one of those edges, preferring edges that have been used by the
   |  least number of other robots before, and leave v by that edge;
11 else
12 |  Return from v by rob_i's the original entrance edge;
13 end
```

vertex, finds the same exits and find these exits in the same sequence (e.g., starting north and enumerating clockwise), we need to store the information for each exit only if a robot has entered the vertex through that particular exit (therefore, the edge is either finished or its original entry edge). If, on the other hand, the robot has not used it, we need to store the number of robots that have left through that edge. This information is sufficient for the algorithm and its analysis; the actual identity of the robots does not need to be stored on the bookkeeping device.

To summarize, each robot running the MR-DFS algorithm follows essentially a tree, starting at the common start vertex. If a robot meets his own tree by a different edge, it will immediately leave along that edge again (line 2–3, Algorithm 4.1). If a robot meets the tree of another robot (i.e., two robots meet at the same vertex entering it through different edges), they will divide the outgoing edges for exploration, each choosing some unexplored edges, as long as possible (line 12–13: Algorithm 4.1). At

any time and for each robot that has visited a vertex, there is at most one edge by which a robot left without returning back. If several robots jointly explore the outgoing edges of a vertex, and a returning robot finds no unexplored edge any more, it will join another robot in the branch the other is still exploring. Only when each edge has been followed by a robot in both directions, the robot returns from that vertex by its original entry edge (line 7: Algorithm 4.1).

Fig. 4.1 shows two robots exploring a graph from a common starting vertex, with their path after five, eight, eleven, and fifteen steps. The dotted blue line represents the path of robot rob_a, and the dashed red line represents the path of robot rob_b.

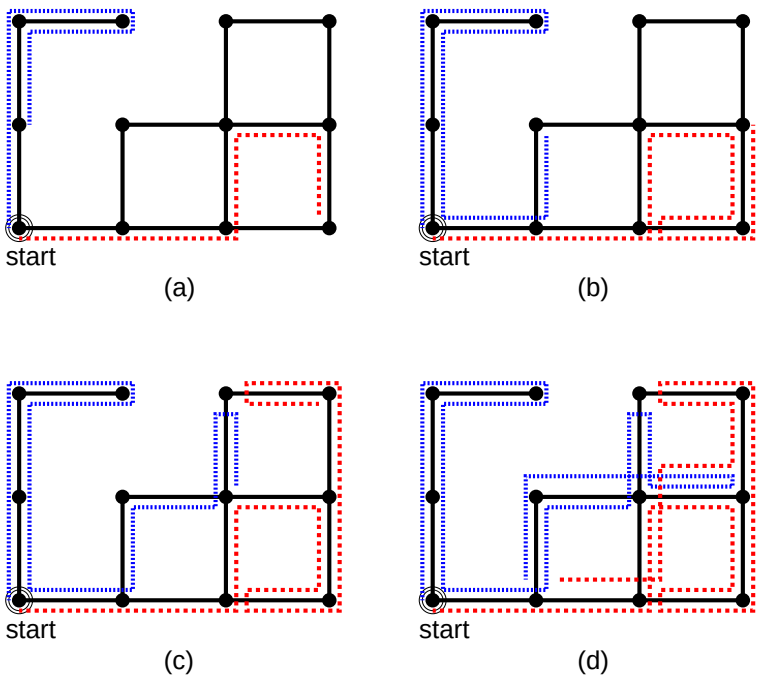


Figure 4.1: Path of two robots after a) five, b) eight, c) eleven, and d) fifteen steps

4.3 Theoretical Analysis

In this section, a theoretical analysis of the MR-DFS algorithm is proposed. The goal is to provide a characterization of the MR-DFS exploration time on general graphs and trees.

4.3.1 Preliminaries

Let us consider a graph $\mathcal{G} = \{V, E\}$ modeling an environment to be explored. The graph is considered to be completely explored if and only if every edge is followed by at least one robot and all the robots return to the starting vertex. This requirement that the robots return to the starting point at most doubles the exploration time, since they could just follow their way back. The number of rounds required in our model to completely explore the graph is the complete exploration time t_c.

If there is only one robot, the exploration time for a graph is at least $t_c = m = |E|$, since every edge needs to be followed. If the underlying graph is a tree, then every edge a robot uses to leave a vertex must be the same one that the robot uses when returning back to the vertex; therefore, the exploration time is at least $2m$. Classical DFS does explore any graph with one robot in $2m$ steps. Thus, the single-robot scenario has an easy solution, which is optimal for trees and at most a factor two slower for arbitrary graphs.

If there are k robots available, the best we can hope for is a speed up factor of k. In each round, k new edges are explored, therefore, we need at least $\left(\frac{m}{k}\right)$ rounds for a general graph, and $\left(\frac{2m}{k}\right)$ rounds for a tree. Unfortunately, this speed up factor is not always possible: if the graph is just one long path of length D_r from the starting vertex, one robot would need to travel all the length D_r and return back, regardless

of the number of robots there might be available at the common starting vertex. If D_r is the diameter of the root of the graph, that is, the longest distance from the starting vertex to any other vertex in the graph, then one of the robots has to reach that vertex at maximum distance, and return back. Therefore, we have the following two lower bounds for the complete exploration time t_c:

- m/k, since each edge needs to be traversed by at least one robot;
- $2D_r$, since a vertex at maximum distance must be visited by at least one robot.

Therefore, for a given graph $\mathcal{G}$ with m edges and root diameter D_r, the *general lower bound* for the exploration time can be defined as

$$\begin{aligned} t_c &\geq \max(m/k, 2D_r), \text{ for general graphs, and} \\ t_c &\geq \max(2m/k, 2D_r), \text{ for trees.} \end{aligned}$$

Since the optimum strategy, which knows the graph in advance and just has to visit all edges, takes at least this time, any algorithm that is within some factor of this lower bound, is competitive and of interest.

4.3.2 Analysis on General Graphs

In order to characterize the exploration time of the MR-DFS on general graphs two important properties must be introduced.

Lemma 4.1 *In the MR-DFS algorithm, each edge is used by each robot at most once in either direction.*

Proof: To proof this lemma, we assume that robot rob_i departs vertex u in direction of vertex v following the edge uv. Let us assume it follows this edge from

u to v twice, at times t_1 and t_2. Between these times, rob_i returns at least once to vertex u. Each time rob_i returns to u by a different edge than vu, it will immediately return back using the edge by which rob_i just enter u (see lines 2 and 3 of algorithm 4.1). Therefore, rob_i must return once by vu, but then it will mark uv as finished and will not follow this edge a second time. □

Lemma 4.2 *In the MR-DFS algorithm, all robots finish their exploration at the same time step.*

Proof: We prove this lemma by contradiction. Let us assume that a robot rob_1 has already returned to the origin and found no further eligible edge, thereby declaring the search finished, whereas rob_2 is still out at a different vertex at that same time step. The robot rob_2 is connected to the start vertex by its return path $[v_p, v_{p-1}, \ldots, v_1]$, with v_p being the current position of rob_2, v_1 the start vertex, and $v_{q-1}v_q$ being the original entry edge of rob_2 to v_q for $q = 2, \ldots, p$.

As per our assumption, rob_1 has already returned to v_1 and found no further eligible edge; that is, the edge v_1v_2 was not eligible for rob_1, otherwise it would have followed that edge. Observe that there are two possible reasons why an edge becomes ineligible; 1) the edge is finished, with a robot leaving and returning through that edge, or 2) the edge is the original entry edge of a robot to that vertex. Note that no edge can be the original entry edge in both directions, since it becomes ineligible in the opposite direction as soon as it is first used. Since the edges along the path $[v_1, v_2, \ldots, v_p]$ are original entry edges of the robot rob_2, they cannot be original entry edges in the opposite direction. Thus, every edge along this path is either finished or eligible. Let $v_{i-1}v_i$ be the last edge on the path $[v_1, \ldots, v_p]$ that is finished, and let rob_3 be the robot that finished this edge. Let us consider the time step (t_{r3}) when rob_3 finished this edge. Since rob_2 used the $v_{i-1}v_i$ edge before it was finished, that edge

is somewhere on the return path of rob_2 at t_{r3}. If rob_2 is not at the same vertex as rob_3, then there is an eligible edge on the return path of rob_2 from v_i in the direction of rob_2. Thus, rob_3 would have followed that edge instead of returning by $v_i v_{i-1}$. As such, rob_2 and rob_3 must be at the same vertex at t_{r3}; they both find no eligible edge to go further down; and as a result they return together.

The same argument applies to any previous edge along the path $[v_1, v_2, \ldots, v_p]$. At the time immediately before rob_2 and rob_3 return together, the edge $v_{i-1}v_i$ was still eligible. However, none of the earlier edges along that path can be finished, since for each vertex there is still one eligible edge available. Consequently, rob_1 at the start vertex has still one eligible edge available, thus giving a contradiction to our initial assumption. □

Let us now state the main result concerning the exploration time of the MR-DFS algorithm on general graphs.

Theorem 4.1 *The algorithm MR-DFS explores any connected graph with m edges, traversing each edge, in at most $2m$ steps.*

Proof: The proof of the theorem is a consequence of the previous lemmas. In particular, according to lemma 4.1 a robot uses each edge at most once in each direction. Therefore, in the worst-case scenario, all the robots are going to traverse $2m$ edges. Furthermore, according to lemma 4.2 all the robots finish the exploration at the same time. At this point, since at each step only one edge can be traversed, the number of edges that each robot can traverse is at most $2m$. □

Remark 4.1 *An important consequence of Theorem 4.1 is that the MR-DFS algorithm explores any graph* completely*, and is* never worse *than classical single-robot DFS.*

4.3.3 Analysis on Trees

The main objective of this section is to obtain an upper bound for the complete exploration time on trees. To this end, we present two different results for the upper bound; the first one is an approximation to a working upper bound, and the second one is an explicit closed-form expression for this upper bound. Let us begin by stating that the MR-DFS algorithm is generally much better on trees than single-robot DFS.

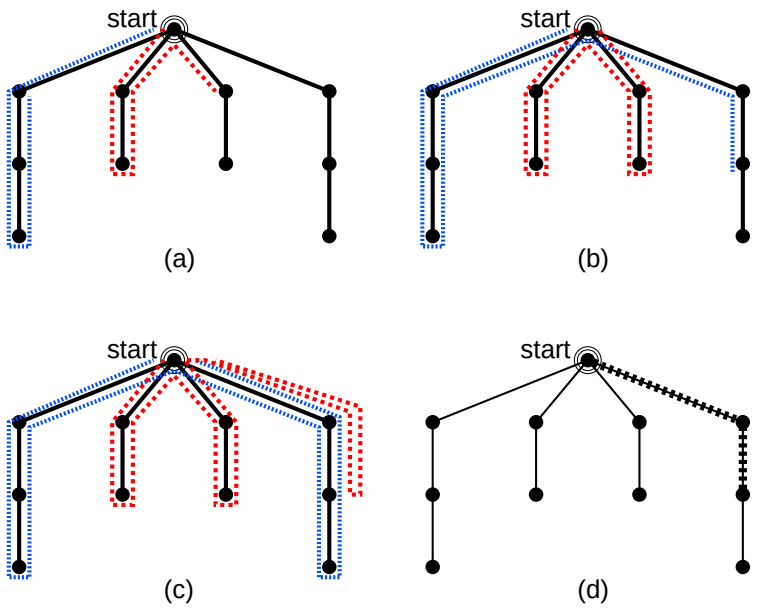


Figure 4.2: Path of two robots after a) five, b) eight, and c) twelve steps on a tree of degree 4. d) Shows the edges traversed by both robots

Fig. 4.2 shows two robots exploring a tree of degree 4, showing the state after 5, 8, and 12 steps, and the edges used by both robots. Again, the dotted blue line represents the path of robot rob_1, and the dashed red line the path of robot rob_2. At the beginning, each robot enters a branch that has not been used before. Only the last branch is entered by both robots. The robot that entered the last branch second (rob_2) meets after two steps the returning robot (rob_1), that entered the branch first,

and they both return together to the starting vertex.

The fundamental property of the MR-DFS algorithm on trees is the decreasing branching property as described in the following lemma. To this end, let us first define an incoming edge of a vertex as the edge in the direction of the root, and all other edges as outgoing edges.

Lemma 4.3 *The edges used by several robots form a subtree. If a vertex v with d outgoing edges is visited by j robots, then among last j outgoing edges, there is at most one edge that is taken by all j robots, and at most $(i+1)$ edges that are taken by at least $(j-i)$ robots, for $i = 0, \ldots, (j-1)$.*

Proof: To prove this lemma, we consider a vertex v that has d outgoing edges and is entered by j robots. Each robot that enters this vertex chooses an outgoing edge, explores a subtree, returns to the vertex and chooses another edge. This process is repeated until it finds no further edges left. Each time it returns from an edge, that edge becomes finished and unavailable for all the other robots that have not already used it. Let us number the outgoing edges $m_1, \ldots, m_d$ in the sequence in which the robots return to vertex v. Thus, the first robot to return will block m_1 for all those robots that have not already entered it by the time m_1 is blocked. Initially one robot will be assigned to enter one of the d edges. Additional robots will be allowed to enter an edge only until all other edges have been used by at least one robot. As such, the following holds:

- m_1 will be used by only one robot if $d \geq j$;
- m_1 will be used by at most $(j-d+1)$ robots if $j > d$.

The first statement holds because at the beginning the first j edges will be used by only one robot, leaving unused the other $(d-j)$ edges. In order for any other robot

to enter m_1, all unused edges must be used by at least one robot. But this will only happen if robots return to v. By definition, the first robot to do it corresponds to the one exploring m_1. Since this will block m_1, then this edges is used only by one robot. The second statement holds because the first d robots will be assigned to one of the available outgoing edges in v. The other $(j - d)$ robots could be distributed among all the edges. In the worst-case, all $(j - d)$ robots could be assigned to m_1. As such, this edges could be used by at most $(j - d + 1)$ robots.

In the same way, for $1 \leq a \leq d$, the edge m_a is blocked for all robots that have not entered it at the time a robot on m_a returns to v. Again, additional robots will enter m_a only if all d edges have been used by at least one robot. Out of the $(j - 1)$ available robots to enter m_a, at least $(d - a)$ robots will be busy in other branches (the ones from m_{a+1} to m_d). Thus, m_a will be used by at most $(j - 1) - (d - a)$ additional robots. As such, we have that

- m_a will be used by only one robot if $j \leq d - (a - 1)$, and
- m_a will be used by at most $(j - a + d)$ robots if $j > d - (a - 1)$.

The second statement holds because if more than one robot enters m_a, then, for sure, $0 < (j - 1) - (d - a)$ which translates in that $d - (a - 1) < j$. Also, the maximum number of robots that can enter m_a is $(j - 1) - (d - a) + 1 = j - a + d$. In any other situation, m_a will be used by only one robot; that is, when $0 \geq (j - 1) - (d - a)$. This results on that $d - (a - 1) \geq j$, corroborating the first statement.

Therefore, we have two cases: firstly, when $j < d$, some edges will be explored by only one robot. After arriving at v, the first j edges will be used by one robot, leaving $(d - j)$ edges empty. Any robot that returns will go into one of these empty edges. Consequently, the first $(d - j)$ robots to return from their edges will go into

the empty ones. Thus, it is guaranteed that at least the first $(d-j)$ edges will be used by only one robot. Next, let us consider the last j branches that at some point will have only one robot in them; only the last edge m_d can be explored by all j robots (once all other $(j-1)$ last branches have been finished); only one edge (m_{d-1}) can be explored by $(j-1)$ robots (the other robot is in m_d); only one edge (m_{d-2}) can be explored by $(j-2)$ robots (the other robots are in m_{d-1} and m_d); and so on. In summary, when $j < d$, at least the first $(d-j)$ edges will be explored by only one robot and the last j edges will be explored by at most $[1, 2, \ldots, j-1, j]$ robots.

Secondly, when $j \geq d$, some edges will, for sure, be explored by more than one robot. At the beginning, each of the first d robots will be assigned to explore one edge. The other $(j-d)$ robots would be assigned to any edge. In the worst-case, let us assume that all $(j-d)$ robots are assigned to the first edge to be finished (m_1). Once they are back at v, all $(j-d+1)$ robots enter the second edge to be finished (m_2), and so on, until the last edge m_d is entered by all j robots. As a result, when $j \geq d$, the d outgoing edges would be used by at most $[j-d+1, j-d+2, \ldots, j-1, j]$ robots.

As can be observed, in the worst-case in either case, at most one edge (m_d) will be used by all j robots; at most 2 edges (m_{d-1} and m_d) will be used by at least $(j-1)$ robots (all other edges will be used by $(j-2)$ robots or less); and so on, until finally we can say that at most the last j edges will be used by at least one robot. This completes the proof of the Lemma. □

Fig. 4.3 illustrates the worst-case situation of Lemma 4.3 for $d = j = 4$.

Let us now introduce the concept of *excess multiplicity* $\mu(m_i)$ of an edge m_i as the number of robots that use that edge in addition to the first one to do it. By Lemma 4.1, each robot uses each edge at most twice, going out and returning; therefore, for

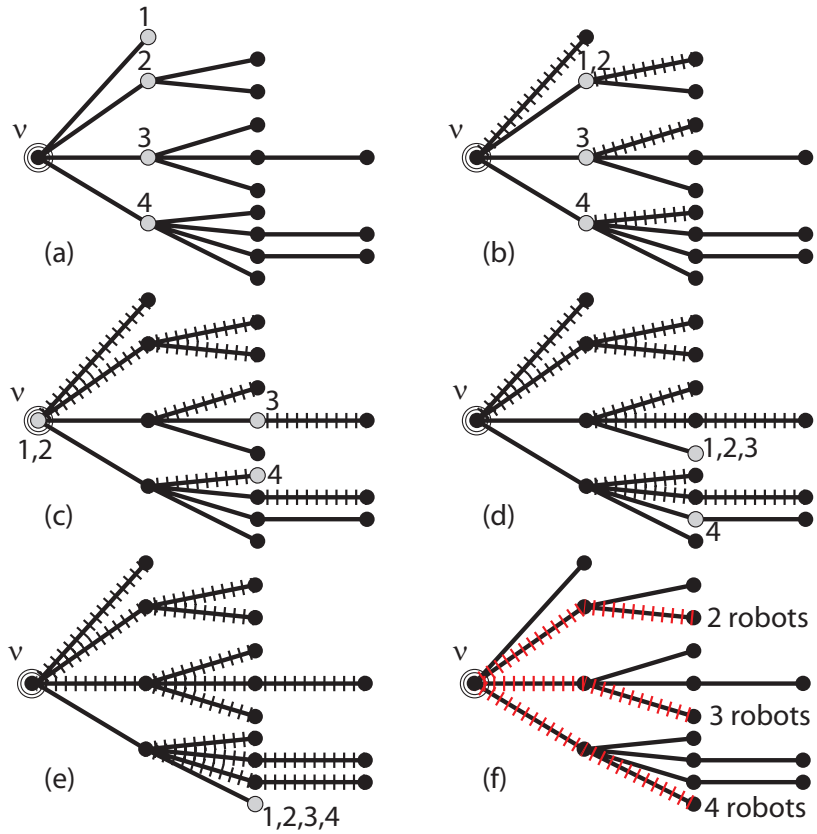


Figure 4.3: Decreasing branching property: worst-case scenario. Path of the robots after a) one step, b) three steps, c) six steps, d) eight steps, and e) twelve steps. In f) the subtree of edges with multiplicity and the number of robots that used those edges is shown

each edge m_i we have $0 \leq \mu(m_i) \leq k-1$, and the edge is used exactly $2 + 2\,\mu(m_i)$ times. Fig. 4.4 shows the multiplicity of a subtree with three outgoing edges being explored by four robots. The *excess multiplicity* plays a key role to define an upper bound for the exploration time of the MR-DFS algorithm as described by the following lemma.

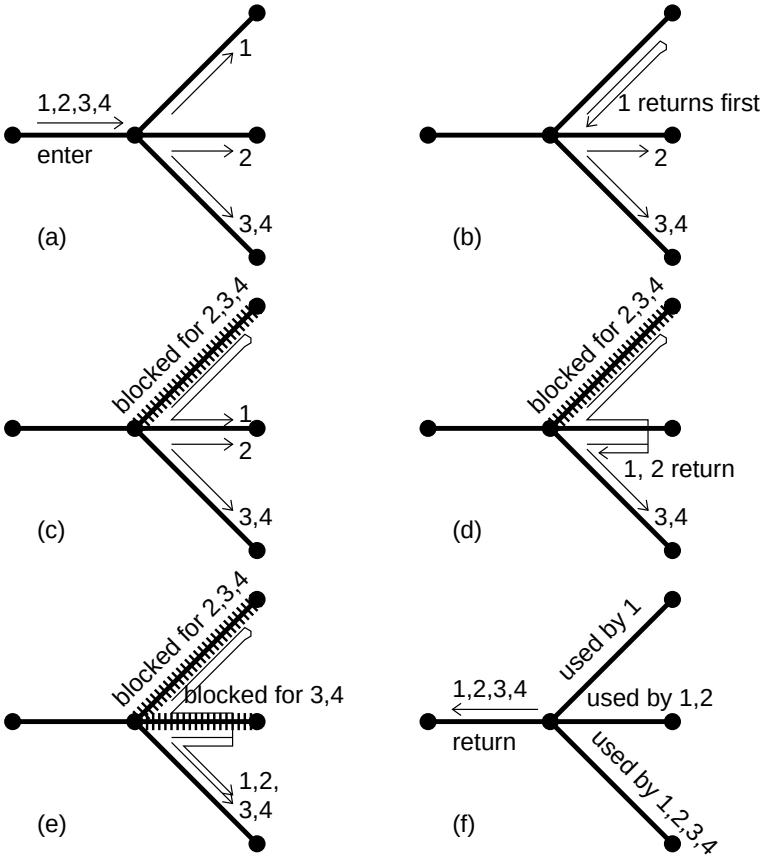


Figure 4.4: Multiplicity of a subtree with three edges being explored by four robots

Lemma 4.4 *A tree with m edges is completely explored with k robots, using the MR-*

DFS algorithm, in time

$$t_c = \frac{1}{k}\left(2m + 2\sum_{m_i}\mu(m_i)\right). \tag{4.1}$$

Proof: To obtain the bound on the total exploration time, we just add up the work done by each robot, and divide by k: since all the robots finish at the same time, we just count the total number of edges walked by the robots when they finish. Each edge was taken at least once in each direction, plus $2\sum_{m_i}\mu(m_i)$ additional edges, taken by several robots (multiplicity). □

Lemma 4.5 *The total* excess multiplicity *of all edges that are used multiple times by the robots satisfies the following recursion:*

$$\begin{aligned} f(k, D_r) \;\leq\; & f(k, D_r - 1) + f(k-1, D_r - 1) + \\ & f(k-2, D_r - 1) + \cdots + f(2, D_r - 1), \end{aligned} \tag{4.2}$$

with boundary conditions $f(2, D_r) = D_r$ *and* $f(k, 1) = (k-1) + (k-2) + \ldots + 1 = \frac{1}{2}k(k-1)$.

Proof: Let us start by recalling that the multiplicity of an edge being traversed by j robots is $(j-1)$. As such, we are interested in finding how many edges are traversed by more than one robot and how many times this happens. Let us restate the problem as follow: what are the different combinations of paths traversed by k robots (in a tree of root diameter D_r) such that those paths overlap the paths of other robots (note that not all the robots overlap the paths of other robots). Define the *dominant branch* as the branch of length D_r that specifies the diameter of the root.

Let us consider the k^{th} robot. The path of this robot may or may not overlap

other paths. If it does not overlap, the problem is reduced to find the overlapping due to the other $(k-1)$ robots. That is: $f(k-1, r)$. If, on the other hand, the path of the k^{th} robot indeed overlap other paths, this will happen for sure in at least one edge. Since we are looking for the upper bound, we are analyzing the worst-case scenario which for a tree with root diameter D_r corresponds to all robots traversing at least the first edge of the dominant branch. Thus, all possible combination of paths will consider that first edge, and as such we can leave that edge out of consideration and reduce the problem to find the overlapping due to k robots in a tree of root diameter $(D_r - 1)$ (because we already know that the first edge of the dominant branch will always have overlapping): $f(k, D_r - 1)$. Hence, the total multiplicity can be defined as the multiplicity when the k^{th} robot does not overlap plus the multiplicity when the k^{th} robot does overlap. That is

$$f(k, D_r) = f(k-1, D_r) + f(k, D_r - 1). \tag{4.3}$$

The same analysis can be repeated for scenarios with less robots:

$$\begin{aligned}
f(k-1, D_r) &= f(k-2, D_r) + f(k-1, D_r - 1),\\
f(k-2, D_r) &= f(k-3, D_r) + f(k-2, D_r - 1),\\
f(k-3, D_r) &= f(k-4, D_r) + f(k-3, D_r - 1),\\
&\vdots\\
f(3, D_r) &= f(2, D_r) + f(3, D_r - 1),\\
f(2, D_r) &= f(1, D_r) + f(2, D_r - 1),\\
f(1, D_r) &= 0.
\end{aligned}$$

The fact that $f(1, D_r) = 0$ is self-evident since with only one robot there is not chance that overlapping will occur. By substituting the previous results into (4.3) we obtain the following recursion:

$$f(k, D_r) \leq f(k, D_r - 1) + f(k - 1, D_r - 1) + f(k - 2, D_r - 1) + \cdots + f(2, D_r - 1),$$

where the inequality is the result of knowing that not all the robots overlap the other paths.

The boundary conditions for this recursion can be obtained when considering the worst-case for the following scenarios:

- When $k = 2$, the worst-case scenario will occur when the tree is a long path of length D_r and both robots must travel that distance. That is

$$f(2, D_r) = D_r. \tag{4.4}$$

- When $D_r = 1$ the worst-case scenario will occur when every branch at starting vertex v has an increasing multiplicity as shown in figure 4.5. In such a case, the total multiplicity can be expressed as

$$f(k, 1) = (k - 1) + (k - 2) + \ldots + 2 + 1 = \frac{1}{2}k(k - 1). \tag{4.5}$$

□

At this point we can introduce the first result on the upper bound for the exploration time on trees.

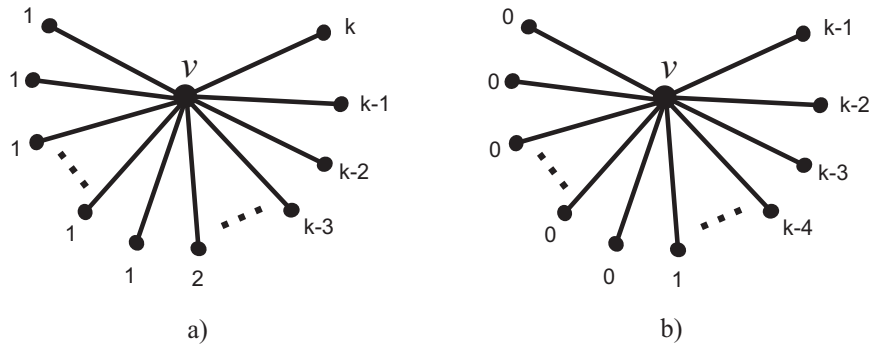


Figure 4.5: Worst-case scenario for multiplicity on a tree rooted at v with $D_r = 1$. The numbers indicate in a) the number of robots that traversed each edge, and in b) the multiplicity of each edge

Theorem 4.2 *The algorithm explores a tree with m edges and root diameter D_r using k robots, in time at most $\frac{2m}{k} + O(D_r^{k-1})$.*

Proof: The proof of the theorem is derived from solving the recursion for the total *excess multiplicity* defined in Lemma 4.5. The recursion can be solved by using generating functions. In general, a generating function for a given sequence a_0, a_1, a_2, ..., a_n is defined to be (Anderson, 1989):

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_3 x^3 + a_2 x^2 + a_1 x + a_0, \tag{4.6}$$

where the coefficients of x^0, x^1, x^2, ..., x^n in $f(x)$ are precisely the terms a_0, a_1, a_2, ..., a_n of the sequence. For the recursion in (4.2), we need to obtain its generating function by using the boundary conditions:

- For the boundary condition in (4.5), its generating function can be expressed

as

$$\begin{aligned} f(k,1) &= (k-1)(1)^{k-1} + (k-2)(1)^{k-2} + \cdots + 3(1)^3 + 2(1)^2 + 1(1) + 0 \\ &= (k-1) + (k-2) + \cdots + 3 + 2 + 1. \end{aligned}$$

- For the boundary condition in (4.4), its generating function can be written as

$$\begin{aligned} f(2,D_r) &= (2-1)(D_r)^{2-1} + (2-2)(D_r)^{2-2} \\ &= D_r. \end{aligned}$$

In general, we can expressed the recursion (4.2) as

$$\begin{aligned} f(k,D_r) &\leq (k-1)(D_r)^{k-1} + (k-2)(D_r)^{k-2} + \cdots + 3(D_r)^3 + 2(D_r)^2 + 1(D_r) + 0 \\ &\leq (k-1)D_r^{k-1} + (k-2)D_r^{k-2} + \cdots + 3D_r^3 + 2D_r^2 + D_r, \end{aligned}$$

which is a polynomial of degree $(k-1)$ in D_r. As a result, the function $f(k,D_r)$, defined by the recursion (4.2) becomes

$$f(k,D_r) \leq O(D_r^{k-1}).$$

Finally, as per Lemma 4.4, we have that

$$t_c = \frac{2m}{k} + \frac{2f(k,D_r)}{k} \leq \frac{2m}{k} + \frac{2O(D_r^{k-1})}{k},$$

that yields

$$t_c \leq \frac{2m}{k} + O(D_r^{k-1}).$$

□

The next step is to look for a closed form solution to the upper bound of the exploration time on trees. To this end, we redefine the function $f(k, D_r)$ in a way that will allow us later to obtain the total *excess multiplicity*.

Lemma 4.6 *The function $f(k, D_r)$ defined by the recursion*

$$f(k, D_r) = \binom{k}{2} + \sum_{i=2}^{k} f(i, D_r - 1), \tag{4.7}$$

with initial condition $f(k, 1) = \binom{k}{2}$ can be solved as

$$f(k, D_r) = \binom{k + D_r}{k - 1} - k. \tag{4.8}$$

Proof: To solve the recursion we rely on the following tools (Chen & Koh, 1992):

- Chu Shih-Chieh's identity,

$$\binom{a}{a} + \binom{a+1}{a} + \cdots + \binom{b}{a} = \binom{b+1}{a+1}, \tag{4.9}$$

  for all a, $b \in N$ with $a \geq b$;

- the combinatorial identities,

$$\binom{a}{0} = \binom{a}{a} = 1, \tag{4.10}$$

$$\binom{a}{b} = \binom{a}{a-b}, \tag{4.11}$$

  for a, $b \in Z$, with $0 \leq b \leq a$; and

- the factorial equation,

$$\binom{a}{b} = \frac{a!}{b!(a-b)!} \quad \text{for} \quad 0 \leq b \leq a. \tag{4.12}$$

We begin the proof by considering the case for $D_r = 2$. Using the recursion (4.7), we have that

$$\begin{aligned} f(k,2) &= \binom{k}{2} + \sum_{i=2}^{k} f(i,1) \\ &= \binom{k}{2} + \sum_{i=2}^{k} \binom{i}{2} \\ &= \binom{k}{2} + \binom{2}{2} + \binom{3}{2} + \ldots + \binom{k}{2}. \end{aligned}$$

Applying Chu Shih-Chieh's identity, (4.9), we get

$$f(k,2) = \binom{k}{2} + \binom{k+1}{3}.$$

Repeating this process for $D_r = 3$, we obtain

$$\begin{aligned}
f(k,3) &= \binom{k}{2} + \sum_{i=2}^{k} f(i,2) \\
&= \binom{k}{2} + \sum_{i=2}^{k} \left(\binom{i}{2} + \binom{i+1}{3} \right) \\
&= \binom{k}{2} + \sum_{i=2}^{k} \binom{i}{2} + \sum_{i=2}^{k} \binom{i+1}{3} \\
&= \binom{k}{2} + \sum_{i=2}^{k} \binom{i}{2} + \sum_{i=3}^{k+1} \binom{i}{3} \\
&= \binom{k}{2} + \binom{k+1}{3} + \binom{3}{3} + \binom{4}{3} + \ldots + \binom{k+1}{3} \\
&= \binom{k}{2} + \binom{k+1}{3} + \binom{k+2}{4}.
\end{aligned}$$

Thus, for any D_r, we prove that

$$f(k, D_r) = \binom{k}{2} + \binom{k+1}{3} + \binom{k+2}{4} + \cdots + \binom{k+D_r-1}{D_r+1}. \tag{4.13}$$

Observe that (4.13) can be written back in recursion form as follows:

$$\begin{aligned}
f(k, D_r) &= \binom{k}{2} + \binom{k+1}{3} + \binom{k+2}{4} + \cdots + \binom{k+D_r-1}{D_r+1} \\
&= \binom{k}{2} + \sum_{j=2}^{D_r} \binom{(k+1)+(j-2)}{j+1} \\
&= \binom{k}{2} + \sum_{j=0}^{D_r-2} \binom{(k+1)+(j-2)+2}{(j+1)+2} \\
&= \binom{k}{2} + \sum_{j=0}^{D_r-2} \binom{k+j+1}{j+3} \\
&= \sum_{j=0}^{D_r-1} \binom{(k+j+1)-1}{(j+3)-1} \\
&= \sum_{j=0}^{D_r-1} \binom{k+j}{j+2}.
\end{aligned}$$

We reduce (4.13) by using (4.11)

$$\begin{aligned}
f(k, D_r) &= \binom{k}{2} + \binom{k+1}{3} + \cdots + \binom{k+D_r-1}{D_r+1} \\
&= \binom{k}{k-2} + \binom{k+1}{(k+1)-3} + \cdots + \binom{k+D_r-1}{(k+D_r-1)-(D_r+1)} \\
&= \binom{k}{k-2} + \binom{k+1}{k-2} + \cdots + \binom{k+D_r-1}{k-2}.
\end{aligned}$$

Now, let us define $f'(k, D_r)$ as

$$f'(k, D_r) = \binom{k-2}{k-2} + \binom{k-1}{k-2} + \binom{k}{k-2} + \binom{k+1}{k-2} + \cdots + \binom{k+D_r-1}{k-2}. \quad (4.14)$$

Using (4.9), and considering that $b = k + D_r - 1$ and $a = k - 2$, we can rewrite (4.14)

as

$$f'(k, D_r) = \binom{(k + D_r - 1) + 1}{(k - 2) + 1} = \binom{k + D_r}{k - 1}. \tag{4.15}$$

Using this result and the identity on (4.10), we can rewrite $f(k, D_r)$ as

$$\begin{aligned} f(k, D_r) &= f'(k, D_r) - \binom{k-1}{k-2} - \binom{k-2}{k-2} \\ &= \binom{k + D_r}{k - 1} - \binom{k-1}{k-2} - \binom{k-2}{k-2} \\ &= \binom{k + D_r}{k - 1} - \binom{k-1}{k-2} - 1. \end{aligned}$$

Finally, using the factorial equation in (4.12), we obtain

$$\begin{aligned} f(k, D_r) &= \binom{k + D_r}{k - 1} - \frac{(k-1)!}{(k-2)!\,((k-1)-(k-2))!} - 1 \\ &= \binom{k + D_r}{k - 1} - \frac{(k-1)(k-2)!}{(k-2)!(1)!} - 1 \\ &= \binom{k + D_r}{k - 1} - (k-1) - 1 \\ &= \binom{k + D_r}{k - 1} - k. \end{aligned}$$

□

Let us now state the closed-form expression for the upper bound of the exploration time of the MR-DFS algorithm, which bounds tighter the exploration time on trees.

Theorem 4.3 *A tree with m edges and root diameter D_r can be explored by k robots using the MR-DFS algorithm in time at most*

$$\min\left(2m, \frac{2m}{k} + \frac{2}{k}\binom{k + D_r}{k - 1}\right) < \frac{2m}{k} + \left(1 + \frac{k}{D_r}\right)^{k-1} \frac{2}{k!} D_r^{k-1}. \tag{4.16}$$

Proof: The proof comes from the observation that the maximum total *excess multiplicity* of all the edges that were used multiple times by the robots is the sum of the excess multiplicities of the subtrees entered from the root, plus the excess multiplicities on the edges from the root to those subtrees. In a tree with root diameter D_r, each subtree entered from the root has root diameter at most $(D_r - 1)$, and by Lemma 4.3 there is at most one subtree entered by all k robots, at most two subtrees entered by $(k-1)$ or k robots, etc., and at most $(k-1)$ subtrees are entered by two or more robots. All other subtrees entered from the root are entered only by one robot; therefore, they do not contribute to the total *excess multiplicity* of the tree. Thus, the maximum total *excess multiplicity* $g(k, D_r)$, as a function of the number of robots k and the root diameter D_r, satisfies the recursion

$$\begin{aligned} g(k, D_r) &\leq \binom{k}{2} + g(k, D_r - 1) + g(k-1, D_r - 1) + \\ &\quad g(k-2, D_r - 1) + \cdots + g(2, D_r - 1) \\ &\leq \binom{k}{2} + \sum_{i=2}^{k} g(i, D_r - 1). \end{aligned}$$

with boundary condition $g(k, 1) = (k-1) + (k-2) + \ldots + 1 = \binom{k}{2}$.

Note that this is the same recursion as in (4.7). From Lemma 4.4, we have that

$$t_c = \frac{1}{k}\left(2m + 2\sum_{m_i} \mu(m_i)\right) \leq \frac{1}{k}\left(2m + 2g(k, D_r)\right).$$

Using Lemma 4.6, we can rewrite t_c as

$$t_c \leq \frac{1}{k}\left(2m + 2\left(\binom{k + D_r}{k-1} - k\right)\right) < \frac{1}{k}\left(2m + 2\left(\binom{k + D_r}{k-1}\right)\right).$$

As such,

$$t_c < \frac{2m}{k} + \frac{2}{k}\binom{k+D_r}{k-1}.$$

Solving the binomial coefficient term $\binom{k+D_r}{k-1}$ with the factorial equation on (4.12),7

$$\begin{aligned}
t_c &< \frac{2m}{k} + \frac{2}{k}\left(\frac{(k+D_r)!}{(k-1)!\,((k+D_r)-(k-1))!}\right) \\
&< \frac{2m}{k} + \frac{2}{k}\left(\frac{(k+D_r)((k-1)+D_r)\ldots(2+D_r)(1+D_r)(1+(D_r-1))\ldots(2)(1)}{(k-1)!(1+D_r)!}\right) \\
&< \frac{2m}{k} + \frac{2}{k}\left(\frac{(k+D_r)((k-1)+D_r)((k-2)+D_r)\ldots(2+D_r)(1+D_r)!}{(k-1)!(1+D_r)!}\right) \\
&< \frac{2m}{k} + 2\left(\frac{(k+D_r)((k-1)+D_r)((k-2)+D_r)\ldots(2+D_r)}{k(k-1)!}\right).
\end{aligned}$$

Evaluating the numerator term, we can rewrite it as

$$\begin{aligned}
(k+D_r)((k-1)+D_r)\ldots(2+D_r) &= (k+D_r)(k+D_r-1)\ldots(k+D_r-(k-2)) \\
&< (k+D_r)(k+D_r)\ldots(k+D_r) = (k+D_r)^{k-1}.
\end{aligned}$$

As a result,

$$t_c < \frac{2m}{k} + 2\left(\frac{(k+D_r)^{k-1}}{k!}\right),$$

which, by simply factorization, becomes

$$t_c < \frac{2m}{k} + \left(\frac{k}{D_r}+1\right)^{k-1}\frac{2}{k!}D_r^{k-1}.$$

□

Remark 4.2 *Note that for a larger k and $D_r \geq k$, we have that*

$$\left(\frac{k}{D_r}+1\right)^{k-1}\frac{2}{k!}D_r^{k-1} \leq D_r^{k-1},$$

as the coefficient of D_r^{k-1} in the LHS decreases rapidly. Consequently, in this situation, the result of theorem 4.3 provides a tighter upper bound than the result of theorem 4.2.

Remark 4.3 *In its dependence on m, this is optimal and improves the $O(\frac{m}{\log k}+D_r)$ result of (Fraigniaud et al., 2006). The dependence on D_r, however, is not. This is an interesting bound for trees with many edges and small root diameter (trees with high branching factors) or for exploration with a small number of robots. As such, this bound is tight for trees with very large m when compared to D_r. The bound on the total excess multiplicity used in the proof above views it only as a function of D_r and k, and leaves m open. To obtain a further improvement along these lines in the bound would require an analysis with m as third parameter.*

Remark 4.4 *Lemma 4.3 can be improved if an equitable distribution of the robots is considered. That is, whenever j robots arrive at a vertex with d outgoing edges, robots will be distributed in such a way that each edge will have approximately $\left(\frac{j}{d}\right)$ robots and where the maximum difference in the number of robots among edges is at most one. This equitable distribution can be achieved by considering a stronger communication model where the bookkeeping devices have a more active role than the one it has been considered so far. Recall that up to this point, the analysis has considered that the bookkeeping devices behave as passive landmarks, providing the robots only with a returning path to the root.*

For two robots (i.e., $k = 2$) the following Theorem shows a type of optimality of

MR-DFS: no strategy can guarantee a better competitive ratio against an optimal explorer, who already knows the tree and always makes the best choices.

Theorem 4.4 *For two robots, all the following statements are true:*

- *The MR-DFS algorithm explores a tree with m edges and root diameter D_r in time at most $m + D_r$.*
- *This upper bound is at most $\left(\frac{3}{2}\right)$ of the optimum exploration time.*
- *No algorithm (for two robots) guarantees a factor less than $\left(\frac{3}{2}\right)$ from the optimum exploration time.*

Proof: For two robots ($k = 2$) Lemma 4.3 implies that there is at most one branch used by both robots. All other branches will be used by only one robot. As such, the subtree used by both robots does not branch; therefore, it is a path with length at most D_r. Thus, $\sum_i \mu(m_i) \leq D_r$. From Lemma 4.4, we have that

$$\begin{aligned} t_c &= \frac{1}{2}\left(2m + 2\sum_{m_i}\mu(m_i)\right) \leq \frac{1}{2}\left(2m + 2D_r\right) \\ &\leq m + D_r. \end{aligned}$$

Furthermore, as explained in Section 4.3.1, the general lower bound of the exploration time of a tree, using two robots, is $\max(\frac{2m}{2}, 2D_r) = \max(m, 2D_r)$. Then,

- if $D_r \leq \frac{1}{2}m$, then $2D_r \leq m$ and as such, $\max(m, 2D_r) = m$. Since, $m + D_r \leq m + \frac{1}{2}m = \frac{3}{2}m$, then, $t_c \leq \frac{3}{2}m$. Thus, t_c is at most $\left(\frac{3}{2}\right)$ of the optimum exploration time (m) for $D_r \leq \frac{1}{2}m$.
- if $D_r \geq \frac{1}{2}m$, then $2D_r \geq m$ and $\max(m, 2D_r) = 2D_r$. Again, $m + D_r \leq$

$2D_r + D_r = 3D_r$. This results in, $t_c \leq 3D_r = \frac{3}{2}(2D_r)$. Thus, t_c is at most $\left(\frac{3}{2}\right)$ of the optimum exploration time $(2D_r)$ for $D_r \geq \frac{1}{2}m$.

Finally, to see that no algorithm can guarantee a better approximation ratio than $\left(\frac{3}{2}\right)$ from the optimum exploration time, we use an adversarial construction. Let us consider a graph that has three branches with two of them of length t and one of length $2t$. This tree can be optimally explored by two robots in time $4t$: one robot explores the two short branches, the other robot explores the long branch. However, any algorithm finds out whether a branch is a short branch or a long branch only after a robot has reached the end of the branch. Thus, an adversary who reveals the graph as it is explored can always make the last branch to be explored a long branch; therefore, any algorithm can be forced to take exploration time at least $6t$. Thus, no algorithm for two robots gives a better competitiveness ratio than the $\left(\frac{3}{2}\right)$ achieved by the MR-DFS algorithm. □

Remark 4.5 *The adversarial construction described above is the special case of a general construction described in (Graham, 1966) and (Fraigniaud et al., 2006), which shows that with k robots, no strategy can guarantee a competitive factor better than* $2 - \left(\frac{1}{k}\right)$.

4.4 Simulation Results

4.4.1 Objectives and Methodology

We implemented the MR-DFS algorithm and perform all simulations in *Matlab*. The objective of the simulations presented in this section are 1) to test the bounds that were obtained analytically (Theorems 4.3 and 4.4) and verify whether they hold, and

2) to observe how tight they are with respect to the actual exploration time.

We run simulations applying the algorithm to randomly generated trees of increasing size using different number of robots. The size of the tree was changed by increasing the number of vertices m. Since for a given m different tree configurations are possible, as shown in Fig. 4.6, and this directly affects how the tree is explored and the time needed to do it, we have run 100 simulations for every value of m. The mean and the standard deviation were then calculated in order to see the qualities of the algorithm.

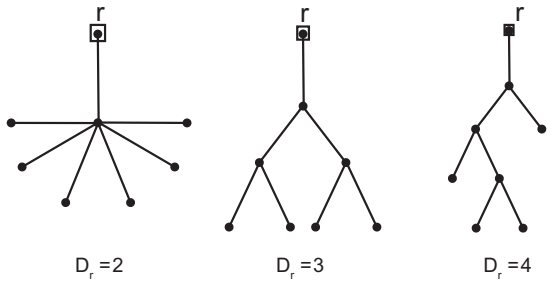
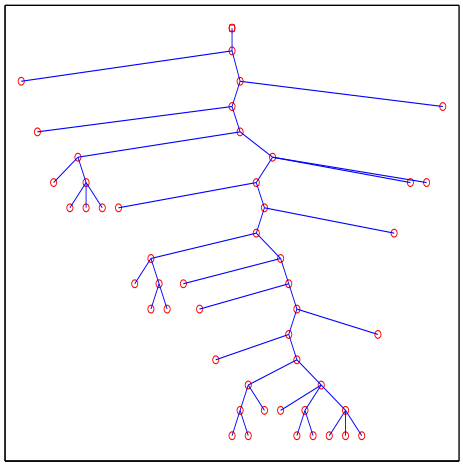


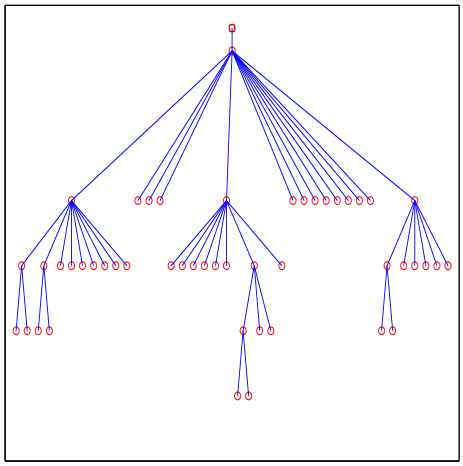
Figure 4.6: Different tree configurations for the same number of edges (m=7)

Additionally, in order to obtain meaningful information out of the simulations, we have categorized the randomly generated trees in three categories: *long*, *wide* and *symmetric trees*. Long trees refer to trees in which increasing the number of vertices is reflected in its root diameter. This kind of tree is characterized by a small number of children per vertex and a big root diameter. Wide trees are those in which increasing the number of vertices is reflected directly in the average number of children per vertex. This yields that the root diameter of wide trees is small. Finally, symmetric trees are those in which the growth of the size is reflected in both, the number of children per vertex and the root diameter. The best example for a symmetric tree is a *full N-ary tree* (called *N-ary tree* from now on) which is the one we use in our

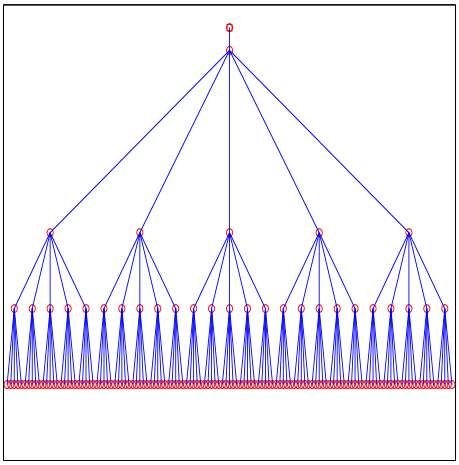
simulations. Examples of these three categories are shown in Fig. 4.7.



(a) Long tree ($D_r = 16$, $n = 50$)



(b) Wide tree ($D_r = 5$, $n = 50$)



(c) N-ary tree ($D_r = 4$, $n = 157$)

Figure 4.7: Examples of long, wide and N-ary trees

The size of the tree was increased (in long and wide trees) by increasing the number of edges. In N-ary trees, the size of the tree was defined by the number of children N that each vertex was allowed to have and by the diameter of the root.

The way the robots were distributed in a vertex is as follows: let us assume k robots arrive at a vertex v that has m_v downward unexplored edges. The k robots

will distribute themselves in the most homogeneous way possible where the maximum difference in the number of robots in every edge is equal to one. As an example consider five robots arriving at a vertex with three unexplored downward edges: two of those edges will be taken by two robots and the last edge will be taken by only one robot. The idea is to obtain the maximum parallelism in the exploration process. For long and wide trees, since the same number of edges m can produce very different configurations of trees (each one with a different exploration time), we performed 100 runs of the simulation per each value of m.

4.4.2 Results

The results of the first set of simulations are shown in Figs. 4.8 and 4.9. The plots show the upper bound defined by Theorem 4.3, the general lower bound (i.e., $\max(2m/k, 2D_r)$), and the exploration time (mean of 100 runs) due to different numbers of robots exploring the tree.

For N-ary trees, only one simulation per tree configuration was run since, due to the symmetry of the tree, the algorithm will make the robots explore the tree in the same way all the time. The plots in Fig. 4.10 show the upper bound (straight line) and the exploration time (dashed line) for this type of tree when the exploration is performed by different number of robots ranging from two to six. The results for the general lower bound were not shown in order to simplify the reading of the plots.

From the results of this set of simulations (Figs. 4.8-4.10) we can observe that the bounds of exploration, as defined in this paper, hold at all times. An interesting result is shown in Figs. 4.8 and 4.9 when using two robots: the curve of the upper bound of Theorem 4.3 matches tightly that of the actual exploration time of the algorithm. In particular in wide trees, the analysis presented in Section 4.3 produces bounds of

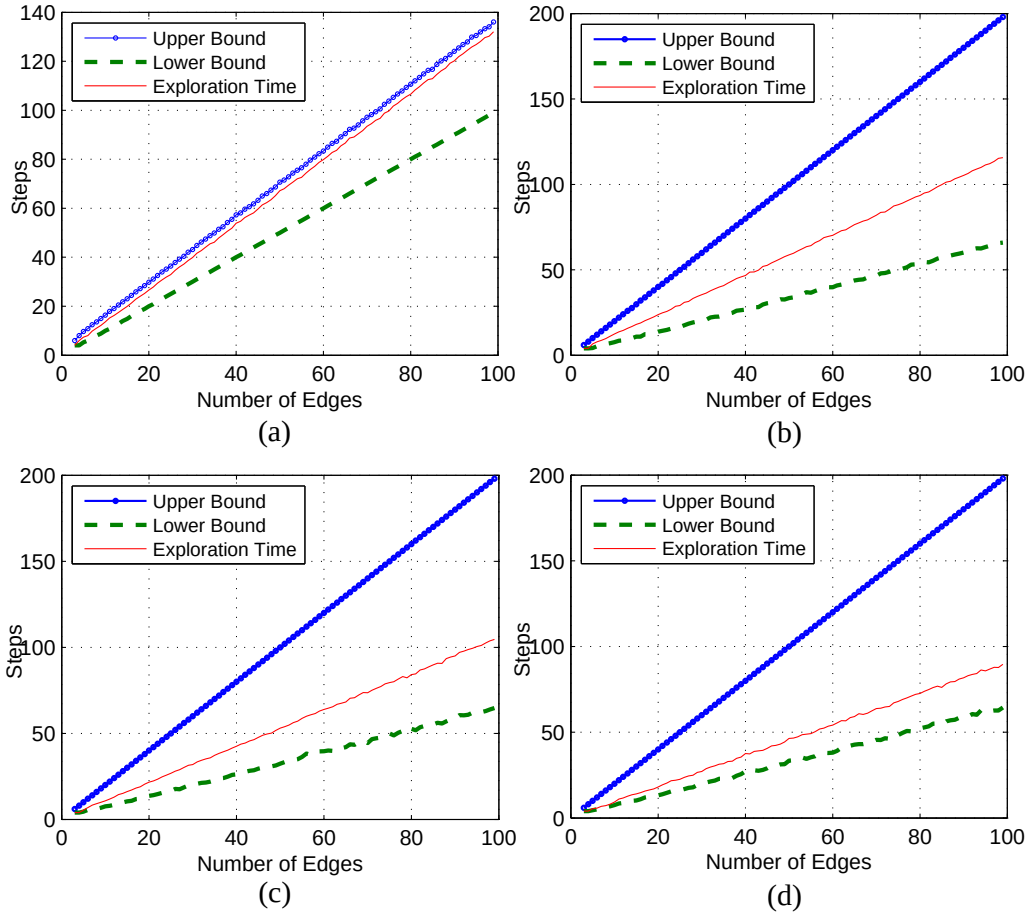


Figure 4.8: Comparison of exploration times and bounds of exploration on long trees of increasing number of vertices using a) two, b) three, c) five, and d) twenty robots.

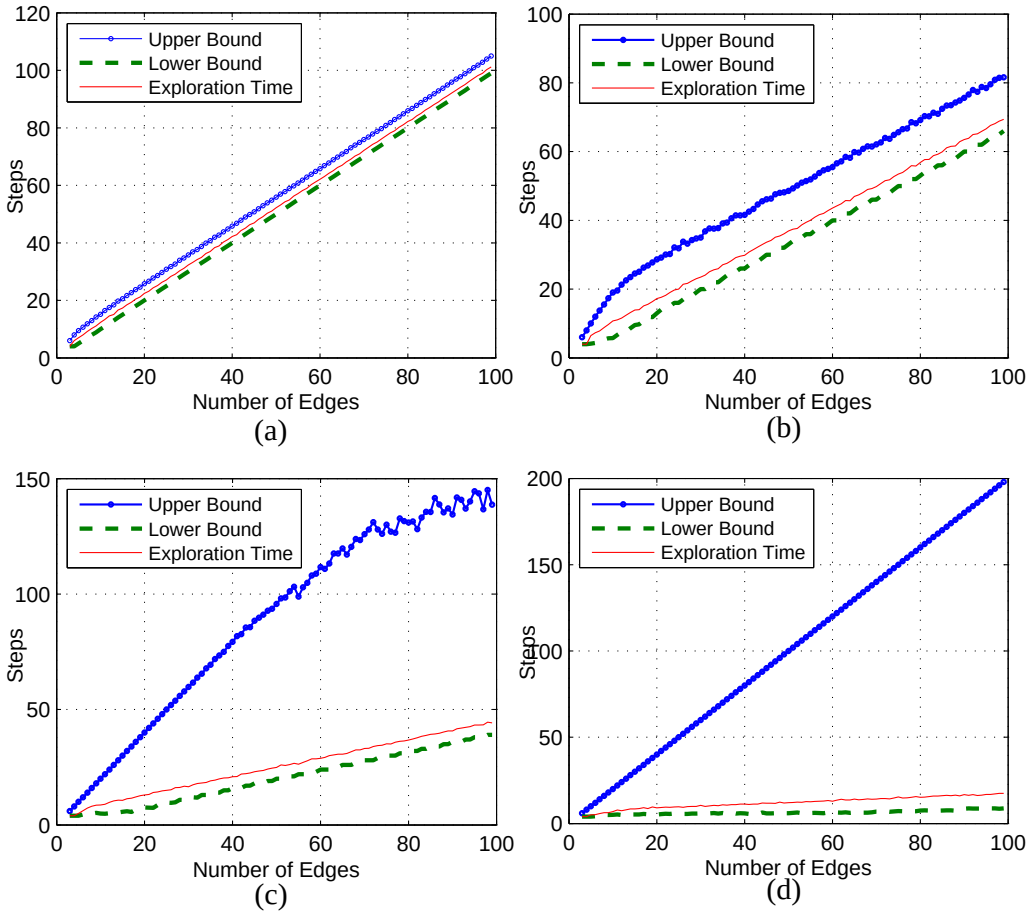


Figure 4.9: Comparison of exploration times and bounds of exploration on wide trees of increasing number of vertices using a) two, b) three, c) five, and d) twenty robots.

exploration that perfectly enclose the exploration time. Let us recall that tightness on the bounds of exploration is desired in order to perform estimations on the actual exploration time when no explicit expression for this exploration time has been found (like in this case).

From the results on wide trees (see Fig. 4.9) it is evident that our lower bound is very close to the exploration time. As such, it suggest the existence of a linear function of k, D_r and m that actually defines the exploration time, or that, at least, upper-bounds it more tightly. The results on all trees corroborate Remark 4.3, since our upper bound is indeed prevalent on trees with many edges and small root diameter (wide trees), particularly when using a small number of robots. For long trees, the upper bound is basically defined by $2m$.

The results on all trees also show that our MR-DFS algorithm is effective in reducing the exploration time when increasing the number of robots and that this exploration time is, at all times, better than the single robot DFS approach (which is a desired characteristic of any multi-robot strategy). Finally, we can observe from the simulations that when increasing the number of robots, the exploration time of the algorithm is brought down closer to the lower bound, that is, the exploration time is reduced closer to the optimal time of exploration.

Fig. 4.11 shows the behavior of the algorithm on a tree with a fixed configuration when using up to fifteen robots. The fixed configuration corresponds to a N-ary tree (N=7) and a root diameter of 5. The plot clearly shows how the exploration time is consistently reduced when more robots are included in the system, which is a topic that will be studied in more detail in chapter 6.

A second set of simulations was performed to corroborate the statement of Theorem 4.4: with two robots the upper bound of the exploration time is $(m + D_r)$. Fig.

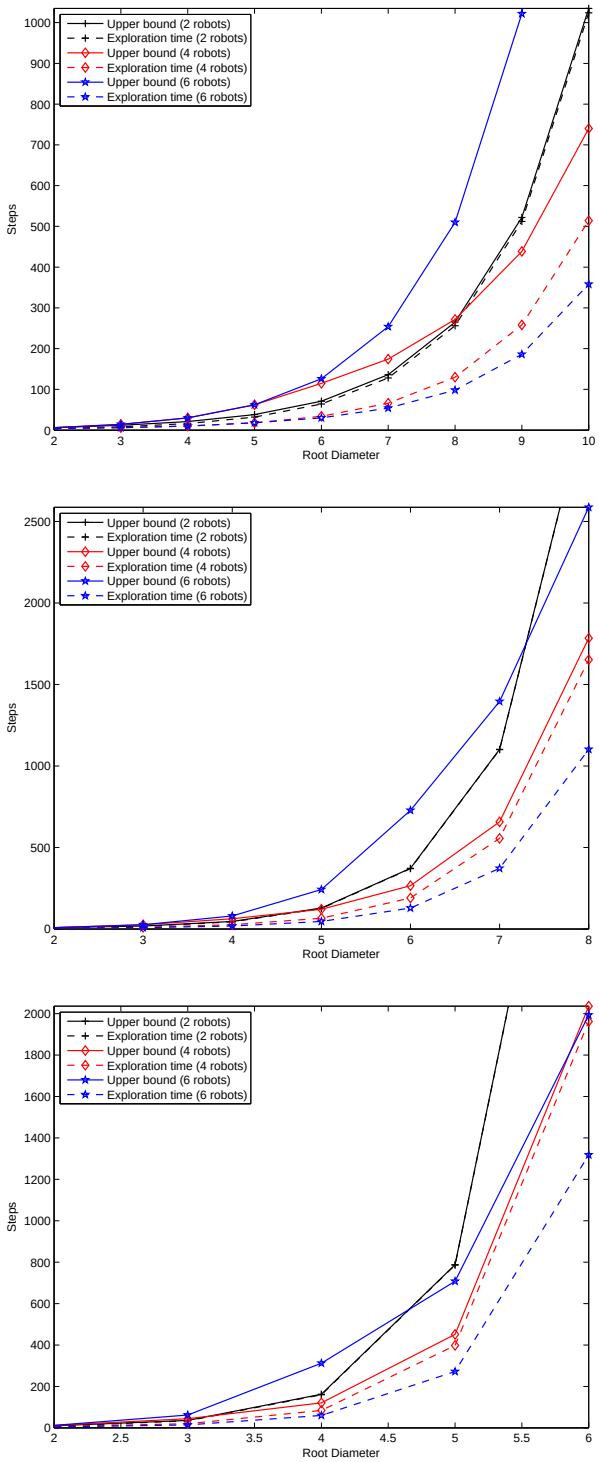


Figure 4.10: Comparison between upper bound (straight lines) and exploration time (dashed lines) on N-ary trees of increasing root diameter and different number of robots. Subfigures from top to bottom respectively show the curves for N=2, N=3 and N=5.

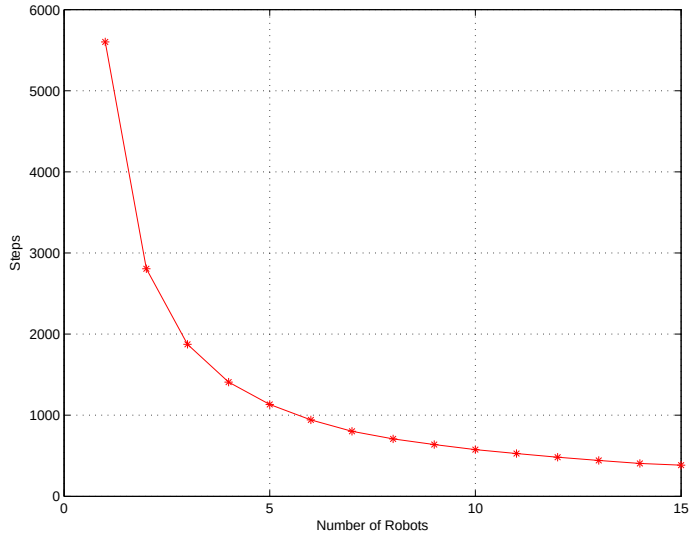


Figure 4.11: Exploration time for increasing number of robots in a tree with a fixed configuration (N-ary tree with N=7 and D_r=5).

4.12 shows the results for long and wide trees.

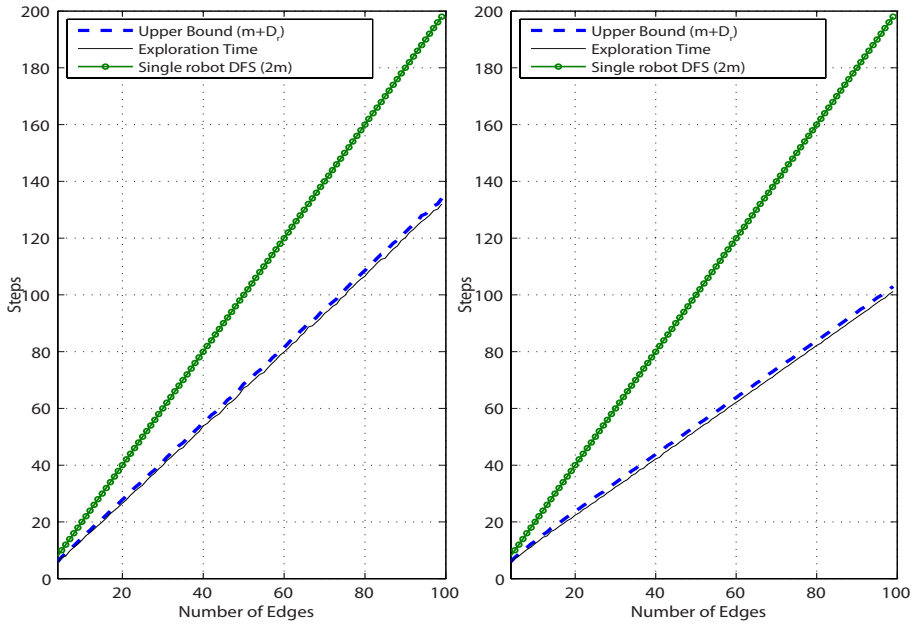


Figure 4.12: Comparison between the exploration time and the upper bound defined in Theorem 4.4 on long (left) and wide (right) trees of increasing number of edges using two robots.

The results show that the upper bound holds at all times and that, as expected, it is tight with respect to the actual exploration time. Fig. 4.12 also allows us to observe, in detail, the performance of the algorithm using two robots and how it contrasts with the result of single robot DFS: in wide trees the average reduction in the exploration time is approximately 50%, whereas in long trees the reduction averages 30%.

Chapter 5

Flooding algorithm

The second algorithm we have developed considers a coordination model that allows a vertex to be occupied by only one robot and an edge to be traversed by only one robot on each step of the exploration process. This coordination model determines that, when the robots enter the environment, they do it one at a time, resembling the way water flows through empty pipes filling them up. This resemblance derives in its name: *flooding* algorithm. The coordination model of the algorithm is achieved in a decentralized manner by the robots using a set of active landmarks that are dropped by them at explored locations.

In this section we present the details of the algorithm, perform a mathematical analysis concerning its performance on trees, prove that the algorithm is never worse than single-robot depth-first search, and perform simulations to corroborate the results of the mathematical analysis.

5.1 General Considerations

The motivation behind the *flooding* algorithm is that, due to its coordination model, it represents the completely opposite to the MR-DFS algorithm presented in chapter 4, in terms of parallelism for multi-robot systems. The *flooding* algorithm deals with a practical problem when multiple robots explore narrow passages where only one robot can pass through at each time. As such, it is expected that the *flooding* algorithm performs slower than the MR-DFS, and yet, faster than single-robot DFS. Additionally, it is expected that the *flooding* algorithm outperforms the MR-DFS algorithm in other aspects of the exploration process. Details of this will be shown in Chapter 6.

Furthermore, having less parallelism, although make it slower, allows the *flooding* algorithm to deal better with the foreseeable problem of deadlocks: lack of sufficient space may cause that the robots exploring an environment obstruct each other, preventing them to reach their destinations. In order to avoid such deadlocks, the movement of one robot cannot be performed without considering the movement of the other robots if they share a common working space. Consequently, multi-robot systems require the coordination among its elements in order to be able to explore an environment efficiently (Sheng et al., 2006). This coordination is only achievable if communication among them is considered (Hsieh et al., 2008).

Thus, the flooding algorithm relies in a communication model where robots and active landmarks are capable to exchange messages only with one-hop neighbors. That is, if a robot is present at a vertex where an active landmark has been deployed, the robot will be able to communicate with the landmark. As for the active landmarks, they will be able to communicate with other active landmarks deployed on parent

and children vertices of its current position. Indeed, this assumption reflects the operational conditions of the majority of multi-robot applications where bandwidth and range limitations are all so evident (Lucarelli & Wang, 2004; Ren & Sorensen, 2008; Zhang et al., 2011; Gasparri et al., 2011).

5.2 Flooding Algorithm

Our flooding algorithm consists of two different strategies that are executed independently by the robots. Initially, the first robot deploys a token at r. Each time a robot reaches a vertex where a token has been deployed previously, it will read the information stored in the token (number of edges in the vertex, ID of robots that have visited the vertex, the edge each robot has taken, available edges to take in the next step). A robot arriving at a leaf will drop a token, marking the place. The token will then transmit to its parent that no branches are available at this end. This will keep robots from entering the edge again and will inform that the robot currently located at the leaf is returning back to its parent in the next step.

At each step, every robot located at a vertex executes the *target edge selection strategy* shown in Algorithm 5.1 in order to obtain an edge e_r that will be its goal for moving in the next step. In one hand, if the selected target is an unused edge, the robot will move through that edge in the next step. On the other hand, if the selected target is an open edge, which implies that there exists a token in the other end of the edge (vertex v_r) of the selected edge, or if the robot expect to remain in the current vertex, the robot will send a request to the token in v_r. This token will collect the requests of all robots that expect to traverse edges converging at v_r. Then the token will send this information to each requesting robot. Upon reception

of the information, each requesting robot executes the *edge allocation strategy* shown in Algorithm 5.2. This strategy deals with any conflict created by robots trying to move to the same vertex in the same step or robots that want to remain static in the next step. As a result, only one robot will be allowed to reach the expected vertex v_r. All other request will be denied and their corresponding robots will need to acquire a new target edge by executing the *target edge selection strategy* once again. This will be done until all requested moves are accepted. Snapshots of the the running process of the robots executing the flooding algorithm can be seen in Fig. 5.1.

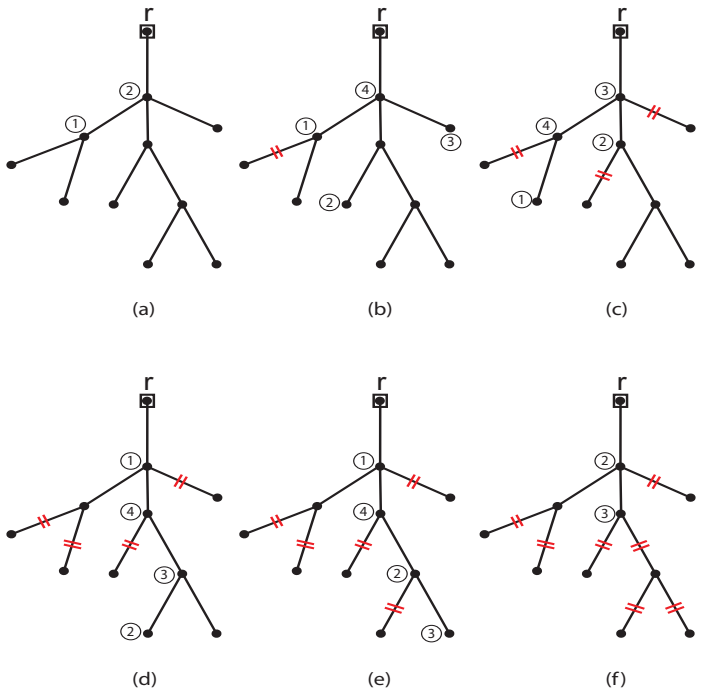


Figure 5.1: Four robots exploring an environment using the flooding algorithm. Snapshots of the position of the robots after (a) two steps, (b) four steps, (c) five steps, (d) seven steps, (e) eight steps, and (f) ten steps. Crossed lines mark that an edge has been finished. Exploration is complete after twelve steps.

In a nutshell, the algorithm gives high priority to proposed moves along unexplored edges. If a vertex has some open edges, the algorithm will send a robot to each open

Algorithm 5.1: Target edge selection strategy

input : denial of any previous request
output: target edge for next step

```
1  let R be a robot arriving at a vertex v through edge e;
2  let e_j be the ID of each edge at v;
3  let j be the total number of edges at v;
4  if there is no token present at v then
5  |  R drops one token at v;
6  end
7  if there are unused edges at v then
8  |  R selects one of the unused edges;
9  else if there are open edges at v then
10 |  if previous request has been denied then
11 |  |  R will choose a different open edge;
12 |  |  if there is at least one open edge at v different from the one selected
   |  |  previously then
13 |  |  |  R selects a different edge;
14 |  |  else
15 |  |  |  R selects to remain in the current vertex;
16 |  |  end
17 |  else
18 |  |  R selects any of the available open edges;
19 |  end
20 else if there are no unused or open edges at v then
21 |  R will choose to return to the parent vertex of v;
22 |  if previous request has been denied then
23 |  |  R selects to remain in the current vertex;
24 |  else
25 |  |  R selects to return to the parent vertex of v using its back-pointer;
26 |  end
27 end
```

Algorithm 5.2: Edge allocation strategy

input : robots request e_r
output: request decision (accepted/denied)

```
1  if more than one robot requests to reach the same vertex v_r in the next step
   then
2      if R is returning to a parent vertex then
3          if more than one robot is returning then
4              if R has the smallest ID then
5                  request of R is accepted;
6                  all other requests are denied;
7              else
8                  request of R is denied;
9              end
10         else
11             request of R is accepted;
12             all other requests are denied;
13         end
14     else
15         request of R is denied;
16     end
17 else
18     request of R is accepted;
19 end
```

edge in increasing label order (recall that every edge in a vertex is uniquely labeled with an integer number). For instance, let assume that vertex v has open edges labeled 4, 5, 6 and 7. The next robot arriving at v will be sent by the algorithm to edge 4 the next step. Any robot that arrives at v later will be sent to edge 5, and so forth, until one robot has been sent to each open edge distributing the robots as evenly as possible.

The next level of priority is for movements along open edges of robots returning to parent vertices. If two or more robots request to move along edges that lead to the same vertex v, the *edge allocation strategy* will select what robot is allowed to do so. If the request of a robot is denied, that particular robot will execute Algorithm 5.1 once again to obtain a new target edge. If no new target edge is possible, the robot will request to remain in its current position during the next step.

The algorithm exhibits three properties:

1. No edge is used more than twice by the same robot.

2. As part of the coordination process, some robots may be asked to remain static at their current positions for a single time step.

3. No robot returns to r before every edge has been traversed and every vertex has been visited at least once.

5.3 Theoretical Analysis

In this section, a theoretical analysis of the flooding algorithm is proposed. The goal is to provide a characterization of the flooding exploration time on trees. We consider a tree environment rather than a graph because the former represents a more

constrained environment where there is only one path between any pair of vertices and where deadlocks are more prone to occur.

5.3.1 Preliminaries

Let us consider that k robots are initially located at the root vertex r of an unknown, finite, undirected tree $\mathcal{T} = \{V, E\}$, formed by n vertices and m edges; and where each robot is uniquely identified with an integer number from 1 to k. The tree is initially unknown and existence of edges becomes known only when a robot sees one end of the edge. The other end becomes known only when the robot traverses the edge and arrives at the other vertex. The distance between vertices (the length of each edge) is determined by the communication range of the active landmarks. The tree is considered to be completely explored if and only if every edge is followed by at least one robot and all the robots return to the root. Define k_m as the total number of robots that actually enter the tree and help in the exploration process. Note the subtle difference between k and k_m: the former is the total number of available robots at r, whereas the latter is the number of robots that actually move inside the tree. Robots explore the tree by moving along the edges that link a pair of vertices. Robots enter the environment one at a time and only when they are allowed to do so, as it was explained in Section 5.2.

The complete exploration time can be generally bounded as follows: if we consider only one robot, the optimal complete exploration time of a tree is achieved DFS (Cormen et al., 2001) and is equal to $2m$. It is expected that with multiple robots this time will be reduced. Ideally, the exploration time can be improved in an inverse proportion to the number of robots used in the exploration (e.g., using twice the number of robots would halved the exploration time). Consequently, the complete

exploration time can be generally bounded as

$$\frac{2m}{k_m} \leq t_c \leq 2m. \tag{5.1}$$

5.3.2 Analysis on Trees

In this section, an analytical investigation of the properties of the algorithm is provided. Consider a tree $\mathcal{T}$ with m edges, rooted at vertex r and define deg_r as the degree of r.

Lemma 5.1 *A robot exploring a tree, using the flooding algorithm, uses an edge at most once in each direction.*

Proof: According to the algorithm 5.1, every time a robot returns to a vertex through the same edge the robot used to leave it, the robot will mark the branch that starts in that edge as finished and will block any robot to enter that edge again. As such, let us consider that a robot rob_i is located initially at vertex v, choosing to enter edge vu in the next time-step. Each time rob_i returns to v by a different edge than uv, it will immediately return back using the edge by which rob_i just enter v. Therefore, when rob_i returns by uv, it will mark that edge as finished and will not follow this edge a second time. Thus, edge vu is used by robot rob_i only once in each direction. □

Lemma 5.2 *Any tree can be considered as being formed by deg_r individual subtrees, all of them rooted at r, where the degree of each root is one.*

Proof: By definition, the algorithm feed one robot into each available unused/open edge at every step. At the start of the exploration, there are deg_r unused edges in r,

along with k robots. The first deg_r robots enter one of the deg_r unused edges in the first step since there is nothing that prevent them to do so. The feeding process starts at the same time on all the subtrees of r. Whenever a subtree is finished, the robots in that subtree will start returning to r, but this will not affect the movement of robots in the other subtrees of r. Thus, the exploration of a subtree of r is independent of the exploration of any other subtree, and as such every subtree can be thought of as an individual tree where its root has degree one. □

Let us define a path $S = [a_0, a_1, a_2, \ldots, a_{l-1}, a_l, a_{l+1}]$ as the path from the root to the vertex that will be explored last, where $a_0 = r$ and the last element is a leaf. Define $\mathcal{T}_s$ as the subtree of r where S is specified.

Lemma 5.3 *The complete exploration time of tree $\mathcal{T}$ equals the time needed to completely explore the subtree $\mathcal{T}_s$.*

Proof: The proof of this lemma follows directly from the claim of lemma 5.2: since each subtree of r is independent of each other, and because the subtree $\mathcal{T}_s$ is the subtree that will be completely explored last, any other subtree will be completed before $\mathcal{T}_s$. Consequently, $\mathcal{T}_s$ will determine the time in which tree $\mathcal{T}$ is completely explored. □

Due to Lemma 5.3, from now on the analysis of tree $\mathcal{T}$ will be referred as the analysis of $\mathcal{T}_s$.

Lemma 5.4 *The number of robots entering a tree is bounded by the root diameter D_r and t_e. That is,*

$$D_r \leq k_m \leq t_e.$$

Proof: Since at each step one robot enters a tree, if conditions are adequate (e.g., no returning robots blocking the movement of other robots), the algorithm will keep feeding robots until time t_e is reached. Since t_e is the time when the last unexplored edge is traversed, after t_e all robots in $\mathcal{T}$ will start returning to r. Thus, $k_m \leq t_e$. Now, consider a path of length D_r. At time t_e, there will be one robot at each vertex of the path. There are D_r edges and $(D_r + 1)$ vertices. But we should not count the root, since any robot in r is not considered as being inside the tree. Then, one robot is stationed only in the last D_r vertices. The existence of branches in the path may require additional robots to be explored. Thus, $D_r \leq k_m$. □

Lemma 5.5 *The total time to complete the exploration t_c equals the total time to finish the exploration t_e plus the number of robots k_m inside the tree at time t_e. That is,*

$$t_c = t_e + k_m.$$

Proof: One of the properties of the algorithm is that all robots that enter the tree will remain in it until the exploration is finished either by moving along open/unexplored edges, or by remaining static in a vertex. Assume that $k_m = 1$ at time t_e. This means that the tree is only a link of two vertices. It will take exactly one step to take that robot back to r. Next, assume the tree is a star with m edges. If we mark one of its leaves as the root r, there will be $(m-1)$ leaves and one internal vertex (recall that an internal vertex is one which is not a leaf). Assume that at t_e there is a robot in each leaf and in the internal vertex. That is, $k_m = m$. Since on each step an edge can be traversed only by one robot, only one robot will reach r in each step, and only one robot will be able to move from one of the leaves to the internal vertex. The other robots will have to wait their turn. The last robot to move

from a leaf to the internal vertex will have to wait that all other robots have done that. That is, it will have to wait $(m-2)$ steps. Since it takes two steps for this robot to reach the root, the last robot to exit the tree will do it in m steps after t_e. As such, $t_c = (t_e + m) = (t_e + k_m)$. □

Theorem 5.1 *Our flooding algorithm completely explores a tree with m edges and root diameter D_r in time at most $(D_r + m)$. That is,*

$$t_c \leq D_r + m. \tag{5.2}$$

Proof: In the first part of the proof we consider how the algorithm perform in small trees. Define *"minimum tree"* as a 3-edge tree with $D_r = 2$ and $m = 3$ that will always be explored by two robots in time $t_e = 3$ (Fig. 5.2). From lemma 5.5, the time for complete exploration will be: $t_c = (t_e + k_m) = 5$. Note that t_c can also be defined, for this configuration, as: $t_c = (D_r + m) = 5$. Define *"extended minimum tree"* as a tree with $D_r = 2$ and $m_e = (j+1)$ edges, where j is the total number of leaves. The extended minimum tree is also explored by at most two robots (see Fig. 5.3). In general, the exploration time of any extended minimum tree with two robots can be defined as: $t_e = m_e$ and $t_c = (t_e + k_m) = (m_e + 2) = (m_e + D_r)$.

Now, consider a tree with root diameter $D_r \geq 3$. According to the definition of an internal vertex, the tree with the minimum number of edges and root diameter $D_r \geq 3$ has the configuration shown in Fig. 5.4a. This tree configuration can be decomposed into $(D_r - 1)$ minimum trees (see Fig. 5.4b). Since each minimum tree is explored in $t'_e = m'_e$ with two robots, then the total exploration time will equal $t_e = (D_r - 1)m'_e$. However, this result is inaccurate: there are some redundant edges,

as shown in Fig. 5.4b. For each $(D_r - 1)$ minimum trees in T there are $(D_r - 2)$ redundant terms. Thus, the total exploration time can be defined as

$$t_e = (D_r - 1)m'_e - (D_r - 2). \tag{5.3}$$

The total number of edges in $\mathcal{T}$ can be obtained from the following analysis: each minimum tree has m'_e edges, and there are $(D_r - 1)$ minimum trees and $(D_r - 2)$

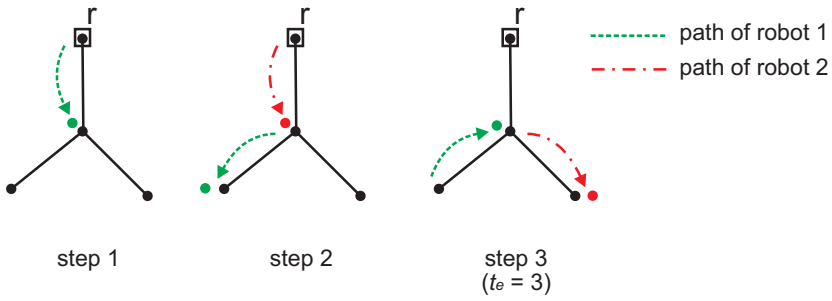


Figure 5.2: Two robots exploring the *"minimum tree"*

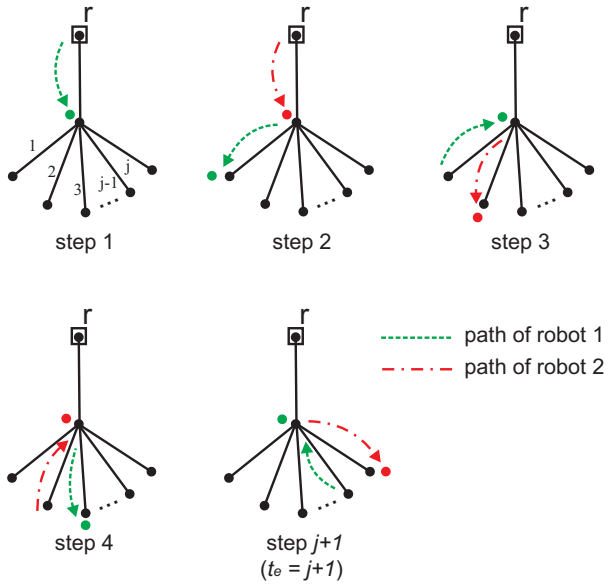


Figure 5.3: Two robots exploring an *"extended minimum tree"*

redundant edges that belong to more than one minimum tree. Thus, the total number of edges in $\mathcal{T}$ can be expressed as

$$m = (D_r - 1)m'_e - (D_r - 2). \tag{5.4}$$

Comparison of (5.3) and (5.4) leads to: $t_e = m$.

The number of robots inside the tree in Fig. 5.4a at time t_e can be obtained following a similar analysis: each minimum trees is explored by two robots. There are some robots that are considered in more than one minimum tree (see Fig. 5.4b). That is, for each pair of minimum trees, there is one redundant robot. Thus, the total number of robots at time t_e can be defined as

$$k_m = 2(D_r - 1) - (D_r - 2) = D_r.$$

Using Lemma 5.5, we can express t_c as

$$t_c = t_e + k_m = m + D_r.$$

Thus, Theorem 5.1 holds for any small tree with root diameter D_r and with the minimum number of edges in it.

In the second part of the proof, we study the behavior of the algorithm in any larger tree. To this end, we use a full N-ary tree. A full N-ary tree is one in which all leaves have the same depth (root diameter) and all internal vertices have degree N, where by definition $N \geq 2$ and $D_r \gg N$. Consider there exists a N-ary tree rooted at a_1 with root diameter h. The path S is defined in the N^{th} subtree as shown in Fig. 5.5.

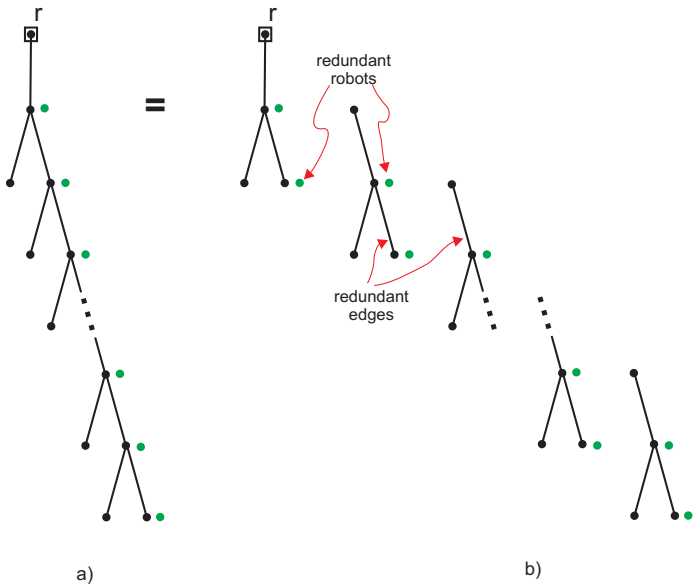


Figure 5.4: a) Tree with minimum number of internal vertices and $D_r \geq 3$ (dots represent robots located at vertices at time t_e) b) Decomposition into $(D_r - 1)$ minimum trees

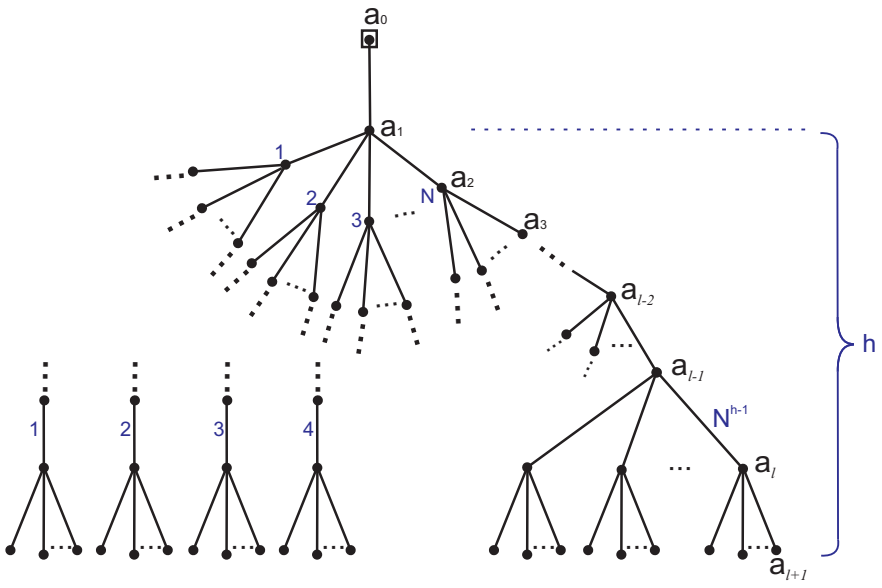


Figure 5.5: N-ary tree rooted at a_1

The total number of leaves in a N-ary tree equals N^h, and the number of internal vertices can be calculated as

$$1 + N + N^2 + \cdots + N^{h-1} = \frac{N^h - 1}{N - 1}.$$

Since it is assumed that the N-ary tree is rooted at a_1, the total number of vertices n in $\mathcal{T}$ equals

$$n = \left(\frac{N^h - 1}{N - 1}\right) + N^h + 1,$$

where the one accounts for vertex a_0. The total number of edges in $\mathcal{T}$ is

$$m = n - 1 = \left(\frac{N^h - 1}{N - 1}\right) + N^h = \frac{N^{h+1} - 1}{N - 1}. \tag{5.5}$$

Since $h = (D_r - 1)$, then (5.5) can be rewritten as

$$m = \frac{N^{D_r} - 1}{N - 1} = N^{D_r - 1} + N^{D_r - 2} + \cdots + N + 1.$$

If our flooding algorithm is used in the N-ary tree, the first subtree of a_1 will be finished first, then the second subtree and so forth. Once finished, a subtree is blocked for other robots to enter. By time t_e, it is likely that some robots have returned to a_1 and entered any other open subtree. This will block robots at r to enter the tree on every step, resulting in the total number of robots in the tree at time t_e being less than the maximum number of robots (recall Lemma 5.4). That is, $k_m < t_e$. As a

result

$$t_c = t_e + k_m < 2t_e. \tag{5.6}$$

Now we can look for t_e: if the N-ary tree has in total N^h leaves, there are N^{h-1} "extended minimum trees" in $\mathcal{T}$. Assume that all extended minimum trees can be labeled with numbers from 1 to N^{h-1}, where the latter corresponds to the one rooted at a_{l-1} that leads to the last vertex in the path S. Assume that all the N^{h-1} extended minimum trees are explored by only one robot. This will guarantee that t_c is maximized. Moreover, any extended minimum tree will be entered by one robot after the robot has traversed at least $(h-2)$ steps (i.e., the time it takes to go from a_1 to the root of the last extended minimum tree of the subtree). Finally, the first robot to enter the $N^{(h-1)th}$ extended minimum tree will do so after a robot has been sent to any other extended minimum tree. That is, the first robot will enter $N^{(h-1)th}$ extended minimum tree after $(N^{(h-1)}-1)$ steps have passed. All these times are with reference to the N-ary tree which is rooted at a_1. To refer them to a_0 we need to add one additional step. As a result, the edge connecting a_l to a_{l+1} will be traversed at time

$$t_e \;\; = \;\; N^{h-1} + 2N + h - 2. \tag{5.7}$$

Using (5.7) into (5.6), and expressing t_c in terms of D_r yields

$$t_c \;\; < \;\; 2N^{D_r-2} + 4N + 2D_r - 6. \tag{5.8}$$

On the other hand, the sum $(D_r + m)$ results in

$$\begin{aligned} D_r + m &= D_r + \left(\frac{N^{D_r} - 1}{N - 1} \right) \\ &= N^{D_r - 1} + N^{D_r - 2} + \cdots + N + 1 + D_r. \end{aligned} \tag{5.9}$$

Comparison of (5.8) and (5.9) reveals that $t_c < D_r + m$. Thus, Theorem 5.1 holds for any large tree. Finally, combining both parts of the proof yields that for any tree $t_c \leq D_r + m$. □

Corollary 5.1 *Our flooding algorithm completely explores a tree in time never worse than single-robot DFS.*

Proof: The proof follows the result of theorem 5.1, since the diameter of the root is at most equal to the total number the edges in the tree. That is, $D_r \leq m$. In fact, the only configuration in which both terms are equal is when the tree is a straight path without branches. As such,

$$D_r + m \leq 2m,$$

and as a result

$$t_c \leq 2m.$$

□

Theorem 5.2 *Our flooding algorithm completely explores a tree with m edges and root diameter D_r using k_m robots in time at least $\left(\frac{2m}{k_m+1}\right) + D_r$. That is,*

$$t_c \geq \left(\frac{2m}{k_m + 1} \right) + D_r. \tag{5.10}$$

Proof: Define the optimal finished exploration time ($t_{e_{min}}$) as the fastest time any tree can be explored using the flooding algorithm. Since t_e is minimized, then t_c is minimized as well. Using Lemma 5.5, we can define the optimal complete exploration time as

$$t_{c_{min}} = t_{e_{min}} + k_m. \tag{5.11}$$

Define Λ_i as the total number of edges traversed by robot i from the time it enters the tree (t_{0_i}) until the time it return back at the root ($t_{c_{min}}$). Note that Λ_i can be expressed as $\Lambda_i = (\Lambda'_i + \Lambda^*_i)$, where Λ'_i is the total number of edges traversed by robot i from t_{0_i} until $t_{e_{min}}$, and Λ^*_i is the total number of edges it traverses from $t_{e_{min}}$ until $t_{c_{min}}$. Then, we can define the total number of edges traversed by all robots as

$$\Lambda_{total} = \Lambda'_{total} + \Lambda^*_{total} = \sum_{i=1}^{k_m} \Lambda'_i + \sum_{i=1}^{k_m} \Lambda^*_i. \tag{5.12}$$

If all robots k_m were to move at every step until $t_{e_{min}}$ is reached, the first robot will traverse in total $t_{e_{min}}$, the second will traverse one less edge and so forth. That is,

$$\begin{aligned} t_{e_{min}} + (t_{e_{min}} - 1) + (t_{e_{min}} - 2) \cdots + 1 &= \sum_{i=1}^{k_m} \Lambda'_i \\ &= \frac{(k_m + 1) t_{e_{min}}}{2}. \end{aligned} \tag{5.13}$$

Equation (5.13) considers that the maximum number of robots enter the tree and as such provides the maximum number of edges traversed by $k_{m_{max}}$ robots. It is expected, though, that the total number of robots that perform the exploration in the optimal time is either less than $k_{m_{max}}$ or that they do not move in every step.

Thus,

$$\Lambda'_{total} \leq \frac{(k_m + 1)t_{e_{min}}}{2},$$

and, from (5.12), we obtain that

$$\Lambda_{total} \leq \left(\frac{(k_m + 1)t_{e_{min}}}{2}\right) + \Lambda^*_{total}.$$

In their return back to the root, the robots will at most traverse the same edges they traverse until $t_{e_{min}}$. Thus,

$$\begin{aligned} \Lambda'_{total} &\geq \Lambda^*_{total} \\ (k_m + 1)t_{e_{min}} &\geq \Lambda^*_{total} + \left(\frac{(k_m + 1)t_{e_{min}}}{2}\right) \\ (k_m + 1)t_{e_{min}} &\geq \Lambda_{total}. \end{aligned}$$

Note that the optimal case for multi-robot exploration will park all but one robot at the root vertex and then use DFS to explore the tree. This will produce an overall traversed distance of $2m$. As such we can bound the total number of edges traversed by all robots as follows:

$$2m \leq \Lambda_{total}. \tag{5.14}$$

Using this result, and solving for $t_{e_{min}}$ we obtain that

$$t_{e_{min}} \geq \frac{2m}{k_m + 1},$$

and from (5.11)

$$t_{c_{min}} \geq \left(\frac{2m}{k_m + 1}\right) + k_m.$$

Finally, using Lemma 5.4, we obtain

$$\begin{aligned} k_m &\geq D_r \\ \left(\frac{2m}{k_m+1}\right) + k_m &\geq \left(\frac{2m}{k_m+1}\right) + D_r \\ t_{c_{min}} &\geq \left(\frac{2m}{(k_m+1)}\right) + D_r, \end{aligned}$$

and this proves the theorem because $t_c \geq t_{c_{min}}$. □

5.4 Simulation Results

5.4.1 Objectives and Methodology

We implemented our flooding algorithm and performed all the simulation in *Matlab*. The objectives of conducting the simulations presented in this section are twofold, 1) to test the bounds that were obtained analytically in order to see if they hold, and 2) to observe how tight they are in comparison with the actual exploration time of the flooding algorithm.

We run simulations applying the algorithm to randomly generated trees of increasing size using the same number of robots. The size of the tree was changed by increasing the number of vertices m.

As it was mentioned in Section 4.4.1, different tree configurations are possible for the same value of m (see Fig. 4.6); a fact that directly affects how the tree is explored and the exploration time. As such, we run 100 simulations for every value of m, and then the mean and the standard deviation were calculated. The results shown on Section 5.4.2 correspond mainly to mean values.

Furthermore, we considered the categorization of the randomly generated trees, described in Section 4.4.1, where trees are grouped as *long*, *wide* and *symmetric trees* (see Fig. 4.7).

The categorization of the random trees is aimed to prevent that the results of the simulations vary so much that makes them useless for any analysis. That is, since the exploration time and the number of traverse edges vary greatly depending on the configuration of the tree (wide or long) and since we are running the simulations several times for each value of m, calculating the average and the standard deviation of a whole set of simulations would produce very different values and it would not be clear if such a variation is the result of the random tree generation algorithm producing widely different trees or due to the actual performance of the exploration algorithms. Thus, the categorization helps the analysis of the results, since similar trees will produce similar results, being the difference how the trees were explored.

A special case is the N-ary trees: since the number of edges increases exponentially with either the root diameter or the number of children, we increase the size of the N-ary trees by changing its root diameter only. We set the number of children per vertex fixed to three, except for the root (recall that in our analysis we consider that the root always has degree one).

The number of robots used in each run of the simulation was determined by the flooding algorithm. That is, both algorithms treat the robots in different ways: in one hand, our flooding algorithm allows robots to enter the environment only one at a time given that the conditions are favorable to do so; on the other hand, MR-DFS allows all the robots to enter the environment all at once. Thus, the flooding algorithm will explore a particular tree with the same number of robots (as stated in Section 5.3.1, we assume that there are sufficient robots at r); whereas MR-DFS can

explore a tree with a fixed number of robots from two to infinite. In order to make both algorithms comparable, we run the simulation for the flooding algorithm first and then we pass the number of robots used by it into the simulation for MR-DFS.

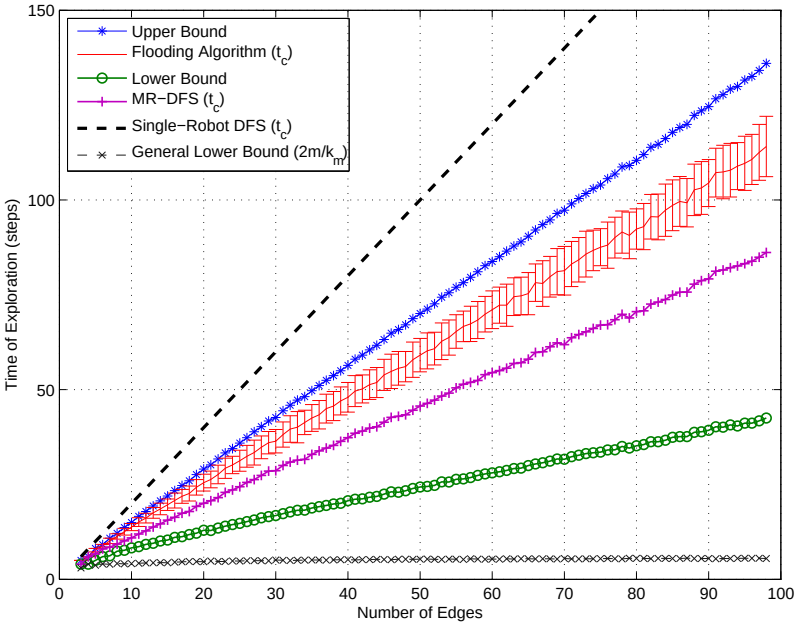
5.4.2 Results

The results for the complete exploration time using both algorithms are shown in Figs. 5.6a and 5.6b. The size of the trees is increased from $m = 4$ to $m = 100$. The results for N-ary trees are shown in Fig. 5.7. All plots show the upper and lower bounds defined on (5.2) and (5.10), the complete exploration time t_c for our flooding algorithm (mean of a 100 runs and standard deviation), for MR-DFS (mean of 100 runs only), the mean value of the exploration time of single-robot DFS (which is also the general upper bound) and the mean value of the general upper bound defined in (5.1).

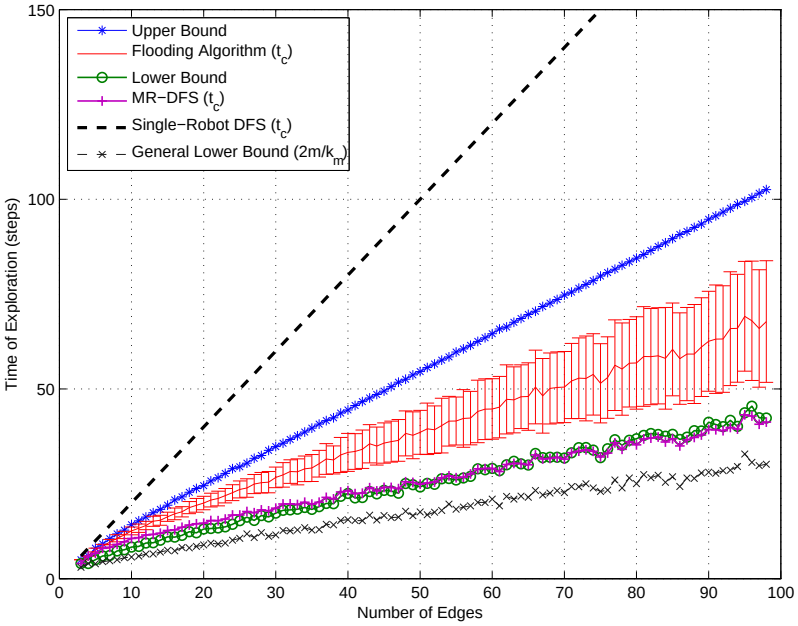
From the figures we can observe that not only both bounds are respected, corroborating the mathematical analysis on trees presented in Section 5.3.2, but also that the bounds are tightly closed to the actual exploration time. The tightness of the bounds is important since it allows for a better prediction of the exploration time for a given environment. It can also be seen that the calculated upper and lower bounds defined in (5.2) and (5.10), improve the general upper bound $2m$ given by single-robot DFS and the general lower bound given by $(2m/k_m)$. This is an important result for the analysis since, in the case of the upper bound, any expression bigger than the general upper bound, although valid, is useless for the analysis of the resources needed for exploring an environment. Comparison of both algorithms shows that MR-DFS is in general faster in performing the exploration in all tree configurations. This is an expected result since MR-DFS has greater parallelism due to the fact that it allows

all the robots available to start the exploration at the same time. Finally, the figures show that the exploration time for trees with the same number of vertices is longer when the root diameter of the tree is larger. This result shows that the exploration time of our flooding algorithm is heavily influenced by the root diameter of the tree.

Fig. 5.8 shows the number of robots required in average to perform the exploration. We observe that long trees require a lot more robots than wide trees. As expected, the number of robots needed for N-ary trees fells just in between the other two configurations, as can be seen when comparing the data presented in Fig. 5.7 with the plots of Fig. 5.8. A surprising result is the small amount of robots used by the algorithm to perform the exploration of wide trees even when the number of edges is high. In fact, from the plot it seems the curve reaches a maximum around seven robots (around the 60 edges mark) and continues in a flat trend from then on. This is a sharp contrast when compared with the curve for long trees which shows a constant increasing tendency. These results show the heavy influence that the root diameter of the tree has over the amount of robots needed to explore it.



(a) Long trees ($m = 4$ to $m = 100$)



(b) Wide trees ($m = 4$ to $m = 100$)

Figure 5.6: Time results on long and wide trees for both algorithms and single-robot DFS

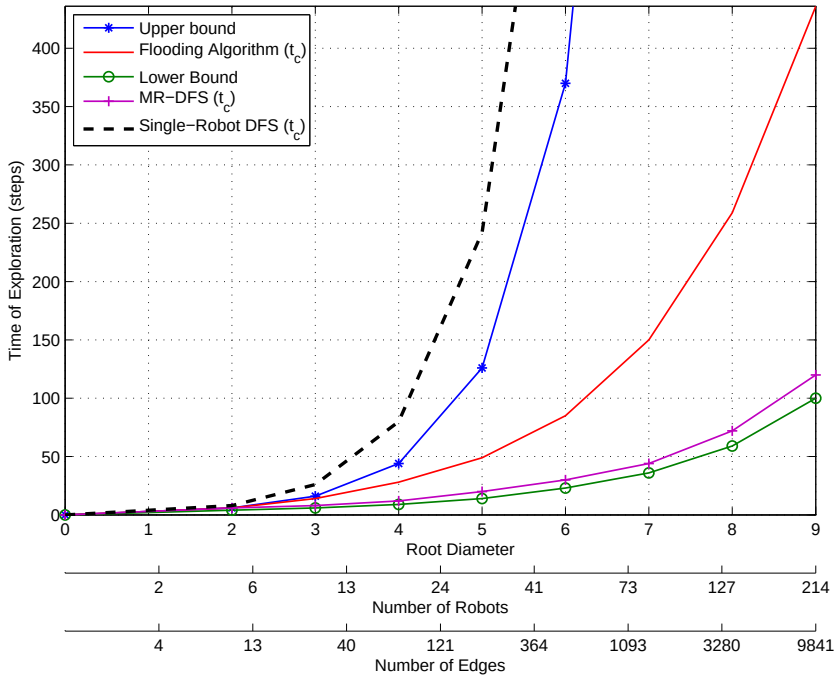


Figure 5.7: Time results for simulations on N-ary trees of different root diameter

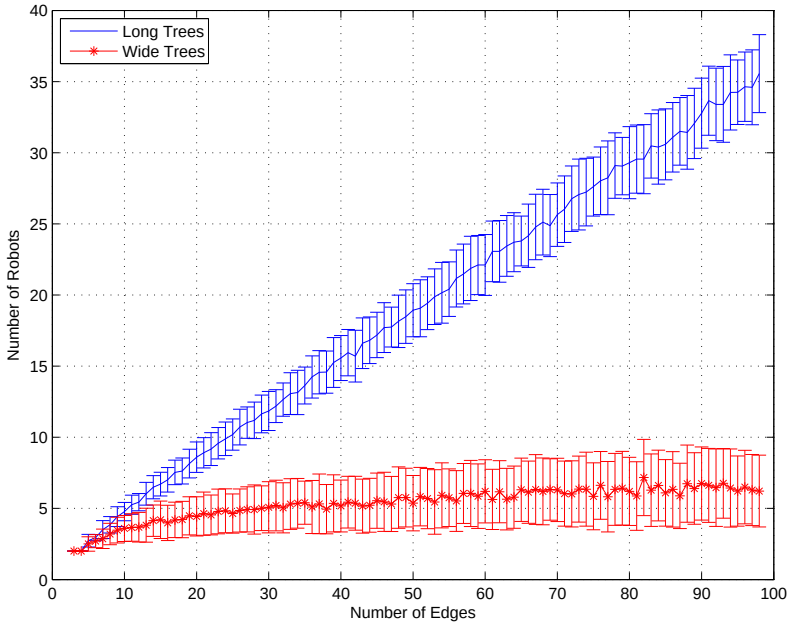


Figure 5.8: Number of robots used to explore long and wide trees

Chapter 6

Analysis of Multi-robot systems

In this chapter we study two important aspects related to multi-robot systems and their application on an exploration process. The first one is the measuring of the efficiency of the exploration algorithms for multi-robot systems when parameters other than time are considered. In our case, we use the total traversed distance of the robots as an additional parameter to represent the exploration cost. The second aspect is the influence of the number of robots on the exploration time and the specification of a limit on this number of robots so that the system produces the maximum reduction on the exploration time.

6.1 Study on the Traversed Distance on Multi-robot Systems

In this section we study the total traversed distance of all the robots performing the exploration of an unknown environment as the major indicator of the exploration overhead (cost). We define three metrics of performance for multi-robot algorithms

and then proceed to apply those metrics to the algorithms we proposed in Chapters 4 and 5. Through simulations we compare their performance along with that of classic single-robot Depth First Search.

6.1.1 General Considerations

As it has been stated throughout this dissertation, multi-robot systems are expected to perform faster not only in exploration but also other areas in robotics such as localization and mapping. Particularly in exploration, time has always been regarded as a critical parameter that needs to be minimized. One application that comes rapidly to mind in which fast exploration is required is search and rescue.

However, time is only one of the constraints that govern a multi-robot system. Take for example space, the most limiting constraint when exploring structured environments: lack of sufficient space may cause the robots to obstruct each other, preventing them to reach their destinations (deadlock). Another important constraint is distance. The overall traverse distance is linked to a variety of other limiting factors such as energy consumption: mobile robots are usually powered by batteries with limited capacity that restricts how much they can move. An exception to this argument could be unmanned aerial vehicles (UAVs) where the energy consumption is linked more to the time of flight than to distance.

Time and distance are usually conflicting optimization objectives: the time it takes one robot to explore a given environment can be reduced by allowing a second robot to help in the exploration; however, this increases the combined traverse distance of both robots. Generally speaking, adding more robots may reduce the exploration time but will definitely increase the overall traverse distance.

Consequently, an optimal exploration strategy should i) explore the environment

in the least amount of time, *ii*) make the robots traverse the overall shortest distance, and *iii*) coordinate the movement of the robots in order to minimize the occurrence of deadlocks.

Note that the overall traverse distance depends on the total number of robots that perform the exploration. For instance, with the coordination model of our flooding algorithm, robots that enter the environment will do it only one at a time. This raises the question of what is the maximum number of robots required to perform the exploration with this particular algorithm. The study of this maximum number of robots could provide us with the maximum overall traverse distance for the algorithm. As such, in this work, we assume that an unlimited number of robots k are stationed at the root vertex, although only a limited number of robots k_m actually explore the environment.

6.1.2 Criteria of Performance

For any multi-robot exploration algorithm we can define three criteria of performance: the total distance overhead μ_{total}, the time efficiency E_t, and the distance efficiency E_d.

Total Distance Overhead or Multiplicity (μ_{total})

It is the amount of additional edges that the robots traverse in order to achieve the same result than that of the optimal approach. Distance-wise, the optimal solution for multi-robot exploration will park all but one robot at the root vertex and then use DFS to explore the tree. This will produce an overall traverse distance of $2m$. Additional robots will always increase this value. On the other hand, the worst that multiple robots can perform is when all the robots traverse all the edges of the tree.

As such, Λ_{total} can be generally bounded as

$$2m \leq \Lambda_{total} \leq k_m 2m. \tag{6.1}$$

Since the optimal solution is $2m$, it is easy to see that Λ_{total} depends on the total number of robots that enters the tree and how much each individual robot moves in it. Recall that in order to complete the exploration, every edge must be traversed at least twice (once in each direction). As such, multiple robots will traverse collectively this amount of edges plus some edges in excess that will not contribute to the overall exploration process. From Definition 3.19 we have that

$$\Lambda_{total} = \sum_{i=1}^{k_m} \Lambda_i = 2m + \mu_{total}. \tag{6.2}$$

The second term (μ_{total}) is the only one that we can expect to reduce in order to make an algorithm more distance efficient. Thus a good metric of performance is to determine how much the distance overhead of the parallel algorithm is.

Time Efficiency (E_t)

It assesses how much faster the parallel algorithm arrives at the same solution than the optimal algorithm. That is,

$$E_t = \frac{t_{opt}}{t_{pal}},$$

where, t_{opt} is the time the optimal algorithm uses to explore an environment and t_{pal} is the time the parallel algorithm uses to do it. Using the general lower bound of (5.1), which gives us the time of the optimum algorithm, the time efficiency can be

defined as

$$E_t = \frac{2m/k_m}{t_c}. \tag{6.3}$$

Distance Efficiency (E_d)

It measures how well the entire system is utilized so that the exploration is performed in a way that less edges are traversed when compared with the optimal approach. That is,

$$E_d = \frac{E_{opt}}{E_{par}},$$

where, E_{opt} is the total traversed distance of all the robots using the optimum algorithm, and E_{par} is the total traversed distance of all the robots using the parallel algorithm. This criterion is related to the total distance overhead defined previously, but allows us to see the proportion in which the overhead relates to the actual total traverse distance. That is, algorithm A can have a small overhead in large trees when compare to algorithm B, but the difference can be more noticeable when both algorithms are compared to the optimal case. Thus, the distance efficiency can be defined as

$$E_d = \frac{2m}{\Lambda_{total}}. \tag{6.4}$$

6.1.3 Simulation Results

Objectives and Methodology

We implemented our flooding algorithm and performed all the simulation in *Matlab*. The objectives of conducting the simulations presented in this section are to study the performance of the flooding algorithm with respect to those of the multi-robot depth first search (MR-DFS) algorithm presented in Chapter 4 and classical single-robot

DFS. We choose MR-DFS for our comparison because it embodies the main property of other multi-robot algorithms in the literature: maximum parallelism in order to minimize the exploration time.

We run simulations applying both algorithms to the same randomly generated trees of increasing size using the same number of robots. The size of the tree was changed by increasing the number of vertices m.

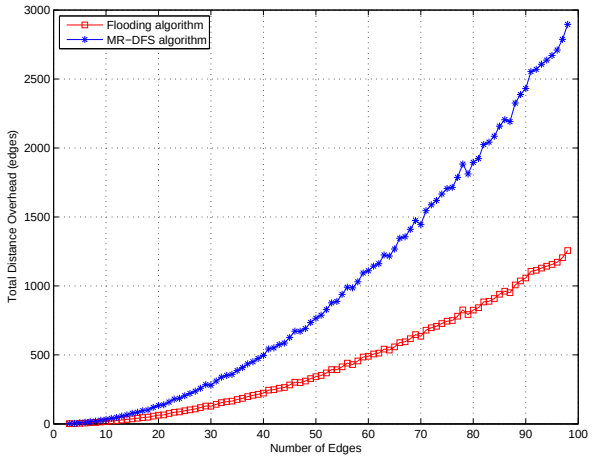
We study the performance of our algorithm using two aspects: traverse distance and time. To this effect, we consider the criteria defined in Section 6.1.2: total distance overhead, time efficiency and distance efficiency.

Results

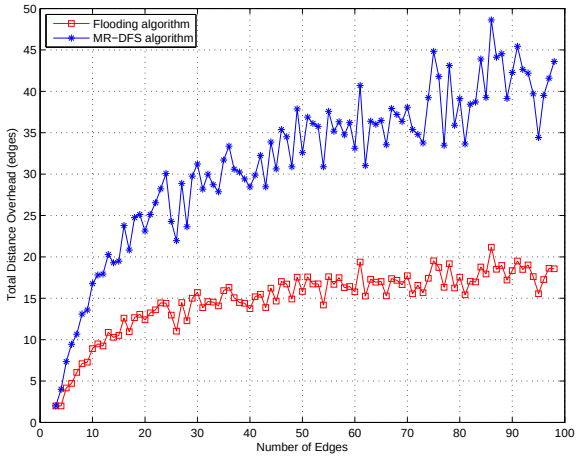
Fig. 6.1 shows the total distance overhead for the three types of trees. From the figure, we can see the main difference between our flooding algorithm and MR-DFS: the number of edges traversed in excess by the system of robots in order to perform the complete exploration. Higher overhead translates in a higher cost of exploration (bear in mind that both algorithms are using the same number of robots). An interesting result is that the curves for both algorithms have the same trend, which is due to the configuration of the trees. This means, that both algorithms behave similarly when facing the same environments, but MR-DFS is costly to implement than our flooding algorithm.

The measurement for time efficiency is shown in Fig. 6.2: for all the scenarios MR-DFS is more time efficient than our flooding algorithm. This comes not as a surprise since it is clear that MR-DFS has a higher parallelism than its flooding counterpart.

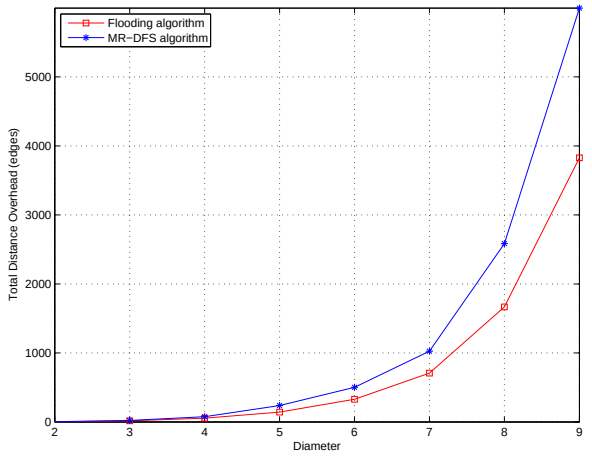
Fig. 6.3 show the measurement for the distance efficiency for both algorithms. This time our flooding algorithm outperforms MR-DFS, a result that is expected



(a) Long trees

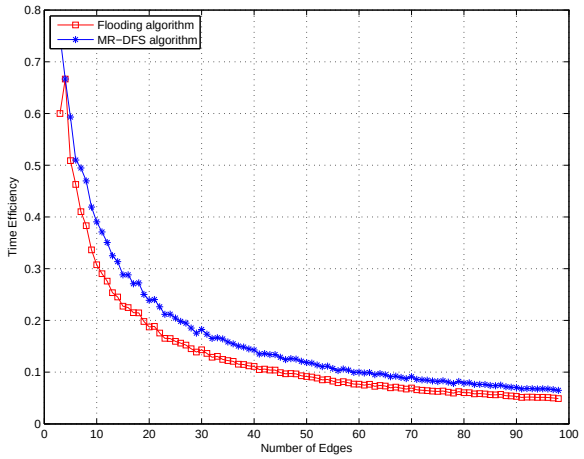


(b) Wide trees

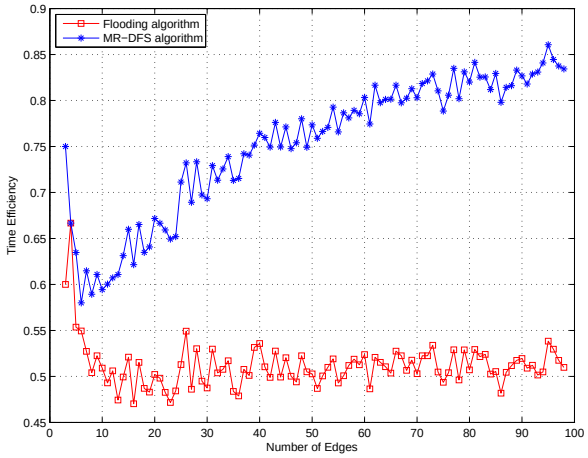


(c) N-ary trees

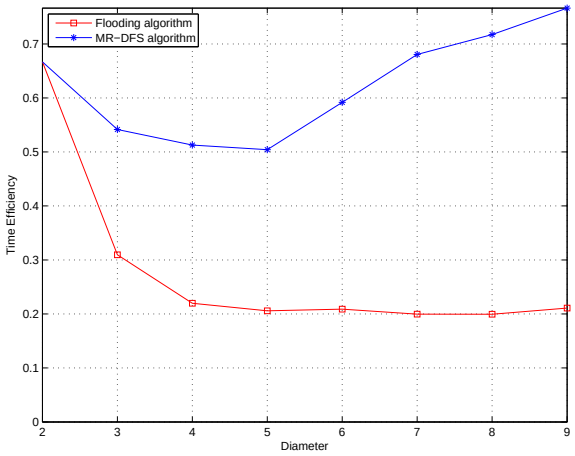
Figure 6.1: Total distance overhead



(a) Long trees



(b) Wide trees



(c) N-ary trees

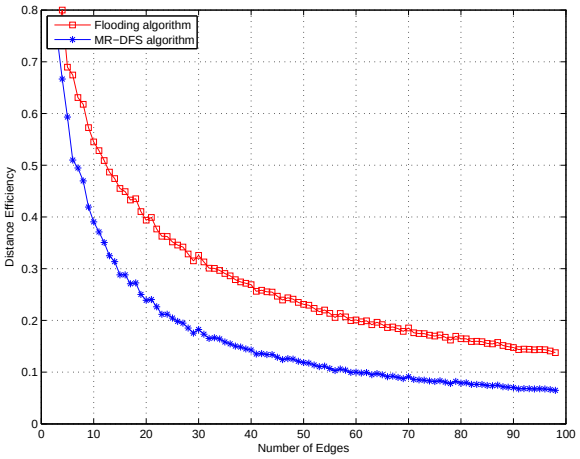
Figure 6.2: Time efficiency of exploration

since the main objective of our algorithm is to reduce the overall number of edges that the robots need to traverse in order to complete the exploration.

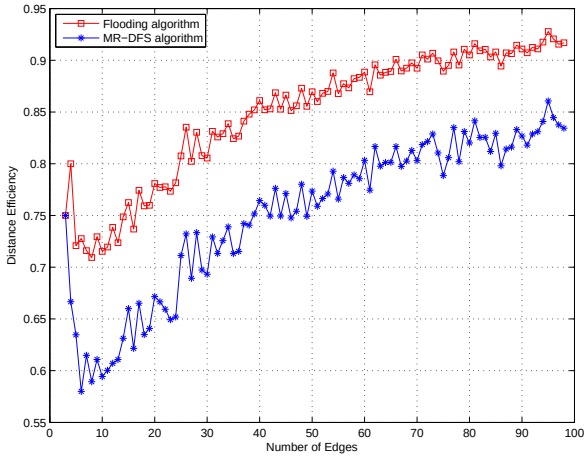
It is interesting to note how the efficiency, both in time and distance, decreases as the trees grow in diameter. Again, this is a direct consequence of having less parallelism in long trees. As it was stated before, the optimal scenario for parallel exploration is that the task is divided exactly into the different robots and with null multiplicity. In long trees, many robots need to traverse the same edges in order to reach a point where their contribution for the exploration is substantial, increasing the multiplicity and reducing efficiency. This is less evident in wide or N-ary trees where the robots start contributing to the exploration faster, yielding a smaller multiplicity.

6.2 Study on the Effects of k on the Exploration Time

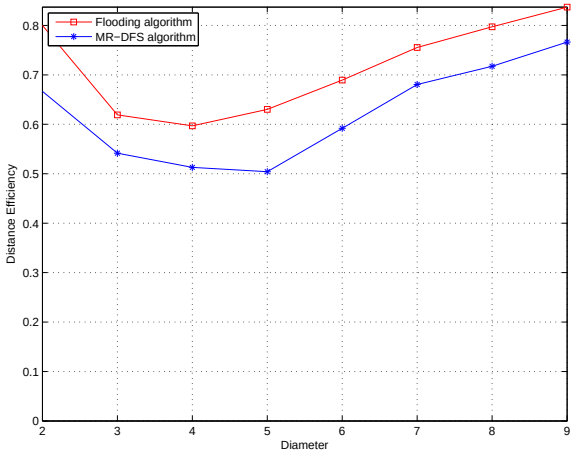
In this section we are interested in studying the effects of increasing the number of robots k on the exploration time of an environment that is modeled as a tree. To do it, we consider the MR-DFS algorithm presented in Section 4 since it allows all the robots in the system to start the exploration at the same time, providing the maximum parallelism for the exploration process. We develop an analysis that allow us to specify the maximum number of robots that provides the maximum reduction on the exploration time for a tree with m edges and root diameter D_r. We perform simulations to corroborate the results and provide a discussion about the findings.



(a) Long trees



(b) Wide trees



(c) N-ary trees

Figure 6.3: Distance efficiency of exploration

6.2.1 General Considerations

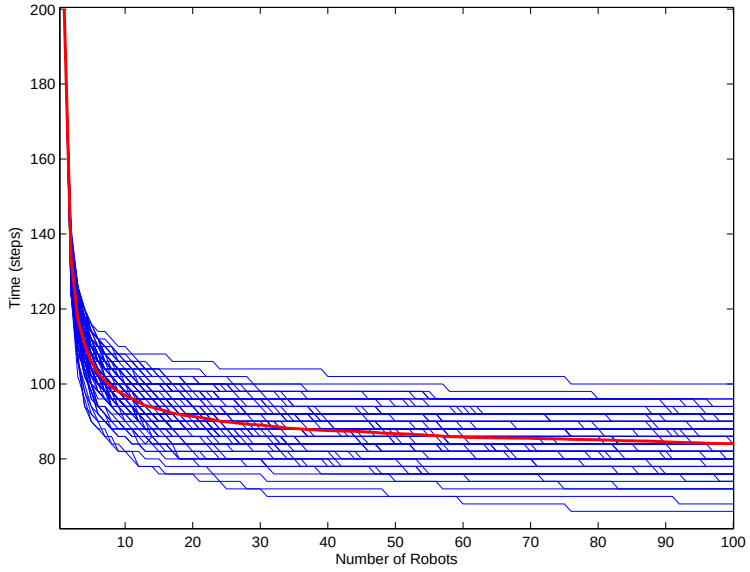
The main attribute of multi-robot systems is the parallelism that allows for a task to be completed faster than when the same task is executed by only one robot. Intuitively, the more robots are allowed to perform the exploration of an environment, the faster the exploration would be completed. In fact, for the particular case of the MR-DFS algorithm and the exploration time of a tree, it was proved in theorem 4.1 that the algorithm never does worse than single-robot DFS. As such, a degree of improvement on the exploration time is expected from the multi-robot system. Yet, this result arises several questions: *i)* how much is the ratio of improvement of the exploration time when the number of robots is increased? *ii)* to what value does the exploration time converges? *iii)* what is the maximum number of robots that produces the maximum reduction in the exploration time? The last two questions are related and are based on the fact that the exploration time cannot be reduced indefinitely, even if the number of robots is increased infinitely. In fact, it is foreseeable that the limit of improvement of the exploration time and its convergence value should be determined by the structure of the tree because in order to be explored, according to our definition of exploration (see Section 3.2), all the edges on the tree need to be traversed at least twice by any robot.

In this section, we are interested in studying the effects of increasing k on the exploration time t_c. In order to do it, we need to obtain an expression that relates t_c with k and the structure of the tree. To define the structure of the tree, we consider only two parameters: the number of edges m of the tree and the diameter of the root D_r. We acknowledge that these two parameters are not sufficient to uniquely define the structure of the tree but it provides a good enough approximation in order to describe how the exploration time is affected by the number of robots. As it was

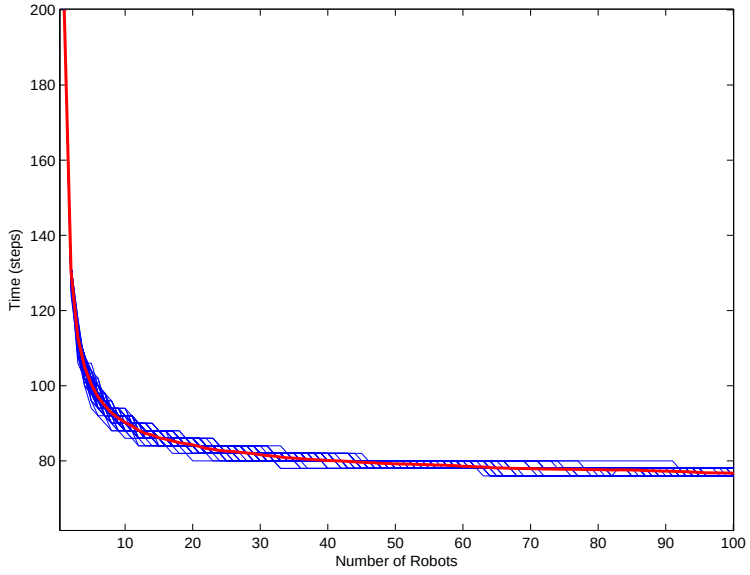
shown in Section 4.4.1 different trees can be constructed out of the same value for m (see Fig. 4.6). All these different trees produce completely different exploration times. As an example, observe Figs. 6.4a and 6.4b. These figures show the effect of increasing the number of robots on the exploration time of different long trees with the same number of edges but different diameters (Fig. 6.4a), and with the same number of edges and diameters (Fig. 6.4b). In red the average exploration time ($\bar{t}_c$) for both cases is shown. As can be seen, by considering only one additional parameter (the diameter of the root), the exploration time can be delimited more precisely (i.e., the standard deviation from the mean value of t_c is clearly reduced). Adding more parameters will allow us to obtain a more precise expression for the exploration time but it will also increase the complexity of the analysis.

In Fig. 6.5 we observe how each pair $\{m,\ D_r\}$ produce a different average value of t_c which is distinct for the result of any other pair. As such, and because its standard deviation is small, the average value of t_c characterizes the exploration time for a given pair $\{m,\ D_r\}$. Since we are looking to find a function that describes how the exploration time is affected generally by the number of robots, we settle for these two parameters, and our analysis will compare our obtained function with the average exploration time $\bar{t}_c$.

Let us define $f(k, m, D_r)$ as such a function. Without further information other than m, D_r and k, finding $f(k, m, D_r)$ becomes an impossible task since there is not direct relation between any of the variables. That is, there does not exists a well-defined function that relates m with D_r, or m with k, or k with D_r. As such, in order to solve this problem, we rely on the seminal work of (Graham, 1969) described in the following section.



(a)



(b)

Figure 6.4: Exploration time for different trees with same number of edges ($m = 100$), explored by an increasing number of robots, with (a) no restriction on the diameter of the root and (b) with the restriction ($D_r = 32$). The mean exploration time is shown in red.

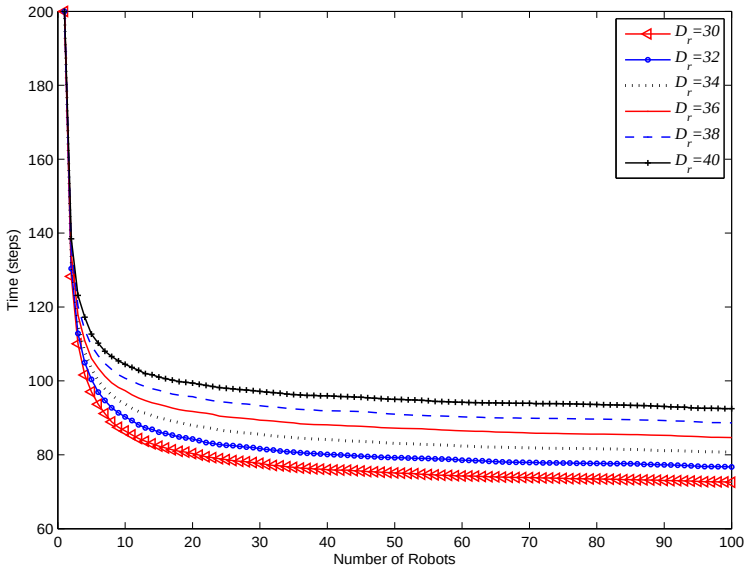


Figure 6.5: Mean exploration time for trees with different pair $\{m, D_r\}$: same number of edges ($m = 100$) but different diameter restriction.

6.2.2 Bounds of Multi-processing Systems and Its Application to Multi-robot Exploration

In (Graham, 1969), a multi-processing system formed by n identical processing units P_i, $1 \leq i \leq n$, is to process a set of tasks $T = \{T_1, \ldots, T_r\}$. Given a partial order $\prec$ on T, $T_i \prec T_j$ denotes that task T_j cannot start until T_i is completed. Also given is a linear order $L : (T_{k_1}, \ldots, T_{k_m})$ of T called a *task list* or *priority list*. Finally, a function $\mu : T \to [0, \infty)$ is defined, in which $\mu(T_j)$ denotes the units of time required to process task T_j.

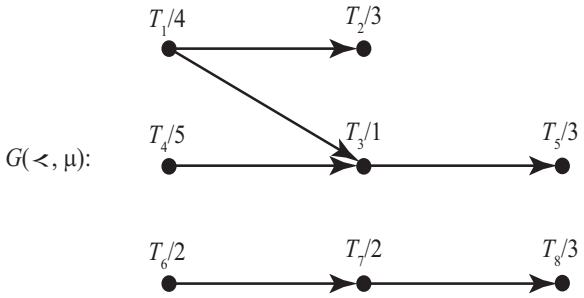
In a system defined in this way, whenever a processor P_i starts executing a task, it will do so until completion. The execution of the task cannot be interrupted. Once a processor P_i finishes a task T_j, it searches the list L to identify the first task T_k which has not being started yet. If all the tasks for which $T_i \prec T_k$ (i.e., the predecessors of

T_k) have been finished, then P_i will start executing T_k. Otherwise, it will look for the next available task T'_k. If, after searching the list L, P_i found no task to executed, then P_i becomes idle, and it will remain in this state until some other processor P_j finishes the task it was executing. At this point, both P_i and P_j start searching the list L for a task to execute. If two processors attempt to execute the same task, a negotiation process starts in which the processor with the lowest index will be allowed to execute the task. The process is repeated until some time ω in which all tasks have been completed.

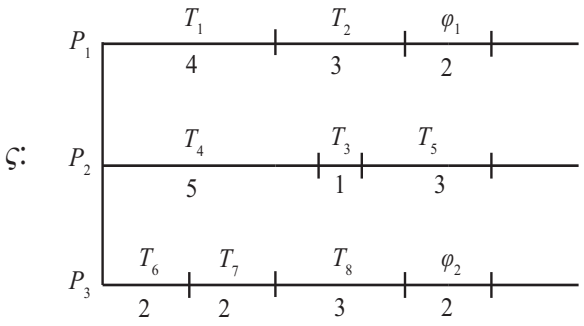
To have a better understanding of this process consider the following example that has been taken from (Graham, 1966). Assume that three processors are required to execute a list of eight tasks. Formally, we can define the priority list for this example as $L : (T_3, T_1, T_2, T_4, T_6, T_5, T_7, T_8)$, that needs to be executed by P_i, where $1 \leq i \leq 3$. The partial order and the function μ are given by a directed graph $G(\prec, \mu)$ shown in Fig. 6.6a. In the graph, the vertices correspond to the tasks T_i, and the directed edges indicate the precedence of each task (i.e., the partial order $T_i \prec T_j$). Each vertex is labeled with the name of the tasks and the time required to execute that task (i.e., $T_i/\mu(T_i)$). For instance, the vertex labeled as $T_1/4$ indicates that the task T_1 requires 4 steps to be executed. The activity of each processor is shown in the Gantt diagram ς of Fig. 6.6b. When a processor becomes idle, the symbol φ_i is attached to the corresponding processor.

At the beginning, each processor scans the priority list L. The first task on the list, T_3, cannot be executed because it is preceded by T_1, as it can be seen on the partial order of Fig. 6.6a. The second task T_1 can be executed and it is assigned to P_1. At this point, the other two processors keep scanning the list. The third task in the list, T_2, cannot be executed because it is preceded again by T_1, but the next

task T_4 can, and it is assigned to P_2. The process is repeated until all processors are assigned a task. The list is completed after nine steps ($\omega = 9$).



(a) Directed graph



(b) Gantt diagram

Figure 6.6: Example of a multi-processing system with three processors executing a list L with 8 tasks. Taken from (Graham, 1966)

This method of defining the works of a multi-processing system allows for the study of different situations, called by the authors "*anomalies*", in which adding more processors do not necessarily result in a shorter executing time. The authors present different cases and their analysis yield bounds on the execution times of set of tasks that can be influenced by such anomalies. One of such cases is of particular

interest for our work and it is described as follows: assume that we execute a set of tasks twice, once with the list L_w that produces the longest execution time ω_L, and once with the list L_o that produces the minimum possible (*optimal*) execution time ω_0. If we were to calculate the ratio between ω_L and ω_0, the best possible bound on this ratio would be given by

$$\frac{\omega_L}{\omega_0} \leq 2 - \frac{2}{n+1}, \tag{6.5}$$

where n is the number of processors.

For our study, we can relate the description of a multi-robot exploration process into the form of a multi-processing system working on different tasks, as can be seen in Table 6.1. This allows us to use the result on (Graham, 1969) shown above for our analysis as it will be described in the next section.

n : number of identical processing units on the system	$\Longleftrightarrow$	k : number of identical robots on the system
P_i : processing unit i for $1 \leq i \leq n$	$\Longleftrightarrow$	r_i : robot i for $1 \leq i \leq k$
T : set of tasks to be processed by P_i that is defined by $T = \{T_1, \ldots, T_r\}$	$\Longleftrightarrow$	E : set of edges that need to be traversed by r_i and that is defined by $E = \{e_1, \ldots, e_r\}$
L : list of tasks that needs to be executed in order of priority	$\Longleftrightarrow$	L : list of edges that need to be traversed in a given sequence in order to perform the exploration.
$\prec$: partial order of tasks	$\Longleftrightarrow$	$\prec$: partial order of edges to be traversed.
$\mu(T_j)$: time it takes a processor P_i to execute task T_j	$\Longleftrightarrow$	$\mu(e_j)$: time it takes a robot r_i to traverse the edge e_j from vertex u to vertex v
ω : time it takes the multiprocessing system to execute all the tasks in T	$\Longleftrightarrow$	t_c : time it takes the multi-robot system to traverse all edges and return back to the root.

Table 6.1: Relationship between multi-processing system and multi-robot exploration process

The relationship in Table 6.1 allows us to redefine the multi-robot exploration process as follows: assume that a tree $\mathcal{T} = \{V, E\}$, formed by n vertices and m edges

is to be explored by k identical robots r_i, $1 \leq i \leq k$. Assume that each edge of the tree is distinctively labeled and that we can construct a set of edges $E = \{e_{01}, e_{11}, \ldots, e_{vw}\}$ that need to be traversed by the robots. Due to the structure of the tree, we can construct a partial order $\prec$ on E such that $e_{01} \prec e_{11}$ denotes that the edge e_{11} cannot be traversed until the edge e_{01} is completely traversed. As such, it is possible to create different priority lists $L : (e_{v_1}, \ldots, e_{v_m})$ of E that determine the movement of the robots inside the tree, all resulting in different exploration times. Additionally, we can define a function $\mu : E \to [0, \infty)$ in which $\mu(e_{vw})$ denotes the time it takes a robot r_i to traverse the edge e_{vw} (for simplicity, we can assume that $\mu(e_{vw}) = 1$ step for all the edges in E). In this scenario, t_c is the time it takes to traverse all edges on the list (i.e., the complete exploration time).

Let us assume that we explore the tree twice using the same number of robots but different priority lists L_o and L_w, where L_o is the list that produces the minimum possible (*optimal*) exploration time t_{c_o}, and L_w is the list that produces the longest exploration time t_{c_w} (of all possible lists). Using the result on (6.5), we can write the ratio between both exploration times as

$$\frac{t_{c_w}}{t_{c_o}} \leq 2 - \frac{2}{k+1}. \tag{6.6}$$

As an example, consider the tree of Fig. 6.7. To explore this tree we can define the set of edges $E = \{e_{01}, e_{11}, e_{12}, e_{13}, , e_{21}\}$ and the following partial orders:

$$e_{01} \prec e_{11},$$

$$e_{01} \prec e_{12} \prec e_{21},$$

$$e_{01} \prec e_{13}.$$

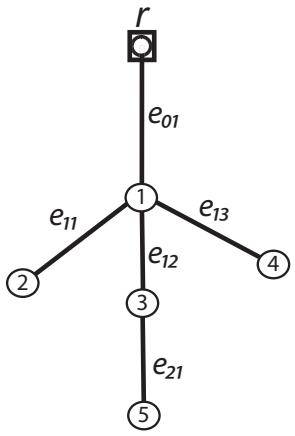


Figure 6.7: Tree with labeled edges

These partial orders are the result of observing that, for example, a robot cannot traverse e_{32} without traversing at least e_{01} and e_{12} first, and so on. if we consider two robots to perform the exploration, it is easy to determine the optimal and worst lists: $L_o : (e_{01}, e_{12}, e_{21}, e_{11}, e_{13})$, $L_w : (e_{01}, e_{11}, e_{13}, e_{12}, e_{21})$. Since we are considering the MR-DFS algorithm, some edges can be traversed at the same time by more than one robot. This tree can be optimally explored in time $t_{c_o} = 6$ steps: upon reaching the vertex *1*, one robot explores the short branches e_{11} and e_{13}, and the other robot explores the longer branch formed by e_{12} and e_{21}. On the contrary, in the worst case, the tree will be explored in time $t_{c_w} = 8$ steps: upon reaching the vertex *1*, one robot takes e_{11}, and the other e_{13}, then both together explore e_{12} and e_{21}. If we calculate the ratio between both exploration times, we can observe that the result validates the expression in (6.6). That is, $\frac{t_{c_w}}{t_{c_o}} = \frac{4}{3} \leq 2 - \frac{2}{k+1}$.

6.2.3 The Function $f(k, m, D_r)$

The main objective of this section is to obtain an expression for $f(k, m, D_r)$ that resembles the behavior of the average exploration time ($\bar{t}_c$) of trees with m edges and root diameter D_r. To this end, we first characterize the behavior of the exploration time of a tree when the number of robots is increased, using the approach presented in section 6.2.2.

Lemma 6.1 *The behavior of the average exploration time ($\bar{t}_c$) of the MR-DFS algorithm on trees, when the number of robots k performing the exploration is increased, can be characterized by an exponential function.*

Proof: We start the proof by noting that an exponential function is characterized by the fact that over equal intervals, the ratio of change (percentage rate of change) of the function is constant. As such, to prove that the average t_c can be written as an exponential function we need to prove that its ratio of change, with respect to k, is constant. That is, we need to prove that

$$\frac{\Delta \bar{t}_c(k)}{\bar{t}_c(k)} = b, \tag{6.7}$$

where $\Delta \bar{t}_c(k) = \bar{t}_c(k + \Delta k) - \bar{t}_c(k)$, and b is constant.

To do it, we can use the result presented in Section 4.3.1 for the lower bound of t_c, and then combine it with the one in (6.6). We use the optimal exploration time because without further information, other than the m and D_r, the behavior of t_c with respect to k cannot be known. The behavior of the optimal and worst exploration times, that create an envelop around the exploration times produced by lists different from L_o and L_w, allow us to have an insight of the behavior of $\bar{t}_c$.

We can rewrite the lower bound defined in Section 4.3.1 for this analysis as

$$t_{c_o} = \max(2m/k, 2D_r). \tag{6.8}$$

As observed, this expression is formed by two parts, one which varies and the other that is constant, where the variable part, $t_{c_o}(k) = 2m/k$, corresponds to the optimal case where maximum parallelism is achieved when exploring a tree with k robots. This expression differs from a regular hyperbola in the fact that after a point, k_{max}, the expression is constant. Note that the lower bound in (6.8) cannot be achieved in all trees, and in those in which is achievable, it does so at the point where $2m/k = 2D_r$ (that is, at k_{max}). For all the other points in a tree, we can expect a constant ratio of change that takes the value of t_{c_o} from $2m$ all the way to $2D_r$. In order to find this ratio of change, let us consider two points: one at $k = 1$ and the other at $k = k_{max}$. The optimal exploration times at those points are $t_{c_o}(1) = 2m$ and $t_{c_o}(k_{max}) = 2m/(k_{max}) = 2m/(1 + \Delta k)$ respectively. If we calculate the discrete derivative $\Delta t_{c_o}(k_1)/\Delta k$ we get

$$\begin{aligned}
\frac{\Delta t_{c_o}(k)}{\Delta k} &= \frac{(t_{c_o}(k_{max}) - t_{c_o}(1))}{\Delta k} \\
t_{c_o}(k_{max}) - t_{c_o}(1) &= \frac{2m}{1+\Delta k} - 2m \\
&= 2m\left(\frac{-\Delta k}{1+\Delta k}\right) \\
&= t_{c_o}(1)\left(\frac{-\Delta k}{1+\Delta k}\right).
\end{aligned}$$

Then, the ratio of change (from $k = 1$ to $k = k_{max}$) can be expressed as

$$\frac{\Delta t_{c_o}(k)}{t_{c_o}(1)} = \left(\frac{-\Delta k}{1+\Delta k}\right). \tag{6.9}$$

We can set this ratio of change as the one for the rest of the curve from $2m$ to $2D_r$. As such, for a $\Delta k = 1$, which is the minimum variation on k allowable, we can set a constant ratio of change of $-1/2$. This ratio is valid since, for each value of k, t_{c_o} decreases faster than a typical exploration time. That is, the ratio indicates that the exploration time is reduced in half for every new robot into the tree, which is in fact an optimal situation as it will be explained in the proof of the following theorem.

The next step is to consider the case for which the exploration time is the worst. To this end, we use (6.6): the worst exploration time can be written as a function of the optimal exploration time as

$$t_{c_w} \leq \left(2 - \frac{2}{k+1}\right) t_{c_o}. \tag{6.10}$$

Since t_{c_w} is proportional to t_{c_o}, which can be described by an exponential function, then t_{c_w} can also be written as an exponential function.

Fig. 6.8 shows a typical behavior of this lower bound along with the upper bound obtained from (6.10), applied to a tree with $m = 30$ and $D = 2$. The figure allows us to observe the exponential decaying nature of the exploration time when more robots are considered. Any other list L, different from the optimal list L_o, will produce an exploration time that will lay in between both curves, maintaining an exponential decay. As such, we can conclude that for any tree configuration, the average exploration time ($\bar{t}_c$) will have a decaying exponential behavior when the number of robots performing the exploration is increased. □

With the following theorem we define $f(k, m, D_r)$.

Theorem 6.1 *The average exploration time of a tree with m edges and diameter of*

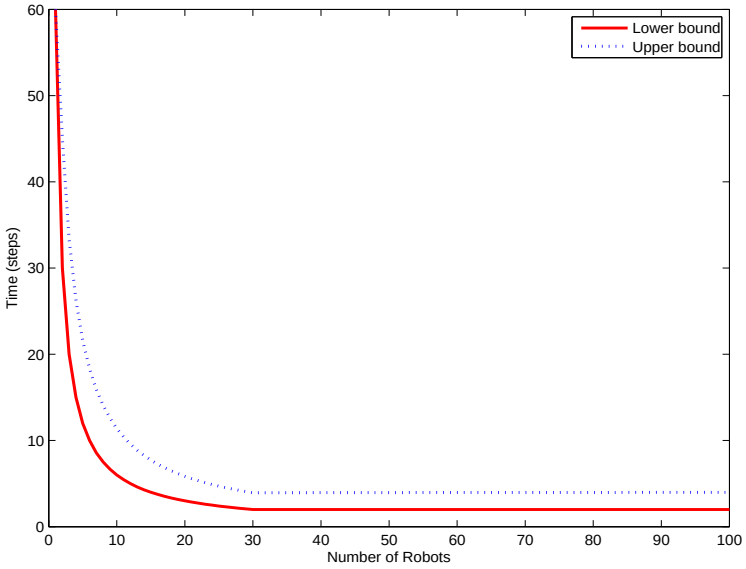


Figure 6.8: Lower (optimal) and upper bounds for exploration time a tree with $m = 30$, $D_r = 2$, and increasing number of robots

the root D_r, being explored by k robots resembles the curve produced by the function

$$f(k, m, D_r) = 2(m - D_r)e^{\tau(k-1)} + 2D_r. \tag{6.11}$$

Proof: The proof starts with the result of lemma 6.1: since the behavior of the average exploration time ($\bar{t}_c$) can be characterized by an exponential function, then our target function is an exponential function as well, of the form

$$g(k) = ab^{k-1}, \tag{6.12}$$

where a is initial value of the function, and b is its ratio of change. We shifted the function one step in order to account for the fact that the exploration time is undefined at $k = 0$. As such, the function starts at $k = 1$ and converges at zero as k increases indefinitely. Evidently, the exploration time can never converge to zero,

either, so we correct our function once again so that it does it to a different value c. The new function $f(k)$ can then be written as

$$f(k) = ab^{k-1} + c = g(k) + c. \tag{6.13}$$

The independent term, c, is the value at which the exploration time converges after enough robots are used to perform the exploration. It can be thought of as the point in which the exploration time cannot be improved any more and as such it does not depend on the number of robots. To find it, let us consider the longest path that runs from the root to a leaf (i.e., the path that defines the diameter), as shown by Fig. 6.9. In a tree, there could be many such paths. If enough robots are present in the system, we can expect that at least one robot (let us call it r_s) will explore this path and return back to the root without traversing any other additional edge. All the other robots will traverse all the other edges. It will take r_s exactly $2D_r$ steps to reach the leaf and return back to the root and this is the fastest a tree can be explored. As such, this is the value the exploration time will converge to when the number of robots is increased.

The initial value a can be found by using the fact that $f(1) = 2m$. Then,

$$\begin{aligned} 2m &= a + 2D_r \\ a &= 2(m - D_r). \end{aligned}$$

The last term we need to look for is the decay ratio b. For simplicity and since this ratio is the same for $f(k)$ and for $g(k)$, we use the latter to obtain it. Assume we have two points $g(k)$ and $g(k + \Delta k)$. Using the definition on (6.12) we can calculate

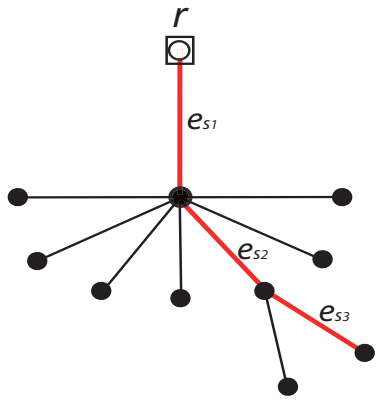


Figure 6.9: Example of a longest path S. In this tree it is formed by edges $\{e_{s1}, e_{s2}, e_{s3}\}$

the discrete derivative of the function as follows:

$$\begin{aligned} \frac{g(k+\Delta k)-g(k)}{\Delta k} &= \frac{(ab^{k+\Delta k-1})-(ab^{k-1})}{\Delta k} \\ g(k+\Delta k)-g(k) &= (ab^{k-1}b^{\Delta k})-(ab^{k-1}) \\ &= ab^{k-1}\left(b^{\Delta k}-1\right). \end{aligned}$$

Dividing by $g(k)$

$$\begin{aligned} \frac{g(k+\Delta k)-g(k)}{g(k)} &= \frac{ab^{k-1}\left(b^{\Delta k}-1\right)}{ab^{k-1}} \\ &= b^{\Delta k}-1. \end{aligned}$$

For $\Delta k = 1$ we have

$$\begin{aligned} b &= \frac{g(k+1)-g(k)}{g(k)}+1 \\ &= \frac{g(k+1)}{g(k)}. \end{aligned}$$

Thus, to find b we need to known at least two point in the exploration time curve. We consider the first interval for $k = 1$ and $k = 2$. When $k = 1$, we already know that $t_c(1) = 2m$. When $k = 2$, according with Theorem 4.4, we have that $t_c \leq (m + D_r)$, which is the best upper bound for algorithms using two robots. As such, we have that $m \leq t_c \leq (m+D_r)$. In the worst case we can assume that $t_c(2) = (m+D_r)$. These two points will be also part of the curve of $f(k)$ and we can use them to calculate b. From (6.13) we have that $g(1) = f(1) - 2D_r = 2m - 2D_r$ and $g(2) = f(2) - 2D_r = m - D_r$, and as a result

$$\begin{aligned} b &= \frac{g(2)}{g(1)} \\ &= \frac{m - D_r}{2m - 2D_r} \\ &= 1/2. \end{aligned}$$

Putting all the results together we get that (6.13) becomes a function not only of k, but also of m and D_r. That is,

$$f(k, m, D_r) = 2(m - D_r)(1/2)^{k-1} + 2D_r. \tag{6.14}$$

Finally, since we can rewrite any exponential function with the natural base (recall, $f(x) = ab^x = a(e^\tau)^x = ae^{\tau x}$, where $\tau = ln(b)$), then we can express (6.14) as follows:

$$f(k, m, D_r) = 2(m - D_r)e^{\tau(k-1)} + 2D_r,$$

where $\tau = \ln(1/2)$. □

6.2.4 The Number of Robots and Its Influence on The Exploration Time

In this section we study how we can define the maximum number of robots (k_0) that produce a significant reduction on the exploration time (i.e., a limit on the number of robots), where the key is to define what *significant reduction* stands for. Significant reduction can be thought of as the minimum reduction in exploration time that we allow our system to get when we use more robots. It can depend on many factors, for instance, at a practical level we can define this parameter by analyzing if the gain in time is proportional to (worthy) the cost of the additional robots. Our study looks for a methodical and analytical way to obtain an expression for k_0.

The following theorem summarizes the result of our analysis.

Theorem 6.2 *In a multi-robot system using the MR-DFS algorithm, the maximum number of robots k_0 that produce a significant reduction on the exploration time can be defined as*

$$k_0 = \left\lfloor 2 - \frac{1}{\ln(2)} \ln\left(\frac{\alpha}{m - D_r}\right) \right\rfloor, \tag{6.15}$$

for $m \neq D_r$.

Proof: let us define α as the minimum reduction in exploration time that is allowed to occur when then number of robots is increased by one. That is,

$$\Delta f(k) \geq \alpha. \tag{6.16}$$

Let us define k_0 as the maximum value of k for which (6.16) holds. If we define k_{0-1}

as the second maximum value (i.e., $k_{0-1} = k_0 - 1$), then we have

$$f(k_{0-1}) - f(k_0) \geq \alpha.$$

Using the definition for $f(k)$ on (6.11) we obtain

$$\begin{aligned}
\left(2(m - D_r)e^{\tau(k_{0-1}-1)} + 2D_r\right) - \left(2(m - D_r)e^{\tau(k_0-1)} + 2D_r\right) &\geq \alpha \\
2(m - D_r)\left(e^{\tau(k_0-2)} - e^{\tau(k_0-1)}\right) &\geq \alpha \\
\left(e^{\tau k_0}e^{-2\tau} - e^{\tau k_0}e^{-\tau}\right) &\geq \frac{\alpha}{2(m - D_r)} \\
e^{\tau k_0}\left(e^{-2\tau} - e^{-\tau}\right) &\geq \frac{\alpha}{2(m - D_r)}.
\end{aligned}$$

To evaluate $e^{-2\tau}$ and $e^{-\tau}$, recall that $\tau = \ln(1/2)$. Then, we have that $e^{-2\tau} - e^{-\tau} = 2$, and as such

$$\begin{aligned}
e^{\tau k_0} &\geq \frac{\alpha}{4(m - D_r)} \\
\tau k_0 &\geq \ln(\alpha) - \ln(4(m - D_r)).
\end{aligned}$$

Since τ is a negative number, we can rewrite k_0 as

$$k_0 \leq \frac{1}{\tau}\left(\ln(\alpha) - \ln(4) - \ln(m - D_r)\right).$$

Using the definition of τ and the fact that $\ln(1/2) = -\ln(2)$ we have

$$\begin{aligned}
k_0 &\leq -\frac{\ln(\alpha)}{\ln(2)} + \frac{\ln(4)}{\ln(2)} + \frac{\ln(m - D_r)}{\ln(2)} \\
&\leq 2 - \frac{1}{\ln(2)}\left(\ln(\alpha) - \ln(m - D_r)\right),
\end{aligned}$$

which can be written as

$$k_0 \leq 2 - \frac{1}{\ln(2)} \ln\left(\frac{\alpha}{m - D_r}\right). \tag{6.17}$$

Finally, since by definition k_0 is not any value but the maximum number of robots for which (6.17) holds, and this number must be an integer, then we can define k_0 as

$$k_0 = \left\lfloor 2 - \frac{1}{\ln(2)} \ln\left(\frac{\alpha}{m - D_r}\right) \right\rfloor. \tag{6.18}$$

□

The result on (6.15) shows the dependence of the value of k_0 on our definition of the parameter α. As mentioned before, this parameter depends on the characteristics of the system and its purpose. Without entering to define it specifically, we can assume the very conservative approach of expecting an improvement of at least one step on the exploration time when one additional robot is included in the process (that is, $\alpha = 1$). This assumption yields that (6.15) becomes

$$k_0 = \left\lfloor 2 + \frac{\ln(m - D_r)}{\ln(2)} \right\rfloor, \tag{6.19}$$

for $m \neq D_r$. That is, our solution excludes the case when the tree is a path, in which case it is clear that $k_0 = 1$.

The result on (6.19) suggests very interesting facts about the exploration process, that will later be corroborated by simulations. For instance, that for trees in which the number of edges is a lot bigger than the root diameter (like in the case of wide trees), more robots will produce a bigger impact on the exploration time since k_0 will be a lot greater than 2. This is due to the greater parallelism that can be achieved on wide

trees. On the other hand, for long trees, where the number of edges is comparable to the root diameter, not many robots are required to produce the maximum reduction on the exploration time. Take for example the case for a long path that only diverges at the very end (a path with two leaves) like the one shown in Fig. 6.10. In this tree

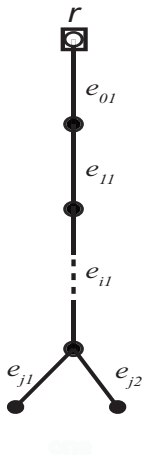


Figure 6.10: A long path with two leaves

we can define $m = D_r + 1$. Then, by using the result on (6.19) we have that $k_0 = 2$. If we consider two robots, the exploration time equals $2D_r$. Including more robots do not improved the exploration time, and as such the result produced by (6.19) is correct.

6.2.5 Simulation Results

Objectives and Methodology

We performed all the simulation in *Matlab*. The objectives of conducting the simulations presented in this section are twofold, i) to test the fitness of the function $f(k, m, d_r)$ defined on theorem 6.1 with respect to the actual average exploration time

$\bar{t}_c(k)$, and *ii*) to observe the accuracy of the function defined by theorem 6.2 to predict the the maximum number of robots k_0.

We run simulations applying the algorithm to randomly generated trees of increasing size using the same number of robots. The size of the tree was changed by increasing the number of vertices m. The size of the trees is increased from $m = 10$ to $m = 150$.

The methodology of the simulations is as follows: first a random tree is generated, next the same tree is explored to completion by one robot, then by two robots, and so forth until the tree is explored to completion by the maximum number of robots. For these simulations we considered 500 robots as the maximum number of robots. After this, a new random tree is generated and the process is started again.

As it was mentioned in Section 4.4.1, different tree configurations are possible for the same value of m (see Fig. 4.6); a fact that directly affects how the tree is explored and the exploration time. As such, we run 100 simulations for every value of m, and then the mean and the standard deviation were calculated. The results shown on Section 6.2.5 correspond mainly to mean values.

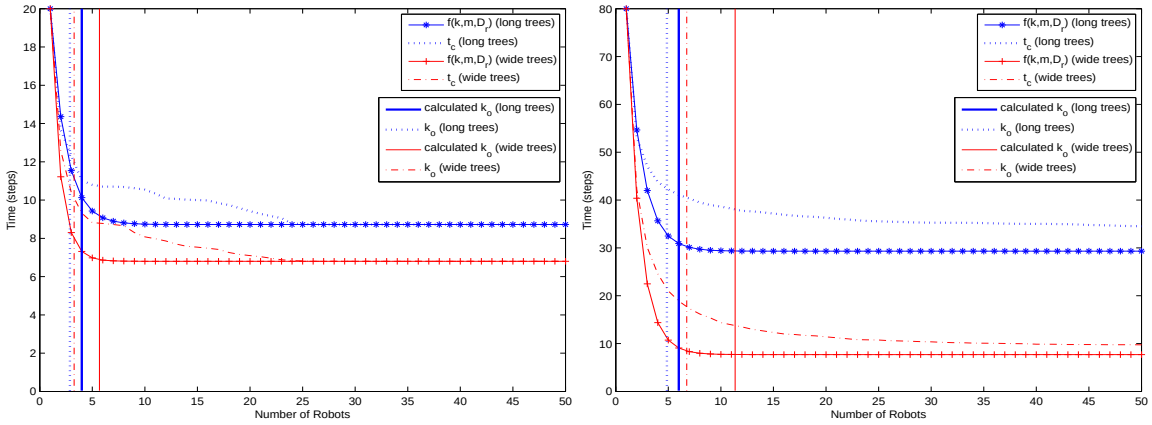
Furthermore, we considered the categorization of the randomly generated trees, described in Section 4.4.1, where trees are grouped as *long* and *wide* trees (see Fig. 4.7). For this analysis we do not consider *symmetric trees* because in this case they do not provide us with meaningful results *different* from the ones obtained for the other two categories.

Results

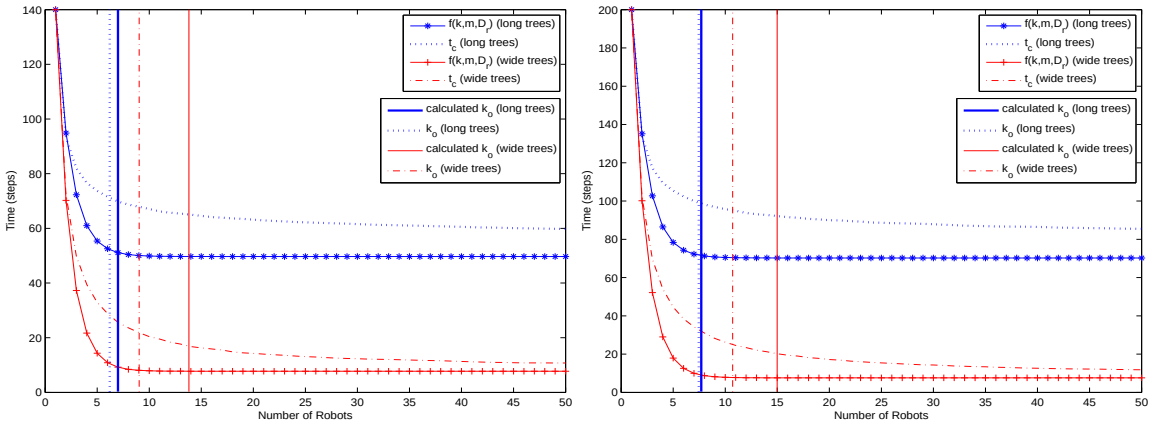
The results for the comparison of the average exploration time and the calculated function $f(k, m, D_r)$ for both, wide and long trees, are shown in Fig. 6.11. Both, the

values for $\bar{t}_c(k)$ and the values for $f(k, m, D_r)$, are the mean of the 100 simulation runs performed for each value of m. Although the number of robots performing the exploration on each randomly generated tree was increased from $k = 1$ to $k = 500$, on the plots of Fig. 6.11 only the results up to $k = 50$ are shown. Also in Fig. 6.11, the comparison between the value of k_0 calculated using (6.15) and the actual value of k_0 obtained from the average exploration time $\bar{t}_c(k)$ is shown. The actual value of k_0 is the mean of the 100 runs (per each value of m). The calculated value of k_0 was obtained by using (6.19), assuming $\alpha = 1$ and with the mean value of the diameter of the root, which is the only parameter that changes on each of the 100 runs for each value of m. In order to verify the accuracy of the model of $\bar{t}_c(k)$ defined by (6.11), we calculate the root mean squared error (RMSE) of their mean values for each m, and for both, long and wide trees, and then normalized it in terms of the average exploration time. As explained before, the simulations consider trees of increasing size from $m = 10$ to $m = 150$ in increments of 10 edges. The results are shown in Fig. 6.12 and Fig. 6.13.

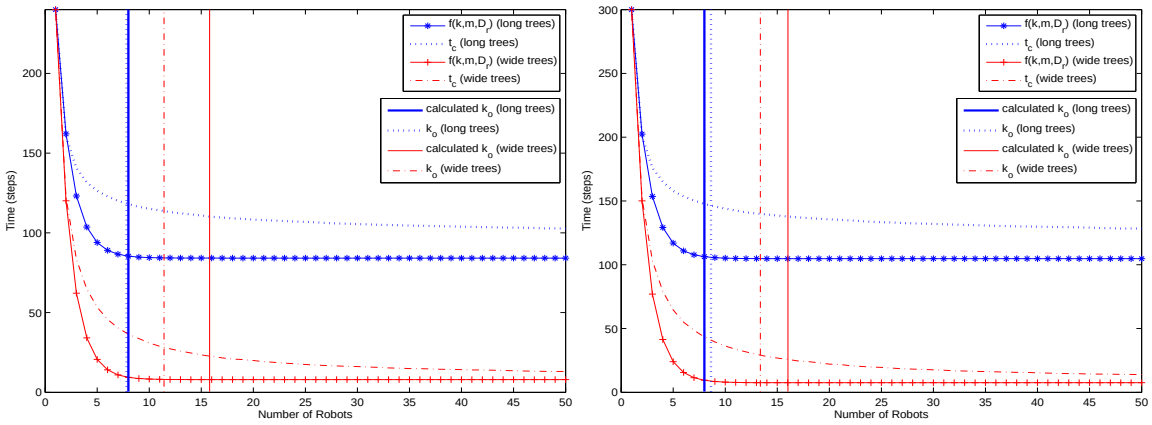
The figures allow us to observe that our model of $\bar{t}_c$, although not perfect, follows closely the trend of the average exploration time, independently if the configuration of the tree is long or wide. Particularly for wide trees, the convergence value of (6.11) lies very close to the actual convergence value of $\bar{t}_c$. The error plots, show us a very interesting result which is that the error of the model ($f(k, m, D_r)$) with respect to the average exploration time increases linearly with the size of the tree. In fact we can argue that the error of the model can be predicted and eventually eliminated, since we could use the results of Fig. 6.12 to obtain an expression for the error. Being beyond the objectives of this work, which is to find the results in an analytical way and not heuristically, we leave this task for future consideration.



(a) $m = 10$ (b) $m = 40$



(c) $m = 70$ (d) $m = 100$



(e) $m = 120$ (f) $m = 150$

Figure 6.11: Comparison between $\bar{t}_c(k)$ and $f(k, m, D_r)$, and between k_0 and the calculated k_0, for long and wide trees with fixed number of edges m. The trees are explored by an increasing number of robots from $k = 1$ to $k = 50$.

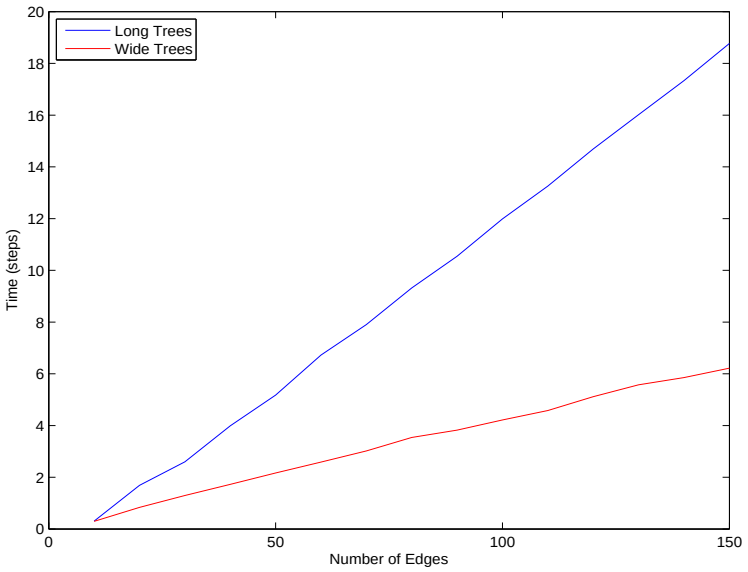


Figure 6.12: Root mean squared error (RMSE) between $\bar{t}_c(k)$ and $f(k, m, D_r)$ for long and wide trees

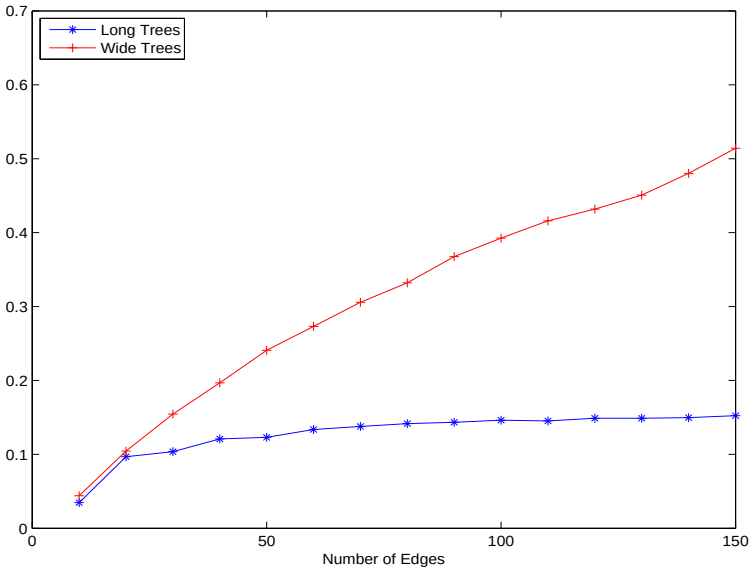


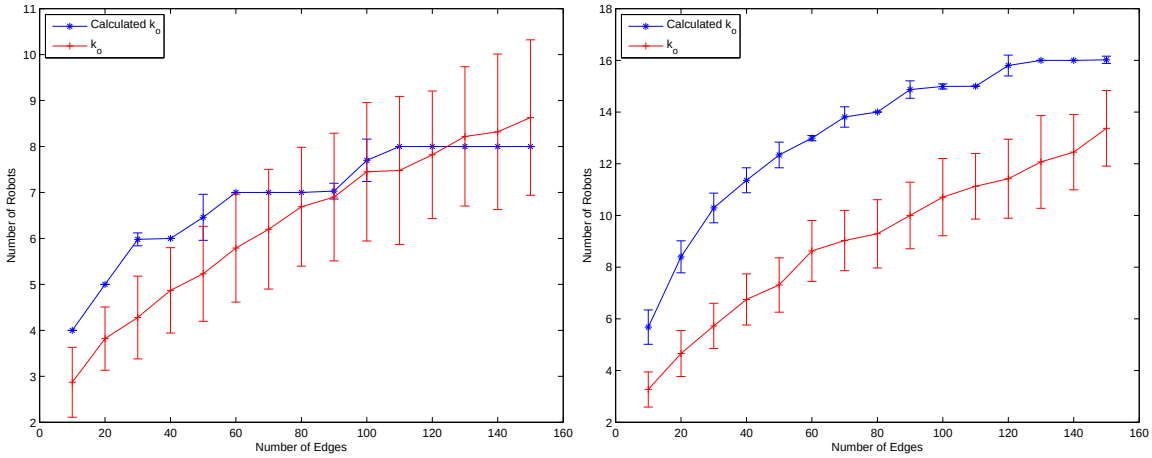
Figure 6.13: Normalized root mean squared error (NRMSE) between $\bar{t}_c(k)$ and $f(k, m, D_r)$ for long and wide trees

The plot of the normalized error shows us the proportion upon which the error increases, being prominent the case for wide trees. On this type of trees, although the error does not increases as steeply as the one for long trees, the error is proportionally greater and as a results, its impact on the overall accuracy of the model is far more noticeable. This fact will not be apparent until we analyze the results of the simulations for k_0.

The plots on Fig. 6.14 show the mean value and standard deviation for both, the value of k_0 obtained using (6.15) and the one obtained from the actual exploration process, for long and wide trees of different size ranging from $m = 10$ to $m = 150$. From the plots, it can be observed that our model produces values of k_0 that are close to the actual values of k_0. In these plots, we observe the impact of the error for our model of $\bar{t}_c$ into the calculation of the number of robots limit (k_0), since the latter was derived using the former. For instance, on wide trees the error of the models make that the estimated value lies way on top of the actual value of k_0. This is not in itself a problem since we can consider that our model acts as an upper bound for the limit number of robots (at least for wide trees).

In order to corroborate this observations, we calculate the root mean squared error (RMSE) of the mean values for each m, and for both, long and wide trees, and then normalized it in terms of the average value of k_0 obtained from the actual exploration process. The results are shown in Fig. 6.15 and Fig. 6.16.

The plot for the error on the estimation of k_0 for wide trees corroborates our statement that our model serves as an upper bound on this type of trees. Again, this is not an undesired results because it still allows us to constraint the number of robots needed to perform the exploration of an environment of a particular size, although no as precisely as with a model with less error. We should highlight the case for long



(a) long trees (b) wide trees

Figure 6.14: Comparison between the mean values of k_0 and the calculated k_0 using (6.19) for long and wide trees

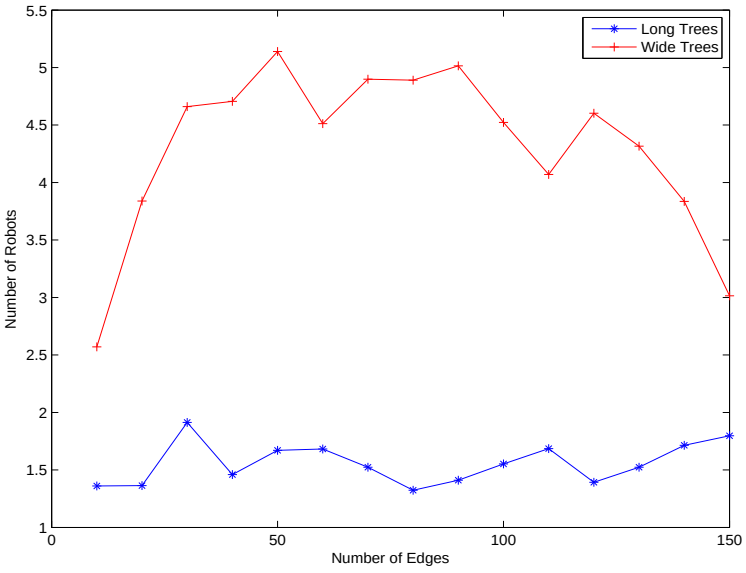


Figure 6.15: Root mean squared error (RMSE) between k_0 and the calculated k_0 for long and wide trees

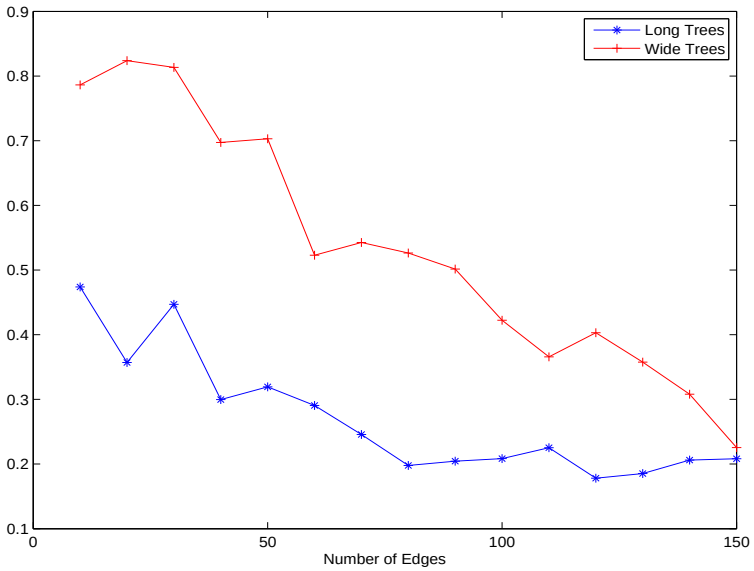


Figure 6.16: Normalized root mean squared error (NRMSE) between k_0 and the calculated k_0 for long and wide trees

trees, where the error is relatively constant for trees of different size.

From the normalized error plots it is important to point out how the error reduces with the size of the tree for both cases (long and wide trees). This is an interesting fact because for trees with large size (either long or wide trees), where determine the limit number of robots could be expected to be more difficult, our model will produce values of k_0 that are more precise to the actual needs of the environment.

Chapter 7

Concluding Remarks

7.1 Conclusions

In this thesis, we have proposed two different algorithms for multi-robot exploration of unknown environments: multi-robot depth first search (MR-DFS) and flooding algorithm.

In chapter 4 we presented the MR-DFS algorithm, which is guaranteed to succeed on any graph, and we proved that the algorithm is never worse than classical single-robot DFS for both general graphs and trees. For the latter, we obtained an expression for the upper bound of the exploration time that improves a previous result presented on (Fraigniaud et al., 2006). For two robots on a tree, we proved that the algorithm is optimal. For k robots we showed that the algorithm has optimal dependence on the size of the tree, but not on its root diameter. In this specific graph-exploration scenario, the robots are initially all located at a common starting vertex, they discover the existence of an edge only when they see one end of it, and know where an edge leads only when they have followed it. Vertices that have been visited before are

recognized.

The proposed algorithm needs only a local communication model, where communication happens only between a robot, and a bookkeeping device left at that node, or between robots standing simultaneously at the same node. So the robots are almost completely unaware of the actions of the other robots. The bookkeeping devices are not in contact with each other; they could be replaced by a piece of chalk leaving marks on the possible exits of the rooms. This is a much weaker communication assumption than global shared information; if global shared information is available, no bookkeeping devices are needed.

In addition to our theoretic analysis, several simulations have been performed in order to corroborate the mathematical results previously described. The result of the simulations showed that our analysis on trees produces upper and lower bounds on the exploration time that are close to the actual exploration time of the algorithm, particularly when considering two robots. The simulations also showed that the algorithm effectively reduces the exploration time when the number of robots is increased and that this exploration time is at all times better than when using the single-robot DFS approach. Moreover, it was shown how the performance of the algorithm reaches closer to the optimal exploration time when more robots are used to perform the exploration.

In chapter 5 we proposed a multi-robot exploration algorithm (flooding algorithm) that aims to minimize the time of exploration, and the overall traverse distance by coordinating the movement of the robots in the system. This coordination is performed in a decentralized manner by the robots that relay in information that is stored in active landmarks. The algorithm differs from other works in the literature by the fact that robots are allowed to enter only one at a time and that edges can

be traversed by only one robot in each step. We analyzed theoretically the flooding algorithm and obtained its bounds on the exploration time (upper and lower bound) on trees. We proved that the algorithm is never worse than classical single-robot DFS for trees. We simulated the behavior of the algorithm on trees of different sizes and showed that our theoretical analysis produced tight upper and lower bounds for the exploration time, improving the trivial upper bound given by the single-robot DFS approach. The simulations showed that both, the growth of t_c and the number of robots k needed to perform the exploration are linearly related to the size of the tree.

In chapter 6, we argued that time is only one of the criteria on which the performance of multi-robots exploration of unknown environments can be evaluated on. As such, we defined three metrics of performance for multi-robot algorithms (the total distance overhead μ_{total}, the time efficiency E_t, and the distance efficiency E_d), and then performed simulations that aimed to compare the performance of the flooding algorithm with that of the MR-DFS algorithm and single-robot DFS. The simulation results showed that the flooding algorithm is effective in performing the exploration while reducing the total distance overhead of the system. In contrast, the time efficiency of our algorithm is not better than that of MR-DFS, but this is an expected result since MR-DFS has higher parallelism which translates in a faster exploration time.

The results on the number of robots used to perform the exploration, particularly in wide trees, suggested that the exploration of any unknown environment can be performed by a small number of robots. In long trees this effect is less evident, but it raised the question if there exists a maximum number of robots for which the reduction of exploration time reaches a maximum, and upon which adding more robots will not be significant for the exploration time.

In chapter 6, we studied this maximum number of robots and obtained analytically an expression that resembles the behavior of the average exploration time $\bar{t}_c$ when the exploration is performed by an increasing number of robots. This analysis was performed on the MR-DFS algorithm on trees of different size. The goal of obtaining such an expression is to be able to define the maximum number of robots k_0 (i.e., a limit on the number of robots) that produce the maximum reduction on the exploration time. This is of particular importance because it will allow us to constrain the size of multi-robot systems. The result of this analysis produced two models, one for $\bar{t}_c$ and the other for k_0.

We performed simulations in order to test the fitness of both expressions and obtained that although not perfect, the models produced results that are consistent with the actual values obtained from the exploration process. For the model of $\bar{t}_c$, we observed that the error on the model increases proportionally with the size of the tree for both configurations. This yields the conclusion that it is possible to eliminate the error in the model by simple modeling this error in terms of m, and then include the value into the model. As for the fitness of our model for k_0, the results showed that the model produces a better fit for long trees than for wide trees where the model resembles more an upper bound. Yet, the model allow us to constraint the number of robots needed to perform a significant reduction on the exploration time. We can observe that this value, even for trees of large size and with the conservative approach of $\alpha = 1$, is really small (around 6% of the total number of edges for long trees and around 10% for wide trees). Another interesting fact about the model of k_0 obtained from the simulations is that for trees with large size, where determine the limit number of robots could be expected to be more difficult, the error is reduced significantly, and as such, our model will produces values of k_0 that are more precise

to the actual needs of the environment.

7.2 Recommendations for Future Work

In this thesis we presented two algorithms that represent the complete opposite in the spectrum of the way we can allow robots to move in an indoor environment: MR-DFS allows the movement of robots with no restrictions, while the flooding algorithm constraints to one the number of robots allowed to traverse an edge and to remain static in a vertex at each step of the process. For both algorithms we performed a theoretical analysis obtaining the bound of the exploration time.

Yet, the analysis of these algorithms was only for trees; the next most-important theoretical problem is to provide an analysis for general graphs. No bounds on multi-robot exploration of general graphs in this scenario are known. The bound for trees could be improved, perhaps even giving optimality for further small numbers of robots, and the most-important problem for the practical applicability of this algorithm is to remove the assumption of robot movement in time steps; the real-robot movement is asynchronous, and the algorithm itself makes no assumption on synchronization, i.e., artifact of the analysis.

For the analysis on trees of both algorithms, there exist still some room for improvement. As it was noted in remark 4.3, the upper bound of the MR-DFS algorithm could be improved if in the bound on the total *excess multiplicity* a third parameter (m) is included. For the flooding algorithm, the result for upper bound can be improved if for the exploration process we consider a fixed number of robots. This will yield the inclusion of k as a third parameter on the upper bound.

Throughout this work we have been able to demonstrate that modeling the envi-

ronment as a graph (or a tree) is convenient for analysis purposes. Then, the analysis of multi-robot systems performing the exploration of unknown environments can be improved by closing the gap between the results produced by theoretical analyses and real applications. This can be done by relieving the constraints and assumptions that this type analyses are subjected to, what will result in the need for the modeling of the movement of the multiple robots and their interaction with the structure of the environment they are exploring. That is, there is a need for creating a unified model for both the environment and the movement of the multi-robot system.

List of Publications

The work that I have performed during the years of my doctoral studies has been documented in the following papers:

General Publications

F. Cabrera-Mora, and J. Xiao, "A Flooding Algorithm for Multi-robot Exploration," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, in press. doi:10.1109/TSMCB.2011.2179799.

P. Brass, F. Cabrera-Mora, A. Gasparri and J. Xiao, "Multi-robot tree and graph exploration," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 707–717, August 2011. doi:10.1109/TRO.2011.2121170.

Peer-reviewed Conference Publications

F. Cabrera-Mora, J. Xiao, and P. Brass, "Multi-robot flooding algorithm for the exploration of unknown indoor environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Anchorage, AK, USA, May 2010, pp. 5478–5483. doi:10.1109/ROBOT.2010.5509522.

P. Brass, A. Gasparri, F. Cabrera-Mora, and J. Xiao, "Multi-robot tree and graph exploration," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, May 2009, pp. 2332–2337. doi:10.1109/ROBOT.2009.5152256.

F. Cabrera-Mora and J. Xiao, "Preprocessing technique to signal strength data of wireless sensor network for real-time distance estimation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Pasadena, CA, USA, May 2008, pp. 1537–1542. doi:10.1109/ROBOT.2008.4543420.

F. Cabrera-Mora, J. Xiao, and Y. Sun, "Effects of communication on mobile sensor networks," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, Beijing, China, October 2006, pp. 1452–1457. doi:10.1109/IROS.2006.281970.

Bibliography

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

Anderson, I.: 1989. *A First Course in Combinatorial Mathematics*. New York, NY (USA): Oxford University.

Awerbuch, B., Betke, M., Rivest, R., and Singh, M.: 1995. Piecemeal graph exploration by a mobile robot. In *Proceedings of the 8th Annual ACM Conference on Computational Learning Theory (COLT)*. New York, NY, USA, pp. 321–328.

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

Awerbuch, B. and Kobourov, S.: 1998. Polylogarithmic-overhead piecemeal graph exploration. In *Proceedings of the 11th Annual ACM Conference on Computational Learning Theory (COLT)*. New York, NY, USA, pp. 280–286.

Batalin, M. and Sukhatme, G. S.: 2004. Coverage, exploration and deployment by a mobile robot and communication network. *Telecommunication Systems Journal, Special Issue on Wireless Sensor Networks*, 26(2-4), 181–196.

Batalin, M. and Sukhatme, G. S.: 2007. The design and analysis of an efficient local

algorithm for coverage and exploration based on sensor network deployment. *IEEE Transactions on Robotics*, 23(4), 661–675.

Bender, M., Fernandez, A., Ron, D., Sahai, A., and Vadhan, S.: 1998. The power of a pebble: exploring and mapping directed graphs. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing (STOC)*. Dallas, TX, USA, pp. 269–278.

Bender, M. A., Fernndez, A., Ron, D., Sahai, A., and Vadhan, S.: 2002. The power of a pebble: Exploring and mapping directed graphs. *Information and Computation*, 176(1), 1–21.

Berhault, M., Huang, H., Keskinocak, P., Koenig, S., Elmaghraby, W., Griffin, P., and Kleywegt, A.: 2003. Robot exploration with combinatorial auctions. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*. Las Vegas, NV, USA, pp. 1957–1962.

Blum, A., Raghavan, P., and Schieber, B.: 1991. Navigating in unfamiliar geometric terrain. In *Proceedings of the twenty-third annual ACM symposium on Theory of computin (STOC)*. New York, NY, USA, pp. 494–504.

Blum, A., Raghavan, P., and Schieber, B.: 1997. Navigating in unfamiliar geometric terrain. *SIAM Journal on Computing*, 26(1), 110–137.

Budach, L.: 1975. On the solution of the labyrinth problem for finite automata. *Elektronische Informations-Verarbeitung und Kybernetik EIK*, 11, 661–672.

Burgard, W., Moors, M., Stachniss, C., and Schneider, F.: 2005. Coordinated multi-robot exploration. *IEEE Transactions on Robotics and Automation*, 21, 376–386.

Chen, C.-C. and Koh, K.-M.: 1992. *Principles and techniques in combinatorics*. Singapore, Singapore: World Scientific Publishing Company.

Chin, W.-P. and Ntafos, S.: 1986. Optimum watchman routes. In *Proceedings of the second annual ACM Symposium on Computational Geometry (SCG)*. Yorktown Heights, NY, USA, pp. 24–33.

Cormen, T. H., Stein, C., Rivest, R. L., and Leiserson, C. E.: 2001. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition.

Das, S., Flocchini, P., Kutten, S., Nayak, A., and Santoro, N.: 2007. Map construction of unknown graphs by multiple agents. *Theoretical Computer Science*, 385, 37–48.

Deng, X., Kameda, T., and Papadimitriou, C. H.: 1991. How to learn an unknown environment. In *Proceedings of the IEEE 32nd Annual Symposium on Foundations of Computer Science (FOCS)*. San Juan, Puerto Rico, pp. 298–303.

Deng, X., Kameda, T., and Papadimitriou, C. H.: 1998. How to learn an unknown environment. i: the rectilinear case. *Journal of the ACM*, 45(2), 215–245.

Deng, X. and Papadimitriou, C. H.: 1990. Exploring an unknown graph. In *Proceedings of the IEEE 31st Annual Symposium on Foundations of Computer Science (FOCS)*. St. Louis, MO, USA, pp. 355–361.

Deng, X. and Papadimitriou, C. H.: 1999. Exploring an unknown graph. *Journal of Graph Theory*, 32(3), 265–297.

Dessmark, A. and Pelc, A.: 2002. Optimal graph exploration without good maps. In *Proceedings of the 10th Annual European Symposium on Algorithms (ESA)*. Rome, Italy, pp. 374–386.

Dessmark, A. and Pelc, A.: 2004. Optimal graph exploration without good maps. *Theoretical Computer Science*, 326(1-3), 343–362.

Detrain, C., Deneubourg, J.-L., and Pasteels, J.: 1999. *Information Processing in Social Insects*. Birkhauser Verlag, 1st edition.

Dias, B. and Stentz, A.: 2002. Opportunistic optimization for market-based multi-robot control. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*. Lausanne, Switzerland, pp. 2714–2720.

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

Duncan, C., Kobourov, S., and Kumar, V.: 2006. Optimal constrained graph exploration. *ACM Transactions on Algorithms*, 2(3), 380–402.

Dynia, M., Kutyowski, J., der Heide, F., and Schindelhauer, C.: 2006. Smart robot teams exploring sparse trees. In R. Krlovic and P. Urzyczyn (Eds.), *Mathematical Foundations of Computer Science 2006*, volume 4162 of *Lecture Notes in Computer Science*. pp. 327–338. Springer Berlin / Heidelberg.

Elizondo-Leal, J., Ramírez-Torres, G., and Toscano Pulido, G.: 2008. Multi-robot exploration and mapping using self biddings and stop signals. In A. Gelbukh and E. Morales (Eds.), *MICAI 2008: Advances in Artificial Intelligence*, volume 5317 of *Lecture Notes in Computer Science*. pp. 615–625. Springer Berlin / Heidelberg.

Fleischer, R., Kamphans, T., Klein, R., Langetepe, E., and Trippen, G.: 2008. Competitive online approximation of the optimal search ratio. *SIAM Journal on Computing*, 38(3), 881–898.

Fleischer, R. and Trippen, G.: 2005. Exploring an unknown graph efficiently. In *European Symposium on Algorithms (ESA)*, volume 3669 of *LNCS*, Springer. pp. 11–22.

Fourcassie, V. and Deneubourg, J.-L.: 1994. The dynamics of collective exploration and trail-formation in monomorium pharaonis: experiments and model. *Physiological Entomology*, 19, 291–300.

Fourcassie, V., Dussutour, A., and Deneubourg, J.-L.: 2010. Ant traffic rules. *Journal of Experimental Biology*, 213, 2357–2363.

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

Fraigniaud, P., Ilcinkas, D., and Pelc, A.: 2008. Tree exploration with advice. *Information and Computation*, 206(11), 1276–1287.

Gasieniec, L., Pelc, A., Radzik, T., and Zhang, X.: 2007. Tree exploration with logarithmic memory. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 585–594.

Gasparri, A., Fiorini, F., Di Rocco, M., and Panzieri, S.: 2011. A networked transferable belief model approach for distributed data aggregation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 99, 1–15.

Gerkey, B. P. and Mataric, M. J.: 2002. Sold!: Auction methods for multirobot coordination. *IEEE Transactions on Robotics and Automation*, 18, 758–768.

Graham, R.: 1966. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45, 1563–1581.

Graham, R.: 1969. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, 17(2), 416–429.

Harris, J. M., Hirst, J. L., and Mossinghoff, M. J.: 2008. *Combinatorics and Graph Theory*. New York, NY (USA): Springer.

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

Hoffmann, F., Icking, C., Klein, R., and Kriegel, K.: 2001. The polygon exploration problem. *SIAM Journal on Computing*, 31(2), 577–600.

Hsieh, M. A., Cowley, A., Kumar, V., and Taylor, C. J.: 2008. Maintaining network connectivity and performance in robot teams. *Journal of Field Robotics*, 25(1-2), 111–131.

H.Wang, Jenkin, M., and Dymond, P.: 2008. Enhancing exploration in graph-like worlds. In *Canadian Conference on Computer and Robot Vision*, IEEE. pp. 53–60.

Kwek, S.: 1997. On a simple depth-first search strategy for exploring unknown graphs. In *Workshop on Algorithms and Data Structures (WADS)*, volume 1272 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg. pp. 345–353.

Lucarelli, D. and Wang, I.-J.: 2004. Decentralized synchronization protocols with nearest neighbor communication. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*. Baltimore, MD (USA).

Moravec, H.: 1988. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, 9, 61–74.

Moravec, H. P.: 1989. *Sensor fusion in certainty grids for mobile robots*, pp. 253–276. Springer-Verlag New York, Inc.: New York, NY, USA.

Ntafos, S. and Gewali, L.: 1994. External watchman routes. *The Visual Computer*, 10, 474–483.

O'Rourke, J.: 1987. *Art Gallery Theorems and Algorithms*. Oxford University Press.

Panaite, P. and Pelc, A.: 1999. Exploring unknown undirected graphs. *Journal of algorithms*, 33, 281–295.

Panaite, P. and Pelc, A.: 2000. Impact of topographic information on graph exploration efficiency. *Networks*, 36(2), 96–103.

Papadimitriou, C. H. and Yannakakis, M.: 1991. Shortest paths without a map. *Theoretical Computer Science*, 84(1), 127–150.

Pearl, J.: 1984. *Heuristics: intelligent search strategies for computer problem solving*. Reading, MA (USA): Addison-Wesley.

Ren, W. and Sorensen, N.: 2008. Distributed coordination architecture for multi-robot formation control. *Robotics and Autonomous Systems*, 56(4), 324–333.

Rooker, M. and Birk, A.: 2007. Multi-robot exploration under the constraints of wireless networking. *Control Engineering Practice*, 15(4), 435–445.

Sheng, W., Yang, Q., Tan, J., and Xi, N.: 2006. Distributed multi-robot coordination in area exploration. *Robotics and Autonomous Systems*, 54(12), 945–955.

Simmons, R. G., Apfelbaum, D., Burgard, W., Fox, D., Moors, M., Thrun, S., and Younes, H. L. S.: 2000. Coordination for multi-robot exploration and mapping.

In *Proceedings of the AAAI National Conference on Artificial Intelligence*. Austin, TX (USA), pp. 852–858.

Vaughan, R., Stoy, K., Sukhatme, G. S., and Mataric, M. J.: 2002. Lost: Localization-space trails for robot teams. *IEEE Transactions on Robotics and Automation*, 18(3), 796–812.

Wu, L., Garcia, M. A., Puig, D., and Sole, A.: 2007. Voronoi-based space partitioning for coordinated multi-robot exploration. *Journal of Physical Agents*, 1, 37–44.

Yamauchi, B.: 1997. Frontier-based approach for autonomous exploration. In *IEEE International Symposium on Computational Intelligence, Robotics and Automation*. Monterrey, CA (USA), pp. 146–151.

Yamauchi, B.: 1998. Frontier-based exploration using multiple robots. In *Second International Conference on Autonomous Agents*. Minneapolis, MN (USA), pp. 47–53.

Zhang, H.-T., Zhai, C., and Chen, Z.: 2011. A general alignment repulsion algorithm for flocking of multi-agent systems. *IEEE Transactions on Automatic Control*, 56(2), 430–435.

Zlot, R., Stentz, A., Dias, M. B., and Thayer, S.: 2002. Multi-robot exploration controlled by a market economy. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Washington, DC (USA), pp. 3016–3023.