

Heterogeneous Multi-Robot Search Algorithms

**Thesis submitted in partial fulfillment
of the requirements for the degree of
“DOCTOR OF PHILOSOPHY”**

by

Shahar

Sarid

Submitted to the Senate of Ben-Gurion University of the Negev

2011

Beer-Sheva

Heterogeneous Multi-Robot Search Algorithms

Thesis submitted in partial fulfillment
of the requirements for the degree of
“DOCTOR OF PHILOSOPHY”

by

Shahar

Sarid

Submitted to the Senate of Ben-Gurion University of the Negev

Approved by the advisors

Dr. Amir Shapiro Amir Shapiro

Prof. Yael Edan Yael Edan

Approved by the Dean of the Kreitman School of Advanced Graduate Studies _____

2011

Beer-Sheva

This work was carried out under the supervision of

Dr. Amir Shapiro

and

Prof. Yael Edan

In the Department of Mechanical Engineering

Faculty of Engineering Sciences

To my family

Acknowledgments

I am excited to have arrived at the moment where I can finally thank all the people who assisted and supported me along the long path toward completing my doctoral dissertation.

First, I would like to thank my advisors, **Amir Shapiro** and **Yael Edan**. My work with **Amir** goes back seven years, and I am grateful to him for his optimism and solid view of reality, for encouraging me when I needed it, for being a friend who was always there for me, for his many ideas for research, and for relieving my worries. Working with Amir was fruitful and rewarding. He was an ideal advisor who I would wish for any doctoral student. To **Yael**, I would like to express my deep gratitude for investing so much time in me, for pushing me forward, and for knowing exactly how to do so. Her dedication and professional assistance were much appreciated.

I thank my students, **Lior Barshan** and **Chen Futerman** who did outstanding work in their final project. I enjoyed working with them during the last year of my research and the last year of their degree on implementing the algorithms in simulations and experiments. Thanks to **Lior Salem**, and **Guy Alon** for their assistance in building the testbed and running the experiments. Thanks to **Yam Geva** and **Hagai Balshai** for their support and help in running the experiments with the mobile robots in the Robot Motion Lab.

I would also like to thank **Dr. Yisrael Parmet** for his help in statistically analyzing the experiments results. Thanks to **Yoav Gabriely**, and **Prof. Elon Rimon** for their helpful contribution to the research. I would like to thank **Prof. Reuven Segev** and **Prof. Ephraim Korach** for being great lecturers, and for giving me their attention and assistance when I needed it. I would like to extend thanks as well to the members of my Ph.D. committee, **Prof. Moshe Kaspi**, and **Prof. Gal A. Kaminka** for their comments, which helped improve my research.

I would like to thank **Hannah Komy Ofir** for her technical editing, which improved the outcome.

Many thanks to **Shahar Berger**, **Asaf Formoza**, **Bella Gurevich**, **Dan Shaine**, **Hanoch Efraim**, **Eddie Zisser**, **Alon Ohev-Zion**, and **Alon Kapoa** who all made my years in the Department of Mechanical Engineering enjoyable and enriching.

When I began to summarize and consider all my years of study towards my Ph.D., I realized that they are almost equal to the age of my oldest child. This period was filled with events not only in my professional life, but also in my personal life. My dear grandfathers, **Sholi** and **Yashka**, ז"ל, passed away during these years. Both were Holocaust survivors, who brought me up with a lot of love, and I miss them very much. I thank my grandmothers, **Eva** and **Frida**, who raised me with much tenderness, but also showed me how bold one can be when necessary. I love and admire them both. I thank my sister **Noga** and my brother **Tal** who are loving, fun, and caring siblings. A special thanks goes to my parents, who raised me with so much love and effort, and always supported my choices. My father, **Israel** taught me that there is nothing that I can not do, and my mother **Rochale** showed me how to do everything I want. You are the greatest family one can have. To **Amalia** and **Micha**, my parents-in-law, many thanks for sensitively and professionally helping me throughout the long period of my studies. To my son, **Uri** and to my daughter, **Ziv** who I love the most in the whole world, thank you for entering my life, with all the joy you have brought. And last, but definitely not least, my wife, the love of my life, **Yael**, who shared this journey with me, a very special thanks.

Contents

List of Tables	IX
List of Figures	XI
Abstract	XV
1 Introduction	1
1.1 Problem Description	1
1.2 Objectives	3
1.3 Research Significance	3
1.4 Contributions and Innovations	4
2 Literature Review	7
2.1 Single Robot Coverage Algorithms	9
2.2 Multi-Robot Coverage Algorithms	13
2.3 MRSAM algorithm	15
2.4 MRBUG algorithm	18
3 Methodology	21
3.1 Overview	21
3.2 Algorithms	21
3.2.1 Heterogeneity	21
3.2.2 Completeness	21
3.2.3 Decentralization, Communication, and Robustness	22
3.3 Performance Measures	22
3.4 Simulation analysis	22
3.4.1 Overview	22
3.4.2 Parameters evaluated	22
3.5 Initial run	25
3.5.1 Performance Measures	25
3.6 Experiments	26
3.6.1 Overview	26
3.6.2 Mobile Platform	26
3.6.3 The environments	27
3.6.4 Localization system	27
3.6.5 The software	27

3.6.6	Parameters	28
3.6.7	Performance Measure	35
3.6.8	Validation	35
4	HMRSTM Algorithm	37
4.1	Introduction	37
4.2	The Problem's Definitions	37
4.3	Time competitive complexity lower bound	38
4.4	The HMRSTM Algorithm	39
4.5	Analytical performance analysis	44
5	HMRBUG Algorithm	49
5.1	Introduction	49
5.2	A Lower Bound for a Known Target	49
5.3	HMRBUG Algorithm	51
5.3.1	<i>PBUG1</i> Motion Planning Algorithm for a Pair of Robots [53]	52
5.3.2	Target Reachability Test	53
5.3.3	<i>HMRBUG</i> Algorithm for a Heterogeneous Group of Robots	53
5.4	HMRBUG Upper Bound Analysis	56
6	Simulations	63
6.1	HMRSTM simulations	63
6.2	HMRBUG simulations	68
7	Experiments	77
7.1	HMRSTM experiments	77
7.1.1	Experiment initialization	77
7.1.2	Experiment example	78
7.1.3	Experiments results analysis	80
7.1.4	Statistical analysis	83
7.2	HMRBUG experiments	85
7.2.1	Experiment example	85
7.2.2	Experiments results analysis	87
7.2.3	Statistical analysis	89
8	Conclusions and Future Research	91
8.1	Conclusions	91
8.2	Future Work	91
	Bibliography	100
	Appendices	103

A	Experimental Results	103
A.1	HMRSTM experimental Results	103
A.2	HMRSTM simulation of experimental results	112
A.3	HMRSTM simulation-experiment comparison	116
A.4	HMRBUG experimental Results	119
A.5	HMRBUG simulation-experiment results and comparison	121
B	DVD content	123
B.1	DVD 1	123
B.1.1	Software	123
B.1.2	Videos	123
B.2	DVD 2	123
B.3	DVD 3	123

List of Tables

2.1	Exploration works	10
2.2	Approximate Cellular Decomposition Coverage works	11
2.3	Exact Cellular Decomposition Coverage works	12
3.1	Simulation Environments	25
3.2	Simulation Parameters	25
3.3	Threads timing	28
3.4	Experiments Parameters	29
5.1	HMRBUG first ellipses search times	59
7.1	<i>HMRSTM</i> experiment configurations	77
7.2	Tests of Between-Subject Effects for HMRSTM experiments	84
7.3	Tests of Between-Subject Effects for Target 3, HMRSTM experiments	84
7.4	<i>HMRBUG</i> experiment configurations	85
7.5	Tests of Between-Subject Effects for HMRBUG experiments	89
A.1	HMRSTM experiments results - "cave" Target 1	103
A.2	HMRSTM experiments results - "cave" Target 2	104
A.3	HMRSTM experiments results - "cave" Target 3	105
A.4	HMRSTM experiments results - "free" Target 1	106
A.5	HMRSTM experiments results - "free" Target 2	107
A.6	HMRSTM experiments results - "free" Target 3	108
A.7	HMRSTM experiments results - "libr" Target 1	109
A.8	HMRSTM experiments results - "libr" Target 2	110
A.9	HMRSTM experiments results - "libr" Target 3	111
A.10	HMRSTM simulation of experiment results - "cave"	113
A.11	HMRSTM simulation of experiment results - "free"	114
A.12	HMRSTM simulation of experiment results - "libr"	115
A.13	HMRSTM simulation-experiment comparison - "cave"	116
A.14	HMRSTM simulation-experiment comparison - "free"	117
A.15	HMRSTM simulation-experiment comparison - "libr"	118
A.16	HMRBUG experiments results - "cave"	119
A.17	HMRBUG experiments results - "free"	120
A.18	HMRBUG experiments results - "library"	120
A.19	HMRBUG simulation-experiment comparison	121

List of Figures

2.1	Execution examples of (a) STC, (b) D-STC	13
2.2	Execution examples of MFC.	14
2.3	Boustrophedon extension to multi-robot by Rekleitis et al.	14
2.4	Multi- Robot demining by Rekleitis et al.	15
2.5	Boustrophedon extension to multi-robot by Kong et al.	15
2.6	The first and final steps of <i>MRSAM</i> execution example	17
2.7	Execution example of <i>MRBUG</i>	20
3.1	<i>HMRSTM</i> Cave environment	24
3.2	<i>HMRSTM</i> 10 targets lopt	24
3.3	<i>HMRBUG</i> 10 Start-Target positions	24
3.4	<i>HMRSTM</i> targets 1,2 in "free" and "cave" environments	30
3.5	<i>HMRSTM</i> target 3 in "library" environment	30
3.6	<i>HMRSTM</i> target 1 in "free" environment	31
3.7	<i>HMRSTM</i> target 2 in "cave" environment	31
3.8	<i>HMRSTM</i> target 3 in "library" environment	32
3.9	<i>HMRBUG</i> "cave" and "lib" environments	32
3.10	<i>HMRBUG</i> S-T 1 in "cave" environment	33
3.11	<i>HMRBUG</i> Start-Target 1,2	33
3.12	<i>HMRBUG</i> Start-Target 2 in "lib" environment	34
3.13	<i>HMRSTM</i> and <i>HMRBUG</i> validation	36
4.1	Difficult environment for the lower bound proof of unknown target position	38
4.2	<i>HMRSTM</i> execution example 1	41
4.3	<i>HMRSTM</i> execution example 2	42
4.4	<i>HMRSTM</i> Pseudocode	43
5.1	Difficult environment for the lower bound proof of known target position	50
5.2	<i>PBUG1</i> execution example	52
5.3	Execution example of <i>HMRBUG</i>	54
5.4	<i>HMRBUG</i> Pseudocode	55
5.5	<i>HMRBUG</i> execution example	56
6.1	<i>HMRSTM</i> simulation example part 1	64
6.2	<i>HMRSTM</i> simulation example part 2	64
6.3	<i>HMRSTM</i> average simulation results for environment and beta	65

6.4	HMRSTM average simulation results for no. of robots and beta	65
6.5	HMRSTM normalized simulation results for no. of robots and beta	66
6.6	HMRSTM simulation initial search radii	66
6.7	HMRSTM finding cluster of targets #90, in "free" environment	67
6.8	HMRSTM normalized simulation results for no. of robots and beta	68
6.9	HMRSTM simulation and analytical upper bound path length	68
6.10	HMRBUG simulation example part 1	70
6.11	HMRBUG simulation example part 2	70
6.12	HMRBUG average simulation results for environment and beta	71
6.13	HMRBUG average simulation results for no. of robots and beta	71
6.14	HMRBUG normalized simulation results for no. of robots and beta	72
6.15	HMRBUG simulation results for initial ellipse and beta, S-T#4, library	72
6.16	HMRBUG simulation results for initial ellipse and beta, S-T#6, library	72
6.17	HMRBUG simulation results for initial ellipse and beta, S-T#6, cave	73
6.18	HMRBUG simulation results for initial ellipse and beta	73
6.19	HMRBUG simulation vs. analytical upper bound path length	74
6.20	HMRBUG ratio between analytical upper bound and maximal simulation path length	74
7.1	HMRSTM main application	78
7.2	HMRSTM simulation and experiment example 1	79
7.3	HMRSTM simulation and experiment example 2	79
7.4	HMRSTM simulation and experiment example 3	80
7.5	HMRSTM simulation and experiment example 4	80
7.6	HMRSTM time to find target 1 with std. Error	81
7.7	HMRSTM time to find target 2 with std. Error	82
7.8	HMRSTM time to find target 3 with std. Error	82
7.9	HMRSTM measured and maximal expected path - target 3	83
7.10	HMRSTM experiments' statistical analysis	84
7.11	HMRBUG simulation and experiment example 1	85
7.12	HMRBUG simulation and experiment example 2	86
7.13	HMRBUG simulation and experiment example 3	86
7.14	HMRBUG time to reach all the targets in all environments	87
7.15	HMRBUG time to reach the targets in "library"	88
7.16	HMRBUG time to reach the targets in "cave"	88
7.17	HMRBUG measured and maximal expected path - target 3	89
7.18	HMRBUG experiments statistical analysis	90

Abstract

This thesis focuses on two on-line motion planning problems where a group of mobile robots with heterogeneous velocity must reach a target located in an a priori unknown and unbounded environment. In the first problem, the target position is unknown and should be found by the robots, while in the second problem the target position is known and a path to it should be found. This thesis focuses on optimizing the task cost in terms of motion time and describes two new on-line navigation algorithms: *Heterogeneous Multi-Robot Search Time Multiplication*, *HMRSTM*, and *Heterogeneous Multi-Robot BUG*, *HMRBUG*, which extends Lumelsky's famous BUG algorithm.

The performance of an on-line algorithm is usually expressed in terms of *competitiveness*, the constant ratio between the on-line and the optimal off-line solutions. Specifically, *competitiveness*, is defined as the ratio between the lengths of the actual path made by the robot that reached the target and the shortest path to the target. We use *generalized time competitiveness*, i.e., the solution is the time to reach the target, and the ratio is not necessarily constant, but could be any function. Classification of a motion planning task in the sense of performance is achieved by finding the upper and lower bounds on the competitiveness of all algorithms solving that task. If the two bounds belong to the same functional class, this is the *competitive complexity class* of the task. We find the two bounds for the aforementioned common on-line motion planning problems, and classify them into competitive classes. It is shown that in general any on-line motion planning algorithm that tries to solve these problems must have at least a quadratic competitive performance. This defines the lower bound of the problems.

We proved that both *HMRSTM*, and *HMRBUG* algorithms have quadratic upper bounds, which prove that the problems they solve have quadratic upper bounds. Thus, it is shown that navigation in an unknown and unbounded environment by a group of robots belongs to a quadratic time competitive class. *HMRSTM* and *HMRBUG* have a quadratic competitive performance and thus have optimal competitiveness.

The algorithms' average case performance is evaluated in simulations. The average performance is much better than the worst-case analytical upper bounds of the algorithms. It is evident that the performance of heterogeneous groups is better than the performance of homogeneous groups. Moreover, it is evident that more heterogeneous velocity distributions perform better. Simulations were validated by experiments in real robots. The results were within the expected bounds. Therefore, the *HMRSTM* and *HMRBUG* algorithms are not only optimal in their upper bounds, but also perform very well in average-case real-world scenarios.

Keywords:

Multi-Robot, Heterogeneous, Motion Planning, On-Line Algorithms, Competitive Complexity, Competitive Complexity Class, Decentralized systems, Navigation, Area Coverage.

This thesis is in part based on the following publications:

Chapters in collective volumes

1. S. Sarid, A. Shapiro, "*H-MRSTM: Competitive Search Algorithm for Heterogeneous Group of Robots*", Emerging Technologies, Robotics and Control Systems, Third edition, pages. 209-216, Editor: Salvatore Pennacchio, International Society for Advanced Research 2009.

Refereed articles

2. S. Sarid, A. Shapiro, "*H-MRSTM: A Competitive Search Algorithm for Heterogeneous Group of Robots*", International Journal of Factory Automation, Robotics and Soft Computing, Issue 3: 51-58, July 2009.
3. S. Sarid, A. Shapiro, "*Classifying the Multi Robot Path Finding Problem into a Quadratic Competitive Complexity Class*", Springer-Verlag Dordrecht publishers, Annals of Mathematics and Artificial Intelligence, Volume 52 (2-4):169-203, April 2008.

Peer reviewed conference papers in conference proceedings

4. S. Sarid, A. Shapiro, E. Rimon, Y. Edan, "*Classifying the Heterogeneous Multi Robot Online Search Problem into Quadratic Time Competitive Complexity Class*", IEEE International Conference on Robotics & Automation (ICRA 2011), May 9-13, 2011, Shanghai, China.
5. S. Sarid, A. Shapiro, "*A Time Competitive Heterogeneous Multi Robot Path Finding Algorithm*", IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS 2010), pages: 4590-4595, October 18-22, 2010, Taipei, Taiwan.
6. S. Sarid, A. Shapiro, Y. Gabriely, "*MRBUG: A Competitive Multi Robot Path Finding Algorithm*", IEEE International Conference on Robotics & Automation (ICRA 2007), pages: 877-882, April 2007, Rome, Italy.
7. S. Sarid, A. Shapiro, Y. Gabriely, "*MRSAM: A Quadratically Competitive Multi-Robot Online Navigation Algorithm*", IEEE International Conference on Robotics & Automation (ICRA 2006), pages: 2699-2704, May 2006, Orlando, FL, USA.

Conference papers in conference proceedings

8. S. Sarid, A. Shapiro, Y. Edan. *Algorithm for finding a path to a target with heterogeneous group of robots*. The 3rd Israeli Conference on Robotics, 10-11 November, 2010, Herzlia, Israel.
9. S. Sarid, A. Shapiro, Y. Edan. "*H-MRSTM: Heterogeneous multi robot time competitive motion planning algorithm, performance analysis and simulations*". The first AUVSI-IEEE Israel Joint Conference, October 6, 2010, Ben-Gurion University, Beer-Sheva, Israel.
10. S. Sarid, A. Shapiro, Y. Edan, "*Algorithm and Simulations for Finding a Target With Robots With Different Velocities*", The 31st Israeli Conference on Mechanical Engineering, Tel-Aviv, Israel, 2-3 June 2010.
11. S. Sarid, A. Shapiro, Y. Gabriely, "*Online Motion Planning Algorithm for a Group of Robots Obtaining a Common Goal*", Israeli Conference of Robotics (ICR), June 2006, Tel Aviv, Israel.

Workshops

13. S. Sarid, "*Online Motion Planning for Multiple Robots Obtaining a Common Goal*", GSC 2006, Graduate Students in Control, Weizmann Institute of Science, April 3, 2006.

Published scientific reports and technical papers

14. S. Sarid, A. Shapiro, Y. Gabriely, "*MRSAM: A Quadratically Competitive Multi-Robot Online Navigation Algorithm*", Dept. of Mechanical Engineering, Ben Gurion University of the Negev, April 2006. <http://www.bgu.ac.il/~ashapiro>

Chapter 1. Introduction

1.1 Problem Description

Mobile robots play a significant role in many domains, among them military and police applications (e.g., demining, armed patrol vehicles, surveillance, bomb disarming), industry (e.g., conveying and handling materials), civil tasks (e.g., search and rescue missions, agriculture), space (e.g., planetary exploration, and material acquisition on distant planets), and service (e.g., cleaning households/pools/grass/others).

Robot motion planning is a fundamental task [41] in mobile robotic operations. The goal of motion planning is to provide the robot with a trajectory to follow in order to accomplish its mission. The trajectory is usually a set of output commands such as direction of movement, distance, and speed, based on input variables such as current position. Many single robot motion planning algorithms exist ([18, 17, 5, 19, 20, 21, 43, 41, 47, 50, 51, 12, 10, 7, 13])

The basic problem of motion planning [41] is to determine which path a robot starting from a specific location should follow in order to reach a known target point in a known environment. The solution includes a set of coordinates the robot should follow, including orientation, velocity, and acceleration. The path planning problem depends on the defined objective (time, length), usually implying a totally different strategy for the solution.

Mobile robot motion tasks that involve target finding include exploration, mapping, coverage, and box pushing. In many real-world situations, the environment's geometry is unknown, for example in disaster areas or space applications, or dynamic, e.g., in populated environments (for example, in offices, hospitals, and factories). In such cases, finding a known target [11] implies finding a path to the target. When no *a priori* knowledge is available, exploration tasks are useful for obtaining information about the environment, and are usually accompanied by mapping. Such tasks require world modeling requisites and communication capabilities. Moreover, the use of advanced sensors such as vision cameras and laser scanners greatly improves the performance of exploration algorithms, but may prove inefficient in worst-case scenarios such as mazes, and congested environments, e.g., disaster areas, since in such cases the multiplicity of obstacles obstructs the long-range sensors and they become as effective as the touch sensors. ***The algorithms developed in this thesis assume limited range sensors, thus they can operate in worst-case scenarios without restrictions.***

When the environment's geometry and the target position are known in advance, the solution is denoted *off-line*, since all motions can be preprocessed prior to the actual execution of the mission. *On-line* solutions are defined when no information about the environment is known in advance. The next step is computed while the robot moves and according to the information it receives from its sensors. ***This thesis focuses on on-line solutions.***

In order to follow a path, a robot must have some information regarding its position and orientation, which can be relative to earth, or to any frame (e.g., a starting point). When no outer source can define the robot's position, the problem is defined as self-localization. Self-localization can be solved using several approaches, including odometry, GPS, *RFID* tags ([68, 16]), vision, and *Line of Sight*

(LOS) ([60, 4]). Odometry accumulates errors and GPS accuracy is insufficient for small-scale robots and cannot be used indoors. The main drawback of the tagging and marking algorithms is the extra equipment they need to hold and deploy the tag devices, which adds weight and physical size to the robots. ***The algorithms developed in this thesis do not rely on communication or on vision sensors. Perfect localization achieved through off-the-shelf solutions is assumed.***

In area coverage missions, the covering robot must pass a tool over all points of a defined area. Exploration missions in which the target position is unknown and can only be identified by the robot upon arrival may be considered partial area coverage problems. Different coverage algorithms exist ([18, 17, 40, 26, 48, 37, 34, 35, 65, 2, 39, 67, 1, 27, 23, 25, 3, 12, 8, 49, 14, 53, 54, 55, 59]). A famous single robot coverage algorithm is the Spanning Tree Covering, (*STC*) [17] algorithm. It is a grid-based coverage algorithm that constructs a spanning tree and deploys the robot to circumnavigate the tree, thus covering the whole environment. *STC* has been proved to be complete and optimal in the sense that the area will be covered only one time. *STC* is designed to work in bounded environments and can be applied either off-line or on-line.

Performance of grid-based coverage algorithms is commonly derived or compared with *STC*. Hazon et al. [26] introduced a *STC* based multi-robot algorithm that focuses on non-redundancy of the area covered by the robots. Thus, the covering time is not optimal and its performance may be reduced to that of *STC* in the worst case. Agmon et al. [1] introduced a distributed construction of a spanning tree while considering different initial locations of the robots. Zheng et al. [67] rely on *STC* as well, and introduced a multi-robot algorithm that uses heuristics to construct partially overlapping trees off-line. In the notable early paper [37], Kurabayashi et al. present an off-line sweeping algorithm for homogeneous multi-robots, which, unlike the above-mentioned algorithms, uses exact cell decomposition. Though it seems complete and efficient, proofs are not provided and it is only shown through simulations and experiments. The algorithms are off-line and cannot be used in an on-line manner without major revisions. Hazon et al. [27] extended their algorithm and created an on-line version of it. Like their previous off-line algorithm, the on-line version focuses on non-redundancy of the covered area, and thus loses optimality in search time. ***In this thesis we present on-line algorithms that optimize the search time.***

Several coverage algorithms are derived from the Boustrophedon decomposition [14], an exact cellular decomposition developed for a single robot. The cells are created by the natural structure of the obstacles in the environment, and each cell is covered by simple back and forth vertical movements. Rekleitis et al. [48] extended the Boustrophedon decomposition to multiple robots, using two types of robots. This solution relies on line-of-sight communication between the explorers and within the teams. In a later study, the Rekleitis solution et al. [49] demands scattering the robots evenly along the edge of the field to be covered and combines an auction mechanism to divide the uncovered area after the initial run. Kong et al. [35] propose a solution that is based on the Boustrophedon decomposition, but does not require line-of-sight between the robots; instead, communication is assumed to be available without restrictions. All of the above-mentioned algorithms are restricted to operating within a bounded environment. ***The algorithms developed in this thesis operate optimally in unbounded environments.***

Multi-robot algorithms can have improved robustness, performance, and efficiency [26]. Some tasks require the robots in a group to have different attributes, such as different velocities, manipulating

capabilities, sensors, or sizes. In some cases, robots are different due to assembly of different types or models, or due to different make quality. The **objective** of this research is to design motion planning algorithms for a group of velocity-heterogeneous robots (which must reach targets whose positions are known *a priori* and targets whose positions are unknown *a priori*).

Research on heterogeneous multi-robot systems focuses on the cooperation of different robots [24]. Coordination is composed of formation [46, 32], task allocation [62, 30, 64], and integration [33, 63]. Controlling swarms of mobile robots [61, 6, 8] is also considered part of this category. Such works focus on generality rather than specifically solving an exploration or a coverage problem. Other heterogeneous multi-robot research directions include systems with human-robot interaction [31, 66, 9], which rely on human operators to partially or fully guide the robots during their missions. Sensor networks are not originally related to robotics, yet sensor reallocation [38], sensor repair by mobile robots [44], and mobile sensor agents [44] add the capability of movement to the sensors and therefore transform such problems into multi-robot and sometimes to heterogeneous multi-robot motion planning problems. In the aforementioned works, the sensors in the network are treated as multiple robots or as multiple targets. Sensors in such networks must have adequate ranges and specific positions in order to completely cover the area they are scattered in, and must form a network. Thus, they must have communication capabilities. ***In this thesis algorithms utilizing a heterogeneous group of robots with different velocities and optimizing the search time were developed.***

This study explores the problem of ***a group of heterogeneous robots starting from a common location finding a target whose position is unknown, in an unknown a priori, unbounded environment.*** This problem has not previously been treated as a whole. The robots are totally autonomous and their onboard setup is composed of short-range, e.g., tactile sensors. Communication between the robots is needed only at the beginning and the end of the execution, permitting implementation with a decentralized system. A deterministic, complete, robust, and optimal solution is pursued.

1.2 Objectives

The research objective is to develop optimal, complete, and deterministic target finding and path finding algorithms for a heterogeneous group of robots operating in unknown and unbounded environments. A robust solution for a decentralized system is developed. Specific objectives are to:

- Find lower bounds of the algorithms
- Analyze worst-case performance
- Analyze time competitive complexity
- Prove optimality
- Evaluate the effect of different parameters on the average-case performance of the algorithms
- Test the algorithms in experiments with real robots

1.3 Research Significance

Target-finding missions are important in many applications. When the target position is unknown to the searcher, the mission may be considered as a partial area coverage problem. When the area to be searched is dangerous to human activity, e.g., in demining, contaminated area cleaning, and

search and rescue missions, robots can be applied in order to avoid risking human lives. Additional advantages include the efficiency achieved by machines, the ability to work long hours, and the relief from tedious work.

The use of a group of robots can yield better performance and reliability than a single robot. A group of robots can find the target faster, and the robots can replace one another in case of malfunction and assist each other to perform better in positioning and in reducing redundant movements. Many tasks can benefit from the use of heterogeneous robots. For example, smaller robots can enter narrow passages and cover confined crevices, while larger ones may carry heavier loads, such as acquired samples and complementary systems such as positioning, communication and sensory systems. Distributing different sensors between the robots may reduce the individual robot weight and thus result in a heterogeneous group that is lighter and possibly has better maneuverability. Furthermore, the different sensors can provide increased functionality, since different sensors have different capabilities.

A heterogeneous group of robots can be formed from early versions of robots and newly acquired ones. Developing an efficient algorithm to handle a group of heterogeneous robots in tasks such as target finding and area coverage is of great importance.

1.4 Contributions and Innovations

Two optimal motion planning algorithms for a group of heterogeneous mobile robots were developed. The first algorithm solves the problem of finding a target whose position is unknown a priori. The second algorithm solves the problem of finding a path to a target whose position is known a priori.

In both problems, the search *environment is unknown a priori*, which implies that the algorithms developed are *online*. Real situations incorporate an unknown search environment, where a map is unavailable, nonexistent, or partially known. Moreover, the search *environment is unbounded*, meaning that the search area can be very large-scale, or arbitrarily large. Despite these limitations, both algorithms developed maintain their optimal performance.

Optimality is obtained through the following stages. In the first stage, in order to measure performance, *generalized time competitive complexity* is defined. This is the functional relation between a solution to the problem (of an online algorithm) in the form of time of search, and the optimal off-line solution. In the second part, a *lower bound* for each problem is proved. The importance of the lower bound is that no algorithm exists that can perform better than this bound. In the third part, an *upper bound* for each algorithm is proved. This upper bound has the same functional relation as the lower bound. Since both bounds have the same functional relation, they *classify the problem* in part five. Finally, since the algorithms developed perform within the bounds of the problems classes, they are proved *optimal* (up to constants).

The performance of the developed algorithms, which use a *heterogeneous velocity* group of robots, is proved to be better than similar algorithms for a group of homogeneous robots and better than single-robot algorithms.

Although the developed algorithms are on-line, run in unknown environments, and run in unbounded environments, they are proved to be *complete*, which means that if a solution exists, they will find it. Moreover, they are proved to be *robust*, so that they stay complete even when the robots experience malfunctions. Furthermore, the algorithms assume that the robots are capable of only short-range sensing, have no communication and no central control unit, and have minimal processing

power and memory onboard.

The upper bounds of the algorithms are *worst-case scenario bounds*. In order to evaluate the *average-case performance* of the algorithms and the effect of several different parameters on the average case performance, *simulations* were conducted. A special software platform was built to serve as the base for the simulation application of the two different algorithms.

In order to validate the simulation application and evaluate the effect of uncertainty of sensors and hardware implementation, *experiments* were conducted. Four robots were designed and built as were other parts of the experiment system.

To summarize, the main assets of the algorithms developed are that they are proved optimal and complete in the sense of search time, for unknown and unbounded environments. They cover two major motion planning problems: searching for a target whose position is unknown, and searching for a path to a known target. The algorithms developed utilize a group of heterogeneous velocity robots. Previous algorithms that solved these problems either do not have all the properties mentioned above (in particular, no algorithm handles an unknown, unbounded environment and proves upper bound on its performance) or, they have such properties, but are designed for single robots ([19], or homogeneous groups of robots[57, 58, 53]).

The main contributions of this research are as follows:

- Two motion planning algorithms for a heterogeneous group of mobile robots searching for unknown and known targets in unknown and unbounded environments
- Upper bound, completeness, and robustness proofs despite major limitations, such as those mentioned in the previous point, and limited robot capabilities, such as short-range sensors, lack of communication, and lack of centralization
- Classification of the two motion planning problems into time competitive complexity classes, including lower bound proofs, upper bound analysis, optimality, completeness, and robustness proofs
- Extensive performance evaluation of the developed algorithms: Average-case performance for a multitude of parameters was evaluated in simulations. Experiments validate the simulations and further evaluate uncertainty of sensors and hardware implementation.

Chapter 2. Literature Review

Motion planning problems can be categorized in many ways. The most common of these are the definition of the problem, the setup and assumptions about the robot, and the nature of the solution. The overall mission goal is another important category. A basic extension of the fundamental motion planning problem is searching for a target whose position is unknown a-priori to the searcher. This problem may be considered a partial area coverage problem or an area exploration problem. Here a certain area must be visited and inspected by the robot. Examples of applications are lawn mowing, cleaning, demining, search and rescue missions, and planetary exploration. A thorough survey on coverage by Choset et al. [12] classifies it according to four types of cell decomposition, *heuristic*, *approximate*, *partial-approximate*, and *exact cell decomposition*. Choset concentrates on single robot coverage and the section on multi-robots focuses on approximate cell and exact cell decompositions. The three tables 2.1, 2.2, and 2.3 summarize the important studies conducted, including their motion planning, exploration, and coverage parameters, and the contribution of this dissertation. A detailed review of the more interesting and relevant works follows.

Computational Complexity

Robot motion planning algorithms can be categorized in many ways, and one of the most important questions is how good the algorithm is. To answer this question, a common measure must be applied. The most common measure is the performance of the algorithm, which depends on computational complexity. Computational complexity attempts to classify computational problems into classes according to their level of difficulty. The level of difficulty of computational problems was first examined in terms of the time it took to solve them. Since the time needed to solve a problem depends on the capabilities of the machine that runs the algorithm, a more general performance measure is the number of steps required to find the solution as a function of the size of the problem, that is, the size of the input [42].

The common notation for the upper bound of an algorithm is the big O notation. 'A function $f(n)$ is $O(g(n))$ whenever there exists a constant c such that $|f(n)| \leq c \cdot |g(n)|$ for all values of $n \geq 0$. A *polynomial time algorithm* is defined as one whose complexity function is $O(p(n))$ for a polynomial function p , where n is used to denote the input length. Any algorithm whose time complexity function cannot be bounded in this way is called an *exponential time algorithm*' [22]. The definition of time complexity is measured in worst-case scenarios. When the size of the problem, the input, is large, this distinction between polynomial and exponential time algorithms is much more significant than it is with small inputs.

For example, on a computer that calculates each step in a microsecond, for $n = 10$, the time to complete the calculation for an algorithm with an upper bound of $O(n)$ will be 0.00001 second, whereas the time to complete the calculation for an algorithm with an upper bound of $O(3^n)$ will be 0.059 second. For $n = 60$ the time will be 0.00006 second for the $O(n)$ algorithm and 1.3×10^{13} centuries for the $O(3^n)$ algorithm. The improvement of technology, specifically computing power, will not significantly enhance performance. For example, the size of the largest problem instance solvable

in 1 hour for a fast computer 1000 times faster than a slow computer will be 1000 times greater for an algorithm with a complexity function of $O(n)$, whereas it will be greater by only 6.29 input items for an algorithm with complexity function of $O(3^n)$ [22].

Problems are considered intractable if no algorithm exists that can solve them with a polynomial upper bound. Thus, generally it is considered best practice to develop and use algorithms with polynomial time complexity. The theory of computational complexity classifies known problems into classes of different levels of difficulty. The P class includes all the problems that can be solved in polynomial time. A more difficult class of problems is called NP, and formally it consists of all the decision problems that can be solved in polynomial time by a nondeterministic computer. In other words, if a solution to the problem exists, by a guess, for example, it can be verified in polynomial time. A major unsolved question is whether the class P is contained in or equal to the class NP.

Stephen Cook, in his study "The Complexity of Theorem Proving Procedures," from 1971, proved that within the class NP there exists a problem called the "satisfiability" problem. All other problems in this class can be reduced to "satisfiability" problems with polynomial time. Thus, this is the "most difficult" problem in NP. Later, it was proved that some other problems in NP are as difficult as the "satisfiability" problem and together they form the class of NP-complete problems. Even more difficult problems exist, which may not even belong to NP. In these problems, solutions cannot be verified in polynomial time, for example, if the solution size is polynomial in the size of the problem. Such problems, if they can be reduced to a problem in the NP-complete class, belong to the class called NP-hard. Among the problems that belong to the NP-complete class, the most relevant to motion planning are the ones consisting of graphs. The most famous of them is the traveling salesman problem.

The traveling salesmen problem is defined as follows: Given a finite set of cities, a distance for each pair of cities and a bound, does a tour of all given cities having a total length no greater than the given bound exist? Another famous graph problem that belongs to the class of NP-complete is the vertex cover problem, which is defined as follows: Given a graph and a positive integer, is there a vertex cover the size of that integer or smaller? Both problems belong to the NP-complete class and thus are considered intractable. In order to overcome the issue of intractable problems that need to be solved, a solution in the form of approximation was introduced. Approximation algorithms run in polynomial time but do not provide an optimal solution. Approximation algorithms can ensure the solution is within a certain bound, not as good as the optimal solution, which, in many cases, is better than an optimal solution that can be obtained in exponential time. Some algorithms have exponential time performance in the worst case, but perform quite well in most of the average cases. An example of this kind is the simplex algorithm, which solves linear programming problems.

In robot motion and path planning problems, due to the advances in technology and computation power, the time it takes a robot to calculate its next step is several orders smaller than the time it takes to actually perform that step. Hence, many studies in this field do not consider the computation time the most important performance measure of their algorithm, but rather the optimality of the solution. Optimality of a solution to a motion or path planning problem is usually measured in terms of the path length produced by the robots executing the algorithm. The actual path length is compared with the optimal off-line solution, and their ratio forms the competitiveness constant.

Search

In many motion and path planning problems, graphs are used to represent either a part of the problem or a part of the algorithm. Searching in graphs has been a well known area of research for many years [36]. An example from this research is the minimum spanning tree problem, for which many solutions have been found. Kruskal's algorithm originally solved this problem in $O(mn)$ time and was later improved to $O(m \log n)$, where m is the number of edges, and n is the number of vertices in the graph. Prim's algorithm originally solved that problem in $O(n^2)$ time and later was improved to $O(m + n \log n)$. The shortest path problem is also very well known and has been thoroughly investigated. Dijkstra's algorithm solved this problem in $O(n^2)$ time and was later improved to $O(m + n \log n)$.

Searching in robotics incorporates a vast variety of problems. Generally, it can be said that searching in robotics sums up to searching for a path. This can be searching for a path to a known target or searching for a target whose position is unknown. The environment's geometry is either known in advance or it is unknown a-priori. Baeza-Yates et al. [5] explored several problems related to the general case of searching in the plane, calculating and proving bounds for the problems' performance. They started with proving that searching for a point on a line using linear spiral search is optimal and the lower bound for that problem is 9 times the optimal off-line solution. Furthermore, they explored problems of searching for a line in planes with different properties.

Other famous algorithms that address the problem of finding a path to a target in an unknown environment are *BUG1* and *BUG2*, presented by Lumelsky et al. [43]. *BUG1* has an upper bound of $D + 1.5 \sum_i p_i$ where D is the straight line distance from the start to the target points, and $\sum_i p_i$ refers to the perimeters of the obstacles intersecting the disc of radius D centered at the target. When searching for a path to a target whose position is unknown with limited capabilities sensors, a certain area must be covered. Thus, the target finding problem can sometimes be referred to as a partial coverage problem.

2.1 Single Robot Coverage Algorithms

Many coverage algorithms use graphs to represent the environment, and deploy the robots to move through the entire graph for complete coverage. The most basic graph walking algorithm is *DFS*, *Depth-first search* algorithm [15], which explores deep in the graph whenever possible, and otherwise backtracks. *STC* algorithm [17] is a single robot coverage that improves *DFS*. Coverage of single robot and multi-robot coverage are the subjects of the following sections.

The single robot covering problem was explored by Gabriely and Rimon [17]. They introduced three versions of the *Spanning Tree Covering (STC)* algorithm, which achieves optimal coverage, *off-line*, *on-line*, and ant-like algorithms. The assumptions and definitions of the *off-line STC* are as follows: First, the robot has a rectangular covering tool of size D and can move only in the four directions orthogonal to the tool's sides. A spanning tree is constructed using *DFS* on a grid of free-from-obstacle cells whose size is $2D$. Finally, each $2D$ cell is further subdivided into four D size cells, and the robot circumnavigates the spanning tree (Fig. 2.1(a)) until it reaches the starting point, implying complete coverage.

The on-line *STC* uses onboard short-range sensors to detect the obstacles of the unknown environment lying in the neighboring cells of the robot. As the robot moves, it incrementally constructs

Table 2.1: Exploration works

Reference	[68]	[16]	[60]	[4]	Sarid PhD
Multi-robot	+	+	+	+	+
heterogeneous	-	-	-	-	+
initial locations	NA ^a	known	known	NA	known
online	+ ^b	+	+ ^c	var. ^d	+
unbounded env.	-	-	-	-	+
grid/graph	graph ^e	grid	grid	NR	NR
cell decomposition	App.	App.	App.	NR	NR
min. param.	time	time	time	time	time
heuristic	+ ^f	-	-	NR	-
sensors	+ ^g	NR	-	NR	+
Short-range	laser	NR	-	NR	+
Long-range	laser	NR	-	unlimited	-
simulations	+	+	+	+	+
sim. no. robots	2-20	1-30	1-4	3-5	1-20
sim. param.	NA	4 ^h	R's No.	R's No.	5 ⁱ
proofs	-	-	-	-	+
non-redundancy	+	+	NR	NR	-
robustness	NR ^j / +	+	NR	NR	+
complete	-	+	NR	-	+
distributed	+ / -	+	+	+	+
centralized	- / +	-	-+ ^k	+	-
localization	+ ^l	cell	NA	NA	- ^m
collaboration	-	+ -	+	+	-
communication	+ ⁿ / +	- ^o	LOS ^p	LOS	-

^anot available^bRFIDs known^ctargets set in advance^d3 cases: online, offline, and between^eoccupancy graph^fA*, Euclid. Dist.^gIMU, RFID reader^hterrain size: 1000-5000 cells, 2-64 rooms, 20-50 obstaclesⁱ3 environments, 1-20 robots, 3 velocity distributions, 10 initial search area, 10 target positions^jnot relevant^ktargets set in advance^lodometry+RFID^massumed idealⁿindirect via RFID^ovia RFID^pline of sight

Table 2.2: Approximate Cellular Decomposition Coverage works

Reference	[1]	[17]	[26]	[67]	[27]	Sarid PhD
Alg. Name		STC	MSTC	MFC	ORMSTC	HMRSTM
multi-robot	+	-	+	+	+	+
heterogeneous	-	-	-	-	-	+
initial locations	arb.	NR	arb.	arb.	arb.	known
online	-	+	-	-	+	+
unbounded env.	-	-	-	-	-	+
grid/graph	grid	grid	grid	grid	grid	NR
min. param.	time	path	non-red. ^a	time	non-red.	time
heuristic	+	-	-	+	-	-
robot size		D ^b	NR	D	NR	D
short-range sensors		+	+	+	+	+
Long-range sensors		NR	NR	-	-	-
simulations	+	+	-		-	+
sim. no. robots	3-30	1	NR	2-20	2-10	1-20
sim. param.	OD ^c		NR	OD, RC ^d	2 envs.	5 ^e
experiments	-	-	-	-	+	+
proofs	+	+ ^f	+ ^g	+ ^h	+	ⁱ
non-redundancy	NR	+	+	-	+	-
robustness	NR	NR	+	-	+	+
complete	NR	+	+	+	+	+
distributed	NR	NR	+	-	+	+
centralized		NR	-	+	-	-
localization	NR	perfect	perfect	perfect	perfect	perfect
collaboration	NR	NR	-	-	+	-
communication	NR	NR	-	-	+	-

^anon-redundancy^bcovering tool^cobstacles' density^dRC - robots' clustering^e3 environments, 1-20 robots, 3 velocity distributions, 10 initial search area, 10 target positions^f T_{STC} is optimal^g $T_{MSTC} \leq 0.5T_{STC}$ for $R \geq 3$ ^h $T_{MFC} \leq T_{STC}, T_{MFC} \leq 8T_{opt} + \epsilon$ ⁱoptimal, quadratic in the optimal off-line solution

Table 2.3: Exact Cellular Decomposition Coverage works

Reference	[37]	[14]	[48]	[49]	[35]	Sarid PhD
Alg. Name		B.C.D.^a				HMRSTM
Multi-robot	+	-	+	+	+	+
heterogeneous	-	-	+	+		+
initial locations	+	-	+	+ ^b		known
online	-	+	+	+	+	+
unbounded env.	-	-	-	-	-	+
grid/graph	graph	graph ^c	graph ^d	graph ^e	graph ^f	NR
min. param.	path	path	path	path	path	time
heuristic	-	-	-	-	-	-
robot size	point	NA	NA	NA	NA	D
Short-range sensors	+	+	+	+	+	+
Long-range sensors	-	-	LOS	-	-	-
simulations	+	+	-	+	+	+
sim. no. robots	5	NR	NA	3	2	1-20
sim. param.	R's r=15,30	NA	NA	NA	NA	5 ^g
Experiments	+	+	-	-	+	+
proofs	-	-	-	-	-	+
non-redundancy	-	-	-	-	-	-
robustness	-	NR	NA	+	+	+
complete	+	+	+	+	+	+
distributed	-	NR	+	+	+	+
centralized	-	NR	-	-	-	-
localization	+	DR ^h	+	+	+	perfect
collaboration	+	NR	+	+	+	-
communication	-	NR	LOS	+	+	-

^aBoustrophedon Cellular Decomposition^binitially deployed along the side of the field at regular intervals^cadjacency graph^dReeb graph^eReeb graph^fadjacency graph^g3 environments, 1-20 robots, 3 velocity distributions, 10 initial search area, 10 target positions^hdead reckoning

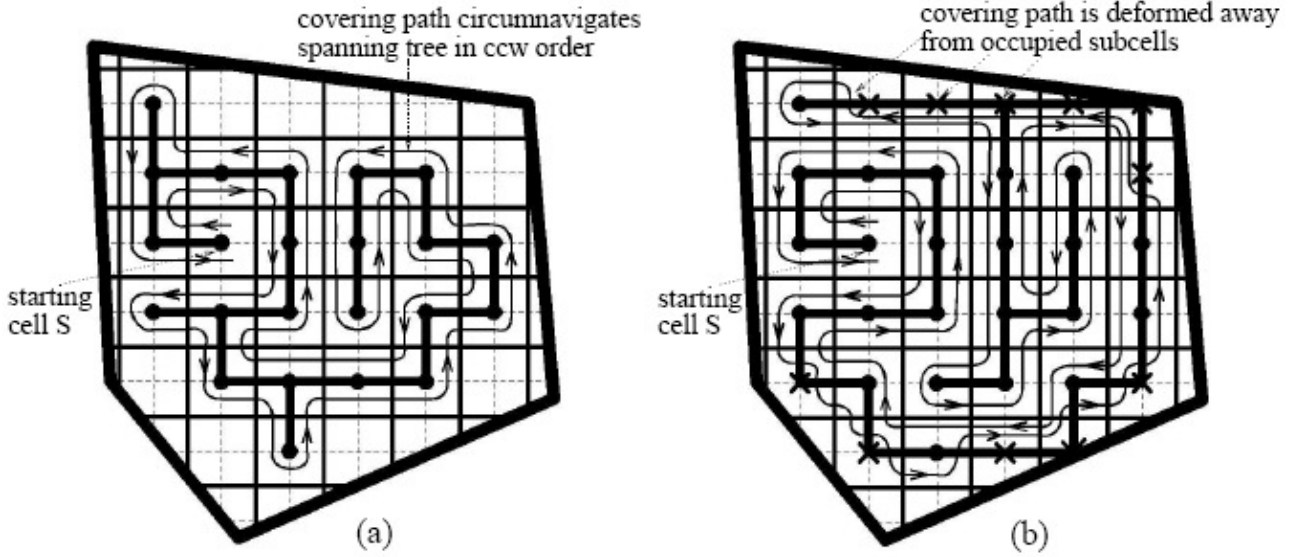


Figure 2.1: Execution examples of (a) STC [17], (b) D-STC [18]

and circumnavigates a spanning tree using *DFS* method, thus covering the environment completely. In this version, the spanning tree is stored in the onboard memory, resulting in a dependency of the search area on memory size. The third version solves this memory problem by leaving markers on visited cells. A major drawback of partially occupied $2D$ cells that resulted in uncovered D cells near obstacle boundaries and non-rectilinear walls was addressed in later research [18]. Gabriely and Rimon introduce *D-STC*, which solves this problem by visiting the previously uncovered cells (Fig. 2.1(b)), resulting, in worst-case scenarios, in a coverage area twice the area of the environment.

2.2 Multi-Robot Coverage Algorithms

Several algorithms extend *STC* to support multi-robots. An example of such an extension is the Multi-Robot Spanning-Tree Coverage algorithm (*MSTC*) [26] of Hazon et al. which assumes different starting positions for the robots. Their algorithm first builds one spanning tree using *STC* algorithm, and then each robot circumnavigates its part of the tree. Since *MSTC* focuses on non-redundancy of the area covered by the robots, the covering time is not optimal when the robots' starting positions are close together. In such worst-case scenarios, $k - 1$ out of k robots may stand while one robot continues to cover the entire environment and *MSTC* performance is reduced to that of *STC*. Allowing the robots to backtrack, *MSTC* achieves a worst-case performance, which is twice as good as that of *STC*.

Zheng et al. [67] introduce the Multi-robot Forest Coverage Algorithm (*MFC*), which uses heuristics to construct partially overlapping trees (Fig. 2.2). The union of the forest completely covers the grid. *MFC* is proved to cover the environment with a time larger than optimal eight times at most. Agmon et al. [1] introduce *Create-Tree*, an off-line grid-based distributed construction of a spanning tree that considers the initial locations of the robots. First, each robot constructs a disjointed spanning tree, trying to create trees of equal lengths, knowing the other robots' spanning trees. Then, using heuristics, the individual trees are merged into one spanning tree, which minimizes the total coverage time.

In the notable early paper [37], Kurabayashi et al. present an off-line sweeping algorithm for homogeneous multi-robots. The algorithm includes three parts. First, it computes separated paths according to the Voronoi method. Next, the paths are connected to create a minimum distance tour

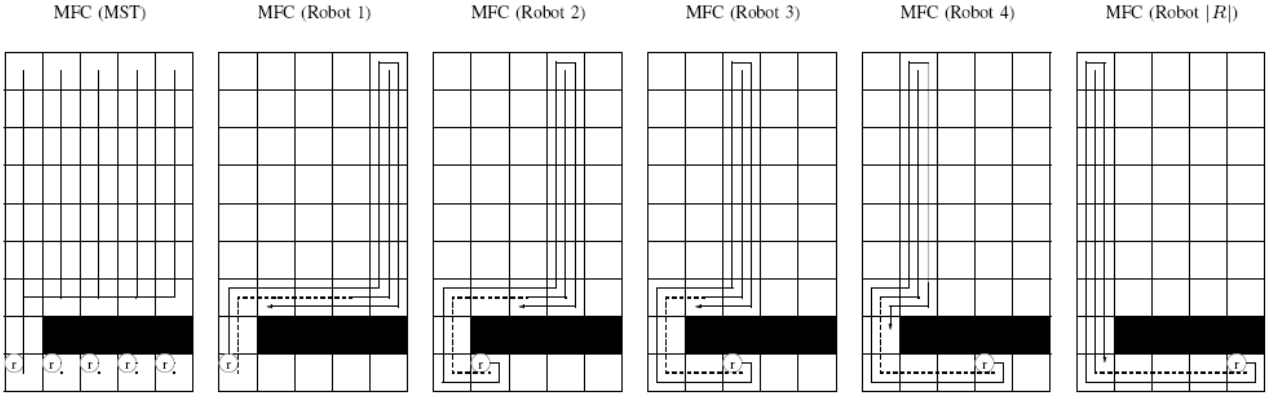


Figure 2.2: Execution examples of MFC [66].

according to a known solution to the analogous Chinese Postman Problem. Finally, the tour is divided into paths of equal lengths, which are assigned to the robots. The assumptions made here are that all robots have equal ability, the sweeping unit of a robot is a circle with a radius of r , a robot can move omni-directionally, only the size of the sweeping unit is considered, a robot can be regarded as a point, and the work area to be swept is represented by polygons. No proof of optimality is made.

All the above-mentioned algorithms *STC*, *MSTC*, *MFC*, *Create-Tree*, use approximate cell decomposition. Moreover, they all are off-line and operate in a known a priori, bounded environment. Hazon et al. [27] further developed *MSTC* and created a distributed on-line version of such an algorithm, called *ORMSTC*. Here, each of the robots gradually constructs a spanning tree, each knowing other robot's current position by communicating with them. Like their previous off-line algorithm, *ORMSTC*, also focuses on non-redundancy of the covered area, thus losing optimality in search time, since in worst-case scenarios the algorithm will lead to blocking $k - 1$ out of the k robots from the beginning of the coverage, leaving one robot to cover the whole environment.

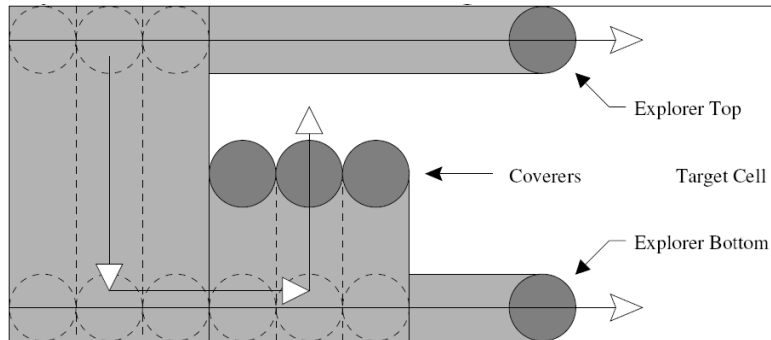


Figure 2.3: Boustrophedon extension to multi-robot by Rekleitis et al. [48]

Rekleitis et al. [48] extended the Boustrophedon decomposition to multiple robots, using two types of robots, explorers, which search the environment for obstacles and decompose it accordingly, and coverers, which cover each cell previously created (Fig. 2.3). This solution relies on line-of-sight (*LOS*) communication between the explorers and inside teams formed by robots that are in line of sight. In a later study, Rekleitis et al. [49], focus on de-mining. Here, the robots are deployed from a vehicle at one side of a field that needs to be de-mined. The initial positions are equally distributed so that each robot is assigned a vertical stripe to cover (Fig. 2.4). At first, each robot tries to encircle its stripe and build a representation of the work area. The solution combines an auction mechanism to divide the uncovered area after the initial run. Kong et al. [35] propose a solution based on

the Boustrophedon decomposition, but it can be considered partially approximate, since, at first, the algorithm decomposes the environment to vertical stripes having a constant width of twice the covering tool size (Fig. 2.5). Next, each robot covers a cell with simple back and forth motion and updates the connectivity graph according to the obstacles it encounters. Finally, a mechanism to introduce new tasks is incorporated in the algorithm to handle cases where one robot finishes its covered area before a new task is introduced and the environment is not yet completely covered.

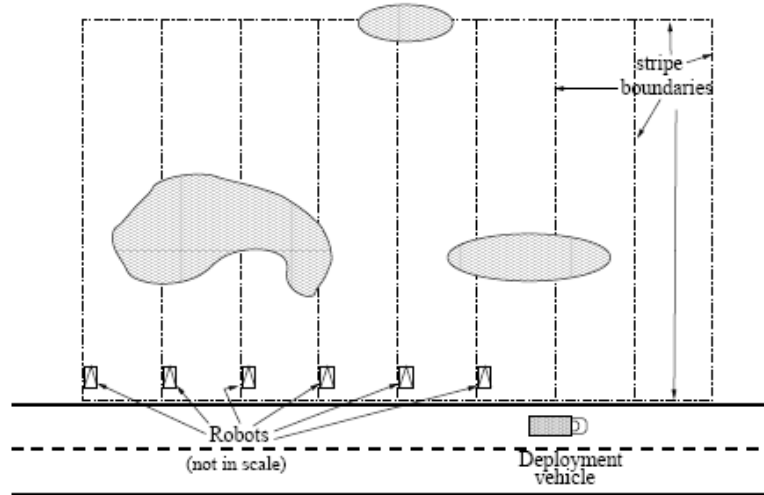


Figure 2.4: Multi- Robot demining by Rekleitis et al. [49]

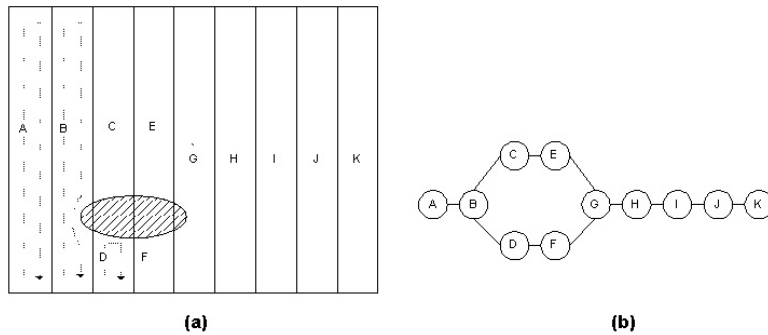


Figure 2.5: (a) Cellular Decomposition with fixed size cell width, (b) Adjacency Graph to represent the decomposition of (a) [35]

2.3 MRSAM algorithm

This study is partially based on our previous *MRSAM* algorithm [57, 53, 52]. *MRSAM* launches multiple robots from a common starting point S and assigns each robot j to a disc to search for the target T in it. All the discs are concentric and S is their center.

The first robot ($j = 1$) is designated to the initial disc of area A_0 , and each of the following robots starts its search in a disc of area larger than the previous disc by a factor of ¹ $\alpha > 1$, namely, the areas of the discs will be $A_0, \alpha A_0, \alpha^2 A_0, \alpha^3 A_0 \dots$. For example, in Figure 2.6, *MRSAM* deploys a group of four robots to search for the target. Robot 1 is initially assigned the task of searching for the target inside a disc of area A_0 , robot 2 is assigned the task of searching inside a disc of area αA_0 , and robots 3 and 4 search initially in discs of areas $\alpha^2 A_0$ and $\alpha^3 A_0$ respectively. After robot 1

¹This is an important property, since the search area must be extended in each step in order to reach the target in case the target is positioned outside the first search disc.

finishes covering the entire portion of disc 1 that is accessible from S and fails to find the target, it starts searching for the target inside disc 5 of area $\alpha^4 A_0$. Robot 2 will continue to search in disc 6, and robot 3 will continue to disc 7. Note that the target lies inside disc 7, but cannot be reached by robot 3 in this current step due to the presence of obstacles. Finally, robot 4 moves on to search in disc 8, where it finds the target and terminates the algorithm. Each robot searches for the target in the accessible portion of the disc allocated to it until the target is detected, or until the entire region accessible from S is explored without finding T . The search process in each disc is as follows: The robot imposes an online discretization of the continuous area into a grid of D -size cells [29, 18]. The grid consists only of free cells and is surrounded by partially occupied cells. The robot executes a standard area coverage tour on the grid of free cells, while scanning each new cell for the target. Upon entering a new cell, the robot additionally scans the neighboring partially occupied cells for T . If the discretization preserves the connectivity of the accessible region (this assumption can be relaxed by a more sophisticated algorithm that monitors local connectivity breakage), then clearly all free and partially occupied cells in the region accessible from S are eventually inspected by the robot.

The robots cover the area of the discs until they reach the target. If the target was not detected in the initial disc, the robot is assigned to the next unoccupied disc.

In [57] it is proved that if the target T is reachable from S , *MRSAM* finds the target using n robots and the path length l_j traveled by the robot that found the target satisfies the quadratic inequality

$$l_j < \frac{2\pi\alpha^{n+1}}{D(\alpha^n - 1)} l_{opt}^2 + \frac{2\pi r_0^2}{D}$$

, where D is the robot size, r_0 is the initial search radius, α is the multiplication factor, which is a function of n only, and l_{opt} is the length of the optimal off-line path from S to T . Later it is proved that the competitive complexity of *MRSAM* is minimal when the multiplication factor is $\alpha = (n + 1)^{1/n}$. It is proved that *MRSAM* algorithm is optimal, complete, robust, and decentralized, and does not need communication.

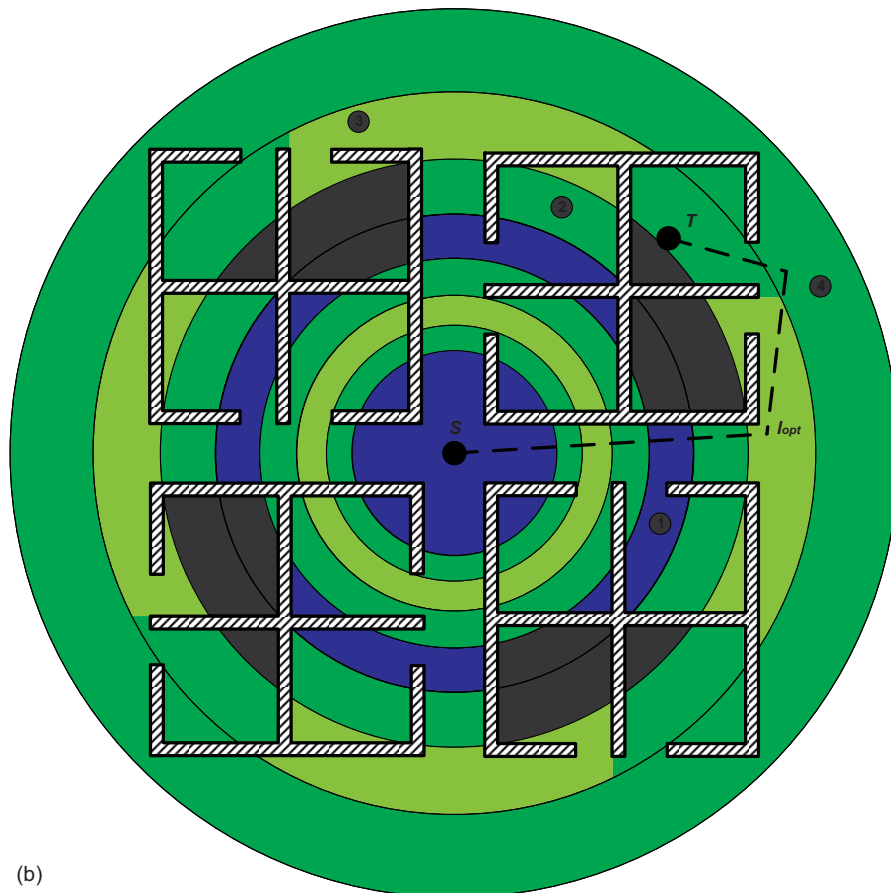
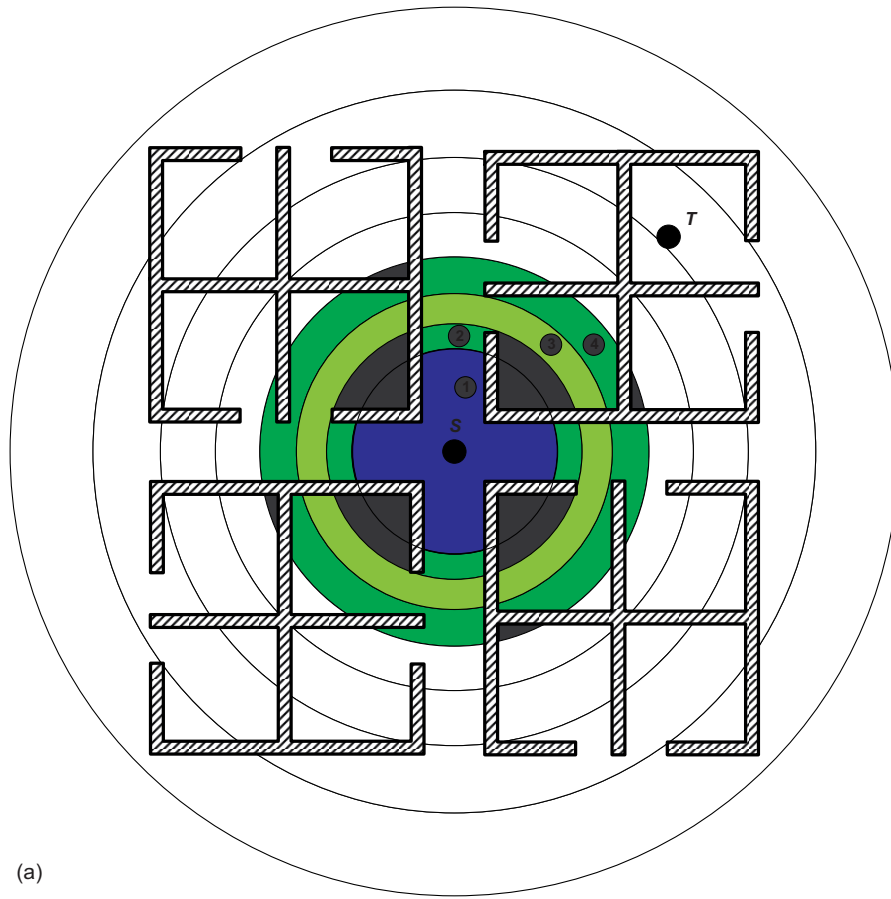


Figure 2.6: The first (a) and final (b) steps of *MRSAM* execution example. The gray area marks the unreachable parts of each robot in its current step. The dashed line denotes the optimal path, l_{opt} (b).

2.4 MRBUG algorithm

This study is partially based on our previous *MRBUG* algorithm [58, 53, 52]. *MRBUG* launches pairs of robots from a common starting point \mathcal{S} to search for a target \mathcal{T} whose position is known in an unknown and unbounded environment. Each pair of robots is assigned to an ellipse whose focal points are \mathcal{S} and \mathcal{T} . Each pair of robots executes *PBUG1*, an extended version of the *BUG1* single robot algorithm.

In *PBUG1* a pair of robots that start from a common start point \mathcal{S} needs to find a path to a target \mathcal{T} whose position is known in an unknown planar environment. The pair of robots will move together toward the target in a straight line until it hits an i^{th} obstacle at a point marked as *Hit point* H^i , $i = 1, 2, \dots$. At this point the pair splits, robot R_L turns left and robot R_R turns right, and they circumnavigate the obstacle from different directions. Each robot encircles half of the obstacle's perimeter. While moving, each robot calculates and remembers the closest point on the obstacle's boundary to the target. Upon meeting, the robots compare the recorded information, decide which point is the closest to the target, join and again move together to that closest point, which they mark as *Leave point* L^i , $i = 1, 2, \dots$. Finally, the robots continue to move together toward the target.

In *MRBUG*, the execution of *PBUG1* regards the ellipse as a virtual obstacle's boundary. If the target is detected, the algorithm terminates; otherwise, the pair of robots repeats the process on the next unassigned ellipse in the series. A formal description of the basic algorithm follows. Each pair of robots executes *PBUG1* within an ellipse of area larger than the previous ellipse's area by a factor of $\alpha > 1$; that is, the areas of the ellipses will be $A_0, \alpha A_0, \alpha^2 A_0, \alpha^3 A_0, \dots$. For example, in Figure 5.3, robots 1_L and 1_R are initially assigned to search for a path to the target inside an ellipse of area A_0 and robots 2_L and 2_R are assigned to search inside an ellipse of area αA_0 . The search for the path inside an ellipse is conducted by the pair of robots assigned to that ellipse using *PBUG1*. In the following example depicted in Fig. 5.3, *MRBUG* launches three pairs of robots, 1, 2, 3 to search for a path to the target in a very simple environment. Each robot pair is initially assigned to a bounding ellipse, e_1, e_2, e_3 to execute *PBUG1* in it, and each robot in a pair is assigned to a different local direction, *Left, Right*: $1_L, 1_R, 2_L, 2_R, 3_L, 3_R$. At first, the robot pairs move directly toward the target, and as they encounter an obstacle they split, each robot moving in its local direction. It can be observed that a part of the path is traversed by all the robots together at the same time, and the robots will move together as long as they are within the boundary of the first ellipse. As depicted in Fig. 5.3(a), while robot pair no. 2 is traversing the second ellipse, robot pair no. 1 finishes traversing the obstacle's boundary, which lies inside the first bounding ellipse and the ellipse itself. It can be seen that robot pair no. 3 does not have to traverse ellipse no. 3 in this example, since it does not intersect the obstacle. After meeting each other, the robots of pair no. 1 move toward their first leave point, which is the closest point to the target they encountered while traversing the obstacle's boundary, where they conclude that they cannot reach the target from ellipse no. 1 (Fig. 5.3(b)). Thus, the robots of pair no. 1 start executing *PBUG1* in ellipse no. 4. Meanwhile, robot pair no. 3 meets on the obstacle's boundary, as can be seen in Fig. 5.3(c). While robot pair no. 1 is busy with its search in ellipse no. 4, robot pair no. 2 meets on ellipse no. 2, and afterward the pair moves together toward the closest point to the target. At this interval, robot pair no. 3 moves together to its leave point on the obstacle's boundary and from there it continues without additional obstacles and reaches the

target (Fig. 5.3(d)).

In [58] it is proved that if the target \mathcal{T} is reachable from \mathcal{S} , *MRBUG* finds the target using n robots and the path length l traveled by the robot that reached the target satisfies the quadratic inequality

$$l \leq \frac{\pi}{D} \frac{\alpha^{n+1}}{\alpha^n - 1} l_{opt}^2 + \|\mathcal{S} - \mathcal{T}\| + 4 \frac{A_0}{D}$$

where D is the robot size, and l_{opt} is the length of the optimal off-line path from \mathcal{S} to \mathcal{T} . Note that the upper bound is scalable, in the sense that both summands have units of length. Later it is proved that the competitive complexity of *MRBUG* is minimal when the multiplication factor α equals $\alpha = (n + 1)^{1/n}$. It is proved that *MRBUG* algorithm is optimal, complete, robust, decentralized, and does not need communication.

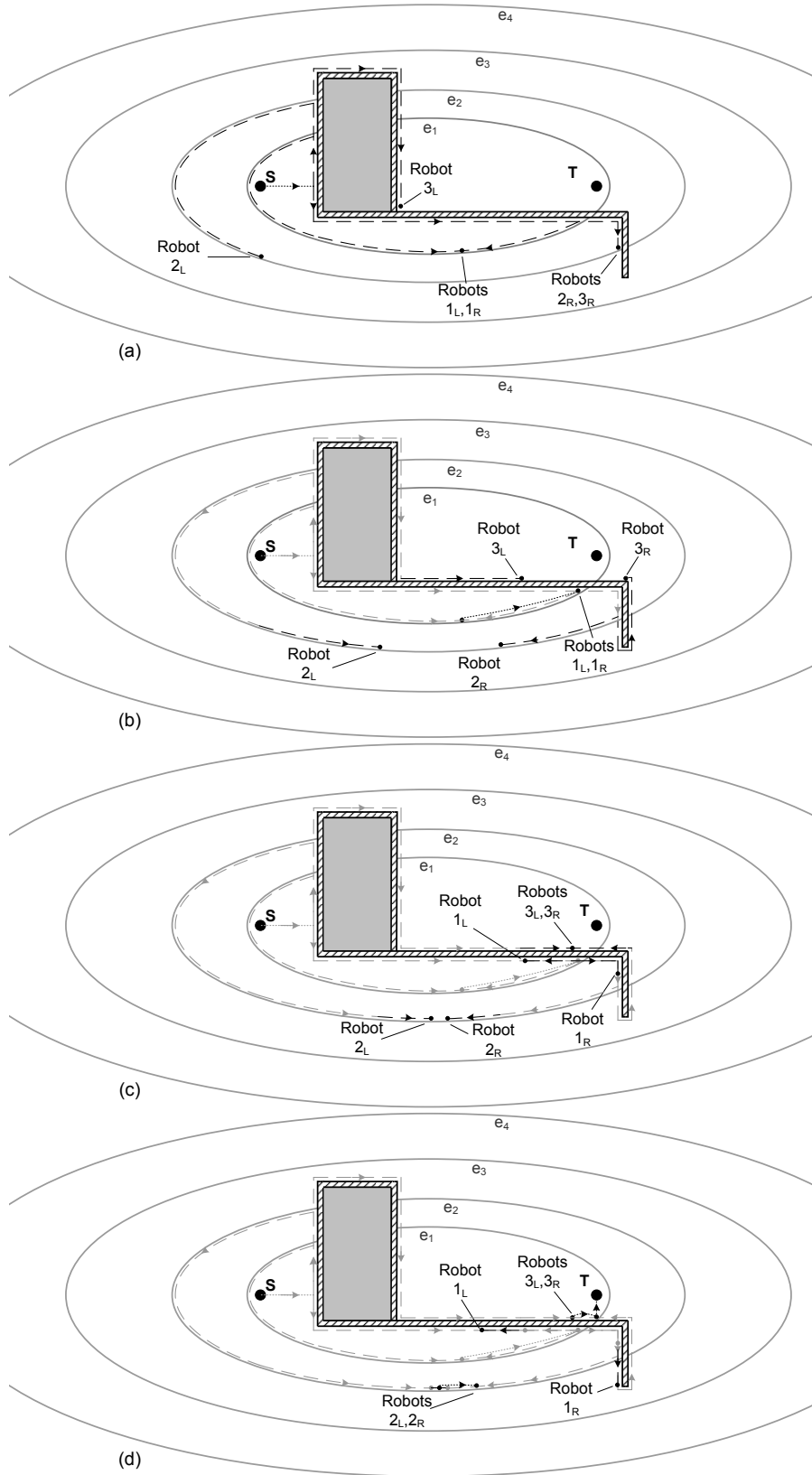


Figure 2.7: Execution example of *MRBUG* with 3 pairs of robots. The dense dashed lines indicate a mutual path of a pair. The path traversed in the previous step is colored grey

Chapter 3. Methodology

3.1 Overview

New algorithms for a group of heterogeneous robots that must find a target for unknown (*HMRSTM*, *Heterogeneous Multi-Robot Search Time Multiplication*) and known (*HMRBUG*, *Heterogeneous Multi-Robot BUG*) positions are developed. Proofs of the algorithms are presented, including completeness, performance bounds, and relative performance compared to other algorithms. The algorithms are based on the area multiplication method of *MRSAM* and *MRBUG* algorithms for a group of homogeneous robots [57, 58]. The area multiplication method computes a multiplication factor, α , according to the number of robots n , and assigns each robot to search for the target in consecutive areas where each area is larger than the previous area by the multiplication factor. The areas selected are discs for an unknown target position (*HMRSTM*) and ellipses for a known target position (*HMRBUG*). Analysis includes theoretical analysis, simulation analysis, and validation in real-world experiments.

3.2 Algorithms

Since the main performance measure is time, a multiplication factor for the time it takes to cover each area (disc and ellipse corresponding to *HMRSTM*, and *HMRBUG*, respectively) is developed. The size of each area is calculated using this time and the robot's velocity. The multiplication factor, denoted α , is calculated with two constraints. The first asserts that the time to cover the next area must be greater than the time to cover the current area. The second constraint is that the next area must be greater than the current area. While the first method yields α calculated at the beginning and remains constant during the rest the execution, the second method considers a varying multiplication factor, namely, α changes from area to area, and more precisely, each robot has its own multiplication factor.

3.2.1 Heterogeneity

The heterogeneity measure explored is different velocities among the robots. First, an algorithm for a group composed of robots with two velocities, slow robots and fast robots, is developed. Subsequently, the algorithm is generalized in such a way that more than two groups of velocities are handled, i.e., any number of velocities smaller than or equal to the number of the robots. The robots' velocities are defined as follows. The slowest robot, numbered robot 1, has a velocity $v_1 = \beta_1 v$, where $\beta_1 = 1$, thus, $v_1 = v$. Each additional robot has a velocity equal to or greater than that of the previous robot, so that, $v_j \geq v_{j-1}$, $\forall j = 2, \dots, n$, where n is the total number of robots. Thus, $v_j = \beta_j v$ and $\beta_j \geq \beta_{j-1} \geq 1$, $\forall j = 2, \dots, n$.

3.2.2 Completeness

An algorithm is said to be complete if it finds a solution, if one exists. The algorithms developed bound the search in each step with a disc or an ellipse until the target is found or reached. Within each disc or ellipse, the algorithms perform the search using sub-algorithms, that is, *STC* to cover a disc area, and *PBUG* to find a path to the target within an ellipse. Since the discs and ellipses grow

in each step until they contain a path to the target, and since the sub-algorithms used are complete, completeness of the algorithms is proved using the completeness of the sub-algorithms they use.

3.2.3 Decentralization, Communication, and Robustness

The algorithms are decentralized in such a way that each robot executes its mission independent of the other robots and of any centralized unit. Communication is not needed, except at the end of the execution, when the target is reached, and all other robots must receive a stop signal. An exception to this rule is the extended version of *HMRBUG* algorithm. Here, the robots work in pairs, and each pair is independent according to the rule above. However, two robots of the same pair cooperate to find the path within the same ellipse, and thus must have some communication capabilities upon meeting.

Since the algorithms are decentralized, and the robots are independent and do not need communication, and since, in each step of the algorithms, the search is conducted within growing discs or ellipses until reaching the target, the algorithms are robust. If one or more robots stop executing the algorithm due to malfunction or sabotage, the algorithms will still be complete since, if at least one functioning robot is left, it will reach the target, and hence robustness is achieved.

3.3 Performance Measures

Performance of on-line motion algorithms is usually measured by the path length traveled prior to finding or reaching the target. In this research, since robots with varying velocities are employed, each will travel different path lengths during the same time. Thus, the main performance measure was defined as the time to reach the target.

The on-line path length of the examined algorithm is compared to the off-line optimal solution. The ratio between the on-line path length and the optimal solution is defined as competitiveness. The general definition of time competitiveness is provided in chapter 4.

The universal lower bounds of the problems are proved environments that are believed to be very difficult for a group of heterogeneous robots, causing all algorithms to perform badly, that is, covering the whole area of the environment prior to finding the target. The upper bounds of the algorithms are found using methods taken from computational geometry. The algorithms' performances are compared to other algorithms solving the same problems in worst-case conditions.

3.4 Simulation analysis

3.4.1 Overview

Simulation analysis is performed on Intel Core2 Duo 2.5 GHz CPU based PC, with 3.5 GB of RAM memory and running Microsoft Windows XP operating system. Simulation software was written in C-Sharp (C#) language using Microsoft Visual Studio .NET 2010.

Simulations test the average-case behavior of the developed algorithms for several different parameters evaluated as defined below. The main performance measure evaluated is the time it took to reach the target.

3.4.2 Parameters evaluated

The computer simulations test various configurations of the framework, with variations in the parameters governing the algorithm performance.

Number of robots

For unknown targets (*HMRSTM*) the number of robots, n , varies from between 1 and 10, where single robot simulations were conducted for purposes of comparison. For known targets (*HMRBUG*) the number of robots pairs, n , varies from between 1 and 10 (2 to 20 robots).

Velocity distribution

Three velocity distributions are considered. The first is linear distribution of the velocities, denoted β_{Linear} , in which the velocities are calculated to have linear distribution and β_n is maximal according to the number of robots and the algorithm's restrictions. In the second velocity distribution, all robots have the same slow velocity $v_j = \beta_j v$, $\beta_j = 1 \forall j = 1, \dots, n-1$, with the exception of the last robot, which has a maximal velocity. This distribution is denoted β_{Max} . Both heterogeneous velocity distributions are compared with homogeneous velocities, where, $v_j = \beta_j v$, $\beta_j = 1 \forall j = 1, \dots, n$, hence, $v_j = v$, $\forall j$.

β_j indicates the velocity ratios among the robots within the group. β_j were calculated to meet the constraints derived from the algorithm $\alpha_j > 1$, $\beta_1 = 1$, $\beta_j > \beta_{j-1} \forall j > 1$, $\beta_{j-1} < \alpha_j \beta_j \forall j > 1$, $\beta_n < \alpha_1$;

The three sets of β -s tested for each run are:

1. β_{Linear} , "beta-linear": β -s with values distributed linearly
2. β_{Max} , "beta-max": β_n with maximal value and the rest of the β -s close to 1
3. β_1 , "beta-homogeneous" : equal β -s

Environments

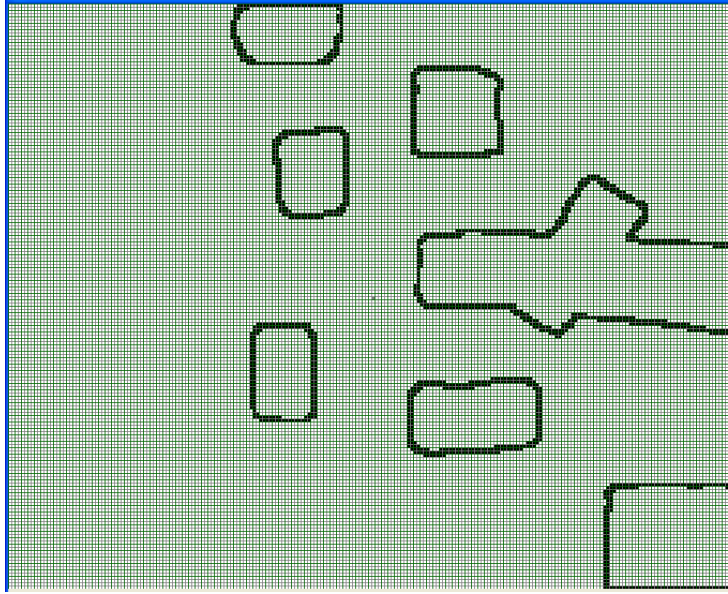
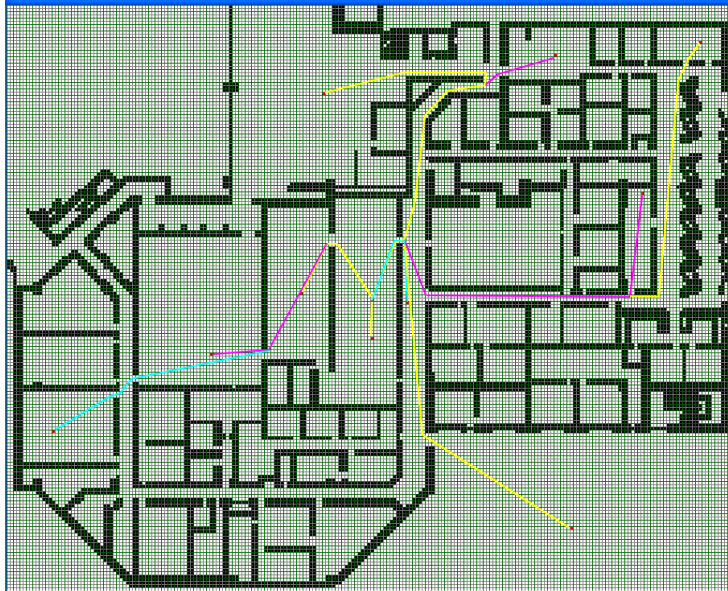
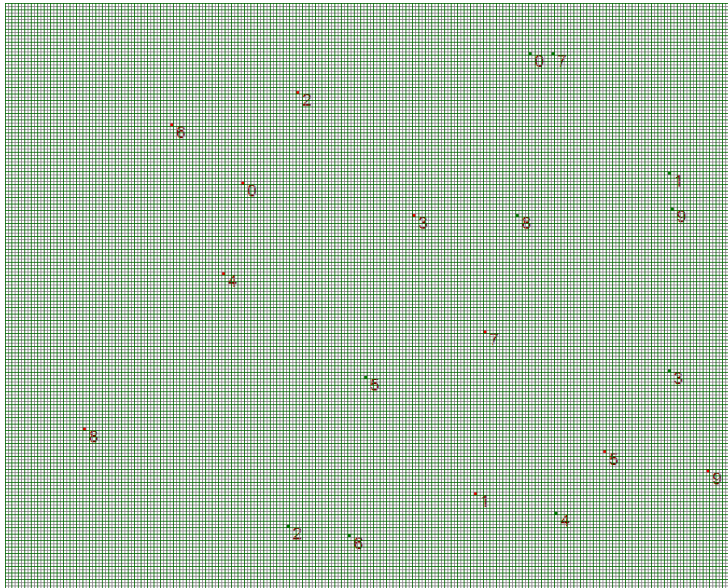
Three environments in which the robots search for a path to the target are evaluated. The first environment, "free," is free of obstacles. The second environment, "cave," is lightly congested and is populated with 7 rock-like obstacles (Fig. 3.1). The third, the "library" environment is a highly congested office-like environment. It is the map of the Vasche library, the 1st floor of the CSU Stanislaus library (Fig. 3.2). The two last environments were obtained from the Robotics Data Set Repository (Radish) [28]¹

Target positions

Ten different target positions were randomly selected and tested for the *HMRSTM* algorithm (searching for unknown position target). Since the search path follows a spanning tree from one side, adjacent targets can be on two sides of the spanning tree and thus the path length to them can differ drastically. To overcome this problem, first, for each target position we further tested two adjacent target positions, one cell to the right and one cell below the original position. Later, for each target position we further tested 8 adjacent target positions, surrounding the original position In such a way that it appears as a cluster of 9 target points. The ten target positions along with the optimal paths are shown in Fig. 3.2;

Ten different start (green) and target (red) positions were randomly selected and tested for the *HMRBUG* algorithm (searching for a known target position) shown in Fig. 3.3. These points were chosen so that all target points would be reachable from their start points in all environments.

¹Thanks to Richard Vaughan and to Ashley Tews for providing this data.

Figure 3.1: *HMRSTM* Cave environment.Figure 3.2: 10 target positions in *HMRSTM* "library" environment with optimal off-line paths marked.Figure 3.3: *HMRBUG* 10 Start-Target positions

Initial search time

For the unknown target simulation (*HMRSTM*), the initial search radius varies from between 3 and 40, with intervals of 4 (intervals of 3 in the second phase). A total of 37,800 simulation runs were analyzed, composed of two parts, as follows: In both parts, 3 environments, 10 robot groups, and 3 beta were tested. The first part consists of simulations on 10 target positions, two adjacent cells that were additional targets, and 13 initial radii. The second part consists of simulations on 10 target positions, eight adjacent surrounding cells that were targets, and 1 initial radius.

For the known target (*HMRBUG*), 10 initial search ellipses were evaluated starting from a minimum value of the minor axis determined by the start and target position. A total of 9,000 simulation runs were analyzed.

3.5 Initial run

The first run took 1 hour and 11 min on an Intel Core2 Duo CPU 2.5 GHz with 3.5 GB of RAM. 16,200 different simulations were conducted during that time. The results were kept in a .csv file of size 1.38MB. The parameters in this run varied as follows: 3 environments, 10 target positions plus 2 adjacent target positions for each target position, 9 robot groups (between 2 to 10), 2 beta distributions (beta-linear,beta-max), 10 initial radii.

The various parameters are presented in Tables [3.1,3.2].

Table 3.1: Simulation Environments

No.	Name	Type	Crowdedness
1	Free	Free from obstacles	None
2	Cave	Open field	Lightly Congested
3	Library	Office	Heavily Congested

Table 3.2: Simulation Parameters

Name	Symbol	Range
No. of Robots (Pairs in <i>HMRBUG</i>)	n	$n = 1, 2, \dots, 10$
Velocity Distributions	β	$\beta_1, \beta_{Linear}, \beta_{Max}$
Initial Search Time	T_{0_i}	$i = 1, 2, \dots, 10$

3.5.1 Performance Measures

Performance is analyzed by calculating the time it took the robot to reach the target. The difference in velocities is implemented in simulation by instructing each robot to move one D -step according to its velocity, where D is the robot's width or diameter. Hence, in the same time period, a robot that is twice as fast will perform twice as many D -steps as the slower robot. Since all robots start and finish the search together concurrently, the execution time of all robots is equal. Thus, time in this context is the number of D -steps of the slowest robot.

This time measure can easily be converted to any physical system, assuming real units are provided. The performance is compared with all the different heterogeneous executions and with exe-

cutions of other optimal deterministic algorithms, with uniform multi-robot algorithms such as *MR-SAM*, *MRBUG* [57, 58] and with single-robot algorithms such as *SAD1* [19].

The universal lower bound of the problem and the upper bound of the algorithm are valid for worst-case scenarios. Consequently, the actual performance, or average performance, is better in all simulation executions.

3.6 Experiments

3.6.1 Overview

Testing the algorithms on real robots in real environments plays an important role in the evaluation process, since some parameters can be considered neither in the analytical proofs nor in the simulation phase. Furthermore, using real robots in real environments enables us to analyze performance capabilities in real world conditions. Significant parameters are the sensors' data errors, communication interferences, and borderline situations, e.g., where the target is very close to an obstacle. Ideally, experiments should resemble the simulation for purposes of comparison, but few limitations are imposed by real-world constraints, that is, financial reasons. For example, testing 20 robots is easy in simulation, but is complicated and expensive to implement.

HMRSTM and *HMRBUG* algorithms are distributed algorithms, and hence do not require communication. Moreover, the algorithms assume perfect localization (position and orientation).

In *HMRBUG*, each pair of robots works as a team when circumnavigating an obstacle, and communicates the shortest point from the obstacles boundary to the target. Apart from that, *HMRBUG* needs perfect localization when moving directly toward the target. In simulations, each pair of robots held a common map and had perfect localization and communication through the software.

In the experiments, in order to achieve perfect localization, we used IR cameras with an error of less than 1 [mm]. The localization data must be communicated to the robots, thus, in applications that use external sources for localization, communication is a necessity.

3.6.2 Mobile Platform

The iRobot Create mobile robot platform was first tested to serve as the foundation of the system. It has a Bluetooth module for communication, encoders for localization, and several short sensors. However, since the area of the lab is approximately 9[m], and the iRobot's diameter is approximately 30[cm], and since our algorithms are optimal for **unbounded** environments, smaller robots were required in order to demonstrate the performance of the algorithms in large environments. The WowWee Rovio mobile robot platform was also tested. It has a NorthStar localization system, wireless communication, and omni-wheels. However, its main disadvantages were the very short battery life (5[min]) and the inaccuracy of the NorthStar system due to the lab's high ceiling and rough surfaces.

Therefore, self-designed robots were developed from off-the-shelf components, and according to our needs. Four platforms were built, each composed of the following components;

- The base of the platform is the 2WD made by DFRobot. It is round and includes two motors, two wheels, and one caster ball. Its diameter is slightly less than 20[cm].
- 1 Arduino UNO microcontroller
- 1 Solarbotics L298 motor driver (with H-Bridge)
- 3 SRF05 Ultrasonic range finders (US sensors)

- 1 XBee wireless communication module
- 1 7.4v 2200mAh LiPo battery

3.6.3 The environments

Experiments were performed in a 3X3 meter area in the Robot Motion Lab of the Department of Mechanical Engineering of Ben-Gurion University of the Negev. The robots' size is 20 centimeters and thus the unbounded environment assumption, even relative to the robots' size, is somewhat limited.

3.6.4 Localization system

The cameras are connected to the OptiTrack Tracking Tools software installed on a single PC. Each robot has at least three round reflective markers positioned uniquely on board. Hence, each robot is defined as a unique rigid body in the Tracking Tools software. The Tracking Tools software broadcasts the position and orientation of each robot to a specified port on the PC using the VRPN server application. Each robot is defined by 7 parameters, composed of 3 coordinates for position x,y,z and 4 quaternions for orientation. A VRPN client application was written in C++ to receive the broadcasted localization data and saved in the Windows registry for compliance reasons. Finally, each robot received its localization data from the main C# application through wireless communication. Since the height of the robots is less than ten centimeters, simple cardboard walls construct the obstacles of the environment, and do not cause any interference to the localization system (no obstruction between the markers onboard the robots and the cameras).

3.6.5 The software

Since the localization data for all the robots is received on a single PC, the motion planning algorithm was implemented in a C# application running on that PC. The C# application orchestrates the reading of the camera's localization data from the registry and its communication to the robots, receives the US sensor's readings and commands the robot's movements according to all of the above information. The C# application is composed of several modules, each run by a different thread.

1. GUI, for presenting the form on screen, visualizing the map of the experiment environment, the robot's positions and shortest points, and the sensor's reading
2. Registry read, for reading the robot's position and orientation from the IR cameras
3. Algorithm, which calculates the motion planning commands for all robots
4. Communication, which transmits localization data and motion commands to the robots
5. Log, which saves path coordinates

The modules run either by a looping thread or by a timer thread. The timing of the timer threads was predetermined to achieve best performance while avoiding execution delays. The looping threads were executed in a loop, some inserting a delay (sleep) for best performance. The timings for each iteration of the non-timer modules were measured using the Stopwatch class. The timings of the different modules are presented in Table 3.3.

Another timer is in charge of communication synchronizing assurance with a timing of 100[ms]. Table 3.3 asserts that the time between each communication message received for each robot depends

Table 3.3: Threads timing

	Name	Type	Avg. timing	Min. timing	Max. timing	Sleep
1	GUI	Timer	100	-	-	-
2	Registry Read	Thread	10	10	10	10
3	Algorithm	Thread	1.03	1	2	1
4	Communication	Thread	4.79	4	6	1
5	Log	Timer	250	-	-	-

on the number of robots. For example, when the number of robots is 4, the time between communications received is 20[ms] on average (24[ms] at most). Since the robot's on-board software is implemented in such a way that the robot executes the command it received through communication, it can be said that the timing of the robot's motion commands update equals $5Xn$ [ms] where n is the number of robots.

Log

The log module saves a global results file, a path file, a sequence of screen capture pictures, and a local results file for each experiment. The log data is saved in a folder whose name is composed from the current experiment parameters, for example, for an experiment in the library environment, searching for target no. 2 with 3 robots, having linear beta distribution, the name for the first repetition will be "librT2R3bL-1".

The path history of each robot is saved in a file named like the folder, with the name ending in "path". Every 250[ms] it saves the coordinates and angle of each robot, along with the time from the beginning. Moreover, it calculates the path length and the current speed.

The local results file saves the following for each robot: the number of D steps, the current disc, the path length when the target was found, and the average and the maximal speed. The number of the robot that found the target is recorded along with the disc in which the target was found, as are the number of steps and total time in minutes, seconds, and total seconds. This file is named like the folder, with the ending "res" added to its name.

Every 500[ms], the log module saves a picture of the GUI part that presents the environment with a grid, the robot's positions, their next grid cell, and their bounding discs. These pictures are saved in .PNG format, which conserves space. The .PNG files are numbered with steps of 5, and saved within a folder named like the parent folder with the addition of "PNG" at the end of its name.

The global results file saves the information of all the experiments. For each experiment it saves a line that logs the following parameters: the environment, the target number, the number of robots, their velocity distribution (beta), the repetition number, which robot reached the target, in which disc, how many steps, the total time in seconds, and the average speed of the robot that found the target.

3.6.6 Parameters

Experiments were conducted in multiple environments using several configurations. The first stage was setting up the test bed and testing two robots in a simple environment, i.e., one that was free from obstacles. Next, two more robots were added and the algorithms were tested in environments

as congested as the environments in the simulation according to the previously defined congestion measure. The different parameters evaluated are presented in Table 3.4

Table 3.4: Experiments Parameters

Name	Symbol	Range
No. of Robots (Pairs in <i>HMRBUG</i>)	n	$n = 1, 2, \dots, 4$ (1, 2)
Velocities Distributions	β	$\beta_1, \beta_{Linear}, \beta_{Max}$
Environment	env	$i = 1, 2, 3$ ("free", "cave", "libr")
Target positions (Start-Target in <i>HMRBUG</i>)	\mathcal{T}_i ($\mathcal{S}_i - \mathcal{T}_i$)	$i = 1, 2, 3$, ($i = 1, 2$)
Repetitions	rep_i	$i = 1, 2$

Number of robots

The total number of robot platforms for experiments is 4. Thus, for the *HMRSTM* algorithm the number of robots, n , varies from between 1 and 4, where single robot experiments were conducted for purposes of comparison. For the *HMRBUG* algorithm the number of robots pairs, n , varies from between 1 and 2 (2 to 4 robots).

Velocity distribution

Velocity distributions in experiments were evaluated according to the simulations. Thus, the three sets of β -s tested for each run are:

1. β_{Linear} , "beta-linear": β -s with values distributed linearly
2. β_{Max} , "beta-max": β_n with maximal value and the rest of the β -s close to 1
3. β_1 , "beta-homogeneous" : equal β -s

Environments

Three environments in which the robots search for a path to the target are evaluated. The first environment is free from obstacles and denoted "free". The second environment is lightly congested and populated with 7 rock-like obstacles, and denoted as "cave". The third environment, denoted as "library," is a highly congested office-like environment (Fig. 3.4 - 3.12).

Target positions

In an unknown position (*HMRSTM*), three different target positions were tested. The three target positions are shown in Fig. 3.4-3.8.

In a known position (*HMRBUG*) two different start and target positions were tested. (3.11)

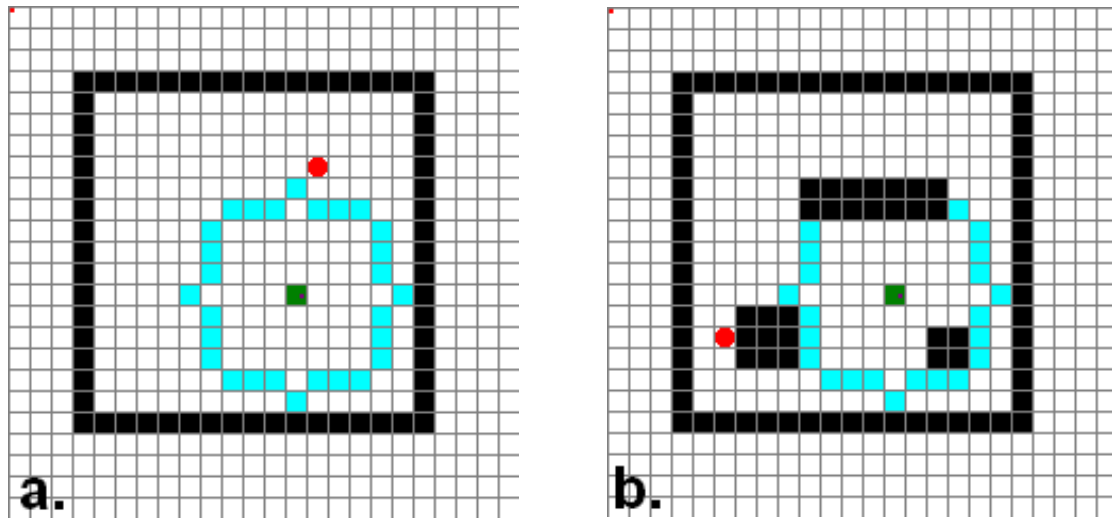


Figure 3.4: a. *HMRSTM* target 1 in "free" environment, b. *HMRSTM* target 2 in "cave" environment.

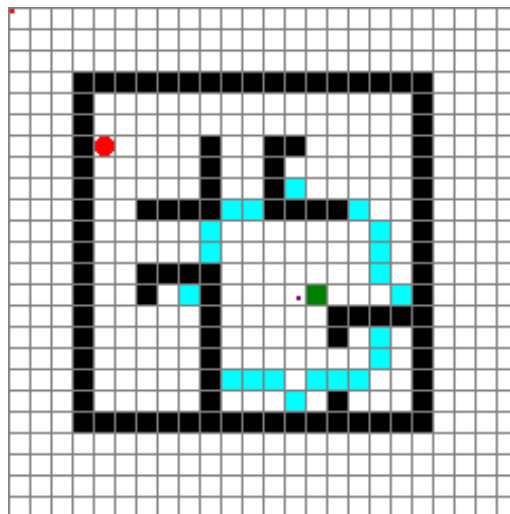


Figure 3.5: *HMRSTM* target 3 in "library" environment.



Figure 3.6: *HMRSTM* target 1 in "free" env.

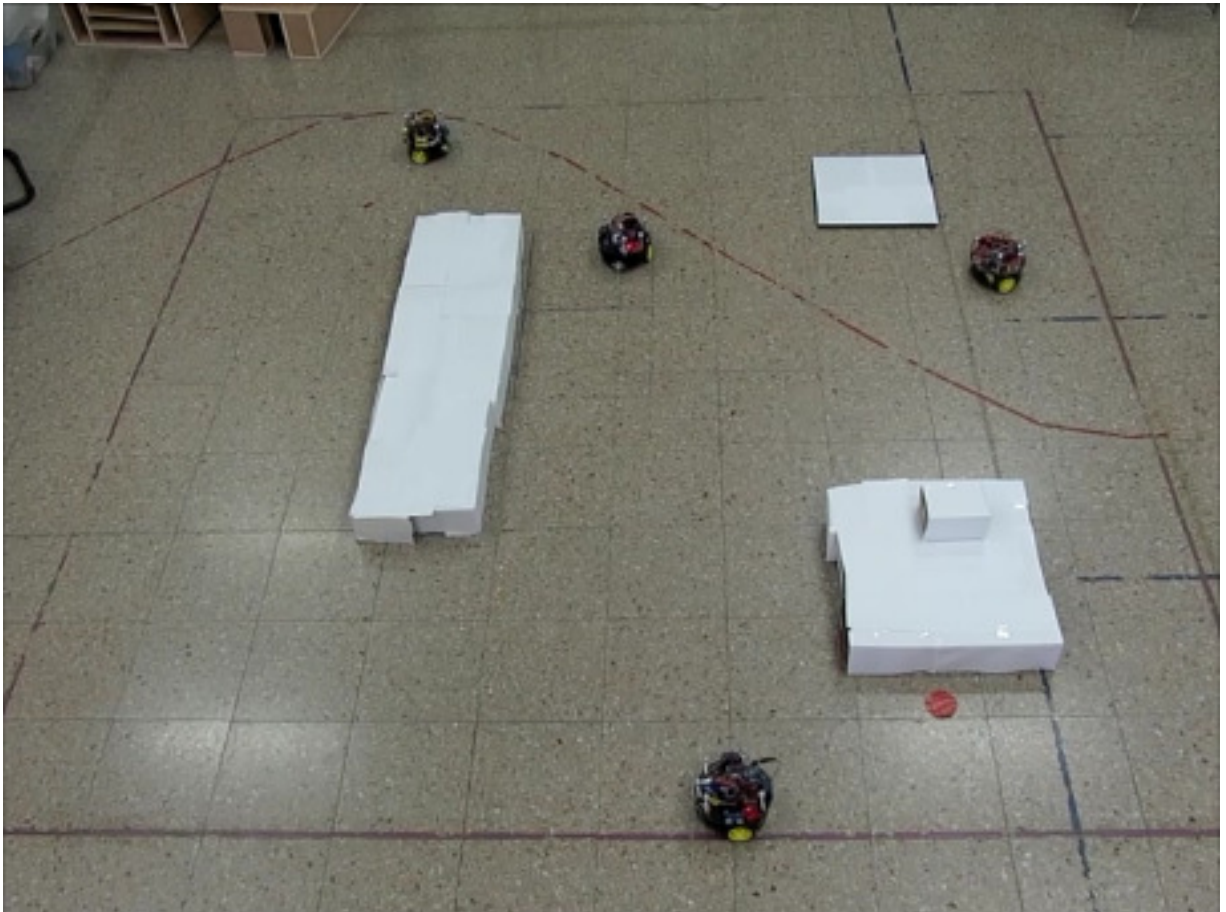


Figure 3.7: *HMRSTM* target 2 in "cave" env.

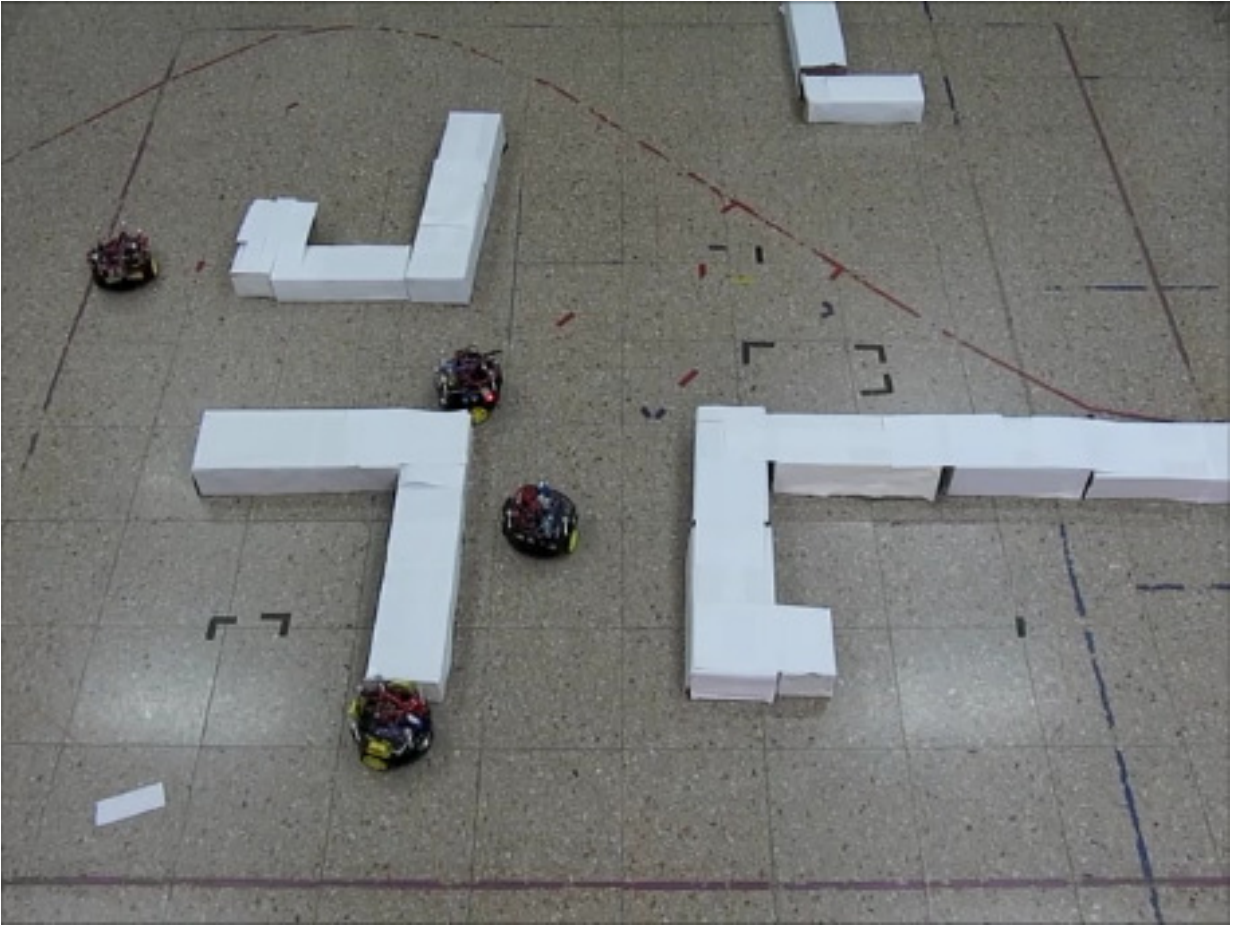


Figure 3.8: *HMRSTM* target 3 in "library" env.

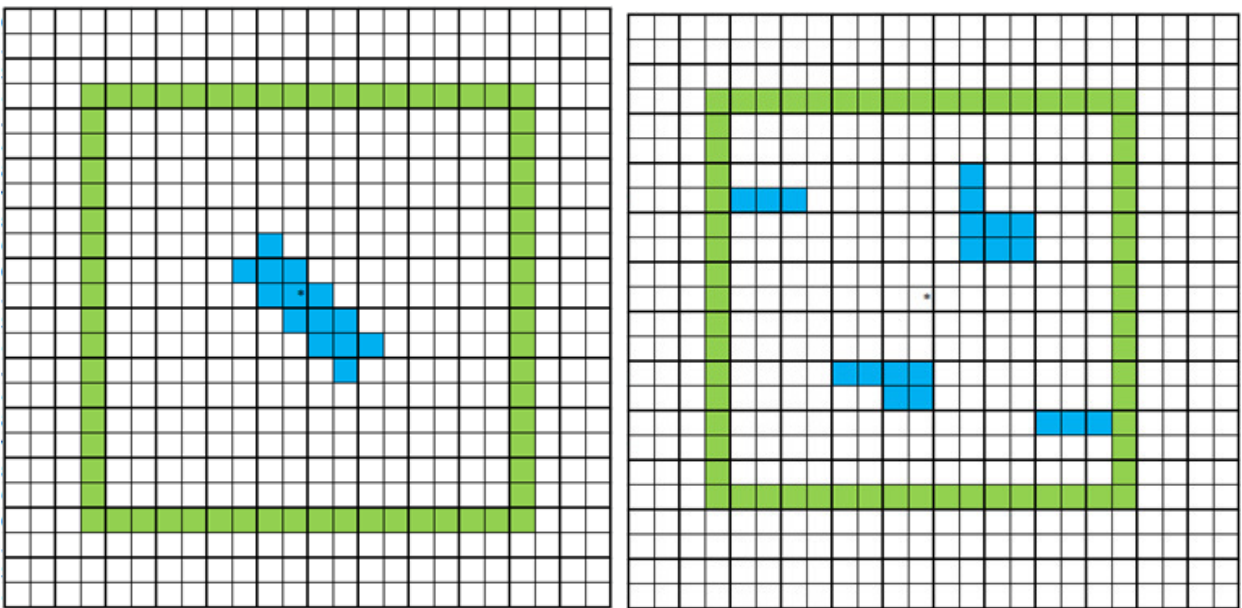


Figure 3.9: *HMRBUG* "cave" and "lib" environments



Figure 3.10: *HMRBUG* Start-Target 1 in "cave" environment

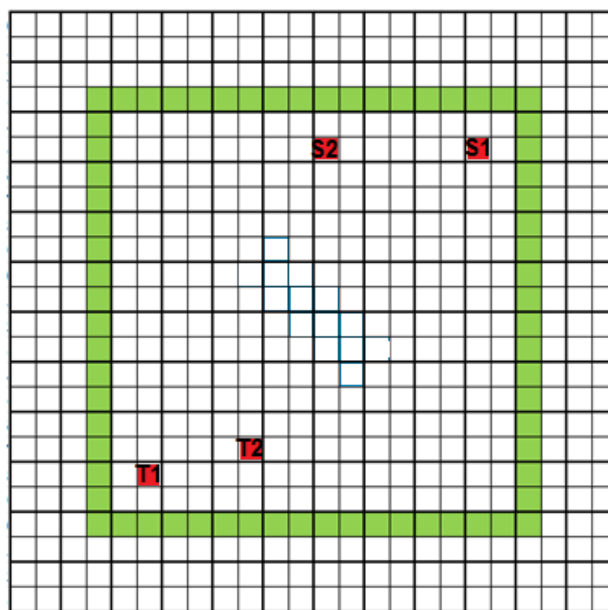


Figure 3.11: *HMRBUG* Start-Target 1,2



Figure 3.12: *HMRBUG* Start-Target 2 in "lib" environment

3.6.7 Performance Measure

The main performance measure in the experiments is similar to the performance measure in the simulation part, that is, the time to find the target. Since we are experimenting with a real physical system, time is measured in seconds. However, since hardware limitations cause deviation from the expected search time, another performance measure is examined, the path length, which is proportional to time.

3.6.8 Validation

The simulation software idealizes all parameters. Several errors are caused in real-world conditions, the most important ones being localization, communication, collision avoidance, and obstacle detection.

For validation, the simulation was executed with environments equivalent to the experiments' environment in size and geometry. The measured and the expected path length and time were compared. The robots in the experiment deviated from the expected search time and path due to hardware limitations that imposed implementation restrictions.

A deviated path was calculated according to the hardware limitations and implementation specifications. Subsequently, the maximal difference factor between the simulation and the experiment path length was yielded. Validation was performed for all experiments, showing that all results are within the calculated bounds of the maximal differences.

HMRSTM Validation

In *HMRSTM*, movement is always between adjacent grid cells, hence, only movements of changing orientation with 90 degrees and movements forward 200[mm] are performed. The robots receive the center of the next cell coordinates as their next target, along with their position and orientation. The onboard micro-controller calculates the angle to the target, directs the robot in place toward it, and then moves forward until it reaches the target, and so on. Since there are hardware limitations, each robot, in the worst case, performs a path that goes diagonally to the corner of the grid cell, and then continues diagonally to the target in the next cell, as shown in Figure 3.13.

HMRBUG Validation

The application of the *HMRBUG* algorithm in the robots yields two factors for deviation from the actual path. The first is that when traveling to a point of known coordinates, the robot heads toward the target, where a deviation of 15 degrees is allowed for each direction. The second is caused by the implementation of the obstacle's edge following. Here, in the worst case, the robot's path will have the shape of half circles of the radius $R = 2D$ around the obstacles, forming a round petal-like path (Figure. 3.13). The second factor is much more significant and can yield a path whose length for a real robot is greater than it is in simulation by a factor of 1.57.

Consider a round obstacle following path of a circular shape with a radius of r . Its perimeter is $P = 2\pi r$. In the implementation, while following obstacle's boundary, each robot tries to keep its position from the boundary at least $0.5D$ and at most $1.5D$. Hence, the followed parameter length equals

$$P^* \leq 2\pi(r + 1.5D) = 2\pi r + 2\pi \cdot 1.5D = P + 3\pi D$$

3. Met

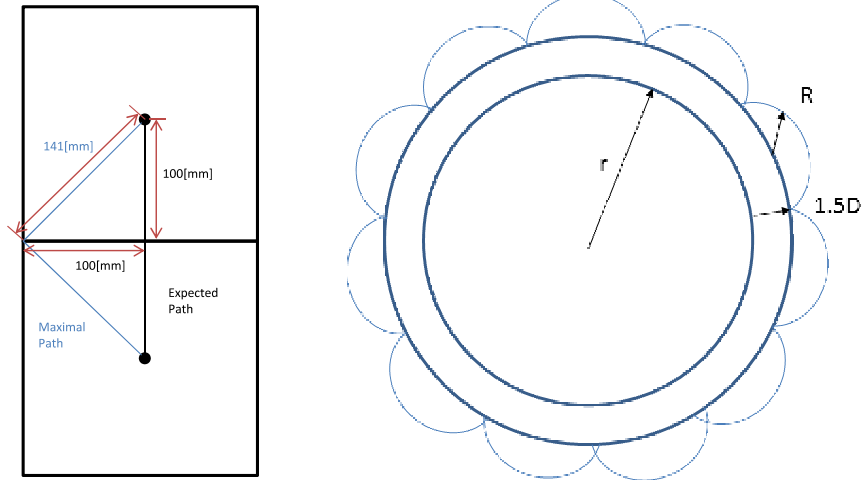


Figure 3.13: HMRSTM and HMRBUG validation.

Now, adding the half circles of radius R , within each new perimeter there are $\frac{P^*}{2R}$ such half circles. Thus, the path length of the half circles equals

$$P^{\star} = \frac{P^*}{2R} \cdot \frac{2\pi R}{2} = P^* \frac{\pi}{2}$$

Substituting p^* and simplification yields,

$$P^{\star} \leq (P + 3\pi D) \frac{\pi}{2} = \frac{\pi}{2}P + 1.5\pi^2 D \approx 1.57P + 1.5\pi^2 D$$

For $D = 200$ [mm], the path length is bounded by

$$P^{\star} \leq 1.57P + 2961 \text{ [mm]}$$

Where P is the obstacles' perimeter.

Chapter 4. HMRSTM Algorithm

4.1 Introduction

Finding a target whose position is unknown is a significant problem in many applications (e.g., search and rescue, de-mining [49], planetary exploration missions [10] and surveillance [45]). In this chapter, the above problem for a group of velocity-heterogeneous robots is defined and assumptions regarding it are presented. Next, the problem's complexity is analyzed, and a lower bound is found. Following this, the *HMRSTM*, *Heterogeneous Multi-Robot Search Time Multiplication* algorithm, which solves this problem, is introduced. Its performance is analyzed in terms of time competitive complexity. Since the lower bound on the search time for any algorithm that solves the problem and the upper bound of the search time using the *HMRSTM* algorithm belong to the same competitive complexity class, the problem's complexity can be classified into that class. Finally, the algorithm is proved to be optimal in terms of its time competitive complexity.

4.2 The Problem's Definitions

The motion planning problem explored is defined as follows: A target must be found by a group of n robots in an unbounded, unknown two-dimensional environment while the target position is unknown. The target can be identified using a specific target detector mounted on each of the robots, for example, a metal detection sensor for metal mines. The target is detected only when the robot is positioned directly above it. The size or diameter is uniform for all the robots and is D , a property important for the performance analysis. The robots are heterogeneous in their velocities, so that the slowest robot has a velocity v , and each robot j has a velocity $v_j = \beta_j \cdot v$, *s.t.* $\beta_j \geq 1$. We use the following definitions of *generalized time competitiveness* (1) and *time competitive complexity class* (2), which extend *generalized competitiveness* and *competitive complexity class* ([19]):

Definition 1 (Generalized Time Competitiveness [54]). *An on-line algorithm solving a task P in time T is $f(t_{opt})$ time competitive when T is bounded from above by a scalable function $f(t_{opt})$ over all instances of P , and t_{opt} is the optimal off-line solution achieved when all the information about the environment's geometry is known. In particular, $T \leq c_1 t_{opt} + c_0$ is the linear time competitiveness, while $T \leq c_2 t_{opt}^2 + c_1 t_{opt} + c_0$ is a quadratic time competitiveness, where the c_i 's are positive constant coefficients that depend on the robots' size D , the robots' velocity, the number of robots, and the geometry of the environment.*

The meaning of scalability is as follows: when performance is measured in physical units such as seconds *sec*, one must ensure that both sides of the relationship $t \leq f(t_{opt})$ possess the same units, so that change of scale does not affect the bound. For instance, the coefficient c_2 in the relationship $t \leq c_2 t_{opt}^2 + c_1 t_{opt} + c_0$ must have units of sec^{-1} , c_1 must be without units, and c_0 must have units of *sec*.

Note that the definition of $f(t_{opt})$ *time competitiveness* focuses on a particular algorithm that solves the task P . However, the objective is to characterize the lowest upper bound that can be achieved over all on-line algorithms for P . This objective requires a universal lower bound on the worst- case

performance of all on-line algorithms for P . If the lower and upper bounds satisfy the same functional relationship, that functional relationship is associated with P itself. Competitive complexity class is composed of two bounds, a universal lower bound, which means that no algorithm exists to solve the problem that can perform better than this bound, and an upper bound of an existing algorithm that solves the aforementioned problem, which is within the same performance function. An algorithm can be proved to be optimal if it can be shown that it performs in the bounds of the problem, if the problem can be classified into a competitive complexity class. The following definitions formalize the previous section:

Definition 2 (Time Competitive Complexity Class). *A lower bound on the time competitiveness of a task P is a lower bound $T \geq g(t_{opt})$ over all on-line algorithms for P under worst-case conditions, where T is the time it takes an algorithm to complete the task. If a time competitive upper bound $f(t_{opt})$ and a universal lower bound $g(t_{opt})$ for P are of the same functional relation up to constant coefficients, this function is said to be the **time competitive complexity class** of the task P .*

4.3 Time competitive complexity lower bound

In this section, a lower bound for the problem will be introduced. This lower bound proves that no algorithm can perform better than that bound. For this purpose, a very difficult environment, depicted in Fig. 4.1.(a) is used. The environment is circular and composed of radial corridors emanating from the start point S . Each part of the obstructed environment is further replaced with more corridors as seen in Fig. 4.1.(b). The width of the corridors is the same as the size of the robots, D . The target is placed in one of the corridors, at the farthest edge from S , whose distance is marked R .

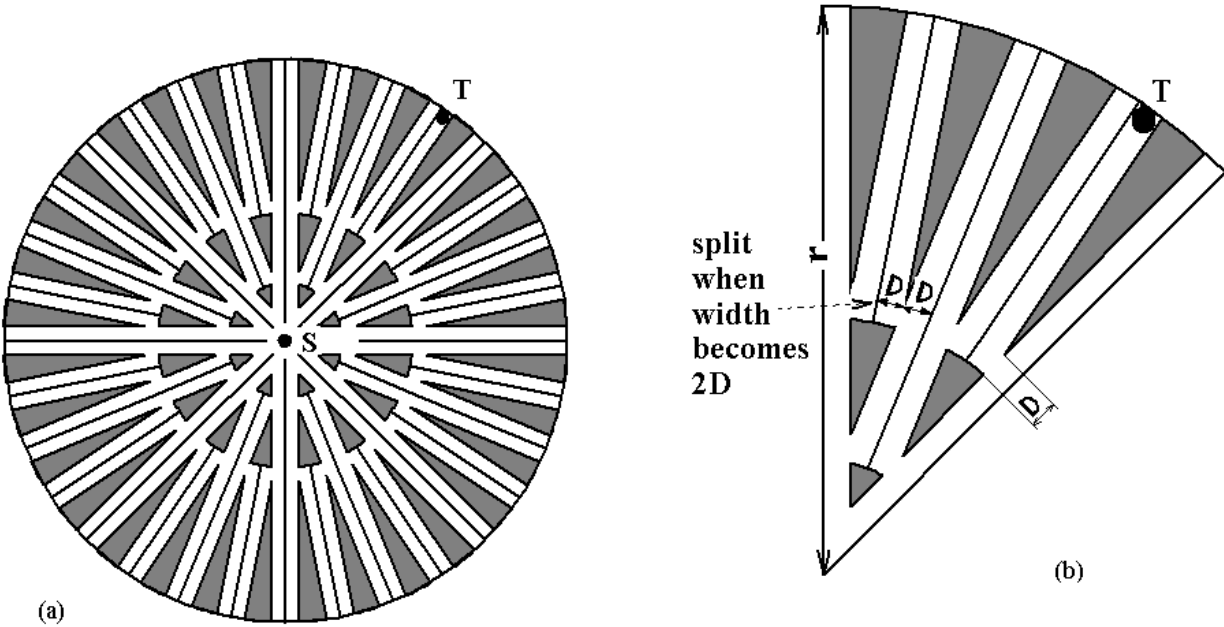


Figure 4.1: [19] (a) The radial corridor's environment. (b) Close-up view of the environment.

Lemma 4.3.1 (Lower Bound). *Let \mathcal{AL} be any navigation algorithm for n robots with velocities $v_j = \beta_j v$, for $j = 1, \dots, n$, where $\beta_k \geq \beta_j \geq 1$, $\forall k > j$, and, $\beta_1 = 1$, in an unknown planar environment, while recognizing the target only upon arrival. Let T_{LB} be the time it took the robot that found*

the target following the path generated by \mathcal{AL} . Let l_{opt} be the length of the optimal off-line path and t_{opt} be the time it took the fastest robot to travel l_{opt} . Let ϵ be an arbitrary small number. Then T_{LB} satisfy the quadratic lower bound,

$$T_{LB} > \frac{2\pi v_n}{3Dn}(1 - \epsilon)t_{opt}^2$$

Proof. In order to find the target, every corridor must be inspected. Assuming the number of corridors is much greater than n , the number of robots in the worst-case scenario, the robots will cover the whole free area of the environment at least once prior to reaching the target. This property is true for deterministic algorithms, since the target will be positioned at the last point the robots visited. If the algorithm is non-deterministic, at least one instance will produce this worst-case behavior. By construction, the obstructed parts cover almost one third of the total environment area, and have a limit of one third as R goes to infinity. Hence, the free area equals $(2/3)\pi R^2$. Thus, the total path length equals $2\pi R^2/(3D)$. In the best-case scenario, no robot traverses the same path of any other robot, and the time it takes for all the robots to cover the whole environment satisfies

$$T_{LB} > \frac{2\pi R^2}{3Dnv_n}$$

where all the robots are considered to be fast, and therefore the inequality is strong. The optimal off-line path length equals $l_{opt} = R + \epsilon'$, where the ϵ' is added due to the transitions between the corridors. Thus, the optimal solution satisfies $t_{opt} = (R + \epsilon')/(v_n)$. Substituting $R = l_{opt} - \epsilon'$, $R = t_{opt} \cdot v_n - \epsilon'$ yields,

$$T_{LB} > \frac{2\pi(l_{opt} - \epsilon')^2}{3Dnv_n} = \frac{2\pi(l_{opt}^2 - 2 \cdot l_{opt}\epsilon' + \epsilon'^2)}{3Dnv_n} = \frac{2\pi}{3Dnv_n}\left(1 - \frac{2\epsilon'}{l_{opt}} + \frac{\epsilon'^2}{l_{opt}^2}\right)l_{opt}^2$$

Substituting $\epsilon = \frac{2\epsilon'}{l_{opt}} - \frac{\epsilon'^2}{l_{opt}^2}$, $l_{opt} = t_{opt} \cdot v_n$ yields,

$$T_{LB} > \frac{2\pi v_n^2}{3Dnv_n}(1 - \epsilon)t_{opt}^2$$

Finally, simplification yields

$$T_{LB} > \frac{2\pi v_n}{3Dn}(1 - \epsilon)t_{opt}^2$$

□

4.4 The HMRSTM Algorithm

The *HMRSTM* motion planning algorithm, which uses multiple heterogeneous robots to search for a target whose position is unknown in an unknown environment is presented. The algorithm is based on our previously developed algorithm, *MRSAM* [57]. *HMRSTM* is introduced and explained, along with a formal description of the algorithm.

In *HMRSTM* the robots are heterogeneous in their velocities. *HMRSTM* deploys each of its robots to search for the target in concentric discs with growing areas. Each robot searches for the target within a disc by covering it. The coverage algorithm should be efficient, e.g., *STC* algorithm [17]. After a robot finishes searching for the target inside a disc, if it has not found the target, it moves to the next unoccupied search disc to search for it. The search is performed, again, by covering the reachable portions of the whole larger disc. Eventually, the search disc will contain a path to the

target if such a path exists and it will be found. Though each consequent disc contains the previous one, the series of the discs' areas form a converging geometric series, thus yielding an upper bound on the path length. The following conditions formalize the latter idea:

Condition 1 (Search disc's area ratio). *Each consequent search discs' area is greater than that of the previous disc.*

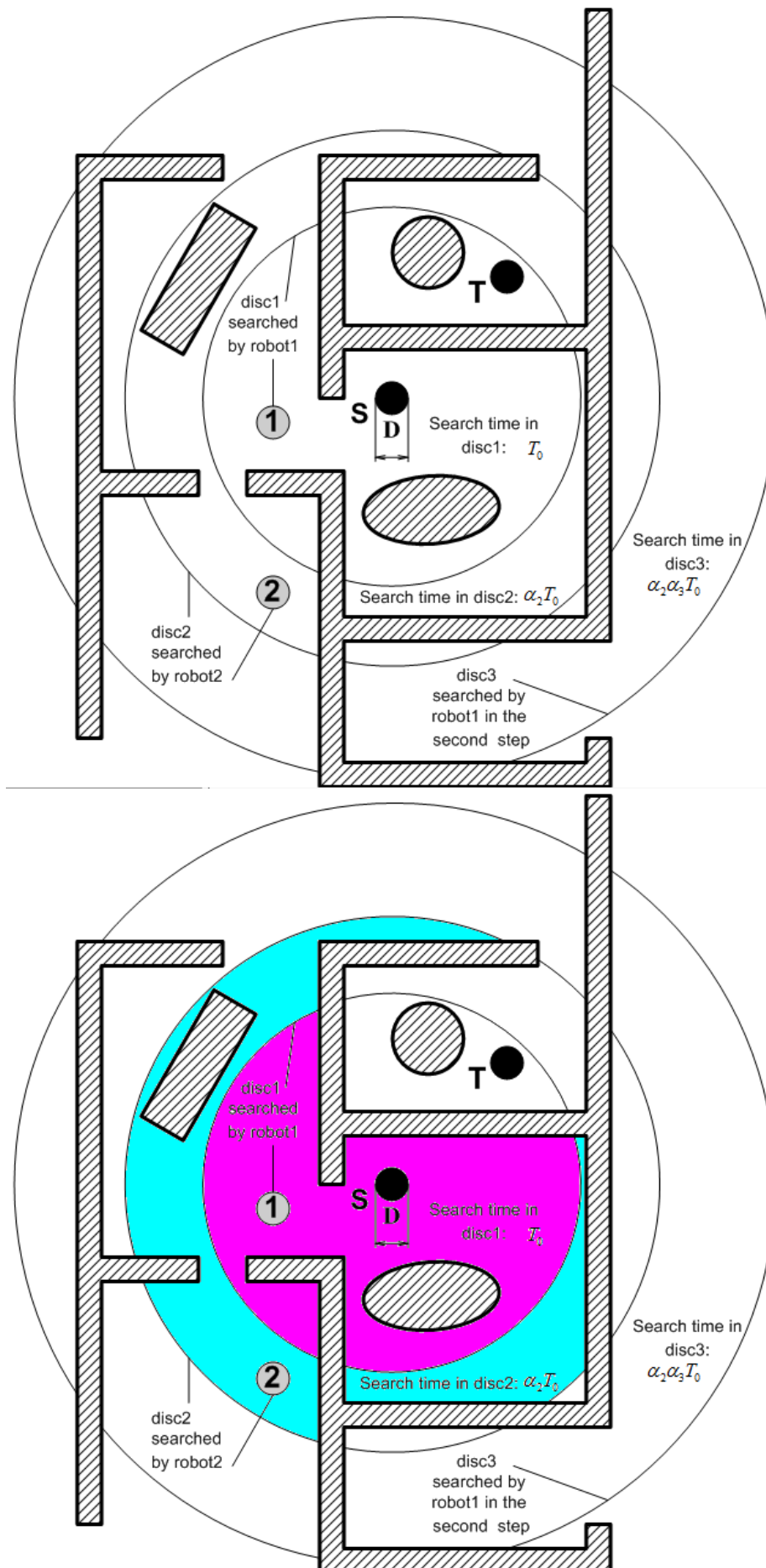
During identical time periods, robots with different velocities will cover different areas, and the ratio of the covered areas will be identical to the ratio of the velocities. Consequently, a fast robot might finish covering the next search disc before the slow robot has finished searching in the previous disc, thus, for *HMRSTM*, condition 1 does not suffice, and the following condition complements it.

Condition 2 (Search time ratio). *The time of search within each consequent search disc is greater than the time of search within the previous search disc.*

HMRSTM algorithm launches multiple robots from a common starting point S and assigns each robot j to a disc to search for the target T in it. All the discs are concentric and S is their center. n robots are deployed, and their velocities are $v_j = \beta_j v$, for $j = 1, \dots, n$, and $\beta_k \geq \beta_j \geq 1$, $\forall k > j$. The first robot, R_1 , is designated to search in the initial disc with search time T_0 . Each of the following robots starts its search within a disc whose search time is larger than the previous disc's search time by a factor of $\alpha_j > 1$. Each robot has its own multiplication factor $\alpha_j > 1$ according to its velocity, hence the search times within the discs will be, $T_0, \alpha_2 T_0, \alpha_2 \alpha_3 T_0, \alpha_2 \alpha_3 \alpha_4 T_0, \dots, \prod_{i=2}^n \alpha_i T_0, \prod_{i=1}^n \alpha_i T_0, \alpha_2 \prod_{i=1}^n \alpha_i T_0, \alpha_2 \alpha_3 \prod_{i=1}^n \alpha_i T_0, \dots, \prod_{i=2}^n \alpha_i \prod_{i=1}^n \alpha_i T_0, \dots$

For example, in Fig. 4.2, 4.3, *HMRSTM* deploys a group of two robots to search for the target. Robot 1 is initially assigned to search for the target inside a disc whose search time is T_0 and robot R_2 is assigned to search inside a disc of search time $\alpha_2 T_0$. After robot 1 finishes covering the entire portion of disc 1 that is accessible from S and fails to find the target, it starts searching for the target inside disc 3, whose search time is $\alpha_1 \alpha_2 T_0$. In this case, robot R_1 will find the target while searching in disc 3, before or after robot R_2 has completed searching in disc 2 and moved on to disc 4, whose search time is $\alpha_1 \alpha_2^2 T_0$. Each robot searches for the target in the accessible portion of the disc allocated to it until the target is detected, or until the entire region accessible from S is explored without finding T . The search process in each disc is the same as in our *MRSAM* algorithm and is based on the *STC* algorithm [57]. Fig. 4.4 contains the pseudocode of *HMRSTM* algorithm.

Implementation of the algorithm requires us to address communication and on-board memory issues. At algorithm initialization, each robot must receive its own id number, and the following parameters: n , the total number of robots, and their velocities, β_j 's. In this way, each robot can calculate its own multiplication factor, α_j , and know in advance in which discs it will be searching next. Upon termination, each robot must receive the stop signal. Thus, communication with and between the robots is not necessary during *HMRSTM* execution. The memory requirements for each robot are as follows: Each robot must remember its own id number, the number of robots, n , the velocities of all the robots ($n - 1$), the multiplication factors of all the robots ($n - 1$), and, during execution, the current search disc number. Additional memory requirements depend on the coverage method within a search disc. For example, if *STC* [17] is being used, each search disc is decomposed into a grid of D -sized cells, and each robot builds a spanning tree on-line and circumnavigates it. If

Figure 4.2: A group of two robots launched by *HMRSTM* searching the target.

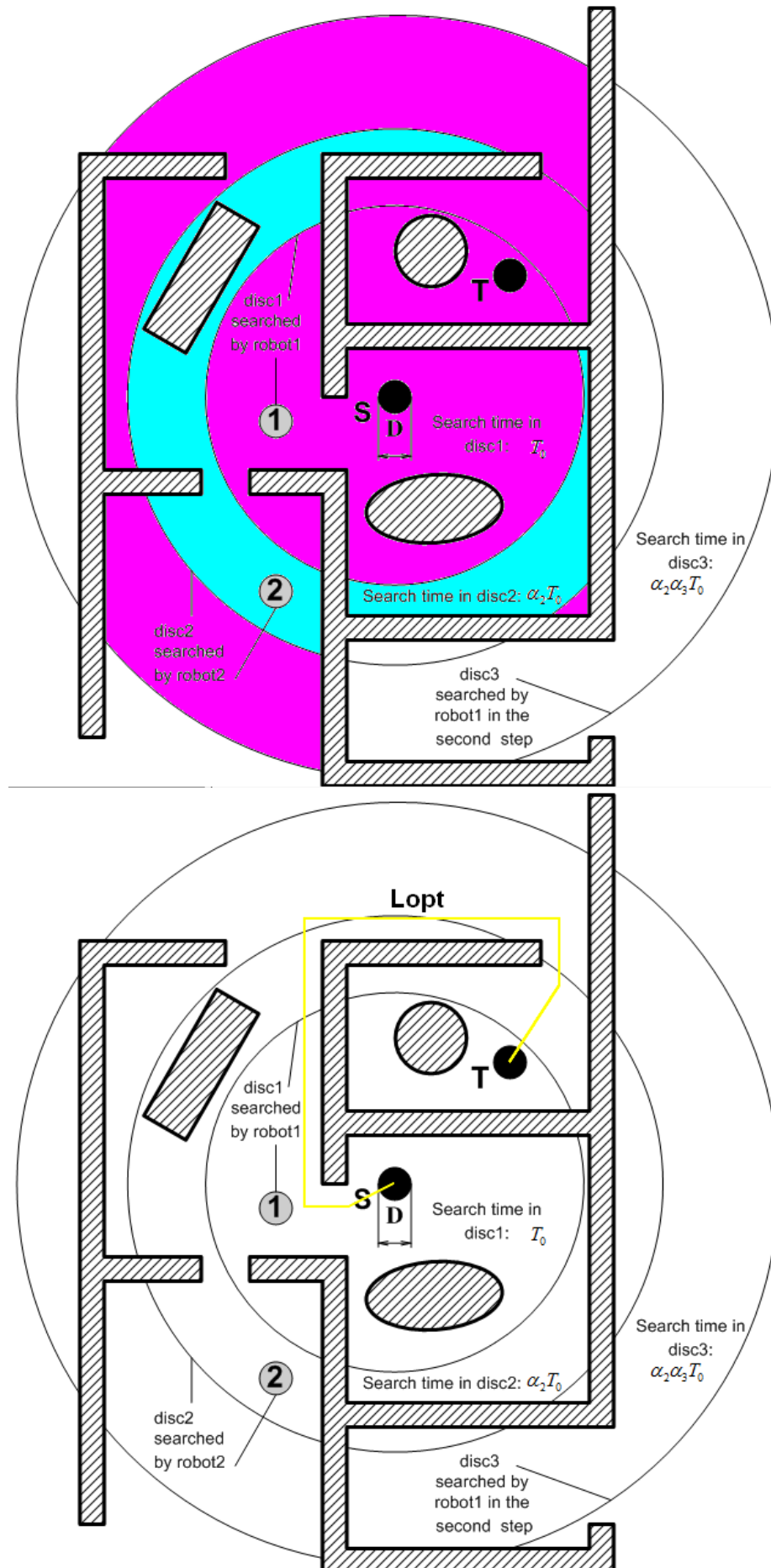


Figure 4.3: A group of two robots launched by *HMRSTM* searching the target.

Basic HMRSTM Algorithm's Pseudocode

Sensors: A position sensor

An obstacle detection sensor

A target detection sensor

Input: A start point S

An initial search time T_0

A group of n searching robots with different velocities,

$$v_j = \beta_j v, \quad i = 1, \dots, n \quad \beta_k \geq \beta_i \geq 1, \quad \forall k > i$$

Initialization:

For each robot R_j , $j = 1, \dots, n$:

Set multiplication factor $\alpha_j = \frac{(n+1)^{1/n} \beta_{j-1}}{\prod_{i=1}^n \beta_j^{1/n}}, \beta_0 = \beta_n, \beta_1 = 1$

Set initial search time $T_{1(R_j)} = T_0$, $j = 1$

$$T_{j(R_j)} = \left(\prod_{i=2}^j \alpha_i \right) T_0, \quad j \neq 1$$

For each robot j ,

Repeat:

Execute a *coverage tour* on the grid contained in the disc of search time T_j centered at \mathcal{S} . Scan each new free cell and its partially occupied neighbor cells for \mathcal{T} .

until one of the following occurs:

- (1) The target is reached: STOP.
- (2) If no new free cell is encountered during the current coverage tour:
STOP, the target is unreachable.
- (3) Otherwise, move to the next unoccupied disc k :
Set $T_{k(R_j)} = \left(\prod_{i=1}^n \alpha_i \right)^{\left(\frac{k-j}{n} \right)} T_0$, $j = 1$.
Set $T_{k(R_j)} = \left(\prod_{i=1}^n \alpha_i \right)^{\left(\frac{k-j}{n} \right)} \left(\prod_{i=2}^j \alpha_i \right) T_0$, $j \neq 1$.

End of Repeat loop

Figure 4.4: HMRSTM Pseudocode.

the number of unoccupied cells equals N , each robot must have a memory of the size of $O(N)$ in order to execute *STC* [17]. Moreover, in the worst case, N is quadratic in the number of cells composing the optimal off-line solution, l_{opt} . Since $t_{opt} = l_{opt}/(\beta v)$, is a linear relation, in the worst case, N is quadratic in the optimal off-line solution t_{opt} .

4.5 Analytical performance analysis

In order to analyze *HMRSTM*'s performance, the worst-case scenario is inspected. Generally, the target lies within a disc whose area is $A_{opt} = \pi l_{opt}^2$, where l_{opt} is the optimal off-line path length from S to T.

Proposition 4.5.1. *If the target T is reachable, HMRSTM finds the target using n robots and the travel time by robot j , which found the target, satisfies the quadratic inequality,*

$$T_{R_j} < \frac{\pi \alpha_j \beta_n^2 v (\prod_{i=1}^n \alpha_i)}{\beta_{j-1} D (\prod_{i=1}^n \alpha_i - 1)} t_{opt}^2. \quad (4.1)$$

where D is the robot size, β_j is the ratio between the velocities of robot j and the slowest robot, α_j is the multiplication factor, which is a function of n and of all the β 's, and t_{opt} is the optimal off-line solution.

Proof. In the first case, R_n searched and totally covered a disc whose area is $A_{opt} - \epsilon$ and search time $T_{opt_n} - \epsilon$ and thus did not find the target. Consequently, R_1 is assigned afterwards to search for it in a disc whose search time is $\alpha_1 T_{opt_n}$ (area is $\pi \alpha_1 l_{opt}^2$) and finds the target with time of

$$T_{i(R_1)} \leq \frac{\alpha_1 \pi (\beta_n v t_{opt})^2}{\beta_n v D} = \frac{\alpha_1 \pi \beta_n v}{D} t_{opt}^2, \quad (4.2)$$

where $T_{opt_n} = A_{opt}/(\beta_n v D)$, and $t_{opt} = l_{opt}/(\beta_n v)$.

The time to find the target by R_1 after searching in i discs equals the sum of the search times

$$\begin{aligned} T_{R_1} &\leq T_1 + T_{1+n} + T_{1+2n} + \dots + T_{i(R_1)} \\ &= T_0 + \left(\prod_{i=1}^n \alpha_i \right)^1 T_0 + \left(\prod_{i=1}^n \alpha_i \right)^2 T_0 + \dots + \\ &\quad + \left(\prod_{i=1}^n \alpha_i \right)^{\left(\frac{i-1}{n} \right)} T_0 \end{aligned} \quad (4.3)$$

The time series is a converging geometric series. According to

$$y + y\lambda + y\lambda^2 + y\lambda^3 + \dots + y\lambda^{w-1} = \frac{y(\lambda^w - 1)}{(\lambda - 1)}, \quad (4.4)$$

substituting $y = T_0$, $\lambda = (\prod_{i=1}^n \alpha_i)$, and $w - 1 = \frac{i-1}{n}$, its sum is

$$T_{R_1} \leq \frac{(\prod_{i=1}^n \alpha_i)^{\left(\frac{i-1}{n} + 1 \right)} - 1}{\prod_{i=1}^n \alpha_i - 1} T_0 < \frac{(\prod_{i=1}^n \alpha_i)^{\left(\frac{i-1}{n} + 1 \right)}}{\prod_{i=1}^n \alpha_i - 1} T_0 \quad (4.5)$$

Comparing the two expressions for the time to cover the last disc, $T_{i(R_1)}$, (4.2) and (4.3), T_0 can be expressed in terms of i ,

$$T_0 = \frac{\alpha_1 \pi \beta_n v}{D (\prod_{i=1}^n \alpha_i)^{\left(\frac{i-1}{n} \right)}} t_{opt}^2 \quad (4.6)$$

Substituting T_0 from (4.6) into (5.10) yields,

$$T_{R_1} < \frac{(\prod_{i=1}^n \alpha_i)^{\left(\frac{i-1}{n}+1\right)}}{\prod_{i=1}^n \alpha_i - 1} \frac{\alpha_1 \pi \beta_n v}{D (\prod_{i=1}^n \alpha_i)^{\left(\frac{i-1}{n}\right)}} t_{opt}^2.$$

Simplification yields

$$T_{R_1} < \frac{\alpha_1 \pi \beta_n v (\prod_{i=1}^n \alpha_i)}{D (\prod_{i=1}^n \alpha_i - 1)} t_{opt}^2. \quad (4.7)$$

Accordingly, for R_2 , in the worst-case scenario, R_1 covered a disc whose area is $A_{opt} - \epsilon$ and search time $T_{opt_1} - \epsilon$ and thus did not find the target. Consequently, R_2 is assigned afterwards to search for it in a disc whose search time is $\alpha_2 T_{opt_1}$ (area is $\pi \alpha_2 l_{opt}^2$) and finds the target in it with covering time of

$$T_{i(R_2)} \leq \frac{\alpha_2 \pi (\beta_n v t_{opt})^2}{\beta_1 v D} = \frac{\pi \alpha_2 \beta_n^2 v}{\beta_1 D} t_{opt}^2,$$

assuming $T_{opt_1} = A_{opt}/(\beta_1 v D)$, and $t_{opt} = l_{opt}/(\beta_n v)$, and $A_{opt} = \pi l_{opt}^2$.

The sum of the search times by R_2 is

$$T_{R_2} \leq T_2 + T_{2+n} + T_{2+2n} + \dots + T_{i(R_2)}$$

The series of time is a converging geometric series. After substituting $y = \alpha_2 T_0$, $\lambda = (\prod_{i=1}^n \alpha_i)$, and $w - 1 = \frac{i-2}{n}$, in (5.9), its sum is

$$T_{R_2} < \frac{\alpha_2 (\prod_{i=1}^n \alpha_i)^{\left(\frac{i-2}{n}+1\right)}}{\prod_{i=1}^n \alpha_i - 1} T_0 \quad (4.8)$$

Comparing the two expressions for the time to cover the last disc, $T_{i(R_2)}$, T_0 can be expressed in terms of i ,

$$T_{i(R_2)} = \frac{\pi \alpha_2 \beta_n^2 v}{\beta_1 D} t_{opt}^2 = \left(\prod_{i=1}^n \alpha_i \right)^{\left(\frac{i-2}{n}\right)} \alpha_2 T_0,$$

$$T_0 = \frac{\pi \beta_n^2 v}{\beta_1 D (\prod_{i=1}^n \alpha_i)^{\left(\frac{i-2}{n}\right)}} t_{opt}^2. \quad (4.9)$$

Substituting T_0 from (4.9) into (4.8) yields

$$T_{R_2} < \frac{\alpha_2 (\prod_{i=1}^n \alpha_i)^{\left(\frac{i-2}{n}+1\right)}}{\prod_{i=1}^n \alpha_i - 1} \frac{\pi \beta_n^2 v}{\beta_1 D (\prod_{i=1}^n \alpha_i)^{\left(\frac{i-2}{n}\right)}} t_{opt}^2.$$

Simplification yields

$$T_{R_2} < \frac{\pi \alpha_2 \beta_n^2 v (\prod_{i=1}^n \alpha_i)}{\beta_1 D (\prod_{i=1}^n \alpha_i - 1)} t_{opt}^2. \quad (4.10)$$

Accordingly, for R_3 ,

$$T_{R_3} < \frac{\pi \alpha_3 \beta_n^2 v (\prod_{i=1}^n \alpha_i)}{\beta_2 D (\prod_{i=1}^n \alpha_i - 1)} t_{opt}^2. \quad (4.11)$$

Accordingly, for R_4 ,

$$T_{R_4} < \frac{\pi \alpha_4 \beta_n^2 v (\prod_{i=1}^n \alpha_i)}{\beta_3 D (\prod_{i=1}^n \alpha_i - 1)} t_{opt}^2. \quad (4.12)$$

Repeating this process for cases where other robots reached the target leads to the generalization (4.1). \square

Lemma 4.5.1. *The time competitive complexity of HMRSTM is minimal when for each of the n robots deployed, the multiplication factor equals,*

$$\alpha_j = \frac{(n+1)^{1/n} \beta_{j-1}}{\prod_{i=1}^n \beta_i^{1/n}}, \beta_0 = \beta_n, \beta_1 = 1 \quad (4.13)$$

Proof. In order to find the optimal multiplication factors, α'_i s, a new objective function that combines the sums of all coverage times is formed:

$$\begin{aligned} T_{tot} &= T_{R_1} + T_{R_2} + \dots + T_{R_n} \\ &= \frac{\pi \beta_n^2 v (\prod_{i=1}^n \alpha_i)}{D (\prod_{i=1}^n \alpha_i - 1)} \sum_{i=1}^n \frac{\alpha_i}{\beta_{i-1}} \end{aligned} \quad (4.14)$$

Substituting $c = \frac{\pi \beta_n^2 v}{D}$ in T_{tot} (4.14) and differentiating according to each of the α'_i s,

$$\begin{aligned} \frac{\partial T_{tot}}{\partial \alpha_k} &= c \frac{\left(\prod_{i=1}^n \alpha_i \sum_{i=1}^n \frac{\alpha_i}{\beta_{i-1}} \right)' (\prod_{i=1}^n \alpha_i - 1)}{(\prod_{i=1}^n \alpha_i - 1)^2} - c \frac{\left(\prod_{i=1}^n \alpha_i \sum_{i=1}^n \frac{\alpha_i}{\beta_{i-1}} \right) (\prod_{i=1}^n \alpha_i - 1)'}{(\prod_{i=1}^n \alpha_i - 1)^2} = \\ &= c \frac{\left(\prod_{i=1}^n \alpha_i \sum_{i=1}^n \frac{\alpha_i}{\beta_{i-1}} \right)' }{(\prod_{i=1}^n \alpha_i - 1)} - c \frac{\left(\prod_{i=1}^n \alpha_i \sum_{i=1}^n \frac{\alpha_i}{\beta_{i-1}} \right) (\prod_{i=1, i \neq k}^n \alpha_i)}{(\prod_{i=1}^n \alpha_i - 1)^2} = \\ &= c \frac{(\prod_{i=1}^n \alpha_i)' \sum_{i=1}^n \frac{\alpha_i}{\beta_{i-1}} + \prod_{i=1}^n \alpha_i \left(\sum_{i=1}^n \frac{\alpha_i}{\beta_{i-1}} \right)'}{(\prod_{i=1}^n \alpha_i - 1)} - c \frac{\left(\prod_{i=1}^n \alpha_i \sum_{i=1}^n \frac{\alpha_i}{\beta_{i-1}} \right) \left(\frac{\prod_{i=1}^n \alpha_i}{\alpha_k} \right)'}{(\prod_{i=1}^n \alpha_i - 1)^2} = \\ &= c \frac{\left(\frac{\prod_{i=1}^n \alpha_i}{\alpha_k} \right) \sum_{i=1}^n \frac{\alpha_i}{\beta_{i-1}} + \prod_{i=1}^n \alpha_i \left(\frac{1}{\beta_{k-1}} \right)'}{(\prod_{i=1}^n \alpha_i - 1)} - c \frac{\left(\prod_{i=1}^n \alpha_i \sum_{i=1}^n \frac{\alpha_i}{\beta_{i-1}} \right) \left(\frac{\prod_{i=1}^n \alpha_i}{\alpha_k} \right)'}{(\prod_{i=1}^n \alpha_i - 1)^2} = \end{aligned}$$

comparing each function to zero and finding the common roots yields,

$$\alpha_j = \frac{(n+1)^{1/n} \beta_{j-1}}{\prod_{i=1}^n \beta_i^{1/n}}, \beta_0 = \beta_n, \beta_1 = 1$$

Substituting α_j back into $\frac{\partial T_{tot}}{\partial \alpha_k}$, yields

$$\begin{aligned} \frac{\partial T_{tot}}{\partial \alpha_k} &= c \frac{\left(\frac{\prod_{i=1}^n \alpha_i}{\alpha_k} \right) \sum_{i=1}^n \frac{\alpha_i}{\beta_{i-1}} + \prod_{i=1}^n \alpha_i \left(\frac{1}{\beta_{k-1}} \right)'}{(\prod_{i=1}^n \alpha_i - 1)} - c \frac{\left(\prod_{i=1}^n \alpha_i \sum_{i=1}^n \frac{\alpha_i}{\beta_{i-1}} \right) \left(\frac{\prod_{i=1}^n \alpha_i}{\alpha_k} \right)'}{(\prod_{i=1}^n \alpha_i - 1)^2} = \\ &= c \frac{\left(\frac{n+1}{\alpha_k} \right) \sum_{i=1}^n \frac{\alpha_i}{\beta_{i-1}} + \left(\frac{n+1}{\beta_{k-1}} \right)}{n} - c \frac{(n+1) \sum_{i=1}^n \frac{\alpha_i}{\beta_{i-1}} \left(\frac{n+1}{\alpha_k} \right)'}{n^2} = \\ &= c \frac{n+1}{\alpha_k n^2} \left[n \sum_{i=1}^n \frac{\alpha_i}{\beta_{i-1}} + \frac{n \alpha_k}{\beta_{k-1}} - (n+1) \sum_{i=1}^n \frac{\alpha_i}{\beta_{i-1}} \right] = \\ &= c \frac{n+1}{\alpha_k n^2} \left[n \left(n \cdot \frac{(n+1)^{\frac{1}{n}}}{\prod_{i=1}^n \beta_i^{\frac{1}{n}}} \right) + n \cdot \frac{(n+1)^{\frac{1}{n}}}{\prod_{i=1}^n \beta_i^{\frac{1}{n}}} - (n+1) n \cdot \frac{(n+1)^{\frac{1}{n}}}{\prod_{i=1}^n \beta_i^{\frac{1}{n}}} \right] = \\ &= c \frac{n+1}{\alpha_k n^2} \left[(n+1) \left(n \cdot \frac{(n+1)^{\frac{1}{n}}}{\prod_{i=1}^n \beta_i^{\frac{1}{n}}} \right) - (n+1) \left(n \cdot \frac{(n+1)^{\frac{1}{n}}}{\prod_{i=1}^n \beta_i^{\frac{1}{n}}} \right) \right] = \\ &= 0 \end{aligned}$$

□

Theorem 1. *The problem of on-line heterogeneous multi-robot navigation to an unknown target in an unknown and unbounded environment belongs to the quadratic time competitive complexity class.*

Proof. As defined in Definition 2, a time competitive complexity class is formed from a lower bound and from an upper bound for the problem. In section 4.3 we found a lower bound for the research problem that is quadratic in the optimal off-line solution, t_{opt} (4.3.1). Then we presented *HMRSTM* algorithm to solve that problem with an upper bound quadratic in t_{opt} (4.1). Thus, the research problem belongs to the quadratic time competitive complexity class. \square

The corollary below asserts that if a path from the start S to the target T exists, *HMRSTM* algorithm will find T .

Corollary 4.5.1. *HMRSTM is complete.*

Proof. The first important property established in Proposition 4.5.1, is that if the target T is reachable, *HMRSTM* will find it. The second property is that *HMRSTM* will find the target in a finite and limited time and is deduced from the upper bound of the algorithm introduced in Proposition 4.5.1. The two properties imply the completeness of *HMRSTM*. \square

The following corollary asserts that *HMRSTM* algorithm is optimal for the problem of finding a target with a group of heterogeneous robots:

Corollary 4.5.2. *HMRSTM is optimal in the sense of the search time.*

Proof. In Theorem 1 we proved that the search problem belongs to the quadratic time competitive complexity class. *HMRSTM* upper bound is time quadratic in the optimal off-line solution (4.1). Thus, *HMRSTM* is optimal up to constant coefficients. \square

The problem of reaching a target whose position is unknown in an a priori unknown and unbounded environment was presented. A lower bound for all algorithms that solve this problem was proved to be quadratic in the optimal off-line solution. *HMRSTM* algorithm for a group of heterogeneous robots was presented. *HMRSTM* algorithm was analyzed and its upper bound found to be quadratic in the optimal off-line solution. Consequently, the aforementioned problem was classified into the quadratic time competitive complexity class, and therefore, *HMRSTM* is optimal. The multiplication factor that minimizes the time of search was found and finally the algorithm was proved to be complete.

Chapter 5. HMRBUG Algorithm

5.1 Introduction

This chapter presents the problem of searching for a target whose position is known, in an environment that is unknown and unbounded. In such cases, a path to the target must be searched for and found. We investigate the use of multiple heterogeneous robots that differ in their velocities. The robots use tactile sensors for obstacle detection and each robot knows its position.

Based on *CBUG* [20] for a single robot search and our *MRBUG* [53] for a homogeneous group of robots, we introduce the *HMRBUG* algorithm [56]. *HMRBUG*, or *Heterogeneous Multi-Robot BUG* algorithm. *HMRBUG* uses a modified *BUG1* [43] technique to search for a path to a known target in an unknown environment. Since *BUG1* may have very poor performance in certain situations, such as long obstacles and unbounded environments, *HMRBUG* bounds *BUG1* search in ellipses whose focal points are the start and target positions. The uniqueness of the ellipse is that it is the locus of all points in the plane whose distances to the two focal points add up to a constant. Using an ellipse means searching with an equal chance for all paths that are of the same length from start to target. *HMRBUG* utilizes multiple heterogeneous robots by deploying pairs of robots in ellipses whose areas and search times grow until they reach the target. In each step, the robot pair's search is bounded to an ellipse. However, eventually, the search area grows until the robot reaches the target.

The structure of this chapter is as follows: First, we introduce the problem of path-finding by a group of robots that are heterogeneous in their velocity, and whose search environment is unknown and unbounded. In the next section, *HMRBUG*, a novel algorithm for the problem of path-finding by heterogeneous robots is introduced and explained. In the following section, *HMRBUG* performance is analyzed, we find its upper bound to be quadratic in the optimal off-line solution, and prove *HMRBUG* is complete and robust.

5.2 A Lower Bound for a Known Target

In this section we establish a universal lower bound on the competitive complexity of a group of heterogeneous robots navigating to a known target in an unknown environment. We would like to show that in difficult, close-to- worst-case scenarios, the robots will cover at least a certain area prior to finding the path to the target, and for this purpose we used the environment depicted in Figure 5.1. The environment is built from radial corridors of the same width as the robots, D , and one circular corridor containing the target with only one entrance. According to the construction of the environment [20], the obstacles occupy one third of the environment's area (Figure 5.1(b)).

This lemma and its proof are inspired by [53]. This lemma establishes a quadratic lower bound on the search time of all online navigation algorithms for a heterogeneous group of robots to a known target in an a priori unknown environment.

Lemma 5.2.1. *Let \mathcal{AL} be any navigation algorithm for n robots with velocities*

$v_j = \beta_j v$, where $\beta_j \geq \beta_{j-1} \geq 1$, $\forall j$, $j = 2, 3, \dots, n$, and $\beta_1 = 1$, in an unknown planar environment to a target whose position is known. Let T_{LB} be the time it took to the robot that reached the target

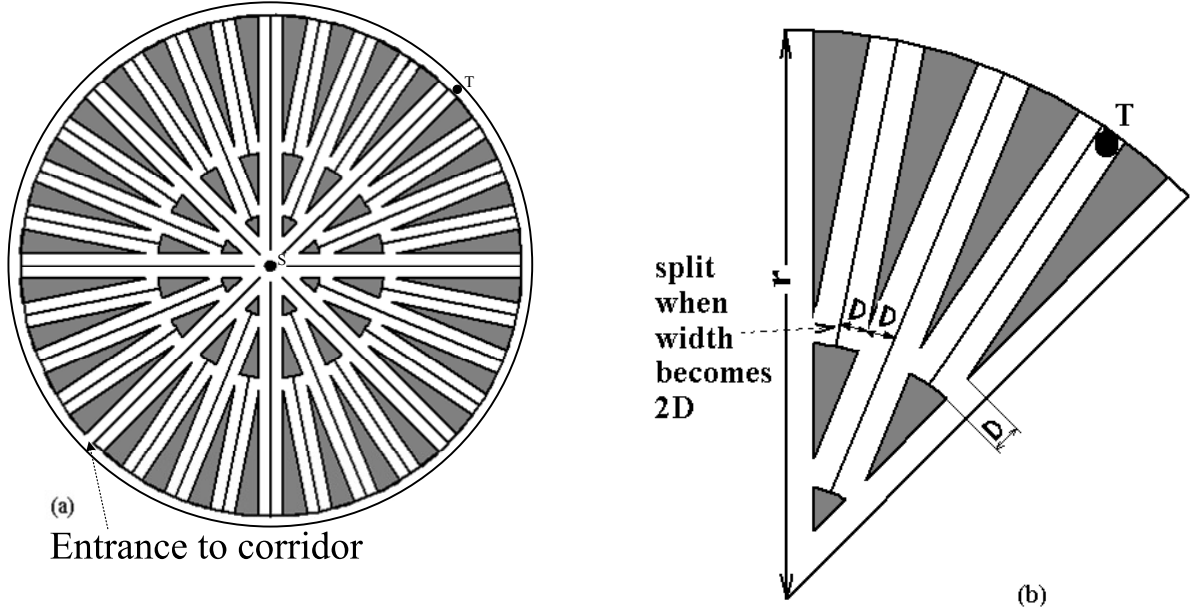


Figure 5.1: [19] (a) The radial corridor environment. (b) A close-up view of the environment

following the path generated by \mathcal{AL} . Let l_{opt} be the length of the optimal off-line path and T_{opt} be the time it took the fastest robot to travel l_{opt} . Then T_{LB} satisfies the quadratic lower bound,

$$T_{LB} \geq \frac{4\pi\beta v}{3nD(1+\pi)^2}(1-\epsilon)t_{opt}^2 \quad (5.1)$$

where D is the robot size and ϵ is a small positive number.

Proof. Consider the corridor environment with the target \mathcal{T} placed in the outer corridor, at a distance r from \mathcal{S} . Since the robots have no knowledge of the environment and no information regarding where the entrance to the outer corridor might be, they must in the worst case inspect every corridor, including all distal corridors. If \mathcal{AL} is deterministic, we can enforce this worst-case scenario by first watching the behavior of \mathcal{AL} , and then placing the entrance to the outer corridor in the last inspected radial corridor. If \mathcal{AL} is non-deterministic, we can only guarantee that one outcome of the algorithm would match this worst-case scenario. By construction, every distal corridor can be approached from \mathcal{S} along a simple radial path. Moreover, assuming the number of corridors in the outermost part is much greater than the number of robots, the robots must eventually move twice through every corridor of the environment—once in order to inspect a distal corridor and once in order to exit the corridor. An exception to this rule is the last corridor of each robot, which is considered below. The total area of the obstacles in the corridor environment is almost one third of the disc area, with the approximation becoming arbitrarily close to one third as the disc's radius increases. The total area inspected by the robots is therefore $\frac{2\pi r^2}{3}$. Since all corridors have a width of D , which is identical to the robots' size, the total length of the path traveled by the robots satisfies in the worst case the inequality $l_{tot} \geq \frac{4\pi r^2}{3D} - nr$, where the subtraction of nr is due to the last corridor for each robot, which need not be traced backward. A good algorithm will distribute the work equally between the robots, in such a way that none of the robots will travel any part of the path of the other robots. Thus, the

total length of the path traveled by one robot satisfies $l \geq \frac{4\pi r^2}{3nD} - r$. Since \mathcal{T} is placed in the circular outer corridor, $l_{opt} \leq (1 + \epsilon')(1 + \pi)r$, where ϵ' is a small positive number that is added due to the transitions between the corridors. This is a worst-case bound for cases where the target is placed in the circular corridor farthest from the entrance. It follows that $r \geq \frac{l_{opt}}{(1+\epsilon')(1+\pi)}$. On the other hand, $r \leq l_{opt}$ in the above-mentioned environment. Substituting the last two inequalities into the lower bound on l gives

$$l \geq \frac{4\pi}{3nD}r^2 - r = \frac{4\pi}{3nD(1+\epsilon')^2(1+\pi)^2}l_{opt}^2 - l_{opt}.$$

We can write the last inequality as

$$l \geq cl_{opt}^2 \left(\frac{1}{(1+\epsilon')^2} - \frac{1}{cl_{opt}} \right) = cl_{opt}^2 \left(1 - \epsilon'' - \frac{1}{cl_{opt}} \right),$$

where $c = \frac{4\pi}{3nD(1+\pi)^2}$ and $\frac{1}{(1+\epsilon')^2} = 1 - \epsilon''$. Since the quantity $\epsilon = \epsilon'' + \frac{1}{cl_{opt}}$ contains the quotient D/l_{opt} , which can be made arbitrarily small for sufficiently large environments, we obtain the lower bound $l \geq c(1 - \epsilon)l_{opt}^2$.

At best, all the robots are assumed to be fast, i.e., $v_j = \beta v$, $\beta = \max \{\beta_j\}$, $\forall j$, $j = 1, 2, \dots, n$. Thus, substituting

$$l \geq \frac{4\pi}{3nD(1+\pi)^2}(1 - \epsilon)l_{opt}^2$$

into $t_j = l_j/(\beta_j v_j)$ yields,

$$T_{LB} \geq \frac{4\pi}{3nD(1+\pi)^2\beta v}(1 - \epsilon)l_{opt}^2$$

Moreover, the optimal off-line solution is assumed to be $t_{opt} = l_{opt}/\beta v$. Thus,

$$T_{LB} \geq \frac{4\pi\beta v}{3nD(1+\pi)^2}(1 - \epsilon)t_{opt}^2$$

□

5.3 HMRBUG Algorithm

We now introduce the *HMRBUG* algorithm for finding a path to a known target. *HMRBUG* uses $2n$ robots (n pairs) with n different velocities, $v_j = \beta_j v$, $j = 1, \dots, n$, where $\beta_{j+1} \geq \beta_j \geq 1$, $j = 1, \dots, n-1$ and v is the velocity of each of the robots in the slowest robot pair and β_j is the ratio between the velocity of robot pair j and the velocity of the slowest robot pair, v .

HMRBUG solves the problem of finding a path to a known target using a group of robots. *HMRBUG* deploys each pair of robots to search for the target in a virtual bounding ellipse, and inside that ellipse uses our *PBUG1* algorithm [53] as a sub procedure. *PBUG1* algorithm is described in the following section.

The following conditions ensure that the search area and time of each consequent robot pair will grow in order to prevent redundant searches and eventually reach the target.

Condition 3 (Search ellipse area ratio). *The area of each consequent search ellipse is greater than the area of the previous search ellipse.*

During the same period of time, robots with different velocities will travel unequal path lengths, and the ratio of the traveled path lengths is the same as the ratio of the velocities. Consequently, a fast robot might finish searching the next search ellipse before the slow robot has finished searching in

the previous ellipse. Thus, for *HMRBUG*, condition 1 does not suffice, and the following condition completes it.

Condition 4 (Search time ratio). *The time of search within each consequent search ellipse is greater than the time of search within the previous search ellipse.*

5.3.1 *PBUG1* Motion Planning Algorithm for a Pair of Robots [53]

We now review our *PBUG1*, a version of *BUG1* [43] for a pair of robots that uses the same problem definitions as *BUG1*. In *PBUG1*, a pair of robots that starts from a common start point \mathcal{S} needs to find a path to a target \mathcal{T} whose position is known, in an unknown planar environment. The pair of robots will move together toward the target in a straight line until it hits an i^{th} obstacle at a point marked as *Hit point* H^i , $i = 1, 2, \dots$. At that point they split, robot R_L turns left, and robot R_R turns right, and they circumnavigate the obstacle from different directions, so that each robot encircles half of the obstacle's perimeter. While moving, each robot calculates and remembers the closest point on the obstacle's boundary to the target. Upon meeting, the robots compare the recorded information, decide which point is the closest to the target, join, and again move together to that closest point, which they mark as *Leave point* L^i , $i = 1, 2, \dots$. Finally, the robots continue to move together toward the target.

Setup and Definitions of *PBUG1*

The basic setup and definitions of *PBUG1* are the same as in [53]. *PBUG1* uses two mobile robots, R_L and R_R . Each has a different pre-defined local direction for moving around an obstacle, *left* and

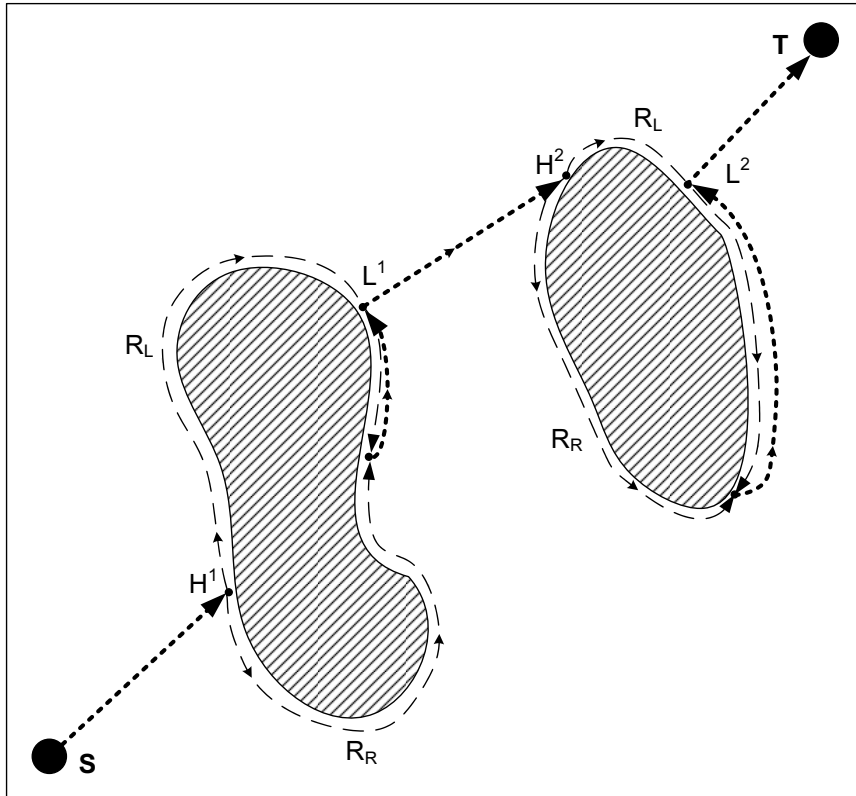


Figure 5.2: A pair of robots, R_L and R_R executing *PBUG1*. The dense dashed lines mark a mutual path of the two robots

right accordingly. The hit and leave points are common for both of the robots. The procedure *PBUG1* needs only one register for each robot, *Reg1*, which is used to store the coordinates of the current point, Q_m , of the minimum distance between the obstacle's boundary and the target. The robots compare their Q_m points and go together to the one with the smaller value.

5.3.2 Target Reachability Test

PBUG1 determines that the target is unreachable and trapped inside an obstacle using the *BUG1* method [43], which checks the direction to the target after circumnavigating an obstacle. If this direction points to the last obstacle, the target is surrounded by that obstacle, since the leaving point is the closest point to the target on the obstacle's boundary. A formal description of *PBUG1* algorithm follows.

***PBUG1* Algorithm**

Sensors: A position and orientation sensor.

An obstacle detection sensor

Input: Position of a start \mathcal{S} and a target \mathcal{T}

A pair of robots: R_L, R_R

Initialization: For each of the robots in the pair R_L, R_R :

Define local direction: *Left* for R_L , *Right* for R_R .

Set $i=1$.

Set initial leave point $L^0 = \mathcal{S}$.

For each of the two robots R_L, R_R , Repeat:

From the point L^{i-1} , move toward the target along a straight line until one of the following occurs:

- (1) The target is reached: STOP.
- (2) An obstacle is encountered: Define a hit point H^i .

Turn in the direction of the predefined local direction and follow the obstacle's boundary according to that direction. While circumnavigating the obstacle, calculate and record the coordinates of the closest point to the target, Q_{m_L} and Q_{m_R} for R_L and R_R respectively, until one of the following occurs:

- (a) The target is reached: STOP.
- (b) Upon meeting each other, exchange information and calculate the closest point to \mathcal{T} : $Q_m = \min(Q_{m_L}, Q_{m_R})$.

Define a new leave point $L^i = Q_m$.

Apply the test for target reachability:

- (i) If the target is not reachable: STOP.
- (ii) Otherwise, move to L^i : If $Q_m = Q_{m_L}$, trace back R_L path. Otherwise, trace back R_R path.

Set $i = i + 1$.

End of Repeat loop

5.3.3 HMRBUG Algorithm for a Heterogeneous Group of Robots

HMRBUG algorithm launches n pairs of robots from a common starting point \mathcal{S} and assigns each pair R_j to a different ellipse to search for a path to the target \mathcal{T} in it. Each ellipse's focal points are

\mathcal{S} and \mathcal{T} .

The first pair of robots R_1 is designated to the initial ellipse of search time T_0 , and each of the following robots starts its search in an ellipse of search time larger than the previous ellipse's search time by a factor of α_j , $\alpha_j > 1$. That is, the search times of the ellipses will be $T_0, \alpha_2 T_0, \alpha_2 \alpha_3 T_0, \alpha_2 \alpha_3 \alpha_4 T_0, \dots$. In the following example, depicted in Fig. 5.3, *HMRBUG* launches two pairs of robots, 1, 2 to search for a path to the target in an office-like environment. Each robot pair is initially assigned to a bounding ellipse, e_1, e_2 , to execute *PBUG1* in it, and each robot in a pair is assigned to a different local direction, *Left, Right*: $1_L, 1_R, 2_L, 2_R$. At first, the robot pairs move directly toward the target, and as they encounter an obstacle they split, each robot moving in its local direction. It can be observed that a part of the path is traversed by all the robots together at the same time, and the robots will move together as long as they are within the boundary of the first ellipse. While robot pair no. 2 is traversing the second ellipse, robot pair no.1 finishes traversing the obstacle's boundary, which lies inside the first bounding ellipse and the ellipse itself. After meeting each other, the robots of pair no. 1 move toward the closest point to the target they encountered while traversing the ellipse, where there they conclude that they cannot reach the target from ellipse no. 1. In Fig. 5.3(a), robot pair 1 is just about to execute *PBUG1* in ellipse no. 3, and robot pair 2 has already met on ellipse no. 2 and is on its way to the closest point to the target. While robot pair no. 1 is busy with its search in ellipse no. 3, robot pair no. 2 reaches the closest point to the target (Fig. 5.3(b)). Next, robot pair 2 is assigned to ellipse no. 4, and while searching in it robot pair no.1 meets on ellipse no.3 (Fig. 5.3(c)). While robot pair no. 2 searches in ellipse no. 4, robot pair 1 moves toward the closest point to the target (Fig. 5.3(d)) and from there it continues without any additional obstacles in its way and reaches the target.

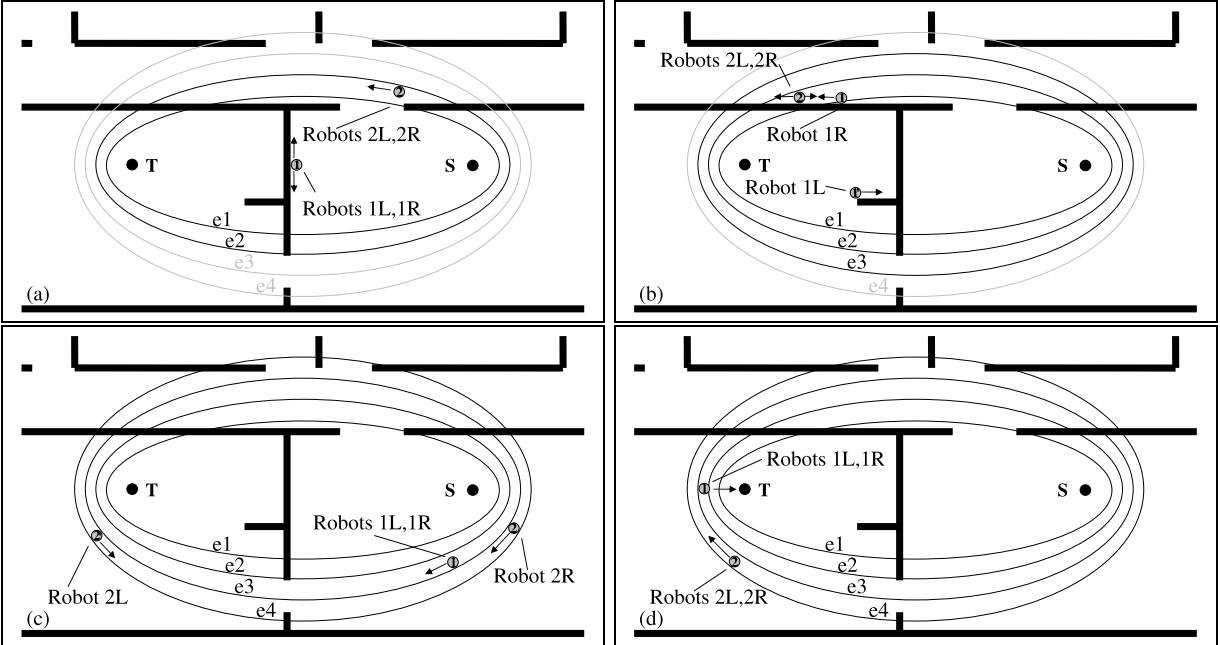


Figure 5.3: Execution example of *HMRBUG*

In *HMRBUG*, the execution of *PBUG1* regards the ellipse as a virtual obstacle's boundary. If the target is detected, the algorithm terminates; otherwise, the pair of robots repeats the process on the next unassigned ellipse in the series. A formal description of the basic algorithm appears in Fig. 5.4.

Basic HMRBUG Algorithm's Pseudocode

Sensors: A position sensor.

An obstacle detection sensor.

Input: Position of a start \mathcal{S} and a target \mathcal{T} points.

An initial ellipse with focal points \mathcal{S} and \mathcal{T}

An initial search time T_0

n pairs of robots $\{1_L, 1_R, 2_L, 2_R, \dots, n_L, n_R\}$

with different velocities, $v_j = \beta_j v$, $j = 1, \dots, n$

$\beta_{j+1} \geq \beta_j \geq 1$, $j = 1, \dots, n-1$, $\beta_1 = 1$

Initialization: For each robot pair R_j , $j = 1, \dots, n$:

Set global step $p = 1$

Set initial leave point $L_{e_j}^0 = \mathcal{S}$,

Set multiplication factor $\alpha_j = \frac{(n+1)(\frac{1}{n})\beta_j^{\frac{n-1}{j-1}}}{\prod_{k=2, k \neq j-1}^n \beta_k^{\frac{1}{n}}}$, where $\beta_{j-1} = \beta_n$ when $j = 1$

Set initial search ellipse parameters:

focal points are \mathcal{S} and \mathcal{T} ,

semi major axis¹ $a_0 = a_0(\alpha_j, T_0, \mathcal{S}, \mathcal{T}, \beta_j, v)$,

semi minor axis¹ $b_0 = b_0(a_0, \mathcal{S}, \mathcal{T})$.

For each robot pair R_j , Repeat:

Initialize *PBUG1* with the following parameters:

Create an outer virtual obstacle's boundary with current search ellipse.

Start point is \mathcal{S} , target is \mathcal{T} .

Set $i = 1$.

Leave point is L^0 .

Execute *PBUG1* until one of the following occurs:

(1) *PBUG1* terminates at \mathcal{T} : STOP, target is found.

(2) \mathcal{T} is trapped inside an obstacle:

(a) If the obstacle does not intersect the eR_j ellipse, STOP, the target is unreachable.

(b) Otherwise, move to the next unoccupied ellipse:

Set $p = p + 1$.

Set current search ellipse parameters:

semi major axis¹ $a_p = a_p(a_0, \alpha_j, p)$,

semi minor axis¹ $b_p = b_p(a_0, a_p)$.

Set L^0 at *PBUG1* termination point.

End of Repeat loop

¹Calculations of a_0, b_0, a_p, b_p is presented in section 5.4.

Figure 5.4: HMRBUG Pseudocode.

Before analyzing the time competitiveness of the algorithm, we make the following remarks: First, during initialization, after obtaining the values of n, T_0, \mathcal{S} , and \mathcal{T} , each robot is assigned to a number j and to a local direction, *Left* or *Right* and thus can calculate its future search ellipse parameters, which means that after a pair of robots has finished searching for a path in an ellipse, it can immediately continue to search in the next ellipse, regardless of the state of the other robots. Second, the method *PBUG1* used to determine that the target is unreachable and trapped inside an obstacle in step 2 is discussed in subsection 5.3.2. *HMRBUG* ensures in step 2a that the robots are not bounded by the ellipse and thus guarantee that the target is unreachable. Third, regarding the memory requirements, in *HMRBUG*, each robot executing *PBUG1* uses the same amount of memory as in *BUG1* with slight modification, plus a constant amount of memory. The target position \mathcal{T} , the current obstacle's hit point, the distance of the closest point on the current obstacle's boundary to the target and the two distances to that point along the obstacle's boundaries from its current position are *BUG1* necessities. *PBUG1* memory modification is evident in the last requirement, where these two distances to the leave point are not necessary and should be recorded by each robot, as was discussed in Subsection 5.3.1. Additional memory requirements of *HMRBUG* are the start position \mathcal{S} and the current ellipse's parameters a_p, b_p .

5.4 *HMRBUG* Upper Bound Analysis

In this section, the performance of *HMRBUG* is analyzed and an upper bound on the traveling time to reach the target is formed. First, two conditions regarding the ellipses' areas and search times are formulated. These conditions assist the calculations and the convergence of the upper bound of *HMRBUG*. Next, using the ellipse's geometrical properties and the optimal off-line solution, the upper bound is achieved.

In the following two lemmas we use terms related to *Configuration Space*. The configuration

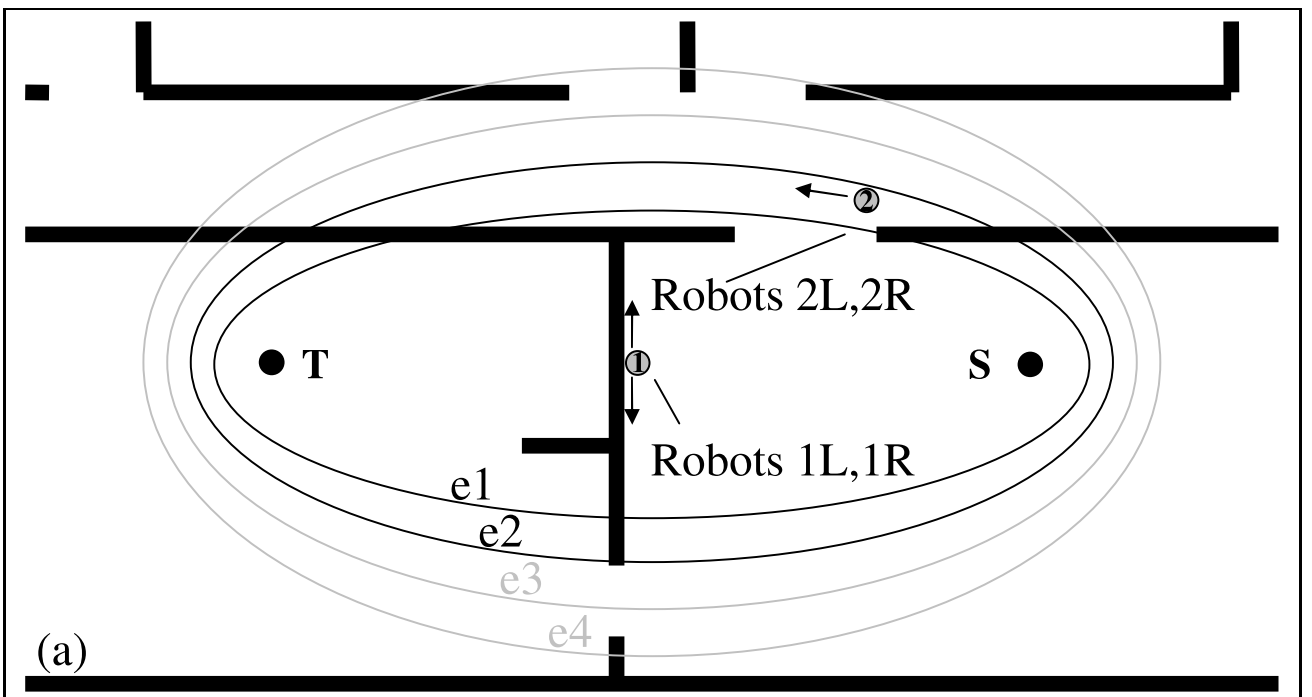


Figure 5.5: Two pairs of robots, executing *HMRBUG*.

space (or \mathcal{C} -space) of a disc-shaped robot is \mathbb{R}^2 , and the \mathcal{C} -space obstacle \mathcal{CB}_i consists of all robot configurations where it intersects the obstacle \mathcal{B}_i .

Definition 3 ([20]). *Let \mathcal{CB}_i be the \mathcal{C} -space obstacle induced by an obstacle \mathcal{B}_i for a disc robot of size D . The traceable obstacle induced by \mathcal{B}_i , denoted \mathcal{B}_i , is obtained by filling any internal holes in \mathcal{CB}_i and then shrinking \mathcal{CB}_i inward by a distance of $D/2$.*

Lemma 5.4.1 ([20]). *Let a planar environment contain z disjointed traceable obstacles \mathcal{B}_i , $i = 1, \dots, z$. Let a disc robot of size D trace the i^{th} obstacle's boundary, and let q_i be the total area swept by the robot during tracing of the i^{th} boundary. Let \mathcal{C} be any simple closed curve that surrounds the z regions swept by the robot. Then $\sum_{i=1}^z q_i \leq 4\mathcal{A}(\mathcal{C})$, where $\mathcal{A}(\mathcal{C})$ is the area of the traceable obstacle-free points enclosed by \mathcal{C} .*

Note that the regions swept during tracing of the individual boundaries may overlap, so that in general the sum $\sum_{i=1}^z q_i$ may be larger than $\mathcal{A}(\mathcal{C})$. The following lemma is written in the spirit of [53].

Lemma 5.4.2. *The path travel time t_{ji} of the the i^{th} ellipse traversed by each robot of the j^{th} pair searching for the path to the target is bounded by*

$$t_{ji} \leq 4 \frac{A_i}{\beta_j v D} + (||L_i^0 - \mathcal{T}|| - ||L_{i+n}^0 - \mathcal{T}||) / (\beta_j v), \quad (5.2)$$

where A_i is the area of the i^{th} ellipse, D is the size of each robot, $\beta_j v$ is the velocity of each robot of the j^{th} 's pair. L_i^0 is the start point at the i^{th} ellipse, n is the number of robot pairs, L_{i+n}^0 is the start point at the next ellipse of the pair of robots, and $||\gamma - \delta||$ denotes the Euclidean distance between γ and δ .

Proof. When moving toward the target in the i^{th} ellipse, each robot of the pair of robots assigned by HMRBUG to that ellipse is executing *PBUG1*. The regions swept by the robots during circumnavigation of obstacles in this ellipse (including the ellipse itself) are surrounded by the ellipse's boundary. Identifying the latter boundary with the curve $\mathcal{A}(\mathcal{C})$ from Lemma 5.4.1, the total path length of the two robots during circumnavigation of the obstacles is at most $4A_i/D$, where A_i is the i^{th} ellipse's area. Since each robot travels exactly half of the way, the path length of one robot is not more than $2A_i/D$. Recall now that under *BUG1*, the robot circumnavigates the boundary of each obstacle at most 1.5 times; here each of the two robots will circumnavigate the boundary of each obstacle only one time at most. Hence, the total length of each robot's path during boundary following is at most $4A_i/D$. Recall, too, that under *BUG1*, motion between obstacles is always conducted directly to the target. The total length of these motion segments equals the net decrease of the distance of the robot from \mathcal{T} , which is $||L_i^0 - \mathcal{T}|| - ||L_{i+n}^0 - \mathcal{T}||$. Adding the two terms and dividing by the robots' velocity gives the result. \square

The following lemma states that the last ellipse search time is bounded from above. This lemma and its proof are inspired by [53].

Lemma 5.4.3. *Let \mathcal{T} be reachable from \mathcal{S} . If the initial ellipse contains no path from \mathcal{S} to \mathcal{T} , robot pair j in HMRBUG reaches the target in an ellipse whose search time T_{f_j} is bounded by*

$$T_{f_j} \leq \alpha_j \frac{\pi t_{\text{opt}} \beta_n}{4\beta_{j-1} D} \sqrt{(t_{\text{opt}} \beta_n v)^2 - ||\mathcal{S} - \mathcal{T}||^2}. \quad (5.3)$$

where, t_{opt} is the travel time of the optimal off-line path from \mathcal{S} to \mathcal{T} by the fastest robot pair, α_j is the multiplication factor of the robot pair that reached the target, β_j is the velocity factor of robot pair j , and for $j = 1$, $\beta_{j-1} = \beta_n$, v is the slowest robot's velocity and D is the robots' width.

Proof. An ellipse with focal points \mathcal{S} and \mathcal{T} satisfies the inequality $\varepsilon = \{x : \|x - \mathcal{S}\| + \|x - \mathcal{T}\| \leq 2a\}$, where $2a$ is the length of the ellipse's major axis. Consider now the optimal off-line path from \mathcal{S} to \mathcal{T} of length l_{opt} . Every point x along this path satisfies the inequality $\|x - \mathcal{S}\| + \|x - \mathcal{T}\| \leq l_{opt}$. It follows that the entire optimal off-line path lies in an ellipse with focal points \mathcal{S} and \mathcal{T} and major axis $2a \leq l_{opt}$. Next, recall that the area of an ellipse is given by πab , where $2b$ is the length of the ellipse's minor axis. In an ellipse with focal points \mathcal{S} and \mathcal{T} , $\|\mathcal{S} - \mathcal{T}\|^2/4 + b^2 = a^2$. Hence, $b \leq \frac{1}{2}\sqrt{l_{opt}^2 - \|\mathcal{S} - \mathcal{T}\|^2}$, where we used the inequality $a \leq l_{opt}/2$. Let A_{opt} denote the area of the smallest ellipse with focal points \mathcal{S} and \mathcal{T} , which contain the optimal off-line path. Substituting the expressions for a and b in πab , the ellipse area computation formula gives the upper bound:

$$A_{opt} = \pi ab \leq \frac{\pi}{4} l_{opt} \sqrt{l_{opt}^2 - \|\mathcal{S} - \mathcal{T}\|^2}. \quad (5.4)$$

According to our assumption, the initial ellipse contains no path from \mathcal{S} to \mathcal{T} . Hence, *HMRBUG* multiplies the search time of the ellipse by a factor of α_j at least once.

We will examine the worst-case scenario for several cases in which different robots reached the target. In the first case, robot pair R_n searched for a path in an ellipse whose area is $A_{opt} - \epsilon$ and search time $T_{opt_n} - \epsilon$, substituting $t_{opt} = l_{opt}/(\beta_n v)$ or $l_{opt} = t_{opt} \beta_n v$, into (5.4) yields

$$A_{opt} \leq \frac{\pi}{4} t_{opt} \beta_n v \sqrt{(t_{opt} \beta_n v)^2 - \|\mathcal{S} - \mathcal{T}\|^2}, \quad (5.5)$$

Then substituting (5.5) into

$$T_{opt_n} = A_{opt}/(\beta_n v D),$$

yields,

$$T_{opt_n} \leq \frac{\pi t_{opt} \beta_n v}{4 \beta_n v D} \sqrt{(t_{opt} \beta_n v)^2 - \|\mathcal{S} - \mathcal{T}\|^2}.$$

Simplification yields

$$T_{opt_n} \leq \frac{\pi t_{opt}}{4D} \sqrt{(t_{opt} \beta_n v)^2 - \|\mathcal{S} - \mathcal{T}\|^2}.$$

Robot pair R_n could not reach the target in the current ellipse. Consequently, pair R_1 is assigned afterwards to search for it in an ellipse whose search time satisfies the inequality $T_{f_1} \leq \alpha_1 T_{opt_n}$. Substituting for T_{opt_n} in the inequality $T_{f_1} \leq \alpha_1 T_{opt_n}$ yields,

$$T_{f_1} \leq \alpha_1 \frac{\pi t_{opt}}{4D} \sqrt{(t_{opt} \beta_n v)^2 - \|\mathcal{S} - \mathcal{T}\|^2}.$$

Accordingly, generally, robot pair R_{j-1} searched for a path in an ellipse whose area is $A_{opt} - \epsilon$ and search time $T_{opt_{j-1}} - \epsilon$, substituting (5.5) into

$$T_{opt_{j-1}} = A_{opt}/(\beta_{j-1} v D),$$

yields,

$$T_{opt_{j-1}} \leq \frac{\pi t_{opt} \beta_n v}{4 \beta_{j-1} v D} \sqrt{(t_{opt} \beta_n v)^2 - \|\mathcal{S} - \mathcal{T}\|^2}.$$

Simplification yields

$$T_{opt_{j-1}} \leq \frac{\pi t_{opt} \beta_n}{4 \beta_{j-1} D} \sqrt{(t_{opt} \beta_n v)^2 - \|\mathcal{S} - \mathcal{T}\|^2}.$$

Robot pair R_{j-1} could not reach the target in the current ellipse. Consequently, pair R_j was assigned afterwards to search for it in an ellipse whose search time satisfies the inequality $T_{f_j} \leq \alpha_j T_{opt_{j-1}}$. Substituting for $T_{opt_{j-1}}$ in the inequality $T_{f_j} \leq \alpha_j T_{opt_{j-1}}$ yields,

$$T_{f_j} \leq \alpha_j \frac{\pi t_{opt} \beta_n}{4 \beta_{j-1} D} \sqrt{(t_{opt} \beta_n v)^2 - \|\mathcal{S} - \mathcal{T}\|^2}.$$

□

The following proposition establishes a quadratic time competitive upper bound on *HMRBUG*.

Theorem 2. (*Quadratic time competitive complexity*) Assume target \mathcal{T} is reachable from \mathcal{S} . Let *HMRBUG* use n robot pairs with velocities: $v_j = \beta_j v$, $\forall j$, $j = 1, 2 \dots n$, where

$$\beta_{j+1} \geq \beta_j \geq 1, \forall j, j = 1, 2 \dots n-1.$$

Then the traveling time of robot pair j , which reached the target, is bounded by:

$$T_{R_j} < \frac{\pi \alpha_j \beta_n (\prod_{i=1}^n \alpha_i) t_{opt} \sqrt{(\beta_n v t_{opt})^2 - \|\mathcal{S} - \mathcal{T}\|^2}}{4 D \beta_{j-1} (\prod_{i=1}^n \alpha_i - 1)}. \quad (5.6)$$

Proof. The path to the target is assumed to lie within an ellipse whose area is given in (5.4) and search time (5.3). First, we will inspect the case in which robot pair no. 1, R_1 reaches the target. In the worst-case scenario, the last robot pair, R_n , searched for a path in an ellipse whose area is $A_{opt} - \epsilon$ and search time $T_{opt_n} - \epsilon$, and thus could not reach the target. Consequently, R_1 was assigned afterwards to search for it in an ellipse whose search time is (5.3)

$$T_{f_1} \leq \frac{\pi \alpha_1 t_{opt}}{4 D} \sqrt{(\beta_n v t_{opt})^2 - \|\mathcal{S} - \mathcal{T}\|^2}. \quad (5.7)$$

The sum of the ellipses' search times of R_1 is

$$\begin{aligned} T_{R_1} &\leq t_{1,1} + t_{1,1+n} + t_{1,1+2n} + \dots + t_{1,i} \\ &= T_0 + \left(\prod_{k=1}^n \alpha_k \right)^1 T_0 + \left(\prod_{k=1}^n \alpha_k \right)^2 T_0 \\ &\quad + \dots + \left(\prod_{k=1}^n \alpha_k \right)^{\left(\frac{i-1}{n} \right)} T_0 \end{aligned} \quad (5.8)$$

The last expression can be explained by example. Consider that *HMRBUG* uses 3 pairs of robots to search for a path to the target. Table 5.1 describes the first search ellipse's search times.

Table 5.1: *HMRBUG* first ellipses search times

<i>ellipse num.</i> (i)	1	2	3	4	5	6	7
Robot pair	1	2	3	1	2	3	1
Saerch time	T_0	$\alpha_2 T_0$	$\alpha_2 \alpha_3 T_0$	$\alpha_1 \alpha_2 \alpha_3 T_0$	$\alpha_1 \alpha_2^2 \alpha_3 T_0$	$\alpha_1 \alpha_2^2 \alpha_3^2 T_0$	$\alpha_1^2 \alpha_2^2 \alpha_3^2 T_0$
T_0 coefficient	1	α_2	$\alpha_2 \alpha_3$	$\alpha_1 \alpha_2 \alpha_3$	$\alpha_1 \alpha_2^2 \alpha_3$	$\alpha_1 \alpha_2^2 \alpha_3^2$	$\alpha_1^2 \alpha_2^2 \alpha_3^2$

This series of times is a converging geometric series. The sum of such a series is

$$y + y\lambda + y\lambda^2 + y\lambda^3 + \dots + y\lambda^{w-1} = \frac{y(\lambda^w - 1)}{(\lambda - 1)}, \quad (5.9)$$

substituting $y = T_0$, $\lambda = (\prod_{k=1}^n \alpha_k)$, and $w - 1 = \frac{i-1}{n}$,

$$T_{R_1} \leq \frac{(\prod_{k=1}^n \alpha_k)^{\left(\frac{i-1}{n}+1\right)} - 1}{\prod_{k=1}^n \alpha_k - 1} T_0 < \frac{(\prod_{k=1}^n \alpha_k)^{\left(\frac{i-1}{n}+1\right)}}{\prod_{k=1}^n \alpha_k - 1} T_0 \quad (5.10)$$

Comparing the two expressions in terms of the time to cover the last ellipse, $t_{1,i}$ from (5.8), and T_{f_1} from (5.3), T_0 can be calculated,

$$\left(\prod_{k=1}^n \alpha_k\right)^{\left(\frac{i-1}{n}\right)} T_0 = T_{f_1} \leq \alpha_1 \frac{\pi t_{opt}}{4D} \sqrt{(t_{opt} \beta_n v)^2 - \|\mathcal{S} - \mathcal{T}\|^2}.$$

$$T_0 \leq \frac{\pi \alpha_1 t_{opt} \sqrt{(\beta_n v t_{opt})^2 - \|\mathcal{S} - \mathcal{T}\|^2}}{4D (\prod_{i=1}^n \alpha_i)^{\left(\frac{i-1}{n}\right)}}.$$

Substituting T_0 into (5.10) yields,

$$T_{R_1} < \frac{(\prod_{k=1}^n \alpha_k)^{\left(\frac{i-1}{n}+1\right)}}{\prod_{k=1}^n \alpha_k - 1} \frac{\pi \alpha_1 t_{opt} \sqrt{(\beta_n v t_{opt})^2 - \|\mathcal{S} - \mathcal{T}\|^2}}{4D (\prod_{i=1}^n \alpha_i)^{\left(\frac{i-1}{n}\right)}}.$$

Simplification yields

$$T_{R_1} < \frac{\pi \alpha_1 t_{opt} \sqrt{(\beta_n v t_{opt})^2 - \|\mathcal{S} - \mathcal{T}\|^2} (\prod_{k=1}^n \alpha_k)}{4D (\prod_{k=1}^n \alpha_k - 1)}.$$

Accordingly, in general,

$$T_{R_j} < \frac{\pi \alpha_j \beta_n (\prod_{i=1}^n \alpha_i) t_{opt} \sqrt{(\beta_n v t_{opt})^2 - \|\mathcal{S} - \mathcal{T}\|^2}}{4D \beta_{j-1} (\prod_{i=1}^n \alpha_i - 1)}.$$

where for $j = 1$, $\beta_{j-1} = \beta_n$. □

The last inequality can also be rewritten as follows,

$$T_{R_j} < \frac{\pi \alpha_j \beta_n t_{opt} \sqrt{(\beta_n v t_{opt})^2 - \|\mathcal{S} - \mathcal{T}\|^2}}{4D \beta_{j-1} \left(1 - \frac{1}{(\prod_{i=1}^n \alpha_i)}\right)}.$$

There are two boundary cases for the last result. The first case is when there are no obstacles between \mathcal{S} and \mathcal{T} . According to the algorithm, all the robots will move directly towards the target and the fastest robot will reach \mathcal{T} first. Here the travel time will be $\|\mathcal{S} - \mathcal{T}\|/(\beta_n v)$. The second case is when the path to the target is found within the first search ellipse. The upper bound in this case will be, according to Lemma 5.4.2,

$$t_{j,j} \leq 4 \frac{A_j}{\beta_j v D} + \frac{\|\mathcal{S} - \mathcal{T}\|}{\beta_j v}$$

Lemma 5.4.1. *The time competitive complexity of HMRBUG is minimal when for each of the n pairs of robots deployed, the multiplication factor equals,*

$$\alpha_j = \frac{(n+1)^{1/n} \beta_{j-1}}{\prod_{i=1}^n \beta_j^{1/n}}, \beta_0 = \beta_n, \beta_1 = 1 \quad (5.11)$$

Proof. In order to find the optimal multiplication factors, α'_j s, a new objective function that combines all the sums of times is formed,

$$\begin{aligned} T_{tot} &= T_{R_1} + T_{R_2} + \dots + T_{R_n} \\ &< \frac{\pi \beta_n (\prod_{i=1}^n \alpha_i) t_{opt} \sqrt{(\beta_n v t_{opt})^2 - \|\mathcal{S} - \mathcal{T}\|^2}}{4D (\prod_{i=1}^n \alpha_i - 1)} \cdot \sum_{i=1}^n \frac{\alpha_i}{\beta_{i-1}}. \end{aligned}$$

Substituting $c = \frac{\pi \beta_n t_{opt} \sqrt{(\beta_n v t_{opt})^2 - \|\mathcal{S} - \mathcal{T}\|^2}}{4D}$ in T_{tot} , And then differentiating T_{tot} according to each of the α'_j s,

$$\begin{aligned} T_{tot} &< c \frac{(\prod_{i=1}^n \alpha_i)}{(\prod_{i=1}^n \alpha_i - 1)} \cdot \sum_{i=1}^n \frac{\alpha_i}{\beta_{i-1}} \\ \frac{\partial T_{tot}}{\partial \alpha_k} &= c \frac{\left(\prod_{i=1}^n \alpha_i \sum_{i=1}^n \frac{\alpha_i}{\beta_{i-1}} \right)' (\prod_{i=1}^n \alpha_i - 1)}{(\prod_{i=1}^n \alpha_i - 1)^2} - c \frac{\left(\prod_{i=1}^n \alpha_i \sum_{i=1}^n \frac{\alpha_i}{\beta_{i-1}} \right) (\prod_{i=1}^n \alpha_i - 1)'}{(\prod_{i=1}^n \alpha_i - 1)^2} = \\ &= c \frac{\left(\prod_{i=1}^n \alpha_i \sum_{i=1}^n \frac{\alpha_i}{\beta_{i-1}} \right)' }{(\prod_{i=1}^n \alpha_i - 1)} - c \frac{\left(\prod_{i=1}^n \alpha_i \sum_{i=1}^n \frac{\alpha_i}{\beta_{i-1}} \right) (\prod_{i=1, i \neq k}^n \alpha_i)}{(\prod_{i=1}^n \alpha_i - 1)^2} = \\ &= c \frac{(\prod_{i=1}^n \alpha_i)' \sum_{i=1}^n \frac{\alpha_i}{\beta_{i-1}} + \prod_{i=1}^n \alpha_i \left(\sum_{i=1}^n \frac{\alpha_i}{\beta_{i-1}} \right)'}{(\prod_{i=1}^n \alpha_i - 1)} - c \frac{\left(\prod_{i=1}^n \alpha_i \sum_{i=1}^n \frac{\alpha_i}{\beta_{i-1}} \right) \left(\frac{\prod_{i=1}^n \alpha_i}{\alpha_k} \right)'}{(\prod_{i=1}^n \alpha_i - 1)^2} = \\ &= c \frac{\left(\frac{\prod_{i=1}^n \alpha_i}{\alpha_k} \right) \sum_{i=1}^n \frac{\alpha_i}{\beta_{i-1}} + \prod_{i=1}^n \alpha_i \left(\frac{1}{\beta_{k-1}} \right)'}{(\prod_{i=1}^n \alpha_i - 1)} - c \frac{\left(\prod_{i=1}^n \alpha_i \sum_{i=1}^n \frac{\alpha_i}{\beta_{i-1}} \right) \left(\frac{\prod_{i=1}^n \alpha_i}{\alpha_k} \right)'}{(\prod_{i=1}^n \alpha_i - 1)^2} = \end{aligned}$$

comparing each function to zero and finding the common roots yields

$$\alpha_j = \frac{(n+1)^{1/n} \beta_{j-1}}{\prod_{i=1}^n \beta_i^{1/n}}, \quad \beta_0 = \beta_n, \beta_1 = 1$$

Substituting α_j back into $\frac{\partial T_{tot}}{\partial \alpha_k}$, yields

$$\begin{aligned} \frac{\partial T_{tot}}{\partial \alpha_k} &= c \frac{\left(\frac{\prod_{i=1}^n \alpha_i}{\alpha_k} \right) \sum_{i=1}^n \frac{\alpha_i}{\beta_{i-1}} + \prod_{i=1}^n \alpha_i \left(\frac{1}{\beta_{k-1}} \right)'}{(\prod_{i=1}^n \alpha_i - 1)} - c \frac{\left(\prod_{i=1}^n \alpha_i \sum_{i=1}^n \frac{\alpha_i}{\beta_{i-1}} \right) \left(\frac{\prod_{i=1}^n \alpha_i}{\alpha_k} \right)'}{(\prod_{i=1}^n \alpha_i - 1)^2} = \\ &= c \frac{\left(\frac{n+1}{\alpha_k} \right) \sum_{i=1}^n \frac{\alpha_i}{\beta_{i-1}} + \left(\frac{n+1}{\beta_{k-1}} \right)'}{n} - c \frac{(n+1) \sum_{i=1}^n \frac{\alpha_i}{\beta_{i-1}} \left(\frac{n+1}{\alpha_k} \right)'}{n^2} = \\ &= c \frac{n+1}{\alpha_k n^2} \left[n \sum_{i=1}^n \frac{\alpha_i}{\beta_{i-1}} + \frac{n \alpha_k}{\beta_{k-1}} - (n+1) \sum_{i=1}^n \frac{\alpha_i}{\beta_{i-1}} \right] = \\ &= c \frac{n+1}{\alpha_k n^2} \left[n \left(n \cdot \frac{(n+1)^{\frac{1}{n}}}{\prod_{i=1}^n \beta_i^{\frac{1}{n}}} \right) + n \cdot \frac{(n+1)^{\frac{1}{n}}}{\prod_{i=1}^n \beta_i^{\frac{1}{n}}} - (n+1) n \cdot \frac{(n+1)^{\frac{1}{n}}}{\prod_{i=1}^n \beta_i^{\frac{1}{n}}} \right] = \\ &= c \frac{n+1}{\alpha_k n^2} \left[(n+1) \left(n \cdot \frac{(n+1)^{\frac{1}{n}}}{\prod_{i=1}^n \beta_i^{\frac{1}{n}}} \right) - (n+1) \left(n \cdot \frac{(n+1)^{\frac{1}{n}}}{\prod_{i=1}^n \beta_i^{\frac{1}{n}}} \right) \right] = \\ &= 0 \end{aligned}$$

□

Theorem 3. *The problem of on-line heterogeneous multi-robot navigation to a known target in an unknown and unbounded environment belongs to the quadratic time competitive complexity class.*

Proof. As defined in Definition 2, a time competitive complexity class is formed from a lower bound and from an upper bound for the problem. In section 5.2 we found a lower bound for the research problem that is quadratic in the optimal off-line solution, t_{opt} (5.1). Then we presented *HMRBUG* algorithm to solve that problem with an upper bound quadratic in t_{opt} (5.6). Thus, the research problem belongs to the quadratic time competitive complexity class. \square

Corollary 5.4.1. *HMRBUG is complete.*

Proof. The first important property established in Theorem 2, is that if the target \mathcal{T} is reachable, *HMRBUG* will find a path to it. The second property is that *HMRBUG* will find that path in a finite and limited time. \square

Corollary 5.4.2. *HMRBUG is optimal in the sense of the search time.*

Proof. In Theorem 3 we proved that the search problem belongs to the quadratic time competitive complexity class. *HMRBUG* upper bound is time quadratic in the optimal off-line solution (5.6). Thus, *HMRBUG* is optimal up to constant coefficients. \square

The problem of reaching a target whose position is known in an a priori unknown and unbounded environment was presented. A lower bound for all algorithms that solve this problem was proved to be quadratic in the optimal off-line solution. *HMRBUG* algorithm for a group of heterogeneous robots was presented. *HMRBUG* algorithm was analyzed and its upper bound found to be quadratic in the optimal off-line solution. Consequently, the aforementioned problem was classified into the quadratic time competitive complexity class, and therefore, *HMRBUG* is optimal. The multiplication factor that minimizes the time of search was found and finally the algorithm was proved to be complete.

Chapter 6. Simulations

The algorithms' analytical analysis provides the upper bound for the performance in worst-case scenarios. However, most scenarios are not worst-case scenarios. Thus, better performance is often evident. Using simulation, it is possible to evaluate the effect of several parameters. This chapter presents simulation results for the *HMRSTM* and *HMRBUG* algorithms.

6.1 HMRSTM simulations

The effect of the following parameters on the average performance of *HMRSTM* was evaluated. According to Sec. 3.4.2, a total of 37,800 simulations with different configurations were executed. The parameters for the simulations were 3 environments, 10 target positions, up to 8 surrounding adjacent cells, 3 beta- velocity distributions, 10 numbers of robots, and up to 10 initial disc sizes.

Since execution time in computer application is not equivalent to the actual running time in real robots, the performance measure chosen was the number of steps made by the slowest robot. Each step is a traveling distance equivalent to the size of the robot, D . Since all robots start at the same time and simultaneously stop when one robot finds the target, the travel time of the slowest robot equals the travel time of any other, possibly faster robot, from the starting point until the first robot reaches the target. In theory and in the simulations, the slowest robots always have the same velocity in all configurations. Thus, measuring the number of D-steps made by the slowest robot in each configuration yields an exact performance measure for comparison.

In the following example, depicted in Fig. 6.1 and 6.2, *HMRSTM* deploys 4 robots to search for target number 98 in the library environment. Since a clear view of the execution is pursued in this example, only robot number 3 is shown along with its path within each search disc. First, each robot was assigned to search for the target in its disc. Thus, robot 3 was initially assigned to search for the target in disc no. 3 (Fig. 6.1 Left). Covering the accessible regions of its first disc took robot 3 635 steps. Afterward, since it did not find the target in its initial disc, it moved on to search the target within its second disc, disc no. 7 (Fig. 6.1 Right). After covering its second disc with 3012 steps and not finding the target, robot 3 continued to search within the next disc, disc number 11 (Fig. 6.2 Left). Finally, after another 4940 steps, robot number 3 found the target and terminated the algorithm (Fig. 6.2 Right).

The first phase of simulations of *HMRSTM*, focused on testing the effect of the initial search disc radius on the average-case performance of the algorithms in addition to testing environment, velocity distribution, and target position. Averaging all parameters except the environment and beta, as in Fig. 6.3, it is evident that the time to find the target (the number of steps) decreases as the environment becomes increasingly congested ("library" is more congested than "cave," which is more congested than "free"). Since *HMRSTM* cover growing portions of the environment until finding the target, in less congested environments there is less free space to cover. Thus, it takes less time to cover the environment and find the target. Moreover, it is evident that the more heterogeneous the group's velocity, the less time it takes for it to find the target. Figure 6.3 shows that in every environment, averaged over all target positions, number of robots, and initial search radii, the beta-max velocity

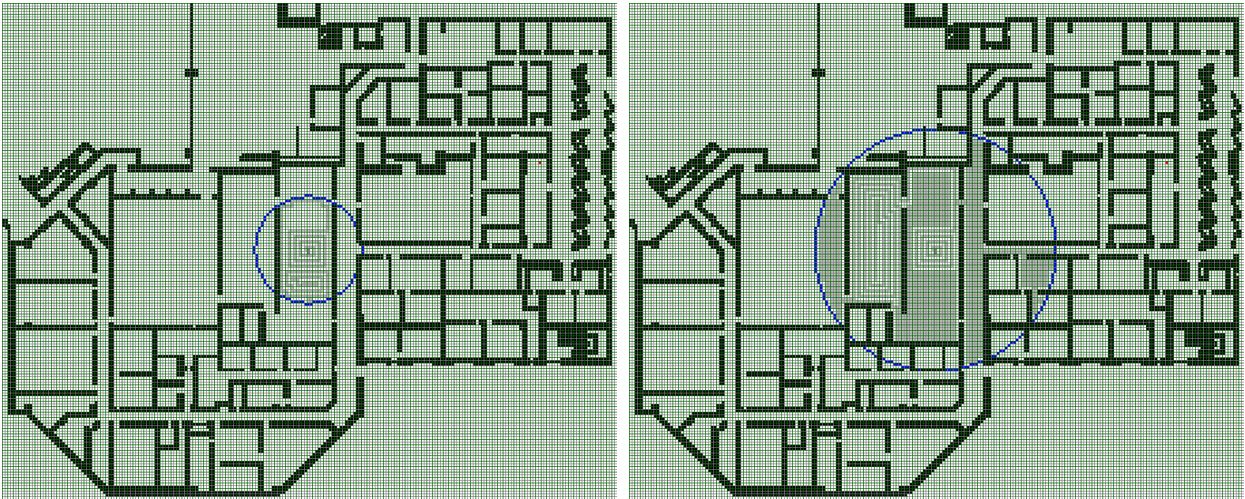


Figure 6.1: Robot no. 3 search for the target, Left: in its initial search disc, no.3. Right: in its second search disc, no. 7.

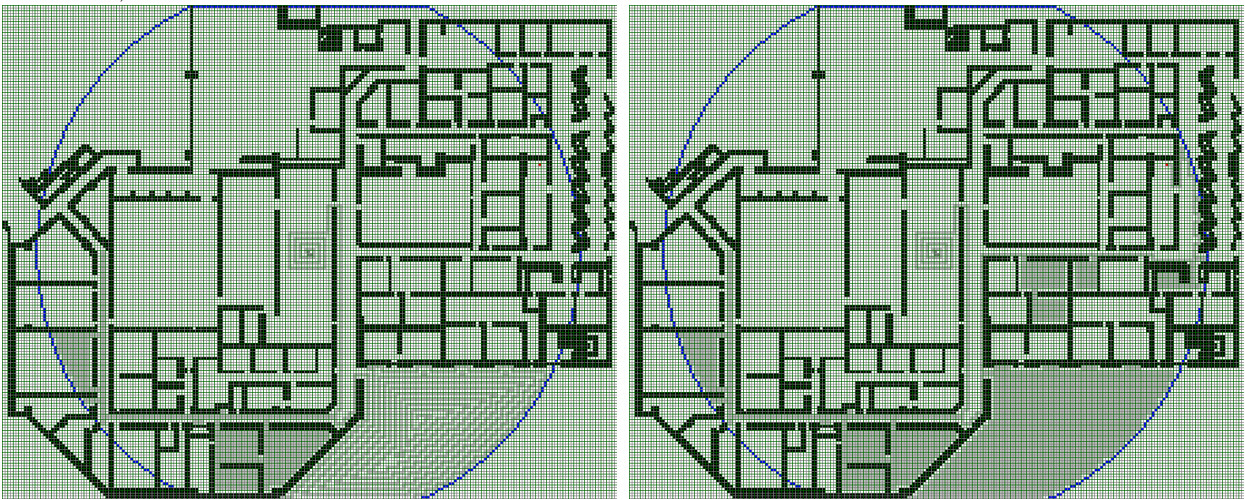


Figure 6.2: Robot no. 3 Left: search for the target in its third disc, no.11. Right: Finds the target.

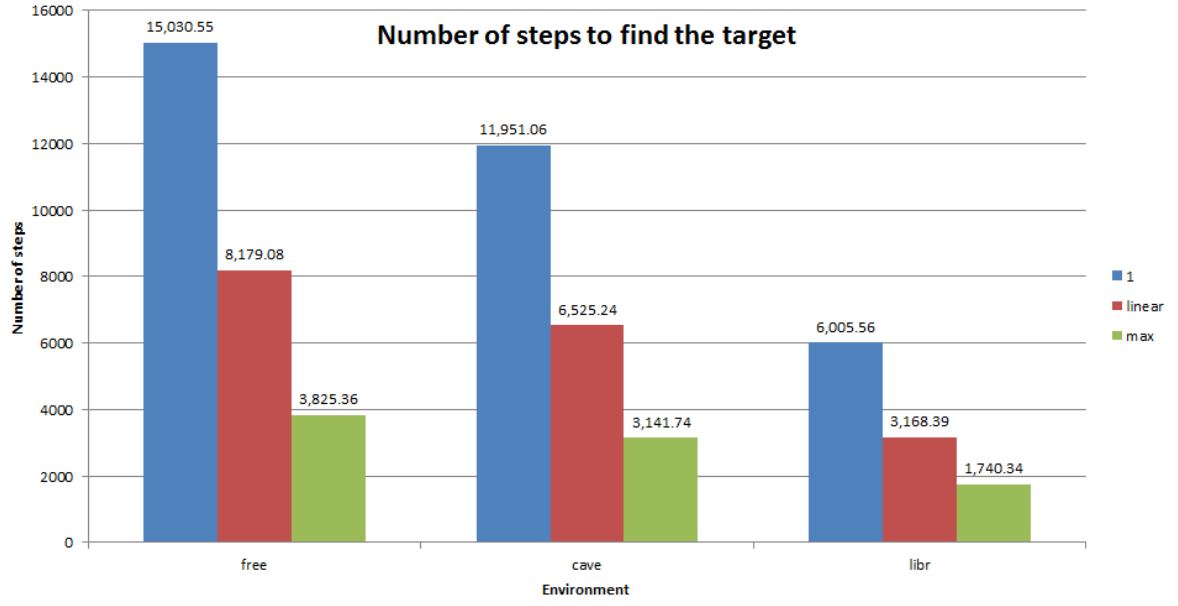


Figure 6.3: HMRSTM average simulation results for environment and beta

distribution performs significantly better than the beta-linear velocity distribution, which performs significantly better than the homogeneous velocity distribution.

The effect of the number of robots is shown in Fig 6.4. It can be seen that as the number of robots increases, the time to reach the target decreases, which shows that generally, the greater the number of robots that search for the target, the sooner it will be found. The latter result is consistent with intuition and with the assumptions made prior to execution of the simulations.

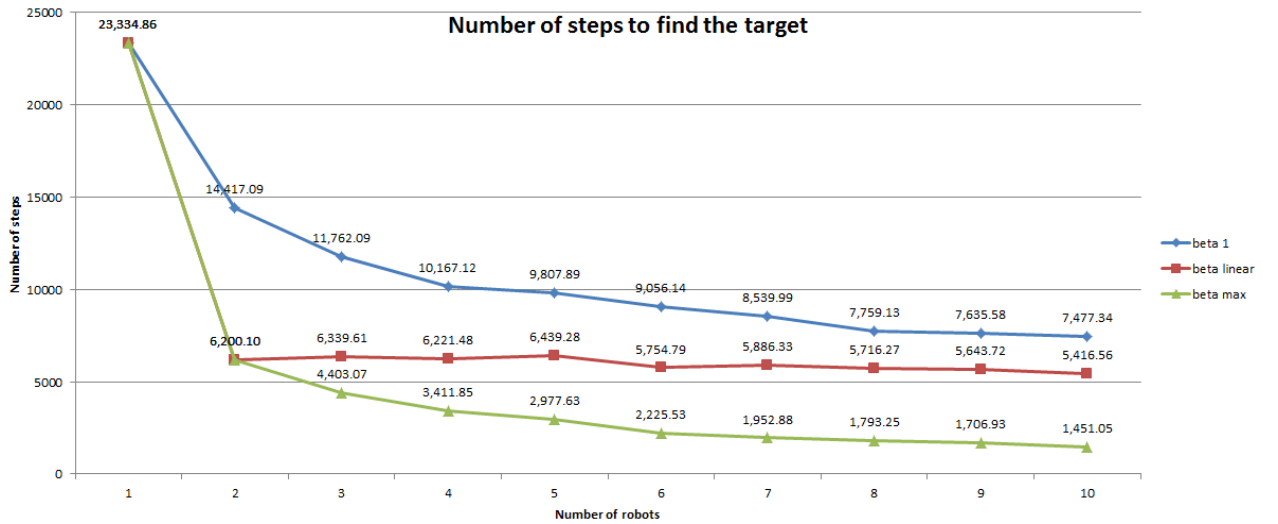


Figure 6.4: HMRSTM average simulation results for no. of robots and beta

For groups with the same number of robots, but different beta, the maximal velocities are different. Moreover, due to analytical restrictions within the linear velocity distribution, the maximal velocity within each group decreases as the number of robots in the group increases. For example, for a group of 2 robots, the fastest robot can have a velocity of 2.99 times the slowest, while in a group of 3 robots, the fastest robot is allowed to be only 2.37 times faster than the slowest. Thus, Fig.6.8 shows the results normalized according to the formula: Normalized number of steps = (number of steps)/(Sum

of velocities). The sum of velocities is a very good measure of the group's total cost, since faster robots consume more energy, and possibly wear out more quickly. Since a single robot has only homogeneous velocity distribution, and a pair of robots has only homogeneous and linear velocity distributions, the graph in Fig. 6.5 shows the normalized results for groups of 3 to 10 robots. It is clear that in the normalized results as well, the more robots used to search for the target, the less time it takes to find it. Again, it is evident that the more heterogeneous the group is, the sooner the target will be found.

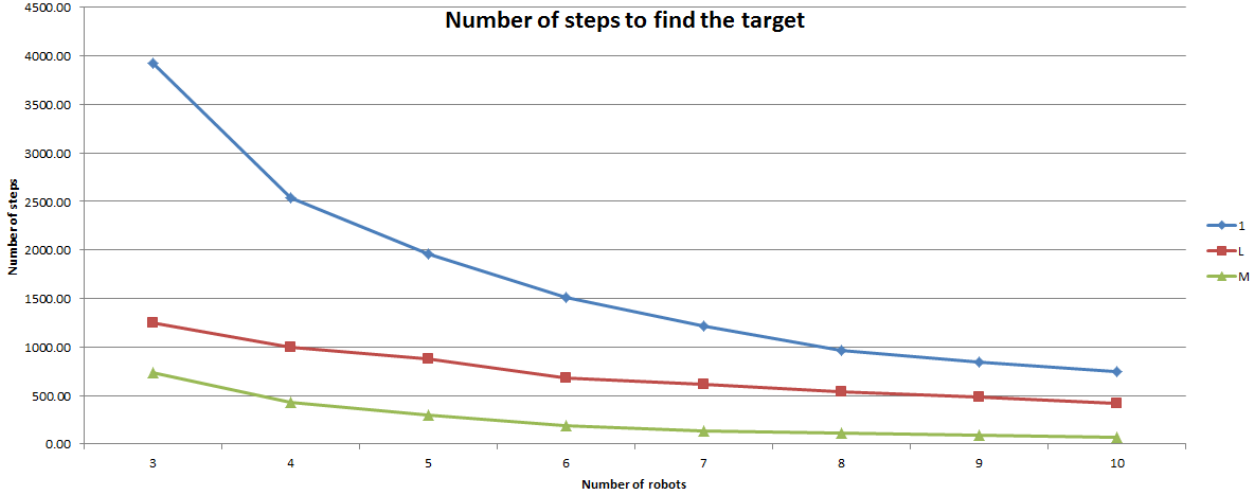


Figure 6.5: HMRSTM normalized simulation results for no. of robots and beta

In theory, the initial disc is supposed to be sufficiently small not to affect the upper bound of the algorithm. In average-case performance, the initial disc may have only a minor effect. The effect of the initial disc in simulations was not consistent, and general rules could not be formed. The general results for varying the initial search disc are depicted in Fig. 6.6. The graphs are fairly horizontal, suggesting that there is only a minor effect of the initial search disc radius. This result is consistent with the theory that dictates that the search time does not depend on the initial search disc radius.

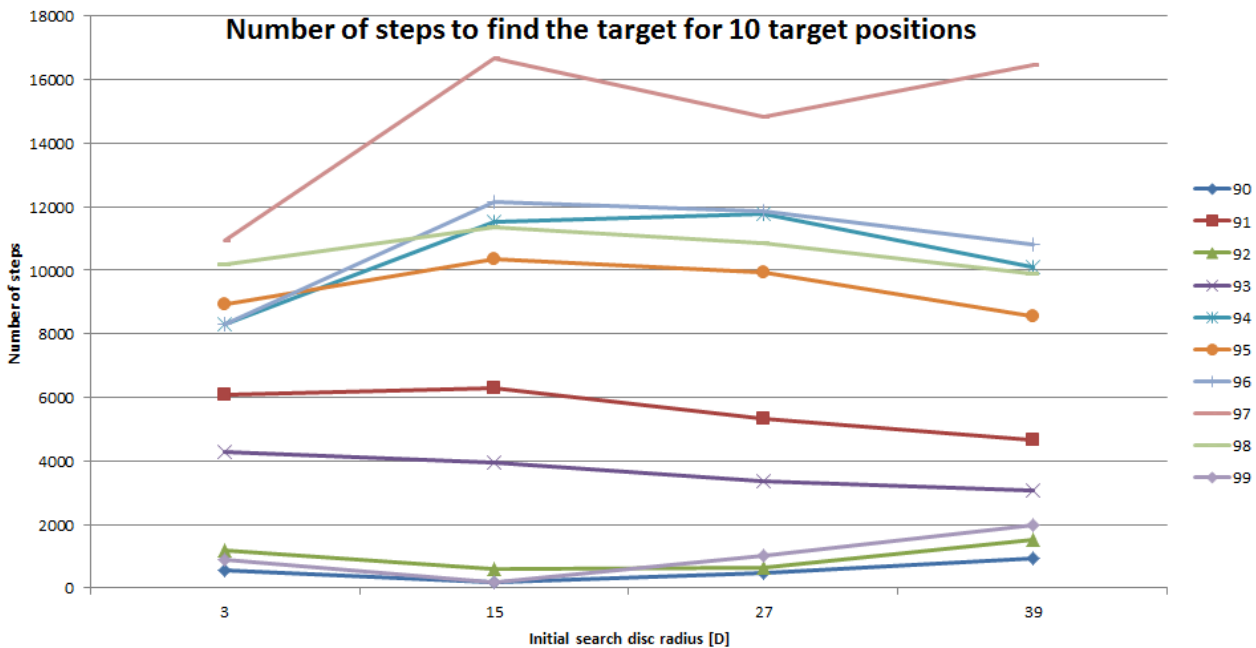


Figure 6.6: HMRSTM initial search radii

The effect of placing a target in an adjacent cell (Sec. 3.4) was evaluated for all 10 targets, for all 3 environments, for robot groups ranging between 1 and 10 robots, and for 3 velocity distributions. Only one initial search disc radius was chosen. For each target position, additional 8 adjacent cells were tested as targets. Figure 6.7 shows the results of finding each of the 9 targets of the cluster of target number 90 in the "free" environment. It is evident that although the targets are 4 cells apart at most (Manhattan distance), the differences between each target are much greater (more than 350 steps). Generally, from the trend lines, it is evident that on average, the more heterogeneous the group, the more quickly the target will be found (beta-max is better than beta-linear, which is better than homogeneous velocity distribution). However, at some points, the beta-max distribution is much poorer than the linear and the homogeneous distributions (e.g., points 2, 5, and 8). This is due to the spiral nature of the inner disc covering algorithm, discussed in 3.4. Fig. 6.8 shows the normalized

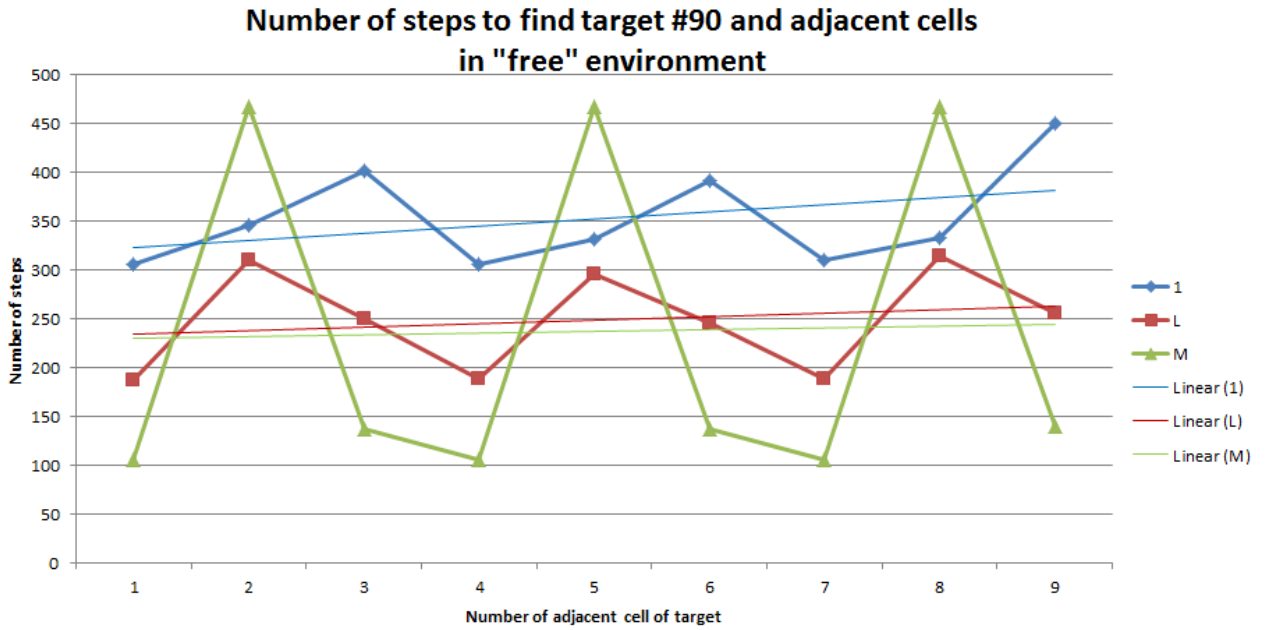


Figure 6.7: HMRSTM finding cluster of targets #90, in "free" environment

results for clusters of 9 adjacent cells for each target. It is clear that in the normalized results as well, the more robots used to search for the target, the less time it takes to find it. Again, it is evident that the more heterogeneous the group, the sooner the target will be found.

The simulation results, which exhibit the average-case performance, were compared with the analytical worst-case upper bound of the algorithm (As presented in Chapter 4). The upper bound of robot j for a certain execution depends on n , α_i , β_i , $i = 1, \dots, n$, $\|\mathcal{S} - \mathcal{T}\|$, D , t_{opt} , v . Thus, it was calculated for each robot (in a pair), for each set of start-target points, for each number of robots, and for each environment. Without loss of generality, $D = 1$, $v = 1$ were assumed. All simulation results were within the upper bound. Moreover, the simulation average-case results were sometimes several orders of magnitude smaller than the analytical worst upper bound. The path length of the simulation for the case of searching for target number 97 in "free" environment by groups of heterogeneous robots with beta-max velocity distributions is compared with the analytical upper bound in Fig. 6.9. Since the differences are so great, the vertical axis is logarithmic.

The tens of thousands of simulation runs show that the average-case performance of *HMRSTM* depends on several parameters. The use of multiple robots significantly reduces the time to find the

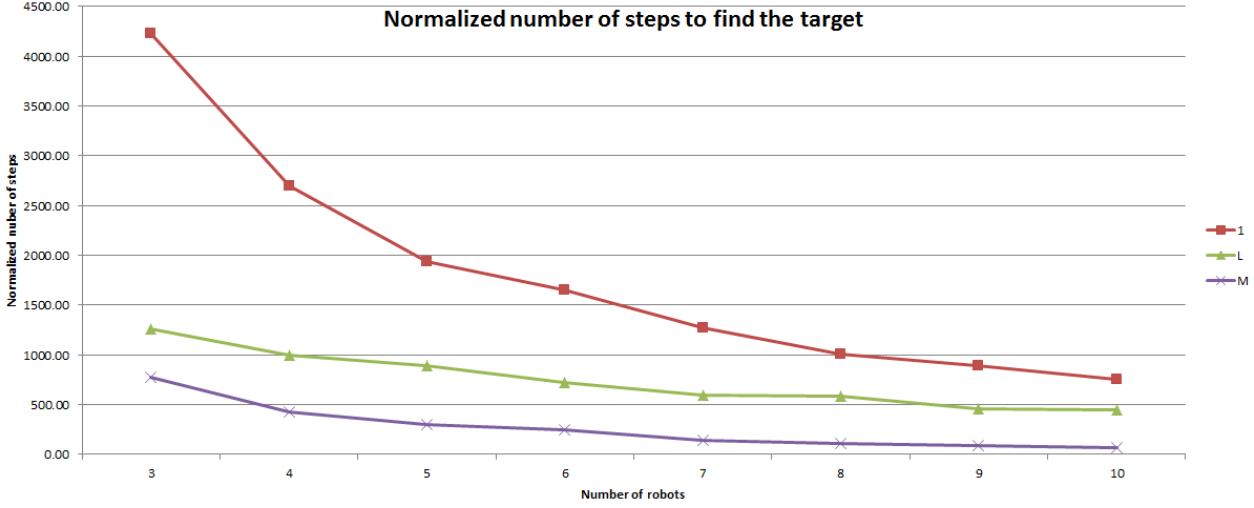


Figure 6.8: HMRSTM normalized simulation results for no. of robots and beta

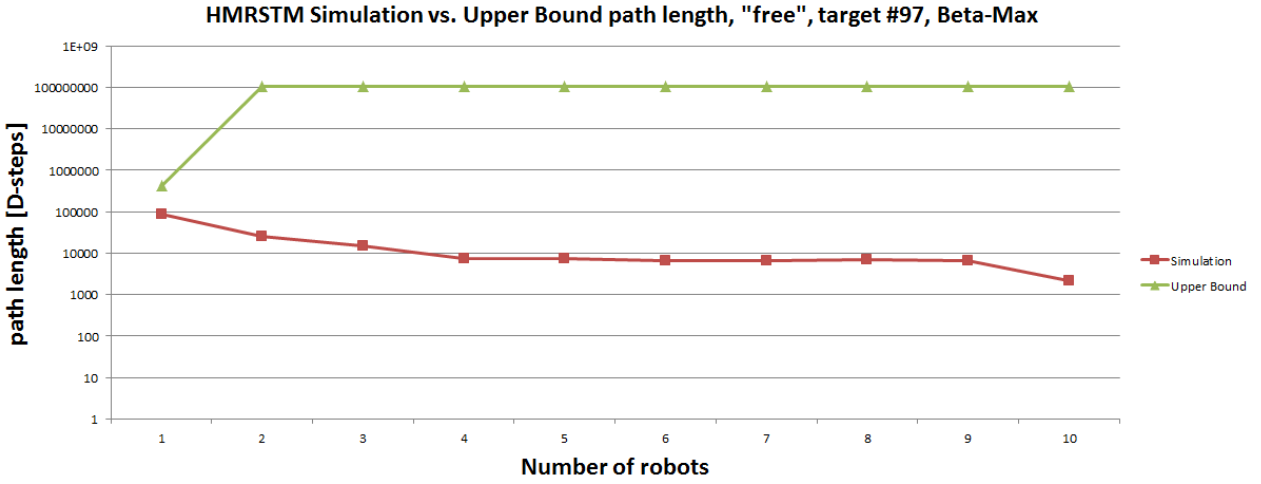


Figure 6.9: HMRSTM simulation and analytical upper bound path length

target, and the use of heterogeneous groups of robots reduces that time even more. Moreover, the more heterogeneous the group is, the better the performance. As predicted, the more congested the environment is, the less time the robots will spend on the search. The distance of the target from the start position obviously affects the time needed to find the target, but due to the nature of the coverage algorithm used, the exact position of the target has a great effect. In accordance with the hypothesis, the initial disc search time does not have a great effect on performance. The average-case performance of *HMRSTM*, as evident from the simulations, is much better than the analyzed worst-case performance.

6.2 HMRBUG simulations

The effect of several parameters on the average performance of *HMRBUG* was evaluated. According to Sec. 3.4.2, a total of 9000 simulation runs with different configurations were executed with the following parameters: 3 environments, 10 sets of start and target positions, 3 beta-velocity distributions, 10 robot pairs, and 10 initial ellipses sizes. As in the previous section, (6.1), the performance measure chosen was the number of steps conducted by the slowest robot.

In the following example, *HMRBUG* deploys two pairs of robots to search for a path from start to

target in the "cave" environment. The first pair's color is green and the second, faster pair's color is red. Each pair's initial bounding ellipse is drawn with its pair color (Fig. 6.10). First, all the robots move directly towards the target. After pair number 2 hits the first obstacle's boundary, it begins to circumnavigate it and its bounding ellipse. The green pair, which is slower, hits the obstacle after the red pair, and begins encircling the obstacle's boundary and its green ellipse. The red pair meets on the red ellipse, and heads towards the shortest point to the target from the boundaries of the obstacle and the ellipse. From the nearest point, figuring their way directly towards the target, which is blocked by an obstacle, pair number 2 moves on to search for a path to the target in a larger ellipse. While pair number 2 circumnavigates the obstacle's boundary within the new, larger ellipse, robot pair number 1 meets on its ellipse's boundary and moves to the nearest point together (Fig. 6.10 Right). After reaching the nearest point and realizing that the way toward the target is blocked by an obstacle, pair number 1 extends its disc and continues to search for a path to the target in the next ellipse (Fig. 6.11 Left). Next, the robots of pair number 2 complete their circumnavigation of the obstacle and their second search ellipse, meet, move together to the closest point on the boundary to the target and, finally, head directly toward the target, reach it and terminate the algorithm (Fig. 6.11 Right). The simulation application shows a message that indicates how many steps each robot made, and the time it took for the simulation to execute the search in seconds.

Averaging all parameters except for the environment and beta, as in Fig. 6.12, it is evident that the time to reach the target (the number of steps) increases as the environment becomes increasingly congested ("library" is more congested than "cave," which is more congested than "free"). This result is consistent with the algorithm that circumnavigates obstacles on its way to the target. Thus, the more congested the environment is, the more obstacles there are for the robots to encircle. Moreover, it is evident that the more heterogeneous the group's velocity is, the less time it takes to reach the target. It is clear that a group with homogeneous velocity distribution will reach the target after the group with linear velocity distribution, and the group with the beta-max velocity distribution will reach the target first. This effect occurs due to the different velocities that influence the ellipse search time multiplication factors, which yield greater ellipses for the heterogeneous groups.

The effect of the number of robots is shown in Fig 6.13. It can be seen that as the number of robots increases, the time to reach the target decreases. This rule does not apply to the linear velocity distribution, due to the constraints on the search time and area within each ellipse (Conditions 3,4). These constraints yield a decrease in the velocity of the fastest robot as the number of robots grows. Thus, although the ellipse within which the search for a path to the target is conducted is larger, the overall performance of the linear velocity distribution decreases as the number of robots grows under this performance measure comparison. Therefore, a normalized comparison yields the results presented in Fig. 6.14. The normalization made according to the sum of the velocities of the group and according to the formula: $\text{Normalized number of steps} = (\text{Number of Steps})/(\text{Sum of Velocities})$. An improvement in performance as the number of robots increases is evident in all velocity distributions. It is also evident that beta-max velocity distribution is better than the linear distribution and that linear distribution is better than homogeneous group even after the velocity normalization.

The initial ellipse is supposed, in theory, to be sufficiently small that it will not affect the upper bound of the algorithm. In average-case performance, the initial ellipse should have an effect. However, the effect of the initial ellipse in simulations was not consistent, and general rules could not be formed.

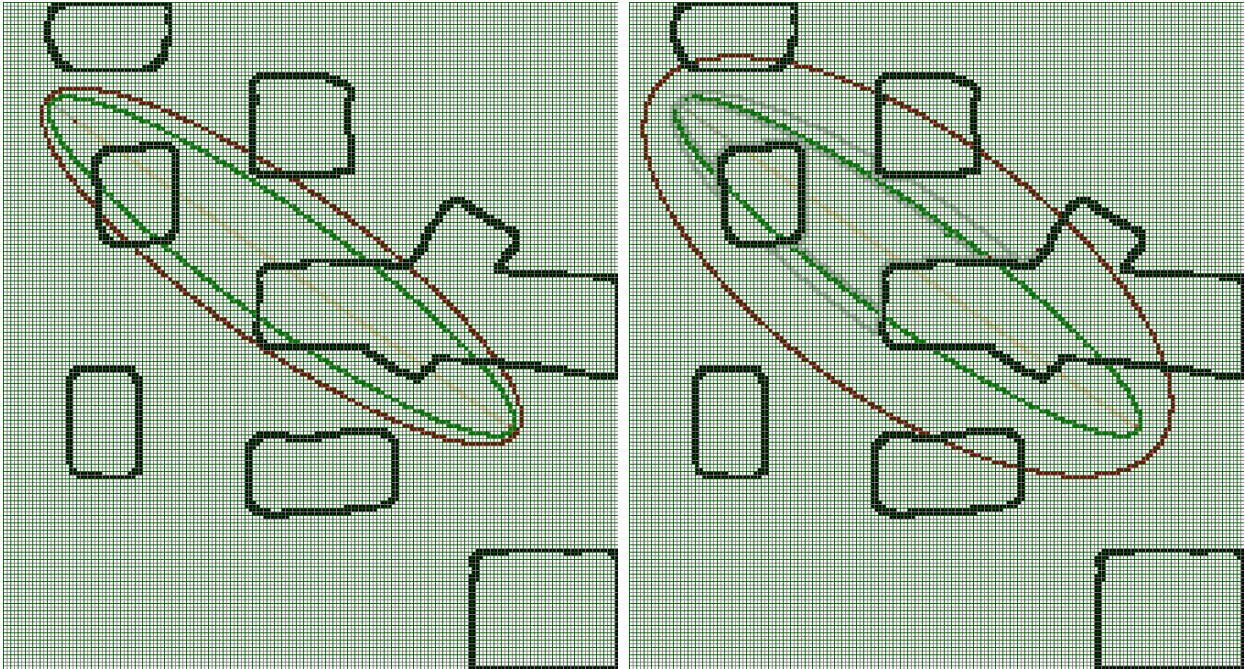


Figure 6.10: Left: Initial search ellipses. Right: fast pair in red moves to a larger ellipse.

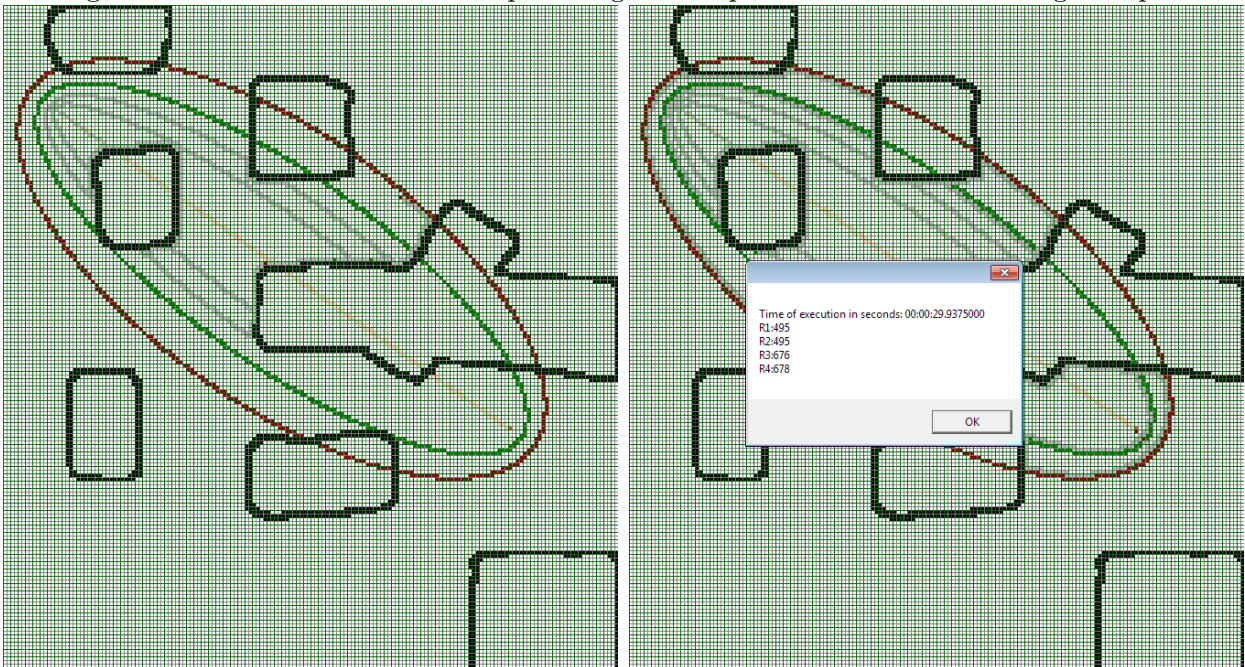


Figure 6.11: Left: Slow pair in green moves to a larger ellipse. Right: fast pair in red reaches that target.

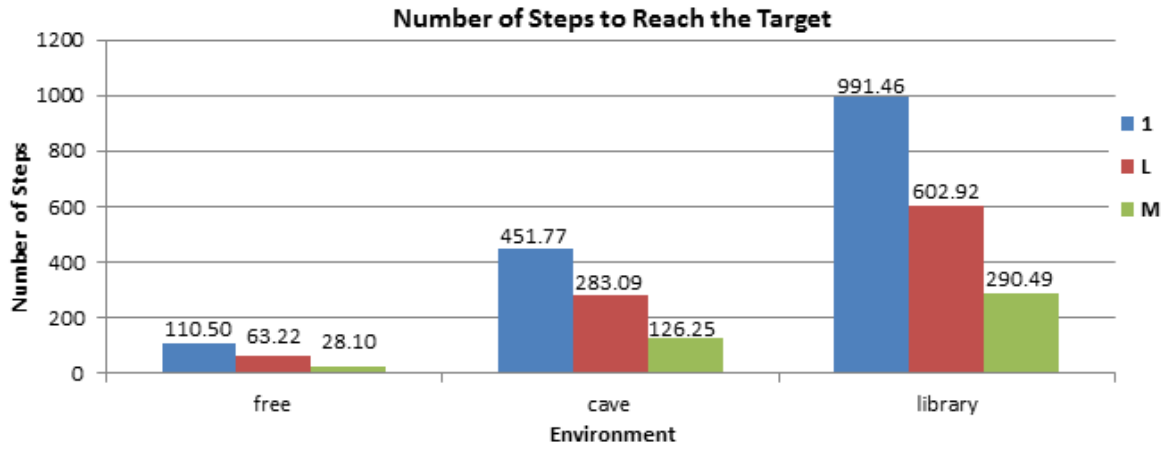


Figure 6.12: HMRBUG average simulation results for environment and beta

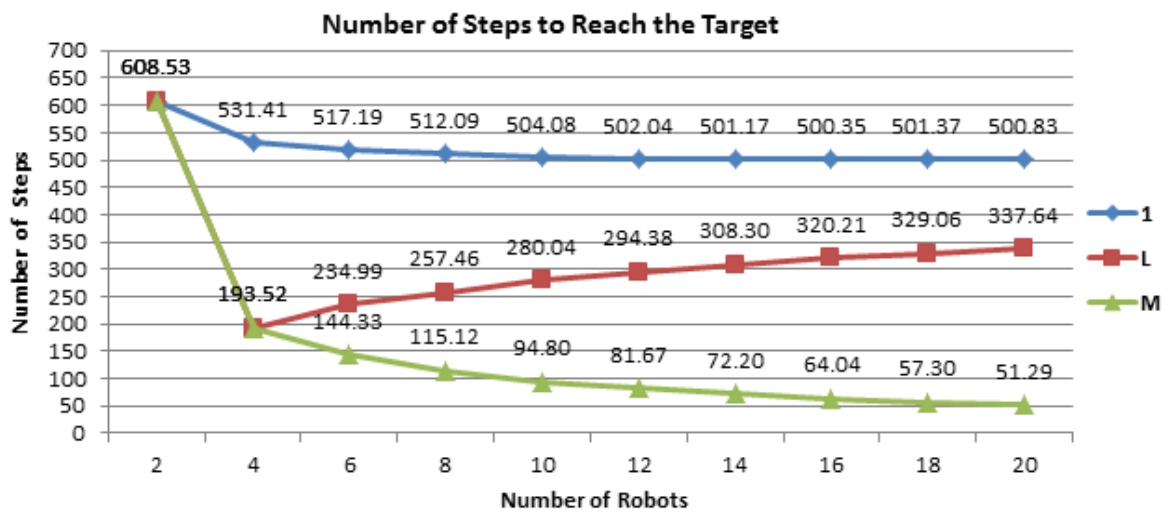


Figure 6.13: HMRBUG average simulation results for no. of robots and beta

Figures 6.15, 6.16, 6.17 show the behavior of the algorithm's average-case performance for different initial ellipses. It is evident that in some cases, a larger initial search ellipse yields poorer results (longer time), e.g., Start-Target configuration number 4 in Library, Fig. 6.15. There are cases where a larger initial search ellipse yields better results (shorter time), e.g., Start-Target configuration number 6 in Library, Fig. 6.16. Additionally, there are cases where the ellipse's initial size does not influence performance, e.g., Fig. 6.17. The general results for varying the initial search ellipse are depicted in Fig. 6.18. In cases where the initial search ellipse is small relative to the S-T distance and to the size of the obstacles on the S-T path, it will be extended many times, and choosing a larger initial ellipse results in better performance. This is the main assumption for large environments. However, if the search ellipse is too large relative to those parameters, as in several cases in the simulation, the performance, in some cases, improves and in others does not, according to the specific arrangement of the obstacles in the environment. For example, sometimes a larger search ellipse yields a larger bounding ellipse to follow. The reason for this behavior stems from the constraints of the size of the simulation system and the additional constraint of the minimal size of the search ellipse.

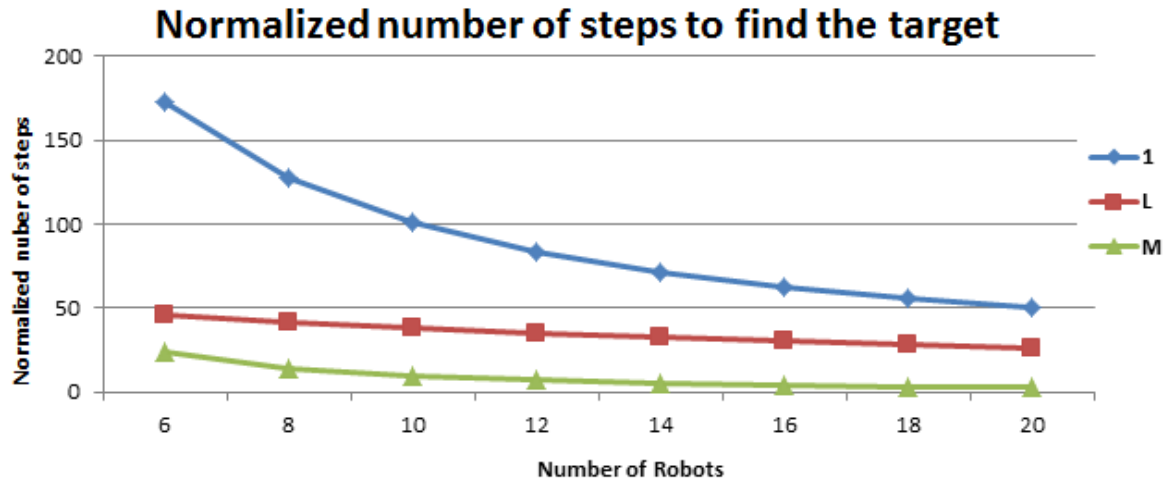


Figure 6.14: HMRBUG normalized simulation results for no. of robots and beta

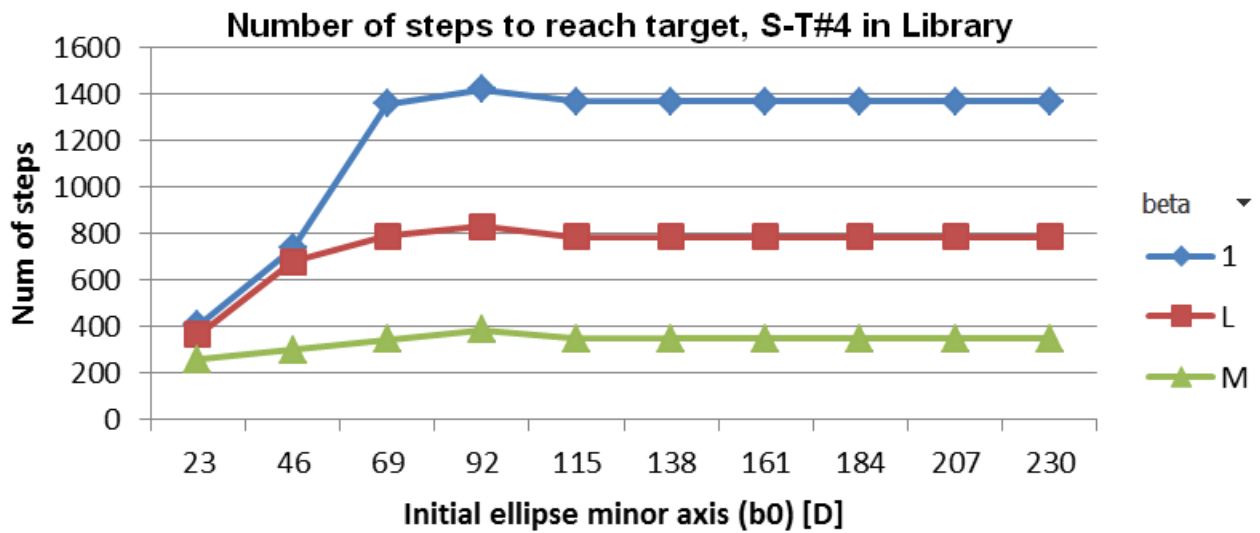


Figure 6.15: HMRBUG simulation results for initial ellipse and beta, S-T#4, library

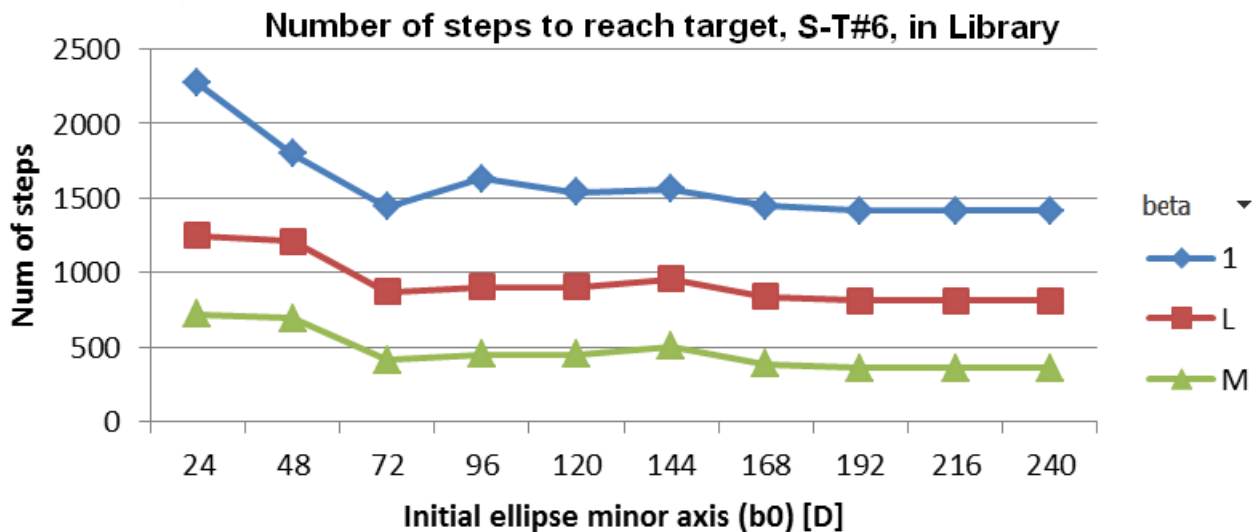


Figure 6.16: HMRBUG simulation results for initial ellipse and beta, S-T#6, library

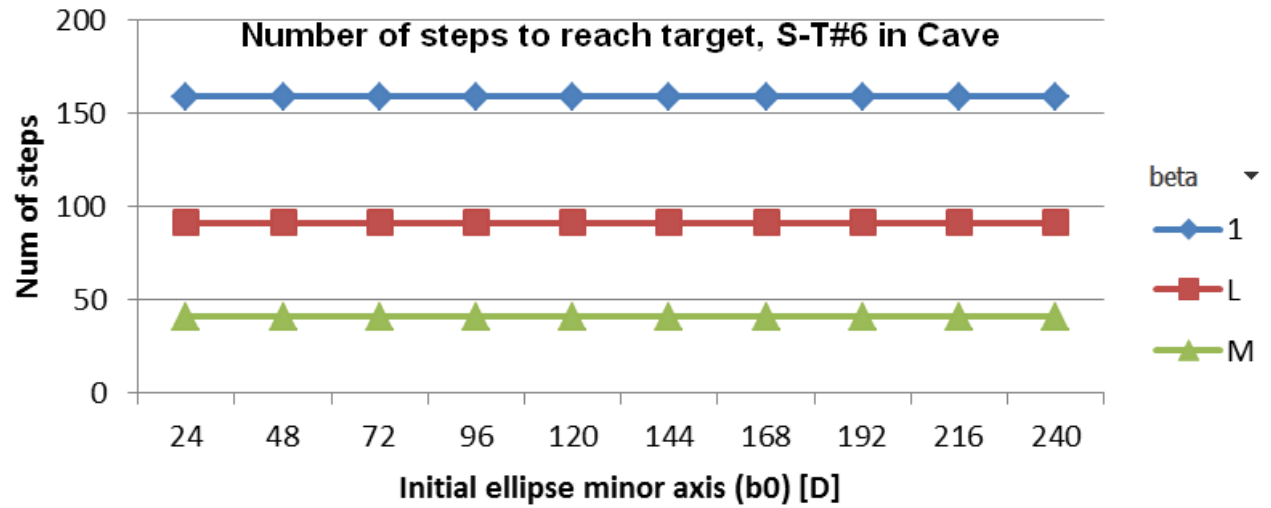


Figure 6.17: HMRBUG simulation results for initial ellipse and beta, S-T#6, cave

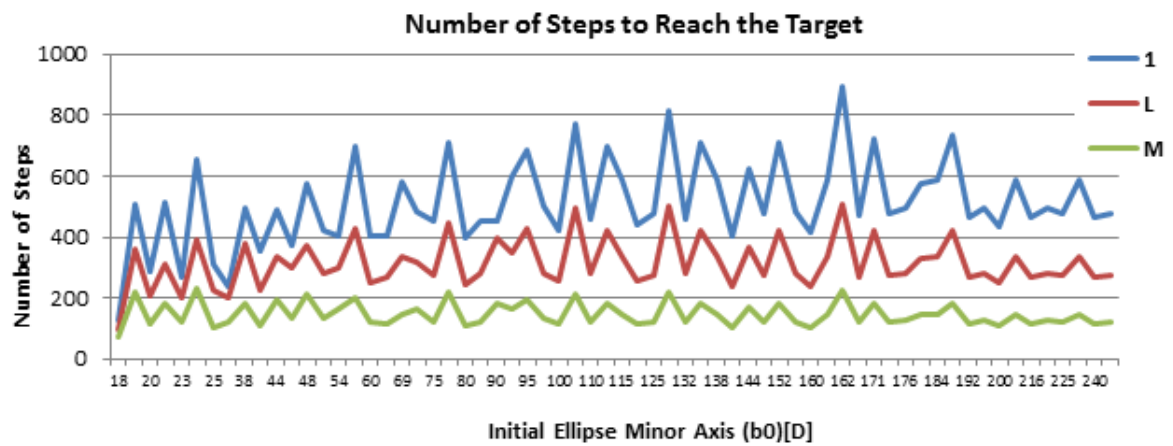


Figure 6.18: HMRBUG simulation results for initial ellipse and beta

The simulation results, which exhibit the average-case performance, were compared with the analytical worst-case upper bound of the algorithm (As presented in Chapter 5). The upper bound of robot j for a certain execution depends on n , α_i , β_i , $i = 1, \dots, n$, $\|\mathcal{S} - \mathcal{T}\|$, D , t_{opt} , v . Thus, it was calculated for each robot (in a pair), for each set of start-target points, for each number of robots, and for each environment. Without loss of generality, $D = 1$, $v = 1$ were assumed. All simulation results were within the upper bound. Moreover, the simulation average-case results were sometimes several orders of magnitude smaller than the analytical worst upper bound. The path length of the simulation for the case of searching for target number 0 in the "cave" environment by groups of heterogeneous robots with beta-max velocity distributions is compared with the analytical upper bound in Fig. 6.19. Since the differences are so great, the vertical axis is logarithmic.

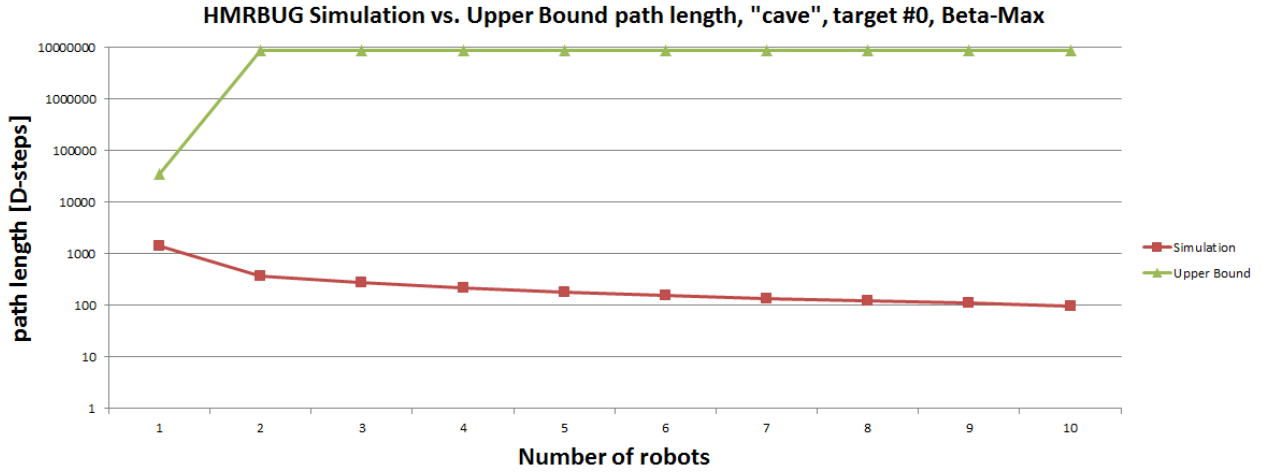


Figure 6.19: HMRBUG simulation vs. analytical upper bound path length

The ratio between the minimal analytical upper bound and the maximal simulation path length is shown in Fig. 6.20. It is evident that this ratio's maximal value in the "cave" environment is 82 and its average is 29, whereas in the "library," the maximal value is 13 and the average is 6.

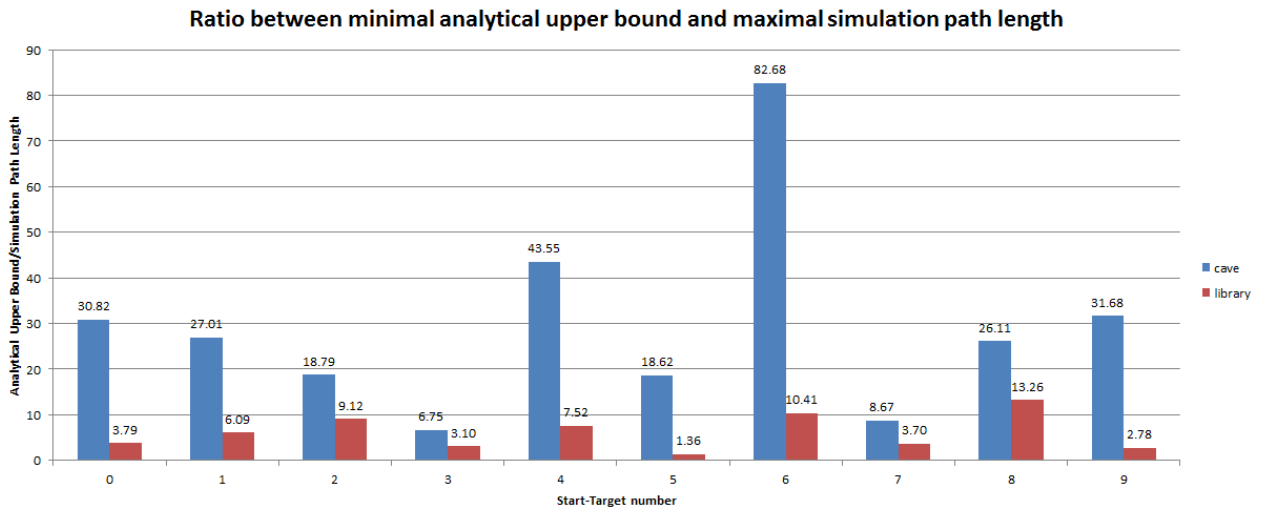


Figure 6.20: HMRBUG ratio between analytical upper bound and maximal simulation path length

Several tens of thousands of simulation runs revealed that the average-case performance of *HMRBUG* depends on several parameters. The use of multiple robots significantly reduces the time to

reach the target, and the use of heterogeneous groups of robots reduces that time even more. Moreover, the more heterogeneous the group is, the better its performance. As predicted, the more congest the environment is, the more time the robots will spend on the search. The distance of the target from the start position obviously affects the time to find the target, but the number and shape of the obstacles between the start and the target positions has a great effect. In accordance with the hypothesis, the initial ellipse search time does not have a great effect on the performance. The average-case performance of *HMRBUG*, as evident from the simulations, is much better than the analyzed worst case-performance.

Chapter 7. Experiments

The experiments aim to validate the simulations and analyze performance in real-world conditions. The main performance measure was defined as the time to find the target in seconds. The test bed and experimental setup are presented in detail in Section 3.6.

7.1 HMRSTM experiments

In *HMRSTM*, the effect of the uncertainty of the positioning system and of the implementation of the algorithm in hardware on the performance of the algorithm was evaluated. Three environments were evaluated, and within each environment three target positions were tested (Figures 3.4 and 3.5). For each configuration, *HMRSTM* deployed between 1 and 4 robots. Three different velocity distributions were evaluated: homogeneous, linear, and "beta-max". Table 7.1 summarizes the parameters tested in *HMRSTM* experiments. For each configuration, two repetitions were conducted, resulting in a total of 162 experiment runs. The experiment's raw results are presented in Appendix A.

Table 7.1: *HMRSTM* experiment configurations

No. of Robots	No. of Environments	Target Positions	Beta tested	Configurations
1	3 (free,cave,libr)	3	1 (H)	9
2	3 (free,cave,libr)	3	2 (H,L)	18
3	3 (free,cave,libr)	3	3 (H,L,M)	27
4	3 (free,cave,libr)	3	3 (H,L,M)	27
Total Configuration				81

7.1.1 Experiment initialization

Prior to the experiment session, the following preparations were made. The hardware equipment, which includes the robots and their batteries, the video camera, the FLEX IR cameras, the PC, the obstacles, and the XBee wireless modules were prepared and activated. The software included the Tracking Tools application, the VRPN client application, the video camera application, and the main experiment C# application. On the main application, the proper parameters which are the environment, the number of robots, the target number, the Beta distribution, and the repetition number were selected. For a detailed explanation of all hardware and software modules see Sec. 3.6.1.

The main application is presented as an example in Figure 7.1. This is the first repetition of the experiment conducted in the "library" environment, where 4 robots with "beta-max" velocity distribution were looking for target number 3. Since the robot does not have a dedicated target sensor, the target position is known to the main application, but unknown to the robot that searches for it.

The map in the top left hand corner shows the current real positions of the robots received from the localization system; each robot is represented by a dot. The next step of each robot according to the algorithm is represented by a colored grid cell. The map shows the target position with a red circle, and each robot's bounding discs in blue. The black rectangle marks the bounds of the localization

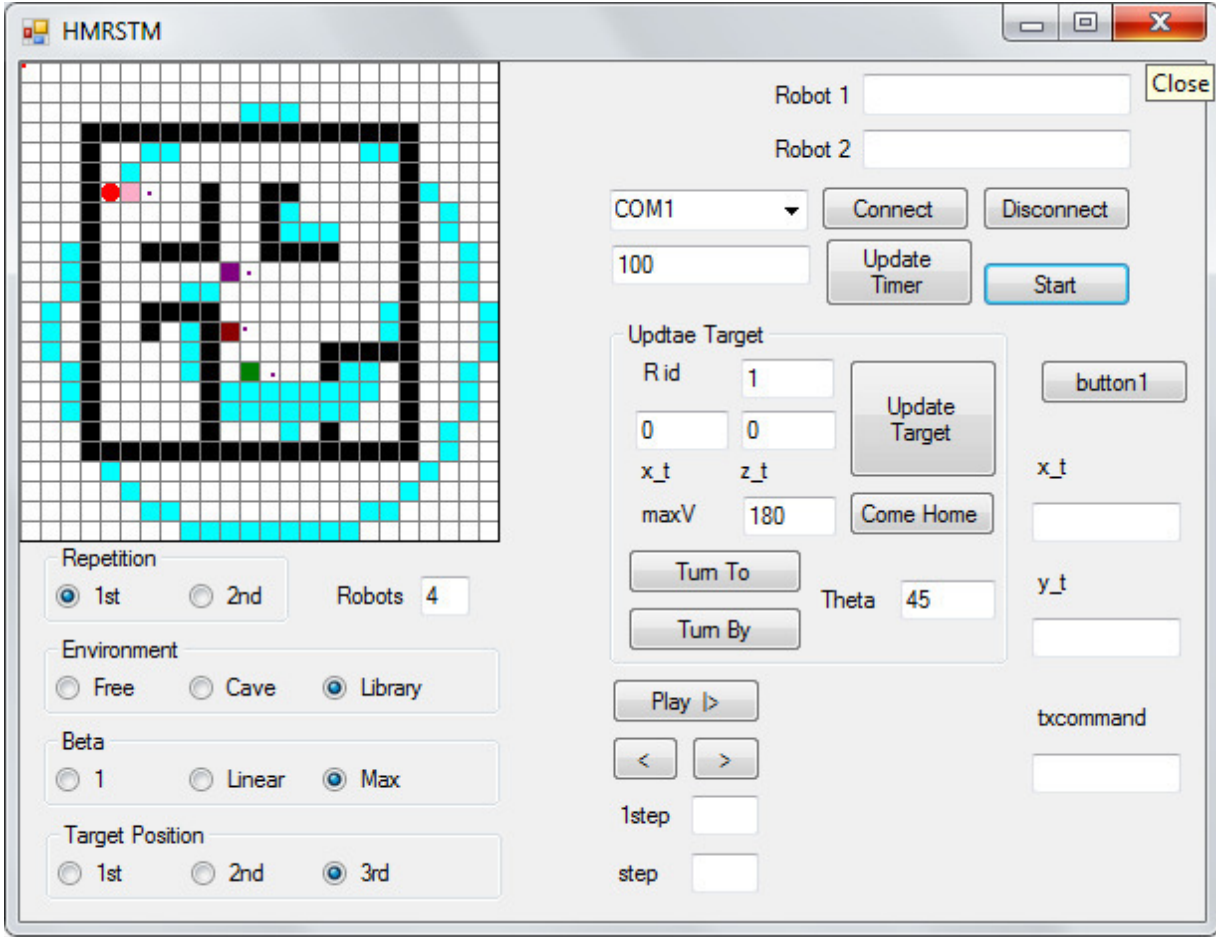


Figure 7.1: HMRSTM main application.

system and thus the bounds of the experiment. The right part contains options to manually control each robot and fields with localization and sensory data (mainly for debugging purposes).

After the aforementioned steps are performed, the robots start to move and search for the target. When one of the robots finds the target, the main application automatically stops all logging and shows a message on screen with the number of the robot who finds the target and the amount of time it took.

7.1.2 Experiment example

Consider the case where a group of 4 robots with linear velocity distribution search for the target positioned in place number 3 in the "library" environment. In simulation, the robots' initial position and discs are depicted in figure 7.2. The colors of the robots are green, red, purple, and pink for robots 1, 2, 3, and 4, respectively. It is evident that the target lies within the fourth disc, but outside of discs 1, 2, and 3.

After robot 1 finished covering its first disc and did not find the target, it moved on to search for it in a larger disc, the next unoccupied disc. The next disc is disc number 5 and is shown in figure 7.3. Robot 2 is in the same position as robot 1, and thus is not shown. It is evident that disc number 5 is slightly larger than disc number 4, and since the starting positions of robots 1 and 4 are adjacent, the discs are not exactly concentric, since each disc is centered at its robot's starting point. In the experiment, after 74 seconds, the first robot finished covering its first disc, and continued to search within disc number 5. Next, after 15 seconds in the experiment, robot number 2 finished covering

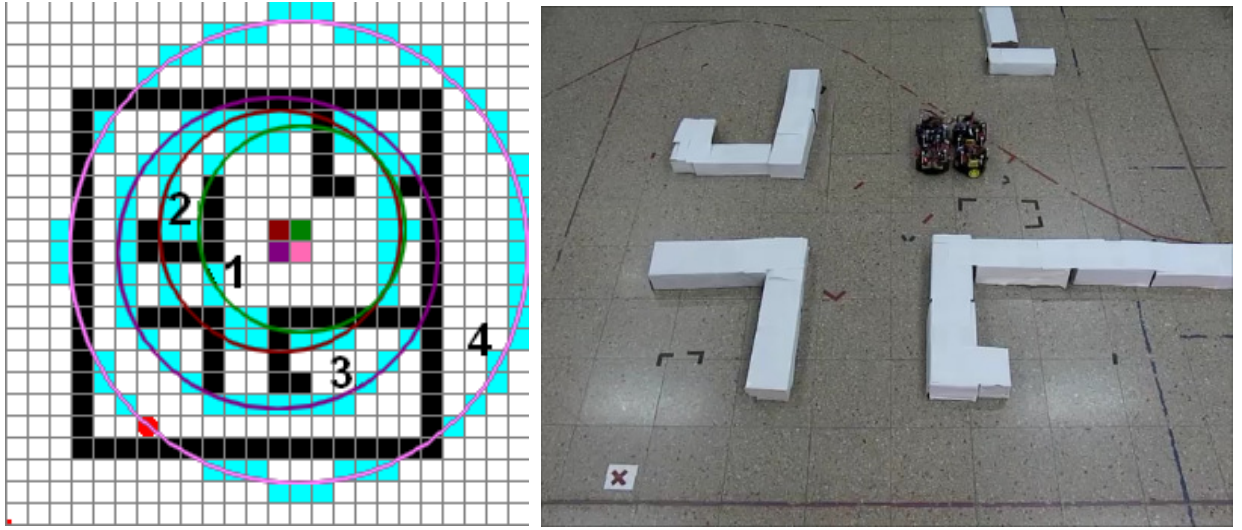


Figure 7.2: Initial position in simulation (left) and experiment (right)

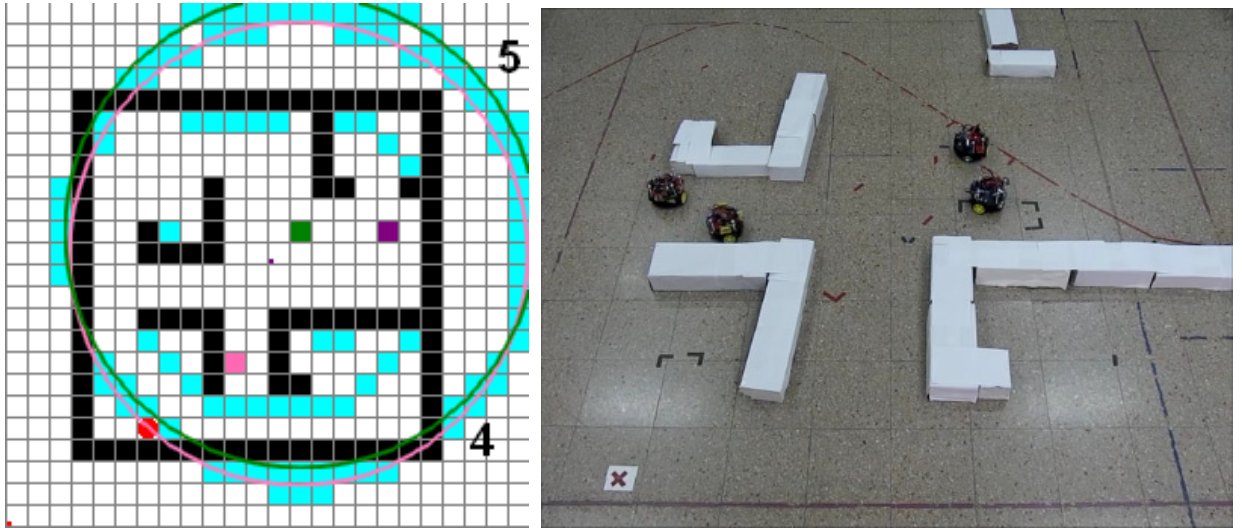


Figure 7.3: Robot 1 expanded its search disc. Left: Robot 2 (red) is in the same position as robot 1 (green) and therefore is not shown.

the second disc without finding the target, so it continued its search in disc 6, depicted in figure 7.4. Finally, robot number four found the target and terminated the algorithm. The positions of all robots a step prior to that moment is depicted in figure 7.5. The total D-steps made by robots 1, 2, 3, and 4 until finding the target were 37, 50, 63, and 77 respectively, which correlates to the linear velocity distribution.

In the experiment, the total D-steps made by robots 1, 2, 3, and 4 until finding the target in the experiment were 60, 75, 65, and 68 respectively. The path lengths that robots 1, 2, 3, and 4 traversed prior to finding the target were 12984, 17053, 15286, and 17095 [mm] respectively, and thus their average speeds were 90.01, 118.22, 105.97, and 118.51 respectively.

The main difference between the simulation and the experiment in this case is that robot 2 found the target in the experiment after 75 D-steps, whereas in the simulation robot 4 found the target after 77 D-steps. The main reason for this difference lies in the fact that in the experiment, the actual velocity distribution was not linear. Moreover, robot 2 was faster than robot 3, and had almost the

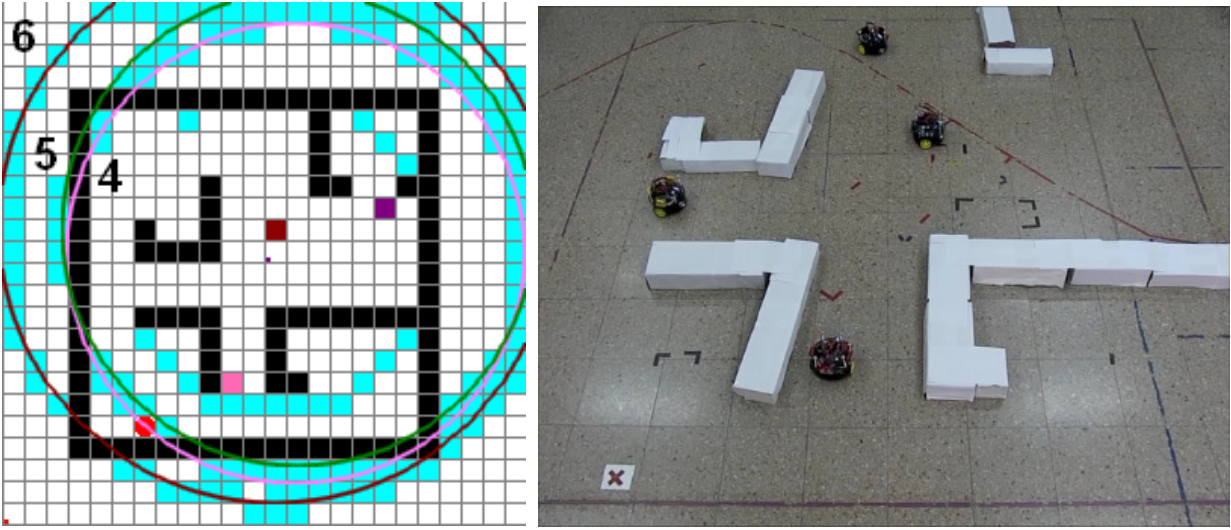


Figure 7.4: Robot 2 expanded its search disc. Left: Robot 1 (green) was in the same position as robot 2 (red) and therefore is not shown.

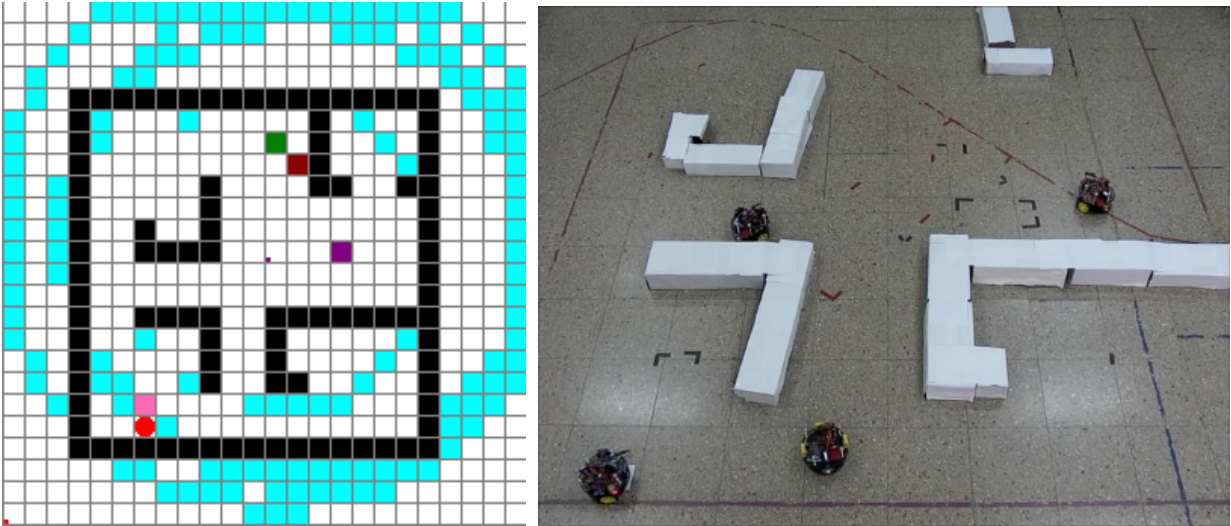


Figure 7.5: The target is found. Left: By robot no.4. Right: By robot no. 2

same velocity as robot 4. Hardware limitations, mainly the robots' motors and the angle control applied, caused the difference in velocity between theory and application. Moreover, in simulation the robots are allowed to pass each other, where actually, in real robots, a mechanism for collision avoidance has been deployed, resulting in delays in travel time.

The main difference between theory and simulation is that in theory the robots start from the same starting point, whereas in simulation and in experiments the robots start from adjacent cells, so their search discs are not concentric.

7.1.3 Experiments results analysis

The experimental results are depicted in Figures 7.6, 7.7, and 7.8. It is evident that the more robots deployed, the less time is needed to find the target. Moreover, it is evident that the more heterogeneous the group is, the more the time to reach the target is reduced. As expected, it is observed that the less congested the environment is with obstacles, the more time is needed to find the target. The last observation is explained by the covering character of the algorithm. Since it covers an area prior to

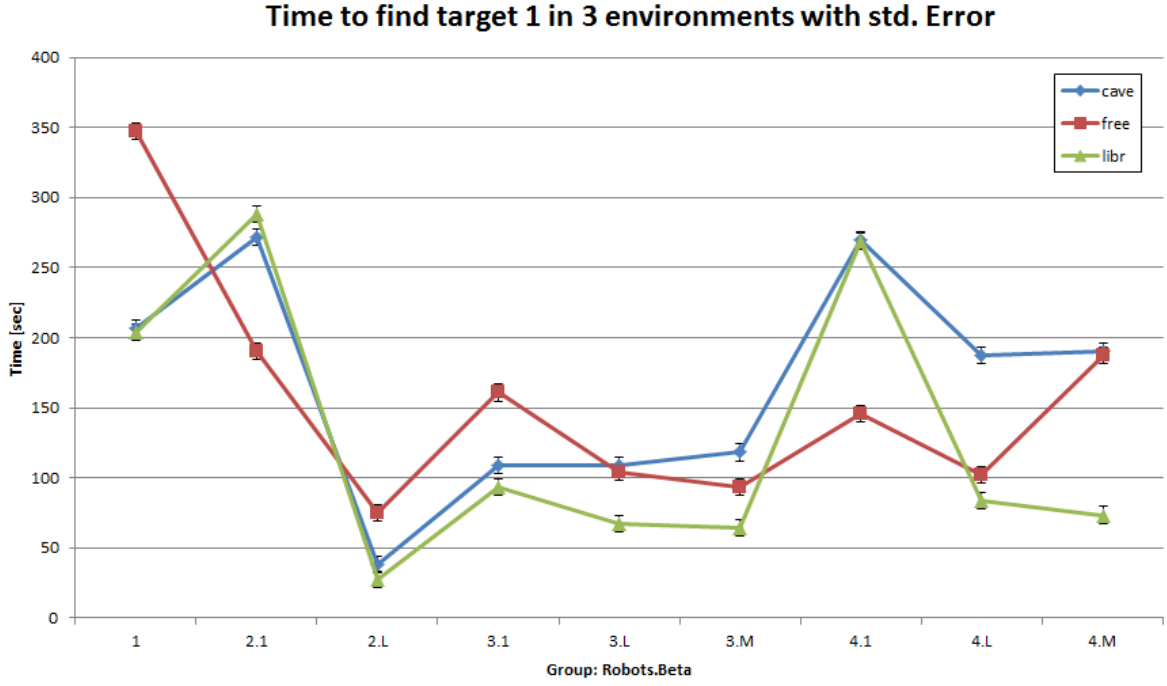


Figure 7.6: Time to find target 1 with std. Error.

finding a target, and since the area in a cluttered environment is smaller than in a non-cluttered one, it will take less time to cover and find a target in a cluttered environment.

Several results do not follow the aforementioned trend. However, none of the paths in the experiments deviates from the maximal expected path (See section 3.6.8). All deviations were investigated, and are explained as follows: From Figure 7.9 it is clear that the group of 3 robots with beta-max (velocity distribution) perform less well than other groups of 3 (beta-1 and beta-linear) and groups of 2 robots (beta-1 and beta-linear). The reason for the deviation stems from the specific case where the velocity distribution is beta-max; thus, the initial search disc is very large and the target is positioned epsilon farther outside the initial search disc of robot 3. Therefore, robot 3 covered the entire disc and did not find the target. Afterwards, it moved on to search for the target in a larger disc where it could reach the target. Thus, this configuration, in which a group of 3 robots with beta-max velocity distribution searches for a target in position 3 and the initial radius is $5D$, is a worst case in terms of performance.

The main performance measure in the experiments is the time to find the target, and is measured in [sec]. However, as noted in 3.6.7, when comparing experiments with simulation time, two problems arise. The first is that the simulation performance measure is normalized time, since we measure path length in [D-steps] and divide it by a virtual velocity [D/sec]. The second, as noted in the previous section, is that, due to hardware limitations, the actual velocity was not as it should be in theory, and thus neither were the velocity distributions in the experiments. For these reasons, we chose to use path length as the performance measure to compare simulation results with experimental results.

Simulation of the experiment environment yielded a *Theoretical* path $l_{Theoretical}$ length. As presented in section 3.6.8, due to hardware limitations, the *Actual* path l_{Actual} was longer. In section 3.6.8, the maximal deviation from the path was calculated. The *Theoretical simulated path* was then factored by the maximal deviation factor to yield the *Maximal* expected path l_{Max} . All path lengths

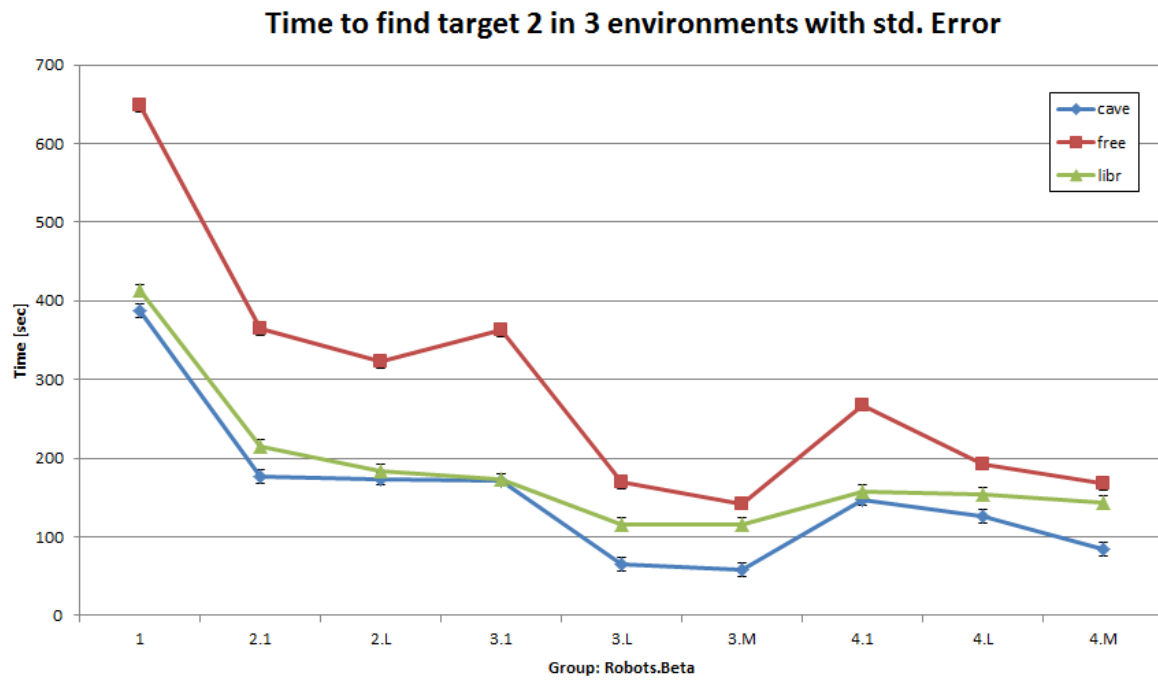


Figure 7.7: Time to find target 2 with std. Error.

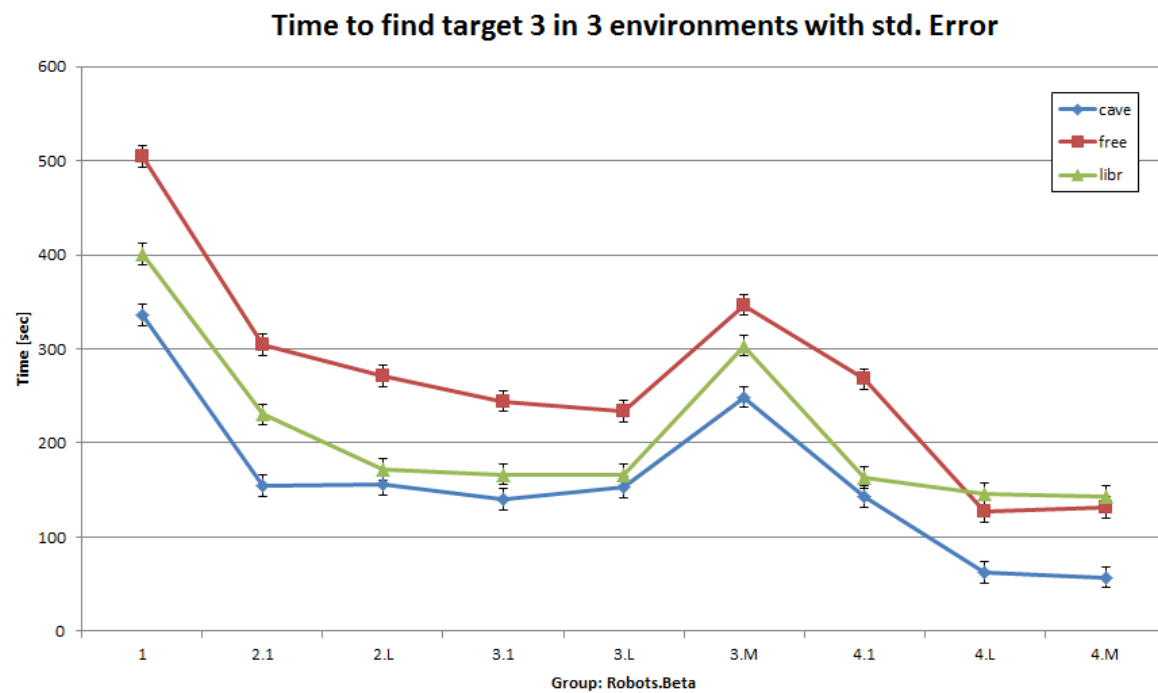


Figure 7.8: Time to find target 3 with std. Error.

in experiments hold the inequality $l_{Actual} < l_{Max}$. In Figure 7.9 both the path length and the maximal expected path for target 3 are presented.

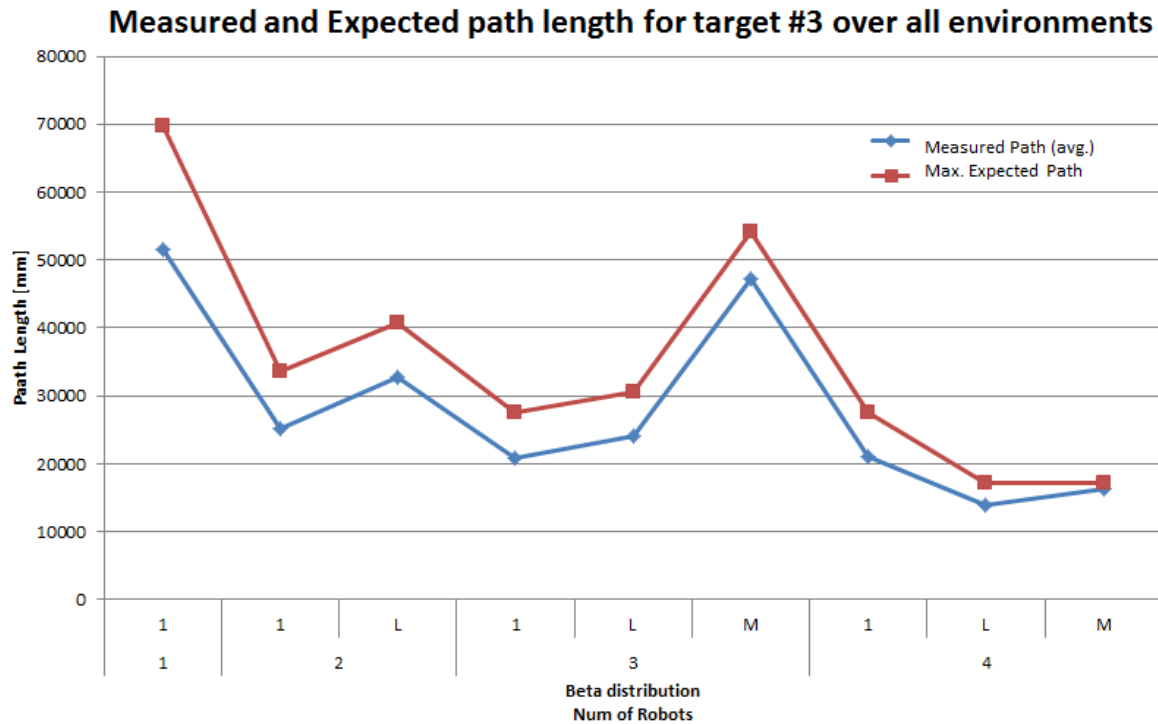


Figure 7.9: Measured and maximal expected path length for target 3.

7.1.4 Statistical analysis

The time to find the target is the variable dependant on the environment's geometry (ENV), the target position (TARGET), and the robot group configuration (GROUP), which is composed of the number of robots and the velocity distribution. The main assumption is that each of the factors is significant. A three way ANOVA of time to find the target was conducted where the three factors were the environment (ENV), the target positions (TARGET), and the group lineup (GROUP) was performed with IBM[®] SPSS[®].

The output, shown in Table 7.2, indicates that all the examined effects were significant, including the main effects, all second order interactions, and the third order interaction (with $p < 0.00001$).

The 3rd order interaction was found to be statistically significant. For example, the results for target 3 are presented in Table 7.3 and in Figure 7.10.

Table 7.2: Tests of Between-Subject Effects for HMRSTM experiments

Source	Type III Sum of Squares	df	Mean Square	F	Sig.
env	191493.836	2	95746.918	622.418	.000
target	136225.631	2	68112.816	442.778	.000
group	943184.333	8	117898.042	766.415	.000
env * target	119220.177	4	29805.044	193.752	.000
env * group	82684.930	16	5167.808	33.594	.000
target * group	391511.987	16	24469.499	159.068	.000
env * target * group	83701.686	32	2615.678	17.004	.000
Error	12460.271	81	153.831		
Corrected Total	1960482.851	161			

Table 7.3: Tests of Between-Subject Effects for Target 3, HMRSTM experiments

Source	Type III Sum of Squares	df	Mean Square	F	Sig.
env	106928.437	2	53464.219	212.159	.000
group	427145.493	8	53393.187	211.877	.000
env * group	22978.553	16	1436.160	5.699	.000
Error	6804.027	27	252.001		
Corrected Total	563856.510	53			

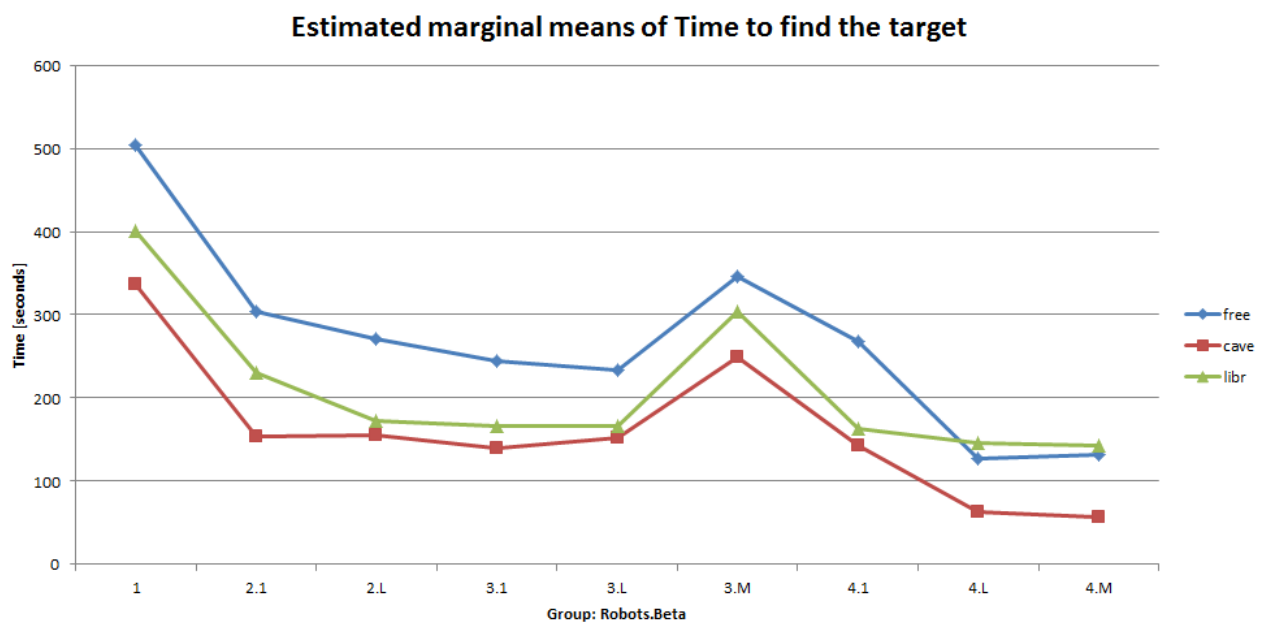


Figure 7.10: HMRSTM experiments' statistical analysis

7.2 HMRBUG experiments

In *HMRBUG*, the effect of the uncertainty of the ultrasonic range sensors, the positioning system, and the implementation of the algorithm in hardware on the performance of the algorithm was evaluated. Three environments were evaluated, and within each environment two target positions were tested (Figures 3.4 and 3.5). For each configuration, *HMRBUG* deployed one and two pairs of robots. Two different velocity distributions were evaluated: homogeneous and linear. Table 7.4 summarizes the parameters tested in *HMRBUG* experiments. For each configuration, two repetitions were conducted, resulting in a total of 36 experiment runs. The experiments' raw results are presented in Appendix A.

Table 7.4: *HMRBUG* experiment configurations

No. of Robot Pairs	No. of Environments	Start-Target Positions	Beta tested	Configurations
1	3 (free,cave,libr)	2	1 (H)	6
2	3 (free,cave,libr)	2	2 (H,L)	12
Total Configuration				18

7.2.1 Experiment example

Experiment initialization in *HMRBUG* is the same as in *HMRSTM* (7.1.1). Consider the case in which a group of 2 robot pairs (4 robots) with linear velocity distribution searches for a path from start to target position 2 in the "library" environment. The robots' initial position and ellipses are depicted in figure 7.11. The colors of the robots are green for robots 1 and 2, which belong to the first

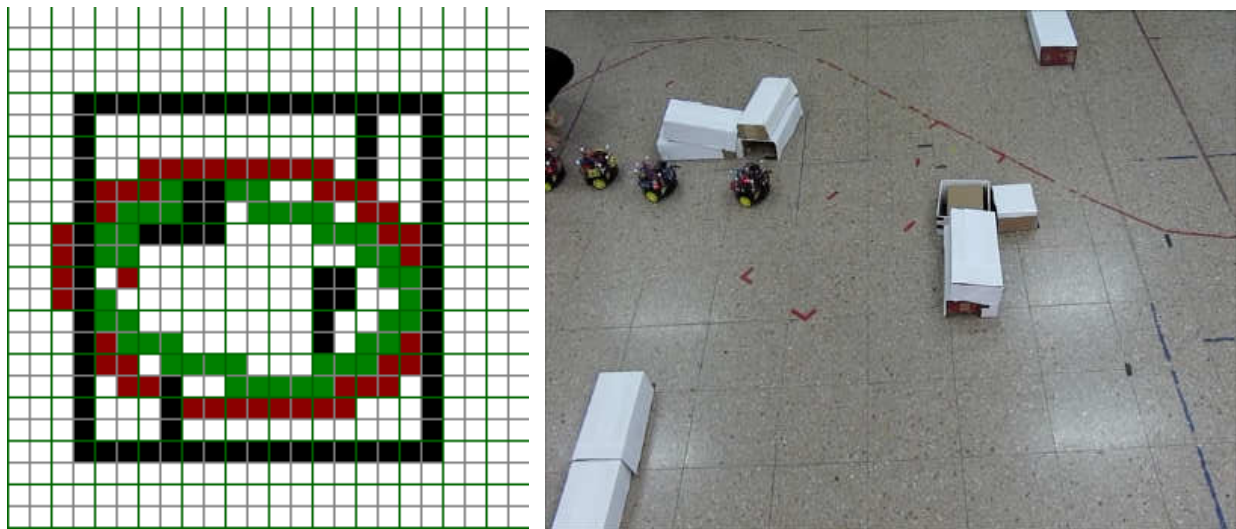


Figure 7.11: Initial position in simulation (left) and experiment (right)

pair, and red for robots 3 and 4 of the second pair. It is evident that a path to the target lies within the second ellipse, but outside the first ellipse. In simulation, all four robots started from the same point, but in the experiment, robot 3 was the first, and after it came robots 4, 1, and 2, in that order.

Pair 1 and 2 moved directly towards the target in a straight line. Pair 2 reached and started circumnavigating the obstacle first. Afterward, pair number 1 reached the obstacle, started circum-

navigating it, but before completing the circumnavigation, met its bounding ellipse and continued circumnavigating it (Fig. 7.12). Meanwhile, robot pair 2 finished encircling the obstacle's boundary, and moved to the closest point to the target. Finally, robot pair number 2 again moved directly

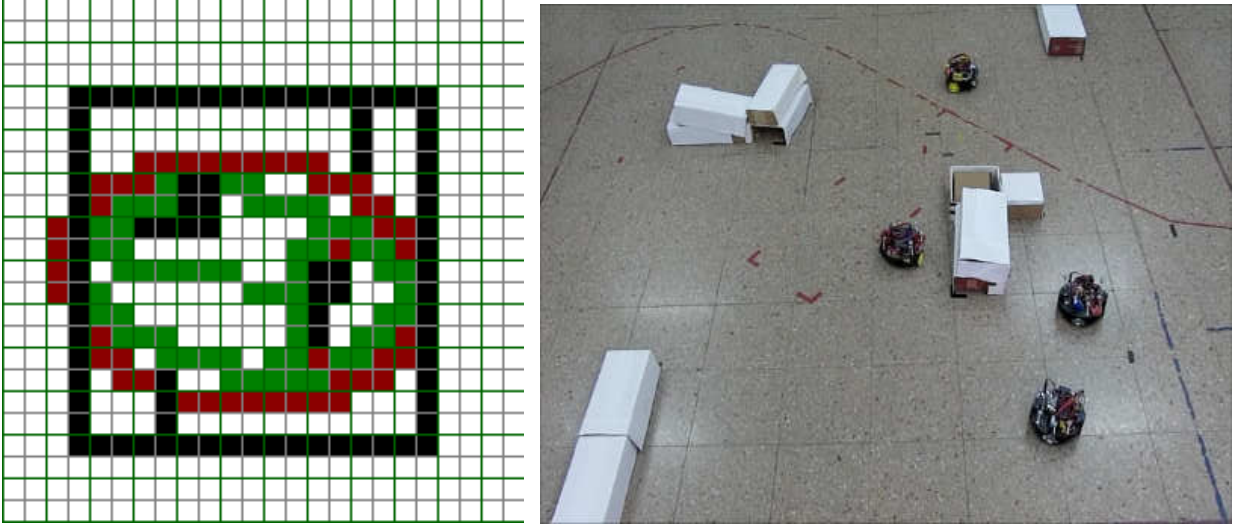


Figure 7.12: Pair number 1 circumnavigates its bounding ellipse, and pair number 2 circumnavigates the obstacle

towards the target and reached it (Fig. 7.13).

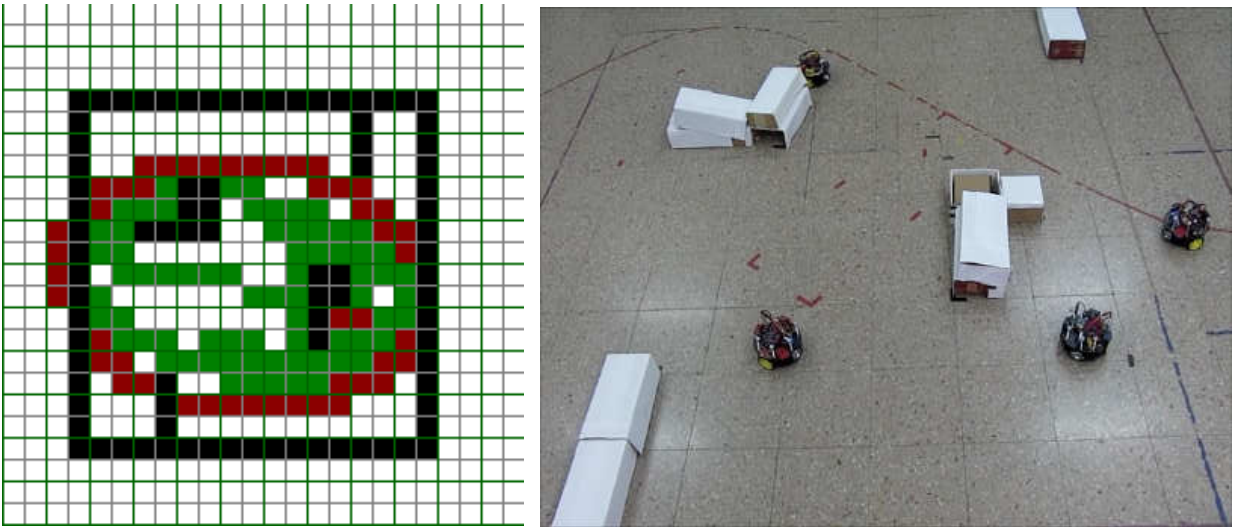


Figure 7.13: Robot pair number 2 reach the target

The total path length in simulation is 3400[mm], and the total path length in the experiment was 5532[mm]. This deviation is within the bounds calculated in section 3.6.8.

The main difference between the simulation and the experiment is due to the actual path traveled while circumnavigating an obstacle. While in simulation the path is touching the obstacle, in the experiment it is farther away. The main reason for this difference lies in the fact that the ultrasonic sensor error imposed a conservative boundary circumnavigation method. Moreover, in the experiment, the actual velocity difference between robot pairs was not as great as it is in theory. Hardware limitations, mainly the robots' motors and the angle control applied, caused the difference in velocity between theory and application. Moreover, in simulation the robots are allowed to pass one another,

where actually, in real robots, a mechanism for collision avoidance has been deployed, resulting in delays in travel time.

The main difference between theory and simulation is that in theory the robots start from the same starting point, where in simulation and in experiments the robots start from adjacent points, yielding a long "start line" in the case of four robots.

7.2.2 Experiments results analysis

The experimental results are depicted in Figures 7.14, 7.15, and 7.16. It is evident that the more robots deployed, the less time is needed to reach the target. Moreover, it is evident that the more heterogeneous the group is, the less time it takes it to reach the target. None of the paths in experiments deviate from the maximal expected path (See section 3.6.8).

Most of the path lengths in the experiment are longer than those in simulation due to the aforementioned reasons. However, in several cases they are shorter. This situation occurs when the robots are circumnavigating an obstacle and accidentally reach the target. In this way, the circumnavigation of the obstacle stops and the algorithm terminates, resulting in a shorter path. In simulations, reaching the target while circumnavigating an obstacle is possible only when the target is adjacent to the obstacle's boundary.

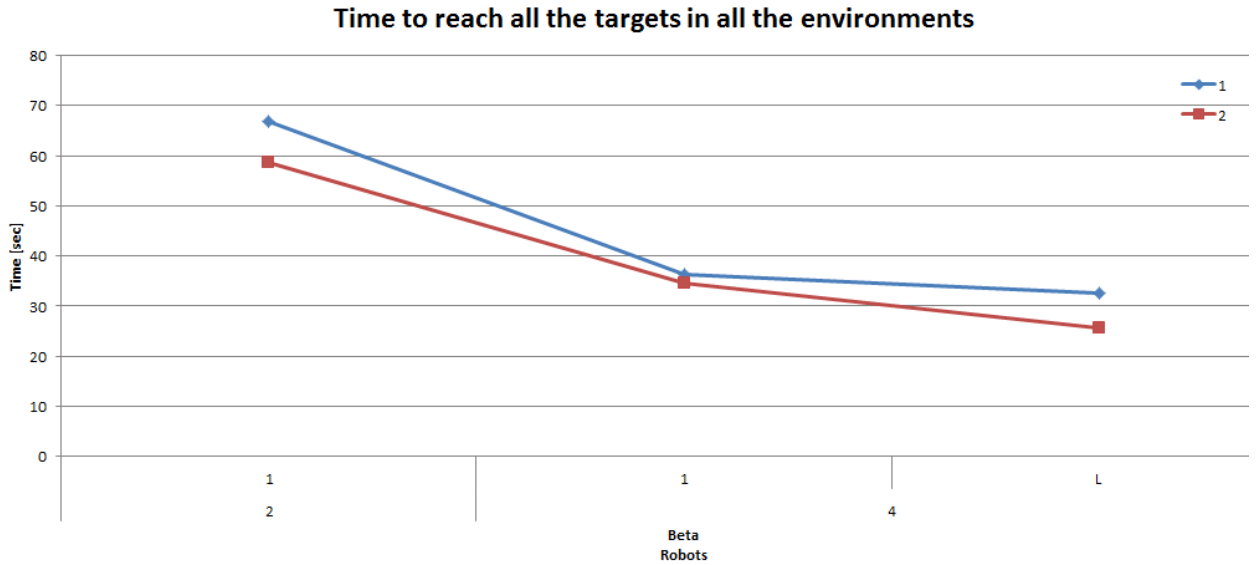


Figure 7.14: Time to reach all the targets in all environments

The main performance measure in the experiments is time to find the target, and is measured in [sec]. However, as noted in 3.6.7, when comparing experiment time with simulation time, two problems arise. The first is that the simulation performance measure is normalized time, since we measure path length in [D-steps] and divide it by a virtual velocity [D/sec]. The second, as noted in the previous section, is that the actual velocity was not as it should be in theory, due to hardware limitations, and thus neither were the velocity distributions in the experiments. For these reasons, we chose to use path length as a performance measure when comparing simulation with experiment results.

Simulation of the experiment environment yielded a *Theoretical* path $l_{Theoretical}$ length. As presented in section 3.6.8, due to hardware limitations, the *Actual* path l_{Actual} was longer. In section 3.6.8, the maximal deviation from the path was calculated. The *Theoretical simulated path* was then

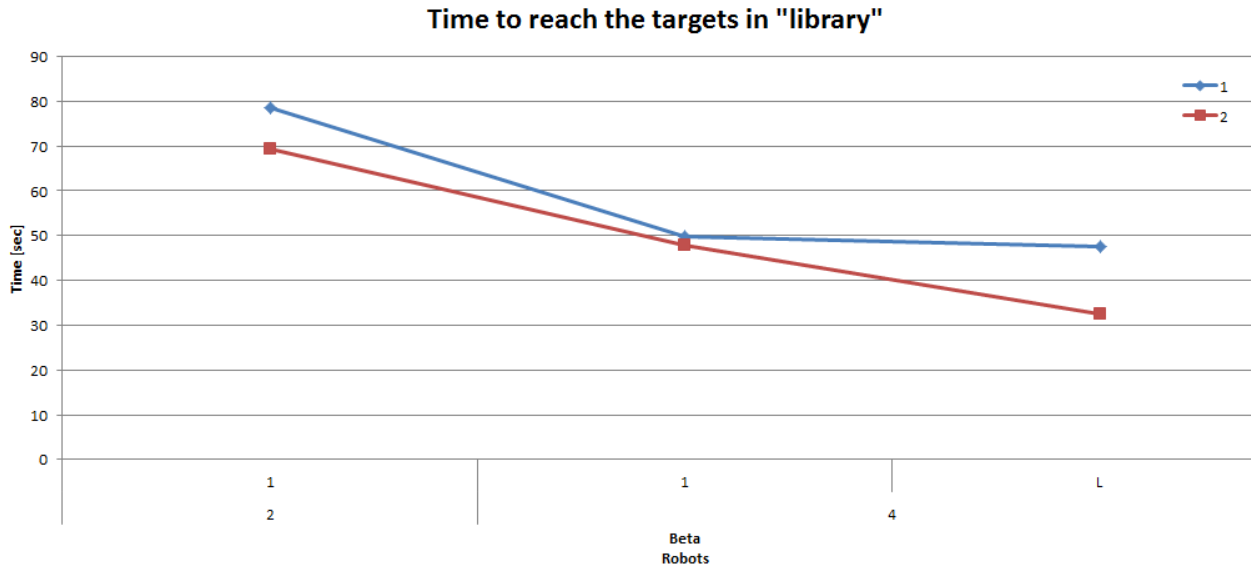


Figure 7.15: Time to reach the targets in "library"

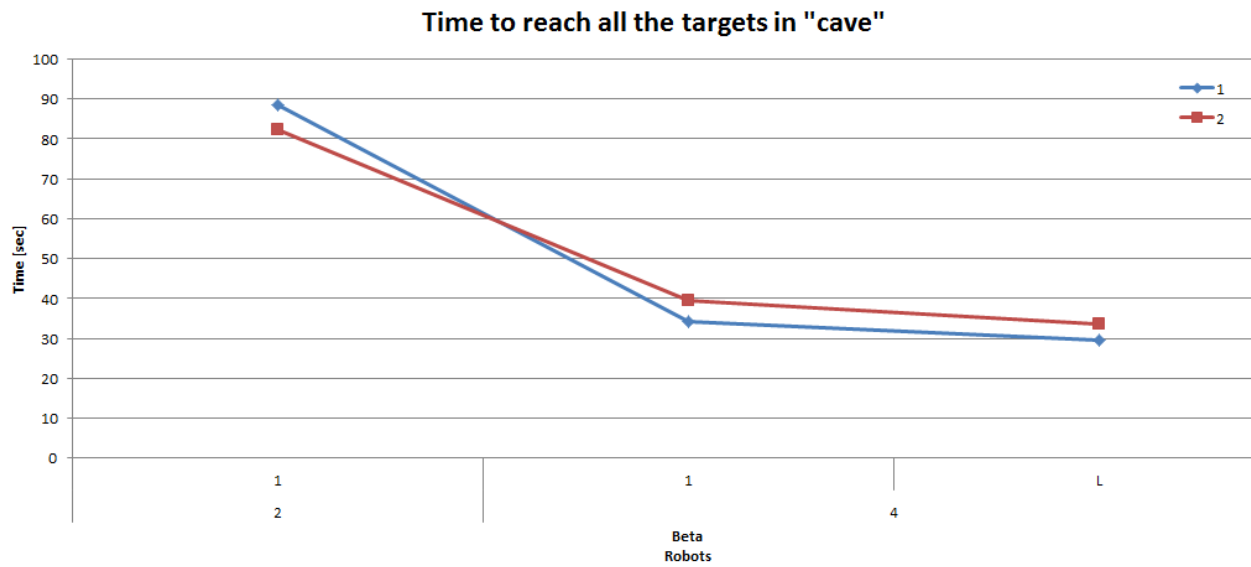


Figure 7.16: Time to reach the targets in "cave"

factored by the maximal deviation factor to yield the *Maximal* expected path l_{Max} . All path lengths in experiments hold the inequality $l_{Actual} < l_{Max}$. In Figure 7.17 both the path length and the maximal expected path for target 3 are presented.

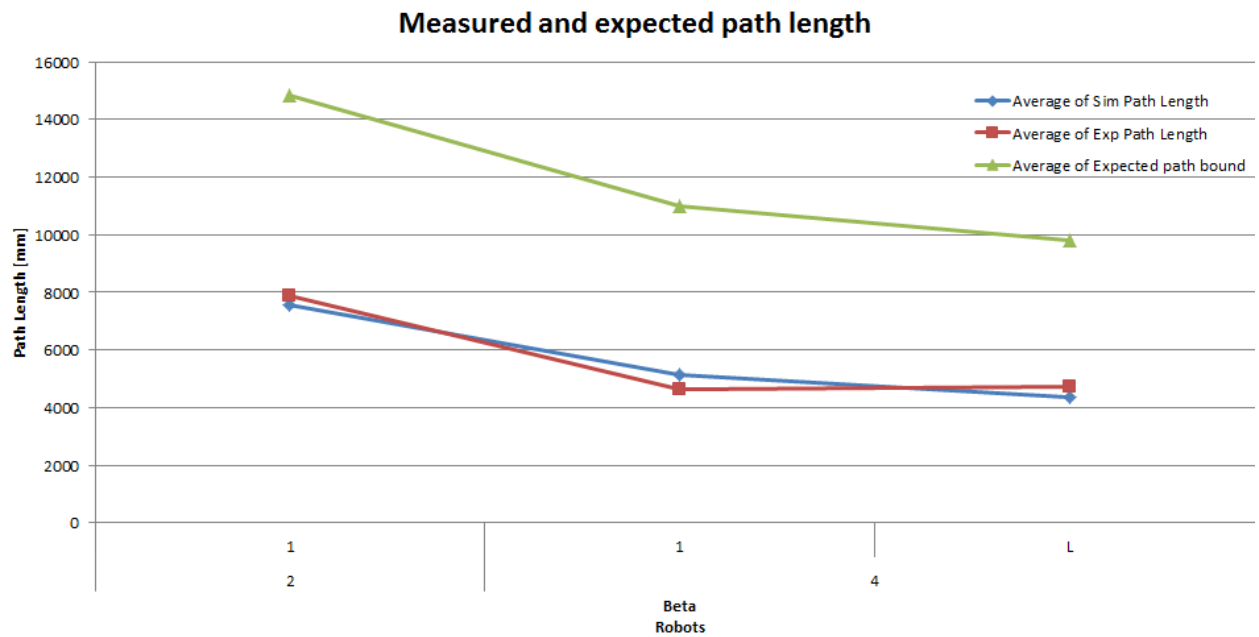


Figure 7.17: Measured and maximal expected path length.

7.2.3 Statistical analysis

The time to find the target is the variable dependant on the environment's geometry (ENV), the target position (TARGET), and the robot group configuration (GROUP), which is composed of the number of robots and the velocity distribution. The main assumption is that each of the factors is significant. A three way ANOVA of time to reach the target was conducted where the three factors were the environment (ENV), the target positions (TARGET), and the group lineup (GROUP) was performed with IBM[®] SPSS[®].

The output, shown in Table 7.5, indicates that the main effects for ENV and GROUP and their interaction are significant (with $p < 0.00001$). However, any effects that involve TARGET are not statistically significant ($p > 0.05$). This phenomenon can be explained by the relatively similar path length of the two Start-Target configurations, which is a limitation of the small experiment environment and algorithm implementation. In a larger environment, different paths would probably yield experiments with significant effect of start-target positions.

Table 7.5: Tests of Between-Subject Effects for HMRBUG experiments

Source	Type III Sum of Squares	df	Mean Square	F	Sig.
env	12281.386	2	6140.693	47.090	.000
target	118.763	1	118.763	.911	.350
group	6942.591	2	3471.295	26.620	.000
env * target	173.425	2	86.713	.665	.524
env * group	3909.173	4	977.293	7.494	.001
target * group	90.850	2	45.425	.348	.710
env * target * group	102.898	4	25.724	.197	.937
Error	2999.290	23	130.404		
Corrected Total	29894.548	40			

The results are presented in Fig. 7.18

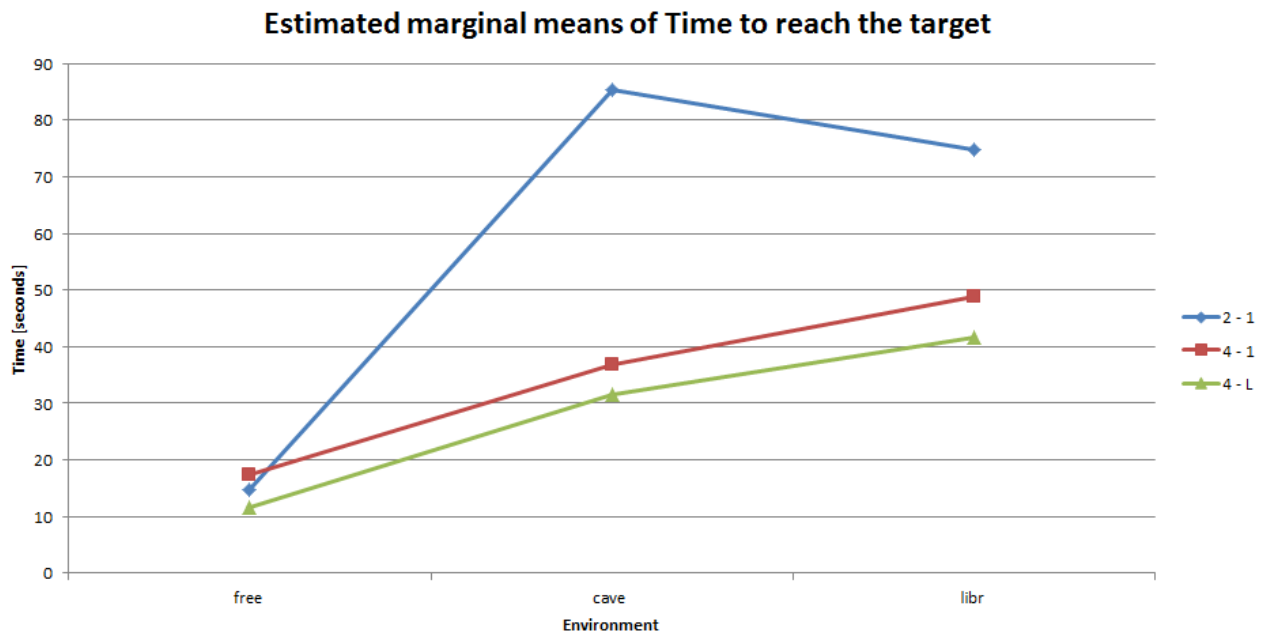


Figure 7.18: HMRBUG experiments statistical analysis

Chapter 8.

Conclusions and Future Research

8.1 Conclusions

This thesis describes the research of two motion planning problems of *finding a path to a target* by a *group of heterogeneous robots*. In the first problem, the target position is unknown, and in the second problem the target position is known. In both problems, *the search environment is a priori unknown and unbounded*. The research includes classification of the problems into *quadratic time competitive complexity classes* formed from lower and upper bounds of the same functional relation. For each problem, a *quadratic time lower bound* was found, and the *quadratic time upper bound* was obtained from a newly developed algorithm. The newly developed algorithms, *HMRSTM*, and *HMRBUG*, utilize a group of velocity heterogeneous robots to find a path to the target. They are proved to be *optimal, complete, and robust*.

The worst-case performance of *HMRSTM* and *HMRBUG* was analyzed. The average-case performance of both algorithms was evaluated in extensive *simulation runs*. It is evident that the average-case performance of *HMRSTM* and *HMRBUG* is much better than the worst-case performance. *It is evident that a heterogeneous group of robots performs better than a homogenous group of robots*. Moreover, it is evident that *maximal velocity distribution* performs better than linear velocity distribution.

Simulations were validated through *experiments* with real robots. The deviations in performance of the experiments from the simulations were caused by hardware limitations and algorithm implementation, and were within the expected bounds. Thus, *the new developed algorithms, HMRSTM, and HMRBUG, not only have an optimal worst-case upper bound, but also perform much better in the average case and are suitable for use in real robots in real environments*.

8.2 Future Work

Future research should evaluate the following:

- **Heterogeneity of robot size and coverage tool:** Heterogeneity in the velocities of the robots can be considered analogous to their size in the sense of the covering tool's area, assuming a covering tool the size of the robot, e.g., a robot with a size $2D$ with a velocity β will cover the same area as a robot with size D and velocity 2β both working the same amount of time. Thus, converting the algorithms from the velocity heterogeneity to the robots' size heterogeneity is straightforward. However, a thorough investigation should be conducted to cover all aspects of this correlation prior to implementation.
- **Define a measure of crowdedness** The average-case performance was evaluated in three environments, which were free from obstacles, lightly congested with obstacles, and highly congested respectively. Defining a measure of crowdedness for an environment according to geometric parameters such as obstacles area versus free area, or obstacle shapes, and measuring the average

performance according to this measure, will enable the anticipation of the performance in other environments, not tested before, according to that measure.

- **Communication heterogeneity** Heterogeneity of communication capabilities should be explored, since minimizing the communication equipment and transmission results in reduced system price and decreased power consumption, implying longer battery life.
- **Improve average-case performance:**
 - * **Use a common map between the robots** In *HMRSTM* and in *HMRBUG* each robot executes the algorithm in a decentralized way, and no communication between the robots is present. Thus, the robots do not know about areas already traveled by other robots, and redundant traversing is evident. If the robots could communicate between themselves, and pass along information regarding the obstacles and the free area in the environment they have already inspected, they could build a common map of the environment and perform much better by minimizing the redundant traverse.
 - * **Search within the ring added in *MRSTM*:** Since each next disc contains the previous search disc, redundancy in the covered area is evident. Enabling the assignment of only the ring added to the last covered disc instead of the whole next disc to the robot currently finished covering its disc will reduce or eliminate this redundancy. Such improvement is possible only through the use of the common map of the environment created and communicated between the robots on the moves
 - * **Improve inner disc search in *HMRSTM* algorithm:** A parameter not included in the theoretical analysis, but which plays an important role in the experiments, is the motion planning method used by the robots to cover the area of a disc, i.e, in the formal description of the algorithm each robot is instructed to make a coverage tour in the disc to which it was assigned. The covering method offered was grid-based *DFS* or *STC* [18] since its upper bound is well defined to be twice the disc's area. However, grid-based covering methods usually restrict the movement of the robots to the neighboring cells and thus break the continuous movement with many turns, especially ninety degrees turns. Such turns do not influence the total path traveled by the robots, but, unless using an omni-wheel platform, physical turning takes a significant amount of time. Consequently, the number of turns is also a performance measure, first by counting them, and second by including the time each turn takes as a fraction of one physical step's time. Replace *Spiral STC* with other coverage algorithms for bounded environments (e.g., *the use of edge-weighting STC*[17]) .
 - * **Improve inner ellipse search in *HMRBUG* algorithm:** Replace *PBUG* with other versions of target finding algorithms (e.g., *BUG2 TangentBUG*).
 - * **Deploy multi robots within each disc in *HMRSTM* algorithm:** Use multi-robot coverage algorithms for bounded environments within each disc (e.g., *MSTC*).
 - * **Utilize long range sensor:** Sensory heterogeneity in this part plays an important role in the obstacle's detection and in the target detection sensors. Touch or short-range sensors for detecting obstacles are employed, and the target is sensed when the robot is positioned

on top of the target or upon arrival at the target. The use of long-range sensors such as laser scanner and vision cameras for such purposes introduces extended capabilities that may result in better performance in the average case. However, in congested environments and worst-case scenarios, such improved sensory equipment will yield the upper bounds of the basic system. Nonetheless, the use of more advanced sensors is important and should be examined. Utilization of the aforementioned sophisticated sensors involves vast modification to the algorithm or writing a completely new one.

- * **Utilize different roles for the robots:** Use fast robots as scouts to partially explore the unknown area prior to deploying the other covering robots. This algorithm makes use of long-range sensors for the scouts and short-range sensors for the coverers. This method may prove efficient in inter-disc searches of multi-robots.
- **Test in simulations:**
 - * **Practical speedups** Examine various versions of practical speedup mechanisms along with the relevant parameters of optimization.
 - * **Unbounded environments** Test much larger environments.
- **Test in experiments:**
 - * **Number of robots:** In simulation up to twenty robots are employed in one execution. In the experiments, up to four robots were deployed. Despite the relatively small number of robots, mechanisms for collision avoidance and sensor synchronization were already needed. The relatively small experiment area limited the performance of multiple robots. A measure of when the increase in the number of robots decreases performance according to the size and other parameters of the environment should be found. Experimenting with more robots, possibly in larger environments, in light of these difficulties and finding the appropriate solutions is an important future direction.
 - * **Comparing with errors in simulations** Introducing into the simulation software errors approximating the size of the errors received from the sensory equipment on board the robots.

Bibliography

- [1] Noa Agmon, Noam Hazon, and Gal A. Kaminka. Constructing Spanning Trees for Efficient Multi-Robot Coverage. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA '06)*, pages 1698–1703, May 2006.
- [2] Mazda Ahmadi and Peter Stone. A Multi-Robot System for Continuous Area Sweeping Tasks. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA '06)*, pages 1724–1729, May 2006.
- [3] Gianluca Antonelli, Filippo Arrichiello, Stefano Chiaverini, and Roberto Setola. A Self-Configuring MANET for Coverage Area Adaptation through Kinematic Control of a Platoon of Mobile Robot. In *Proceedings of 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'05)*, pages 1307–1312, Edmonton, Alberta, Canada, August 2005.
- [4] Ronald C. Arkin and Jonathan Diaz. Line-of-sight constrained exploration for reactive multiagent robotic teams. In *7th International Workshop on Advanced Motion Control*, pages 455–461, 3-5 July 2002.
- [5] Ricardo A. Baeza-Yates, Joseph C. Culberson, and Gregory J. E. Rawlins. Searching in The Plane. *Information and Computation*, 106(2):234–252, 1993.
- [6] Spring Berman, Adam Halasz, Vijay Kumar, and Stephen Pratt. Bio-Inspired Group Behaviors for the Deployment of a Swarm of Robots to Multiple Destinations. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA '07)*, pages 2318–2323, Rome, Italy, April 2007.
- [7] Jeffrey J. Biesiadecki and Mark W. Maimone. The mars exploration rover surface mobility flight software: Driving ambition. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA '06)*, Big Sky, Montana, USA, March 2006.
- [8] Bradley Bishop. On the Use of Capability Functions for Cooperative Objective Coverage in Robot Swarms. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA '07)*, pages 2306–2311, Rome, Italy, April 2007.
- [9] Jonathan Brookshire, Sanjiv Singh, and Reid Simmons. Preliminary results in sliding autonomy for coordinated teams. In *Proceedings of The 2004 Spring Symposium Series*, March 2004.
- [10] Joseph Carsten, Arturo Rankin, Dave Ferguson, and Anthony Stentz. Global path planning on board the mars exploration rovers. In *IEEE Aerospace Conference*, pages 1–11, Big Sky, Montana, USA, March 2007.

- [11] Jenhui Chen and Li-Ren Li. Path planning protocol for collaborative multi-robot systems. In *Proceedings 2005 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 721–726, Espoo, Finland, 2005.
- [12] Howie Choset. Coverage for robotics - a survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31:113 – 126, 2001.
- [13] Howie Choset, Kevin M. Lynch, Seth Hutchinson, GeorgeA Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Cambridge, MA, June 2005.
- [14] Howie Choset and Philippe Pignon. Coverage path planning: The boustrophedon decomposition. In *International Conference on Field and Service Robotics*, 1997.
- [15] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 2001.
- [16] Ettore Ferranti, Niki Trigoni, and Mark Levene. Brick&Mortar: an on-line multi-agent exploration algorithm. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA '07)*, pages 761–767, Rome, Italy, April 2007.
- [17] Yoav Gabriely and Elon Rimon. Spanning-tree based coverage of continuous areas by a mobile robot. *Annals of Mathematics and Artificial Intelligence*, 31:77–98, 2001.
- [18] Yoav Gabriely and Elon Rimon. Competitive on-line coverage of grid environments by a mobile robot. *Computational Geometry Theory and Applications*, 24(3):197–224, 2003.
- [19] Yoav Gabriely and Elon Rimon. Competitive complexity of mobile robot on line motion planning problems. In *Workshop on Algorithmic Foundations of Robotics*, pages 249–264, 2004.
- [20] Yoav Gabriely and Elon Rimon. CBUG: A quadratically competitive mobile robot navigation algorithm. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA '05)*, pages 954–960, 2005.
- [21] Yoav Gabriely and Elon Rimon. Competitive Disconnection Detection in On-Line Mobile Robot Navigation. In *Proceeding of Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2006.
- [22] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, January 1979.
- [23] Shuzhi Sam Ge and Cheng-Heng Fua. Complete Multi-Robot Coverage of Unknown Environments with Minimum Repeated Coverage. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA '05)*, pages 727–732, April 2005.
- [24] Brian P. Gerkey and Maja J. Mataric. A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems. *International Journal of Robotic Research*, 23(9):939–954, 2004.

- [25] Enrique Gonzalez, Oscar Alvarez, Yul Diaz, Carlos Parra, and Cesar Bustacara. BSA: A Complete Coverage Algorithm. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA '05)*, pages 2052–2056, April 2005.
- [26] Noam Hazon and Gal A. Kaminka. Redundancy, Efficiency, and Robustness in Multi-Robot Coverage. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA '05)*, April 2005.
- [27] Noam Hazon, Fabrizio Mieli, and Gal A. Kaminka. Towards Robust On-line Multi-Robot Coverage. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA '06)*, pages 1710–1715, May 2006.
- [28] Andrew Howard and Nicholas Roy. The Robotics Data Set Repository (Radish). <http://radish.sourceforge.net/>, 2003.
- [29] Christian Icking, Thomas Kamphans, Rolf Klein, and Elmar Langetepe. On the Competitive Complexity of Navigation Tasks. In *Revised Papers from the International Workshop on Sensor Based Intelligent Robots*, pages 245–258, London, UK, 2002. Springer-Verlag.
- [30] Luca Iocchi, Daniele Nardi, Maurizio Piaggio, and Antonio Sgorbissa. Distributed Coordination in Heterogeneous Multi-Robot Systems. *Auton. Robots*, 15(2):155–168, 2003.
- [31] Songmin Jia and Kunikatsu Takase. A RTM-Based Home Service Robots Monitoring System. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA '07)*, pages 2880–2885, Rome, Italy, April 2007.
- [32] Edward Jones, Brett Browning, M. Bernardine Dias, Brenna Argall, Manuela Veloso, and Anthony(Tony) Stentz. Dynamically Formed Heterogeneous Robot Teams Performing Tightly-Coordinated Tasks. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA '06)*, pages 570–575, Orlando, FL, May 2006.
- [33] Gal A. Kaminka and Inna Frenkel. Integration of Coordination Mechanisms in the BITE Multi-Robot Architecture. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA '07)*, pages 2859–2866, Rome, Italy, April 2007.
- [34] Sven Koenig and Yaxin Liu. Terrain Coverage with Ant Robots: A Simulation Study. In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, pages 600–607, New York, NY, USA, 2001. ACM Press.
- [35] Chan Sze Kong, New Ai Peng, and Ioannis Rekleitis. Distributed Coverage with Multi-Robot System. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA '06)*, pages 2423–2429, May 2006.
- [36] Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer, Germany, 3rd edition, 2006.
- [37] Daisuke Kurabayashi, Jun Ota, Tamio Arai, and Eiichi Yoshida. Cooperative sweeping by multiple mobile robots. In *Proceedings of the 1996 IEEE International Conference on Robotics and Automation (ICRA '96)*, Minneapolis, Minnesota, April 1996.

- [38] Miu-Ling Lam and Yun-Hui Liu. Heterogeneous Sensor Network Deployment Using Circle Packings. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA '07)*, pages 4442–4447, Rome, Italy, April 2007.
- [39] Svetlana Larionova, Nuno Almeida, Lino Marques, and A. T. Almeida. Olfactory coordinated area coverage. *Autonomous Robots*, 20(3):251–260, 2006.
- [40] DeWitt Talmadge Latimer, IV, Siddhartha Srinivasa, Vincent Lee Shue, Samuel Sonne, Howie Choset, and Aaron Hurst. Towards sensor based coverage with robot teams. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation (ICRA '02)*, volume 1, pages 961 – 967. IEEE, May 2002.
- [41] Jean-Claude Latombe. *Robot Motion Planning*. Kluwer, 1991.
- [42] Harry R. Lewis and Christos H. Papadimitriou. *Elements of the Theory of Computation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1997.
- [43] Vladimir J. Lumelsky and Alexander A. Stepanov. Path-Planning Strategies for a Point Mobile Automaton Moving Amidst Unknown Obstacles of Arbitrary Shape. *Algorithmica*, 2:403–430, 1987.
- [44] Kyle Luthy, Edward Grant, and Thomas C. Henderson. Leveraging RSSI for Robotic Repair of Disconnected Wireless Sensor Networks. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA '07)*, pages 3659–3664, Rome, Italy, April 2007.
- [45] Luis E. Navarro-Serment, Robert Grabowski, Christiaan J.J. Paredis, and Pradeep K. Khosla. Millibots, the development of a framework and algorithms for a distributed heterogeneous robot team. *IEEE Robotics and Automation Magazine*, pages 31–40, December 2002.
- [46] Lynne E. Parker and Fang Tang. Building Multirobot Coalitions Through Automated Task Solution Synthesis. In *Proceedings of the IEEE, special issue on Multi-Robot Systems*, volume 94, pages 1289–1305, 2006.
- [47] Nageswara S. V. Rao, Srikumar Karet, Weimin Shi, and S. Sitharama Iyengar. Robot Navigation in Unknown Terrains: Introductory Survey of Non-Heuristic Algorithms. Technical report, Oak Ridge National Laboratory, July 1993.
- [48] Ioannis Rekleitis, Vincent Lee-Shue, Ai Peng New, and Howie Choset. Limited Communication, Multi-Robot Team Based Coverage. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation (ICRA '04)*, pages 3462–3468, New Orleans, LA, April 2004.
- [49] Ioannis Rekleitis, Ai Peng New, and Howie Choset. Distributed coverage of unknown/unstructured environments by mobile sensor networks. In Alan C. Schultz, Lynne E. Parker, and Frank Schneider, editors, *3rd International NRL Workshop on Multi-Robot Systems*, pages 145–155, Washington, D.C., March 14-16 2005. Kluwer.
- [50] A. Sankaranarayanan and Isao Masuda. A New Algorithm For Robot Curve-Following Amidst Unknown Obstacles, And A Generalization Of Maze-Searching. In *Proceedings of the 1992 IEEE International Conference on Robotics and Automation (ICRA '92)*, pages 2487–2494, May 1992.

- [51] A. Sankaranarayanan and M. Vidyasagar. Path Planning For Moving A point Object Amidst Unknown Obstacles In A Plane: The Universal Lower Bound On Worst Case Path Lengths And A Classification Of Algorithms. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation (ICRA '91)*, pages 1734–1741, April 1991.
- [52] Shahar Sarid. Multiple Robots Motion Planning Obtaining a Common Goal. Master's thesis, Department of Mechanical Engineering, Ben-Gurion University of the Negev, Beer Sheva, Israel, 2007.
- [53] Shahar Sarid and Amir Shapiro. Classifying the Multi Robot Path Finding Problem into a Quadratic Competitive Complexity Class. *Annals of Mathematics and Artificial Intelligence*, 52(2-4):169–203, April 2008.
- [54] Shahar Sarid and Amir Shapiro. H-MRSTM: A Competitive Search Algorithm for Heterogeneous Group of Robots. *International Journal of Factory Automation, Robotics and Soft Computing*, (3):51–58, 2009.
- [55] Shahar Sarid and Amir Shapiro. *H-MRSTM: Competitive Search Algorithm for Heterogeneous Group of Robots*. Emerging Technologies, Robotics and Control Systems, Third edition. International Society for Advanced Research, pages 209-216, 2009.
- [56] Shahar Sarid and Amir Shapiro. A Time Competitive Heterogeneous Multi Robot Path Finding Algorithm. In *Proceedings of 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'10)*, pages 4590–4595, Taipei, Taiwan, October 18-22 2010.
- [57] Shahar Sarid, Amir Shapiro, and Yoav Gabriely. MRSAM: a quadratically competitive multi-robot online navigation algorithm. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA '06)*, pages 2699– 2704, Orlando, FL, May 2006.
- [58] Shahar Sarid, Amir Shapiro, and Yoav Gabriely. MRBUG: A Competitive Multi-Robot Path Finding Algorithm. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA '07)*, pages 877–882, Roma, Italy, April 2007.
- [59] Shahar Sarid, Amir Shapiro, Elon Rimon, and Yael Edan. Classifying the Heterogeneous Multi Robot Online Search Problem into Quadratic Time Competitive Complexity Class. In *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA '11)*, Shanghai, China, May 9-13 2011.
- [60] Antonio Sgorbissa and Ronald C. Arkin. Local navigation strategies for a team of robots. *Robotica*, 21(05):461–473, 2003.
- [61] Dong Sun and Can Wang. Controlling Swarms of Mobile Robots for Switching between Formations Using Synchronization Concept. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA '07)*, pages 2300–2306, Rome, Italy, April 2007.
- [62] Fang Tang and Lynne Parker. A Complete Methodology for Generating Multi-Robot Task Allocation using ASyMTRe-D and Market-Based Task Allocation. In *Proceedings of the 2007 IEEE*

International Conference on Robotics and Automation (ICRA '07), pages 3351–3358, Rome, Italy, April 2007.

- [63] Patrick Ulam, Yoichiro Endo, Alan Richard Wagner, and Ronald Arkin. Integrated Mission Specification and Task Allocation for Robot Teams - Design and Implementation. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA '07)*, pages 4428–4435, Rome, Italy, April 2007.
- [64] Antidio Viguria, Ivan Maza, and Anibal Ollero. SET: An Algorithm for Distributed Multirobot Task Allocation with Dynamic Negotiation Based on Task Subsets. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA '07)*, pages 3339–3344, Rome, Italy, April 2007.
- [65] Kjerstin Williams and Joel Burdick. Multi-robot Boundary Coverage with Plan Revision. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA '06)*, pages 1716–1723, May 2006.
- [66] Tao Zhang and Haruki Ueno. Knowledge model-based heterogeneous multi-robot system implemented by a software platform. *Know.-Based Syst.*, 20(3):310–319, 2007.
- [67] Xiaoming Zheng, Sonal Jain, Sven Koenig, and David Kempe. Multi-Robot Forest Coverage. In *Proceedings of 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'05)*, pages 3852–3857, Aug 2005.
- [68] Vittorio Amos Ziparo, Alexander Kleiner, Bernhard Nebel, and Daniele Nardi. RFID-Based Exploration for Large Robot Teams. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA '07)*, pages 4606–4613, Rome, Italy, April 2007.

Appendices

Appendix A. Experimental Results

A.1 HMRSTM experimental Results

The following tables present the experimental results. In each of the tables A.1 to A.9 the results of one of the environments and of one of the targets are presented. The first 5 columns represent the configuration of each experiment. The first column represent the environment, the second is the target number, the third is the number of robots and the fourth is their velocity distribution. For each configuration, two repetitions were conducted (fifth column). The results appear in the sixth column, which presents the number of robot that reached the target. Next, appears the disc number in which the target was found, followed by (col. 8) the number of grid steps took to reach the target. Finally, (cols. 9, 10) appears the total time in seconds that took to reach the target, and average speed of the robot that reached the target in mm/second.

Table A.1: HMRSTM experiments results - "cave" Target 1

env	target	robots	beta	repetition	reached	inDisc	steps	totalSeconds	avgSpeed
cave	1	1	1	1	1	3	115	212.6563	111.4051
cave	1	1	1	2	1	3	115	200.5469	117.0051
cave	1	2	1	1	2	4	158	269.9219	123.5061
cave	1	2	1	2	2	4	158	272.6963	123.368
cave	1	2	L	1	2	2	30	43.70313	158.7987
cave	1	2	L	2	2	2	30	33.125	197.8566
cave	1	3	1	1	1	4	59	114.7031	106.8446
cave	1	3	1	2	1	4	59	103.4063	117.4951
cave	1	3	L	1	1	4	59	107.1875	115.8689
cave	1	3	L	2	1	4	59	110	111.9182
cave	1	3	M	1	1	4	59	125.125	98.998
cave	1	3	M	2	1	4	59	111.0313	111.693
cave	1	4	1	1	1	5	135	263.3125	105.3853
cave	1	4	1	2	1	5	135	275.2188	102.8171
cave	1	4	L	1	4	4	96	197.6875	107.7719
cave	1	4	L	2	4	4	96	177.6406	117.0381
cave	1	4	M	1	4	4	96	197.9531	127.7367
cave	1	4	M	2	4	4	96	182.7344	136.1866

Table A.2: HMRSTM experiments results - "cave" Target 2

env	target	robots	beta	repetition	reached	inDisc	steps	totalSeconds	avgSpeed
cave	2	1	1	1	1	4	222	386.7129	117.0791
cave	2	1	1	2	1	4	222	388.3281	116.4402
cave	2	2	1	1	2	4	101	175.625	119.4192
cave	2	2	1	2	2	4	101	177.4688	122.1511
cave	2	2	L	1	2	4	133	173.875	171.7728
cave	2	2	L	2	2	4	133	173.9219	172.957
cave	2	3	1	1	2	5	93	165.6094	116.8472
cave	2	3	1	2	2	5	93	178.7383	112.3766
cave	2	3	L	1	3	3	44	66.03125	148.2783
cave	2	3	L	2	3	3	44	64.73438	150.5383
cave	2	3	M	1	3	3	44	57.04688	174.418
cave	2	3	M	2	3	3	44	58.90625	175.7878
cave	2	4	1	1	1	5	74	152.9531	100.9067
cave	2	4	1	2	1	5	74	142.3594	107.8046
cave	2	4	L	1	4	4	51	143.4375	80.29281
cave	2	4	L	2	4	4	51	109.5156	98.53389
cave	2	4	M	1	4	4	51	87.85938	132.9056

Table A.3: HMRSTM experiments results - "cave" Target 3

env	target	robots	beta	repetition	reached	inDisc	steps	totalSeconds	avgSpeed
cave	3	1	1	1	1	4	208	339.625	127.7173
cave	3	1	1	2	1	4	208	332.2813	130.7748
cave	3	2	1	1	2	4	87	154.5156	118.2275
cave	3	2	1	2	2	4	87	154.2813	119.1072
cave	3	2	L	1	2	4	119	154	174.4351
cave	3	2	L	2	2	4	119	157.875	170.0903
cave	3	3	1	1	2	5	79	141.0156	118.9797
cave	3	3	1	2	2	5	79	138.9063	120.9809
cave	3	3	L	1	2	5	87	156.6543	122.5948
cave	3	3	L	2	2	5	87	149.2188	129.0253
cave	3	3	M	1	3	6	162	257.2969	159.0342
cave	3	3	M	2	3	6	162	239.8281	163.9007
cave	3	4	1	1	2	6	79	142.1719	119.5454
cave	3	4	1	2	2	6	79	143.125	118.3371
cave	3	4	L	2	4	4	37	54.8125	151.7172
cave	3	4	L	1	4	4	37	70	122.8143
cave	3	4	M	1	4	4	37	55.32813	169.7509
cave	3	4	M	2	4	4	37	59.25	163.308

Table A.4: HMRSTM experiments results - "free" Target 1

env	target	robots	beta	repetition	reached	inDisc	steps	totalSeconds	avgSpeed
free	1	1	1	1	1	3	183	358.2969	103.9864
free	1	1	1	2	1	3	183	335.6406	111.0265
free	1	2	1	1	1	3	91	189.7813	97.42796
free	1	2	1	2	1	3	91	190.7188	97.98198
free	1	2	L	1	2	2	46	79.96875	128.3001
free	1	2	L	2	2	2	46	69.48438	145.4428
free	1	3	1	1	3	3	73	169.0156	90.80817
free	1	3	1	2	3	3	73	152.4531	99.70934
free	1	3	L	1	3	3	57	115.5156	105.0248
free	1	3	L	2	3	3	57	92.96875	129.0541
free	1	3	M	1	3	3	57	94.54688	144.2988
free	1	3	M	2	3	3	57	92.78125	141.8713
free	1	4	1	1	4	4	68	144.8125	99.70134
free	1	4	1	2	4	4	68	147.1875	96.9172
free	1	4	L	1	3	3	53	104.5156	110.7107
free	1	4	L	2	3	3	53	100.3594	115.5149
free	1	4	M	1	2	6	82	180.4063	95.70068
free	1	4	M	2	2	6	82	194.2031	89.3343

Table A.5: HMRSTM experiments results - "free" Target 2

env	target	robots	beta	repetition	reached	inDisc	steps	totalSeconds	avgSpeed
free	2	1	1	1	1	4	378	659.5625	118.1632
free	2	1	1	2	1	4	378	637.4688	122.3668
free	2	2	1	1	2	4	193	368.75	111.9376
free	2	2	1	2	2	4	193	361.6094	114.195
free	2	2	L	1	2	4	237	337.1094	158.2276
free	2	2	L	2	2	4	237	307.1719	171.1485
free	2	3	1	1	2	5	169	343.8086	107.045
free	2	3	1	2	2	5	169	382.2969	97.60216
free	2	3	L	1	3	3	100	155.75	138.2408
free	2	3	L	2	3	3	100	184.3281	120.139
free	2	3	M	1	3	3	100	138.3125	161.7207
free	2	3	M	2	3	3	100	146.4688	151.9915
free	2	4	1	1	1	5	122	266.0156	94.05463
free	2	4	1	2	1	5	122	270.0625	92.40454
free	2	4	L	1	4	4	99	197.4063	111.4909
free	2	4	L	2	4	4	99	188.4531	119.9237
free	2	4	M	1	4	4	99	189.4531	132.8244
free	2	4	M	2	4	4	99	147.0781	157.2158

Table A.6: HMRSTM experiments results - "free" Target 3

env	target	robots	beta	repetition	reached	inDisc	steps	totalSeconds	avgSpeed
free	3	1	1	1	1	4	324	500.375	134.5491
free	3	1	1	2	1	4	324	508.25	131.5691
free	3	2	1	1	2	4	163	270	124.8593
free	3	2	1	2	2	4	163	338.25	99.66593
free	3	2	L	1	2	4	199	260.2969	168.0888
free	3	2	L	2	2	4	199	282.1094	152.9052
free	3	3	1	1	2	5	135	250.0313	114.1937
free	3	3	1	2	2	5	135	238.4375	120.065
free	3	3	L	1	2	5	147	225.7031	138.7841
free	3	3	L	2	2	5	147	240.8906	133.9737
free	3	3	M	1	3	6	254	314.9219	186.5161
free	3	3	M	2	3	6	254	378.375	163.6498
free	3	4	1	1	2	6	135	293.2656	100.4039
free	3	4	1	2	2	6	135	242.2188	118.0957
free	3	4	L	1	4	4	69	133.2344	123.9695
free	3	4	L	2	4	4	69	120.4375	129.2787
free	3	4	M	1	4	4	69	130.8125	142.173
free	3	4	M	2	4	4	69	131.3594	143.9029

Table A.7: HMRSTM experiments results - "libr" Target 1

env	target	robots	beta	repetition	reached	inDisc	steps	totalSeconds	avgSpeed
libr	1	1	1	1	1	3	111	202.8887	111.608
libr	1	1	1	2	1	3	111	204.4756	111.0695
libr	1	2	1	1	2	4	170	285.5	123.8529
libr	1	2	1	2	2	4	170	291.4375	123.5908
libr	1	2	L	1	2	2	18	29	139.931
libr	1	2	L	2	2	2	18	25.26563	158.7137
libr	1	3	1	1	1	4	47	96.65625	100.7902
libr	1	3	1	2	1	4	47	90.54688	105.9893
libr	1	3	L	1	3	3	45	70.75	143.8869
libr	1	3	L	2	3	3	45	64.35938	158.9046
libr	1	3	M	1	3	3	45	63.48438	168.0886
libr	1	3	M	2	3	3	45	65.73438	163.5674
libr	1	4	1	1	1	5	139	280.7119	104.4238
libr	1	4	1	2	1	5	139	256.5469	112.8215
libr	1	4	L	1	4	4	48	84.70313	126.2645
libr	1	4	L	2	4	4	48	82.45313	131.2139
libr	1	4	M	1	4	4	48	76.4375	145.1513
libr	1	4	M	2	4	4	48	70.46875	163.1929

Table A.8: HMRSTM experiments results - "libr" Target 2

env	target	robots	beta	repetition	reached	inDisc	steps	totalSeconds	avgSpeed
libr	2	1	1	1	1	4	230	420.1875	111.7216
libr	2	1	1	2	1	4	230	405.7344	116.2066
libr	2	2	1	1	2	4	121	211.0469	120.1013
libr	2	2	1	2	2	4	121	217.7031	118.5789
libr	2	2	L	1	2	4	137	185.7656	164.0239
libr	2	2	L	2	2	4	137	181.25	170.8248
libr	2	3	1	1	2	5	101	175.2813	120.4521
libr	2	3	1	2	2	5	101	169.6094	125.8952
libr	2	3	L	1	3	3	76	114.4531	150.8915
libr	2	3	L	2	3	3	76	116.7188	145.9663
libr	2	3	M	1	3	3	76	116.6094	154.756
libr	2	3	M	2	3	3	76	113.7344	156.1709
libr	2	4	1	1	1	5	82	160.9844	106.1656
libr	2	4	1	2	1	5	82	153.7813	110.5661
libr	2	4	L	1	4	4	91	152.4844	128.6689
libr	2	4	L	2	4	4	91	154.5781	129.0804
libr	2	4	M	1	4	4	91	143.8438	150.5175
libr	2	4	M	2	4	4	91	143.625	151.3316

Table A.9: HMRSTM experiments results - "libr" Target 3

env	target	robots	beta	repetition	reached	inDisc	steps	totalSeconds	avgSpeed
libr	3	1	1	1	1	4	208	396.2432	111.482
libr	3	1	1	2	1	4	208	405.5586	110.026
libr	3	2	1	1	2	4	107	225.4473	103.6473
libr	3	2	1	2	2	4	107	234.8555	100.4575
libr	3	2	L	1	2	4	115	178.3008	158.5803
libr	3	2	L	2	2	4	115	165.3662	166.1041
libr	3	3	1	1	2	5	79	166.583	104.7886
libr	3	3	1	2	2	5	79	166.8594	104.3094
libr	3	3	L	1	2	5	91	167.4971	126.6888
libr	3	3	L	2	2	5	91	164.5684	127.4729
libr	3	3	M	1	3	6	158	306.7822	134.3298
libr	3	3	M	2	3	6	158	300.1113	135.6097
libr	3	4	1	2	2	6	79	169.9102	101.7714
libr	3	4	1	1	2	6	79	157.1074	110.1794
libr	3	4	L	1	2	6	75	147.3711	116.6375
libr	3	4	L	2	2	6	75	144.5117	118.0043
libr	3	4	M	2	4	4	77	141.9043	141.9971
libr	3	4	M	1	4	4	77	143.4189	144.9808

A.2 HMRSTM simulation of experimental results

Tables A.10, A.11, and A.12 presents the results of the simulations conducted with the experiments' conditions. The first 4 columns present the configuration, which includes, the environment (col. 1), the target number (col. 2), the number of robots and their velocity distributions (cols. 3, and 4 respectively). Then, columns 5, 6, 7 present results, starting with the number of robot that found the target, the disc number in which the target was found, and the number of steps made until finding the target by the robot which found it. The last two columns presents the number of steps made by the slowest robot, and the path length of the robot that found the target, calculated by the number of steps multiplied by the robots' width, D .

Table A.10: HMRSTM simulation of experiment results - "cave"

env	target	robots	beta	reached	inDisc	steps	1steps	steps*D
cave	1	1	1	1	3	115	115	23000
cave	1	2	1	2	4	158	158	31600
cave	1	2	L	2	2	30	10	6000
cave	1	3	1	1	4	59	59	11800
cave	1	3	L	3	3	89	38	17800
cave	1	3	M	3	3	89	23	17800
cave	1	4	1	2	6	134	134	26800
cave	1	4	L	4	4	96	46	19200
cave	1	4	M	4	4	96	20	19200
cave	2	1	1	1	4	222	222	44400
cave	2	2	1	2	4	101	101	20200
cave	2	2	L	2	4	133	44	26600
cave	2	3	1	2	5	93	93	18600
cave	2	3	L	3	3	44	19	8800
cave	2	3	M	3	3	44	11	8800
cave	2	4	1	1	5	74	74	14800
cave	2	4	L	4	4	51	24	10200
cave	2	4	M	4	4	51	11	10200
cave	3	1	1	1	4	208	208	41600
cave	3	2	1	2	4	87	87	17400
cave	3	2	L	2	4	119	40	23800
cave	3	3	1	2	5	79	79	15800
cave	3	3	L	2	5	87	51	17400
cave	3	3	M	3	6	162	41	32400
cave	3	4	1	2	6	79	79	15800
cave	3	4	L	4	4	37	18	7400
cave	3	4	M	4	4	37	8	7400

Table A.11: HMRSTM simulation of experiment results - "free"

env	target	robots	beta	reached	inDisc	steps	1steps	steps*D
free	1	1	1	1	3	183	183	36600
free	1	2	1	1	3	91	91	18200
free	1	2	L	2	2	46	15	9200
free	1	3	1	3	3	73	73	14600
free	1	3	L	3	3	57	24	11400
free	1	3	M	3	3	57	15	11400
free	1	4	1	4	4	68	68	13600
free	1	4	L	3	3	53	30	10600
free	1	4	M	4	4	128	26	25600
free	2	1	1	1	4	378	378	75600
free	2	2	1	2	4	193	193	38600
free	2	2	L	2	4	237	79	47400
free	2	3	1	2	5	169	169	33800
free	2	3	L	3	3	100	42	20000
free	2	3	M	3	3	100	25	20000
free	2	4	1	1	5	122	122	24400
free	2	4	L	4	4	99	47	19800
free	2	4	M	4	4	99	20	19800
free	3	1	1	1	4	324	324	64800
free	3	2	1	2	4	163	163	32600
free	3	2	L	2	4	199	66	39800
free	3	3	1	2	5	135	135	27000
free	3	3	L	2	5	147	87	29400
free	3	3	M	3	6	254	64	50800
free	3	4	1	2	6	135	135	27000
free	3	4	L	4	4	69	33	13800
free	3	4	M	4	4	69	14	13800

Table A.12: HMRSTM simulation of experiment results - "libr"

env	target	robots	beta	reached	inDisc	steps	1steps	steps*D
libr	1	1	1	1	3	111	111	22200
libr	1	2	1	2	4	170	170	34000
libr	1	2	L	2	2	18	6	3600
libr	1	3	1	1	4	47	47	9400
libr	1	3	L	3	3	45	19	9000
libr	1	3	M	3	3	45	12	9000
libr	1	4	1	1	5	139	139	27800
libr	1	4	L	4	4	48	23	9600
libr	1	4	M	4	4	48	10	9600
libr	2	1	1	1	4	230	230	46000
libr	2	2	1	2	4	121	121	24200
libr	2	2	L	2	4	137	46	27400
libr	2	3	1	2	5	101	101	20200
libr	2	3	L	3	3	76	32	15200
libr	2	3	M	3	3	76	19	15200
libr	2	4	1	1	5	82	82	16400
libr	2	4	L	4	4	91	43	18200
libr	2	4	M	4	4	91	19	18200
libr	3	1	1	1	4	208	208	41600
libr	3	2	1	2	4	107	107	21400
libr	3	2	L	2	4	115	38	23000
libr	3	3	1	2	5	79	79	15800
libr	3	3	L	2	5	91	54	18200
libr	3	3	M	3	6	158	40	31600
libr	3	4	1	2	6	79	79	15800
libr	3	4	L	4	4	77	37	15400
libr	3	4	M	4	4	77	16	15400

A.3 HMRSTM simulation-experiment comparison

Tables A.13, A.14, A.15 present the results of the simulations and the experiments. The first four columns are the environment, the target position, the number of robots, and their velocity distribution. The last two columns are the average path of the two repetitions of the experiments, and the expected path calculated from simulations and expected deviations.

Table A.13: HMRSTM simulation-experiment comparison - "cave"

env	target	robots	beta	avg Path	expected Path
cave	1	1	1	23578	32522
cave	1	2	1	33489.5	44682.4
cave	1	2	L	6747	8484
cave	1	3	1	12154.5	16685.2
cave	1	3	L	12340	25169.2
cave	1	3	M	12360	25169.2
cave	1	4	1	27969.5	37895.2
cave	1	4	L	21001.5	27148.8
cave	1	4	M	25024	27148.8
cave	2	1	1	45246.5	62781.6
cave	2	2	1	21325.5	28562.8
cave	2	2	L	29974	37612.4
cave	2	3	1	19718.5	26300.4
cave	2	3	L	9768	12443.2
cave	2	3	M	10152.5	12443.2
cave	2	4	1	15390.5	20927.2
cave	2	4	L	11154	14422.8
cave	2	4	M	11563.5	14422.8
cave	3	1	1	43415	58822.4
cave	3	2	1	18322	24603.6
cave	3	2	L	26858	33653.2
cave	3	3	1	16791.5	22341.2
cave	3	3	L	19229	24603.6
cave	3	3	M	40113.5	45813.6
cave	3	4	1	16966.5	22341.2
cave	3	4	L	8456.5	10463.6
cave	3	4	M	9534	10463.6

Table A.14: HMRSTM simulation-experiment comparison - "free"

env	target	robots	beta	avg Path	expected Path
free	1	1	1	37261.5	51752.4
free	1	2	1	18588.5	25734.8
free	1	2	L	10183	13008.8
free	1	3	1	15274.5	20644.4
free	1	3	L	12065	16119.6
free	1	3	M	13403	16119.6
free	1	4	1	14351.5	19230.4
free	1	4	L	11582	14988.4
free	1	4	M	17307	36198.4
free	2	1	1	77970.5	106898.4
free	2	2	1	41285.5	54580.4
free	2	2	L	52956	67023.6
free	2	3	1	37058	47793.2
free	2	3	L	21838	28280
free	2	3	M	22315	28280
free	2	4	1	24987.5	34501.6
free	2	4	L	22304.5	27997.2
free	2	4	M	24143.5	27997.2
free	3	1	1	67097.5	91627.2
free	3	2	1	33712	46096.4
free	3	2	L	43444.5	56277.2
free	3	3	1	28590	38178
free	3	3	L	31798.5	41571.6
free	3	3	M	60329.5	71831.2
free	3	4	1	29025	38178
free	3	4	L	16043.5	19513.2
free	3	4	M	18750.5	19513.2

Table A.15: HMRSTM simulation-experiment comparison - "libr"

env	target	robots	beta	avg Path	expected Path
libr	1	1	1	22677.5	31390.8
libr	1	2	1	35689.5	48076
libr	1	2	L	4034	5090.4
libr	1	3	1	9669.5	13291.6
libr	1	3	L	10203.5	12726
libr	1	3	M	10711.5	12726
libr	1	4	1	29128.5	39309.2
libr	1	4	L	10757	13574.4
libr	1	4	M	11297.5	13574.4
libr	2	1	1	47046.5	65044
libr	2	2	1	25581	34218.8
libr	2	2	L	30716	38743.6
libr	2	3	1	21233	28562.8
libr	2	3	L	17153.5	21492.8
libr	2	3	M	17904	21492.8
libr	2	4	1	17047	23189.6
libr	2	4	L	19786.5	25734.8
libr	2	4	M	21693	25734.8
libr	3	1	1	44398	58822.4
libr	3	2	1	23480	30259.6
libr	3	2	L	27871.5	32522
libr	3	3	1	17430.5	22341.2
libr	3	3	L	21099	25734.8
libr	3	3	M	40954	44682.4
libr	3	4	1	17301	22341.2
libr	3	4	L	17121	21775.6
libr	3	4	M	20471.5	21775.6

A.4 HMRBUG experimental Results

The following tables present the experiments results. In each of the tables A.16 to A.18 the results of one of the environments are presented. The first 5 columns represent the configuration of each experiment. The first column represent the environment, the second is the target number, the third is the number of robots and the fourth is their velocity distribution. For each configuration, two repetitions were conducted (fifth column). The results appear in the sixth column, which presents the number of robot that reached the target. Next, appears the ellipse number in which the path to the target was found. Finally, (cols. 8, 9) appears the total time in seconds that took to reach the target, and average speed of the robot that reached the target in mm/second.

Table A.16: HMRBUG experiments results - "cave"

env	target	robots	beta	repetition	reached	in Ellipse	total time [sec]	avg. Speed
cave	1	2	1	1	1	2	93.22	126.47
cave	1	2	1	2	2	2	91.33	108.98
cave	1	2	1	3	2	2	80.59	143.17
cave	1	4	1	1	3	4	32.81	158.42
cave	1	4	1	2	3	2	35.64	157.74
cave	1	4	L	1	3	2	28.98	174.37
cave	1	4	L	2	3	2	30.36	200.70
cave	2	2	1	1	1	2	91.09	141.50
cave	2	2	1	2	1	2	83.27	129.20
cave	2	2	1	3	1	2	72.73	148.80
cave	2	4	1	1	3	4	45.92	131.18
cave	2	4	1	2	3	4	33.09	152.11
cave	2	4	L	1	3	2	33.55	177.60
cave	2	4	L	2	3	2	33.66	158.10

Table A.17: HMRBUG experiments results - "free"

env	target	robots	beta	repetition	reached	in Ellipse	total time [sec]	avg. Speed
free	1	2	1	1	1	1	17.31	147.18
free	1	2	1	2	1	1	16.77	151.98
free	1	4	1	1	3	2	17.42	156.24
free	1	4	1	2	4	2	18.88	147.39
free	1	4	L	1	3	2	14.73	186.10
free	1	4	L	2	3	2	10.59	261.76
free	2	2	1	1	1	1	11.95	206.22
free	2	2	1	2	1	1	12.95	189.45
free	2	4	1	1	3	2	14.73	165.60
free	2	4	1	2	3	2	18.84	131.87
free	2	4	L	1	3	2	13.22	188.82
free	2	4	L	2	3	2	8.11	304.96

Table A.18: HMRBUG experiments results - "library"

env	target	robots	beta	repetition	reached	in Ellipse	total time [sec]	avg. Speed
libr	1	2	1	1	2	1	116.84	122.25
libr	1	2	1	2	1	1	51.84	153.67
libr	1	2	1	3	2	2	66.55	121.81
libr	1	4	1	1	4	2	46.80	106.78
libr	1	4	1	2	4	2	54.50	143.06
libr	1	4	1	3	4	2	47.94	124.98
libr	1	4	L	1	3	2	54.42	131.12
libr	1	4	L	2	4	2	51.27	120.72
libr	1	4	L	3	3	2	37.22	153.36
libr	2	2	1	1	1	1	75.06	135.98
libr	2	2	1	2	1	1	63.56	144.30
libr	2	4	1	1	3	2	50.30	124.04
libr	2	4	1	2	3	2	45.14	104.23
libr	2	4	L	1	3	2	34.86	164.95
libr	2	4	L	2	3	2	29.88	177.91

A.5 HMRBUG simulation-experiment results and comparison

Table A.19 presents the results of the simulations and the experiments. The first four columns are the environment, the number of robots, the start-target position, the velocity distribution. The next two columns are simulation path length and the average path length of the two repetitions of the experiments. The following two columns present the relative error, and the absolute value of the relative error. Finally, the last column presents the expected path bound calculated from simulations and expected deviations.

Table A.19: HMRBUG simulation-experiment comparison

Env	robots	S-T point	beta	Simulation Path Length	Experiment Path Length	Relative Error	Absolute Rel. Err.	Expected path bound
cave	2	4	1	11200	11093.667	0.009	0.009	20542
cave	2	1	1	10200	11480.667	-0.126	0.126	18972
cave	4	4	1	7800	5410	0.306	0.306	15204
cave	4	1	1	6600	5529	0.162	0.162	13320
cave	4	4	L	5600	5573.5	0.005	0.005	11750
cave	4	1	L	4600	5639.5	-0.226	0.226	10180
free	2	4	1	3200	2548	0.204	0.204	7982
free	4	4	1	3200	2752	0.14	0.14	7982
free	4	4	L	3200	2757.5	0.138	0.138	7982
free	2	1	1	2600	2459.5	0.054	0.054	7040
free	4	1	1	2600	2462.5	0.053	0.053	7040
free	4	1	L	2600	2484.5	0.044	0.044	7040
library	2	4	1	9600	10119	-0.054	0.054	18030
library	2	1	1	8600	9689.5	-0.127	0.127	16460
library	4	4	1	7200	6261.667	0.13	0.13	14262
library	4	4	L	6800	6344.333	0.067	0.067	13634
library	4	1	1	3400	5472	-0.609	0.609	8296
library	4	1	L	3400	5532.5	-0.627	0.627	8296

Appendix B. DVD content

B.1 DVD 1

B.1.1 Software

1. Arduino c and assembly code
 2. HMRBUG C# simulation application
 3. HMRBUG C# serial simulation application
 4. HMRBUG C# simulation of experiments application
 5. HMRBUG C# experiment application
 6. HMRSTM C# experiment application
 7. HMRSTM C# environment files
 8. HMRSTM C# simulation of experiments application
 9. HMRSTM C# simulation application
 10. VRPN c++ client application
-
1. HMRBUG experiments results
 2. HMRBUG simulation results
 3. HMRSTM experiments results
 4. HMRSTM simulation of experiment results
 5. SPSS statistical analysis

B.1.2 Videos

- HMRBUG experiments videos
- HMRSTM library experiments videos

B.2 DVD 2

- HMRSTM cave experiments videos

B.3 DVD 3

- HMRSTM free experiments videos

תקציר

עבודת מחקר זו מתמקדת בשתי בעיות תכנון תנועה מקוונות (*on-line*) בהן קבוצת רובוטים ניידים בעלת מהירות הטרוגנית צריכים להגיע למטרה הנמצאת בסביבה שאיננה ידועה מראש ואיננה חסומה. בבעיה הראשונה, מיקום המטרה איננו ידוע, והרובוטים צריכים למצוא אותה, ובבעיה השנייה מיקום המטרה ידוע ועל הרובוטים למצוא דרך אל המטרה. התיזה מתמקדת באופטימיזציה של עלות המשימה במובן של זמן התנועה ומציגה שני אלגוריתמים מקוונים לתכנון תנועה:

Heterogeneous Multi-Robot Search Time Multiplication, HMRSTM ו-

Heterogeneous Multi-Robot BUG, HMRBUG.

כאשר מודדים את הביצועים של אלגוריתם מקוון משתמשים בדרך כלל במושג *תחרותיות* (*competitiveness*), אשר הינו היחס המיטבי בין פתרון מקוון ופתרון לא מקוון. בהקשר של בעיות תכנון תנועה, *תחרותיות* מוגדרת כיחס בין אורך המסלול שבוצע בפועל על ידי הרובוט שהגיע אל המטרה לבין אורך המסלול הקצר ביותר אל המטרה. אנו משתמשים ב*תחרותיות* *זמן מוכללת*, כלומר, הפתרון הוא הזמן להגעה אל המטרה, והיחס אינו חייב להיות קבוע, אלא כל יחס פונקציונלי. סיווג בעיות תכנון תנועה מבחינת ביצועים מתקבל על ידי מציאת החסמים העליון והתחתון של התחרותיות של כל האלגוריתמים הפותרים את הבעיה. אם שני החסמים שייכים לאותה מחלקה פונקציונלית, מחלקה זו היא *מחלקת הסיבוכיות התחרותית* של הבעיה. במסגרת התיזה נמצאו שני החסמים עבור שתי בעיות תכנון התנועה הנזכרות לעיל, והבעיות סווגו למחלקות תחרותיות. אנו מראים שבאופן כללי, לכל אלגוריתם מקוון המנסה לפתור בעיות אלו יהיו ביצועים בעלי תחרותיות ריבועית. כך מוגדר הגבול התחתון של הבעיות.

לשני האלגוריתמים *HMRSTM* ו-*HMRBUG* יש גבול עליון ריבועי, מה שמוכיח שלבעיות אותן הם פותרים יש גבול עליון ריבועי. לפיכך, אנו מראים שתכנון תנועה בסביבה שאיננה ידועה ואיננה חסומה שייכת למחלקת סיבוכיות זמן ריבועית. ל-*HMRSTM* ול-*HMRBUG* יש ביצועים תחרותיים ריבועיים ולכן הם בעלי תחרותיות מיטבית.

ביצועי האלגוריתמים במקרה הממוצע נמדדו בסימולציות. הביצועים הממוצעים טובים בהרבה מהחסם העליון של האלגוריתמים. אנו מראים שהביצועים של קבוצה הטרוגנית טובים מהביצועים של קבוצה הומוגנית. יתר על כן, נראה שככל שהקבוצה יותר הטרוגנית ניכר שיפור בביצועים. הסימולציות תוקפו על ידי ניסויים שבוצעו בעזרת קבוצת רובוטים אמיתית. תוצאות הניסויים עומדות בכל הגבולות המצופים. לפיכך, האלגוריתמים *HMRSTM* ו-*HMRBUG* לא רק בעלי חסמים עליונים המבטאים את האופטימליות שלהם, אלא גם בעלי ביצועים טובים מאד ביישום בעולם האמיתי במקרה הכללי.

מילות מפתח:

ריבוי רובוטים, הטרוגניות, תכנון תנועה, אלגוריתמים מקוונים, סיבוכיות תחרותית, מחלקת סיבוכיות תחרותית, מערכות מבוזרות, ניווט, כיסוי שטח.

העבודה נעשתה בהדרכת

ד"ר אמיר שפירא

ופרופ' יעל אידן

במחלקה להנדסת מכונות

בפקולטה למדעי ההנדסה

אלגוריתמי חיפוש לקבוצת רובוטים הטרוגנית


מחקר לשם מילוי חלקי של הדרישות לקבלת תואר "דוקטור לפילוסופיה"

מאת

שחר שריד

הוגש לסינאט אוניברסיטת בן-גוריון בנגב

אישור המנחים

ד"ר אמיר שפירא 

פרופ' יעל אידן 

אישור דיקן בית הספר ללימודי מחקר מתקדמים ע"ש קרייטמן

2011

תשע"א

באר שבע

אלגוריתמי חיפוש לקבוצת רובוטים הטרוגנית

מחקר לשם מילוי חלקי של הדרישות לקבלת תואר "דוקטור לפילוסופיה"

מאת

שריד

שחר

הוגש לסינאט אוניברסיטת בן-גוריון בנגב

2011

תשע"א

באר שבע