6-17-2011

Multi-Robot Coalition Formation for Distributed Area Coverage

Ke Cheng
University of Nebraska at Omaha

Multi-Robot Coalition Formation for Distributed Area Coverage

By

Ke Cheng

A DISSERTATION

Presented to the Faculty of

The Graduate College at the University of Nebraska at Omaha

In Partial Fulfillment of Requirements

For the Degree of Doctor of Philosophy

Major: Information Technology

Under the Supervision of Dr. Prithviraj (Raj) Dasgupta

Omaha, Nebraska

June 17, 2011

Multi-Robot Coalition Formation for Distributed Area Coverage

Ke Cheng, Ph.D. in IT

University of Nebraska at Omaha

Advisors: Dr. Prithviraj(Raj) Dasgupta

The problem of distributed area coverage using multiple mobile robots is an important problem in distributed multi-robot sytems. Multi-robot coverage is encountered in many real world applications, including unmanned search & rescue, aerial reconnaissance, robotic demining, inspection of engineering structures, and automatic lawn mowing. To achieve optimal coverage, robots should move in an efficient manner and reduce repeated coverage of the same region that optimizes a certain performance metric such as the amount of time or energy expended by the robots. This dissertation especially focuses on using mini-robots with limited capabilities, such as low speed of the CPU and limited storage of the memory, to fulfill the efficient area coverage task. Previous research on distributed area coverage use offline or online path planning algorithms to address this problem. Some of the existing approaches use behavior-based algorithms where each robot implements simple rules and the interaction between robots manifests in the global objective of overall coverage of the environment. Our work extends this line of research using an emergent, swarming based technique where robots use partial coverage histories from themselves as well as other robots in their vicinity to make local decisions that attempt to ensure overall efficient area coverage. We have then extended this technique in two directions. First, we have integreated the individual-robot, swarming-based technique for area coverage to teams of robots that move in formation to perform area coverage more efficiently than robots that move individually. Then we have used a team formation technique from coalition game theory, called Weighted Voting Game (WVG) to handle situations where a team moving in formation while performing area coverage has to dynamically reconfigure into sub-teams or merge with other teams, to continue the area coverage efficiently. We have validated our techniques by testing them on accurate models of e-puck robots in the Webots robot simulation platform, as well as on physical e-puck robots.

Keywords: Multi-robot Area Coverage, Weighted Voting Games, Multi-robot Coalition Formation

Dedication

To my darling wife, **YAQI(LISA) ZHOU** and my elfish son,

YIXUAN(DEVON) CHENG

Acknowledgments

I would like to express my deepest gratitude to my advisors, Dr. Prithviraj(Raj) Dasgupta, who has continuous supported me in the Ph.D. program in IT at College of Information Science and Technology since 2006. It would have been next to impossible to write this dissertation without his professional academic guidance and inspiring encouragement. I also thank the other committee members, Dr. Zhenyuan Wang, Dr. Qiuming Zhu, and Dr. Zhengxin Chen, who gave me valuable support and suggestions.

I also would like to thank all the faulty members and colleague students of College of Information Science and Technology.

I am grateful to the C-MANTIC Research Lab directed by Dr. Dasgupta for providing me with financial support for fulfilling this dissertation. In addition, during the course of this work, I was involved in the COMRADES project: COoperative Multi-Robot Autonomous DEtection System for Humanitarian Demining, Sponsor: Office of Naval Research (ONR) and the COMSTAR project: COoperative Multi-agent System for automatic TArget Recognition (Phase II), STTR Grant awarded jointly with 21st Century Systems Inc. Sponsor: DoD-NavAir.

Special thanks also go to my good friends: Dr. Nian Yan, Dr. Kun Hua, Dr. Honggang Wang, Dr. Wei Wang, and others in Omaha for their continuous encouragement and friendship.

Finally, I would like to give my greatest thanks and love to my most important family members: my wife, Yaqi Zhou; my son, Yixuan(Devon) Cheng; my father, Binghuang Cheng; my mother, Yafang Liu; and my parents in law, Shouxin Zhou and Yumei Li. Without their unconditional support and love, this dissertation would never have been accomplished.

Contents

1 INTRODUCTION 1

1.1 Area coverage 2

1.2 Modeling Multi-Robot Systems 3

1.3 Coalitional Game Theory for Multi-Robot Decision-Making 5

1.4 Research Objectives 6

1.5 Organization of the Dissertation 7

2 BACKGROUND 10

2.1 Multi-robot coverage 10

2.1.1 Multi-robot coverage with individual robot coordination 10

2.1.2 Team based multi-robot coverage 14

2.2 Coalitional Games 15

2.3 Solutions of Coalitional Games 16

2.4 Voting Games 18

2.5 Computational Complexity of Coalitional Games 20

3 E-PUCK ROBOT: DESCRIPTION AND LOCALIZATION 21

3.1 E-puck Robot 22

3.2 Overhead Camera-based Localization 24

3.3 Realistic Simulation 26

4 MULTI-ROBOT COVERAGE: INDIVIDUALLY COORDINATED ROBOTS USING SWARMING 28

4.1 Multi-Robot Distributed Coverage Problem 28

4.1.1 Distributed Coverage with Constrained Mini-robots 29

4.2 Area Coverage Algorithms 33

4.2.1 Node Counting Strategies 38

4.3 Agents with Communication Faults 41

4.4 Experimental Results in Webots Simulator 43

4.5 Physical Robot Experimental Results 55

5 MULTI-ROBOT COVERAGE: TEAM-BASED ROBOTS USING FLOCKING 60

5.1 Single Team Flocking 61

5.2 Single-Team Coverage Technique 62

5.2.1 Formation Maintenance 63

5.2.2 Team Reconfiguration 64

5.3 Experiment in Webots Simulator 66

6 MULTI-ROBOT COVERAGE: COALITION GAME-BASED ROBOT TEAM FORMATION 72

6.1 System Modeling 72

6.1.1 Best Coalition Formation 73

6.1.2 The other coalition formation algorithm 77

6.2 Dynamic Team Reconfiguration and Weighted Voting Games 83

6.3 Experimental Results in Simulator 91

6.3.1 Team Reformation Experiments 93

6.4 Physical Experimental Results 103

7 SUMMARY OF CONTRIBUTIONS AND FUTURE DIRECTIONS 109

7.1 Summary of Contributions 109

7.2 Future Directions 111

A Refereed Publications by Ke Cheng **124**

List of Figures

3.1 The area coverage platform: Epuck robot [30]. 22

3.2 (a) an e-puck with a red marker, (b) 5 e-pucks moving in the 'V' shape team. . . . 23

3.3 Screen shot of five e-pucks within our experiment arena in the image processing software user console. 23

3.4 Screen shots of roborealm console: (a)the raw image with red triangle, (b) the image after RGB filter processing, (c) the final analyzed image with a coordinate location. 24

3.5 Screen shot of a geometric statistics window. 25

3.6 A epuck in Webots simulator. 26

4.1 Different configurations of robots in the neighborhood of a robot r. The outer circle represents robot r's communication range ξ and the inner circle represents robot r's coverage map radius χ. (a) There are no other robots within robot r's communication and coverage maps; (b)There are some robots in robot r's communication map, but no robot in its coverage map; (c) There are some robots within robot r's communication and coverage maps. 33

4.2 Algorithm used by a robot r to select its next action if there are some robots within its communication map but no other robots in its coverage map. . . . 34

4.3 Algorithm used by a robot to select its next action when there are robots both in its communication map and coverage map. 37

4.4 Selection condition for agent action for different pheromone update strategies. . . 42

4.5 Parameters used in the experimental settings for the simulations. 44

4.6 Percentage of area covered with 5 agents in a 100×100 square environment for different coverage strategies. 45

4.7 Redundancy in area covered as the number of agents in the environment changes. 46

4.8 Redundancy in area covered as the coverage map radius MR of agents changes. . . 47

4.9 Percentage of the coverage with the communication range radius CR of agents changes. 47

4.10 Redundancy in area covered as the communication range radius CR of agents changes. 48

4.11 Percentage of the coverage with and without the GPS sensor noise. 48

4.12 Redundancy in area with and without the GPS sensor noise. 49

4.13 Percentage of the coverage with different size of mazes. 50

4.14 5, 10, 25 percentage of obstacles. 50

4.15 Percentage of the coverage with obstacles. 50

4.16 Redundancy in area with obstacles. 51

4.17 Percentage of the coverage with different weight values 52

4.18 Redundancy in area with different weight values. 52

4.19 Percentage of the coverage if agents malfunction with different recover probabilities. 53

4.20 Percentage of the coverage if agents malfunction with different failure probabilities. 54

5.1 Algorithm used by a robot to realize the leader-referenced formation control. 62

5.2 (a) The leader robot (id=0) in a team of five robots encounters an obstacle. (b) A new leader is selected (id=3) and the old leader robot (id=0) calculates the positions for every robot in the new configuration under the new leader. (c) Each robot assumes its new position and the new leader robot selects its heading from randomly between $-\alpha \pm \beta$. 64

5.3 Four different environments of five robots with V shape in Webots simulation platform: (a) a square, (b) a triangle, (c) a corridor, (d) two diamonds connected by a corridor. 66

5.4 (a) Number of times an obstacle is encountered by the team, (b) Average time required for team reconfiguration after encountering an obstacle, (c) Number of times the team has to perform formation maintenance while not encountering an obstacle, and, (d) Average time required for formation maintenance. 68

6.1 (a) 6 robots calculate two minimal winning coalitions in the communication range , (b) 4 robots form a best minimal winning coalition and head to a new team direction. 73

6.2 Run time in millisecond (in the logarithm scale) with different number of robots and $Q_f = 0.7$ by using Vig's Multi-Robot Coalition Formation Algorithm and Team Formation using DYN-REFORM Algorithm. 79

6.3 Number of coalitions (in the logarithm scale) with different number of robots and different values of Q_f by using Vig's Multi-Robot Coalition Formation Algorithm and Team Formation using DYN-REFORM Algorithm. 79

6.4 Percentage of environment covered by different numbers of robot for using Dr.Vig's Multi-Robot Coalition Formation Algorithm and Team Formation using DYN-REFORM Algorithm. within a 2×2 m^2 environment where 10% of the total area of the environment space is occupied by obstacles. The run time is 30 minutes. . . 82

6.5 (a) A robot team showing the position identifiers of each robot. The angular separation in the team is u, the separation between adjacent robots is d_{sep} and α is the heading of the team. (b)-(c) A scenario where a single team in formation encounters a T-shaped obstacle and needs to split. (d)-(e) A scenario where two teams in close proximity of each other encounter each other and need to merge. 83

6.6 (a) Algorithm to find veto players in a WVG. (b) Algorithm to find the best minimum winning coalition (BMWC) from a set of MWCs. 86

6.7 State transition diagram for a robot participating in the dynamic reconfiguration algorithm. 89

6.8 (a) Initial configuration of a team of 5 robots moving in a wedge-shaped formation. (b) Reformed team moving in new direction after encountering a flat wall obstacle. (c) Mean error in the desired position of the robot team during the reconfiguration. 93

6.9 (a) Initial configuration of a team of 7 robots moving in a wedge-shaped formation. (b) Reformed team moving in new direction after encountering a non-uniform wall obstacle. (c) Mean error in the desired position of a robot team during the reconfiguration. 95

6.10 (a) Initial configuration of a team of 7 robots moving in wedge-shaped formation. (b) Reformed team moving in new direction after encountering a narrow obstacle that causes the team to split. (c) Mean error in the desired position of a robot team during the reconfiguration. 96

6.11 (a) Initial configuration of two teams of 3 and 4 robots moving in wedge-shaped formations. (b) Reformed team moving in new direction after encountering each other and merging into a new team. (c) Mean error in the desired position of a robot team during the reconfiguration. 97

6.12 (a) Time spent by the robots in the STOP_AND_WAIT state and CONTINUE_PREVIOUS_MOTION state of the DYN-REFORM algorithm for the three different types of obstacles. (b) Percentage of the environment covered by a set of 5 robots initially configured as a team without and with DYN-REFORM algorithm. The environment is 2×2 m^2 with $0\%, 10\%$ or 20% of the area of the environment occupied by obstacles. Each experiment was run for a period of 30 mins. Error bars were omitted for legibility. 99

6.13 Percentage of environment covered by different numbers of robot for different values of the quota for the WVG within a 2×2 m^2 environment where 10% of the total area of the env. space is occupied by obstacles. . . . 100

6.14 5 robots in a team visited in a $4 \times 4\ m^2$ square environment. The region is colored by light gray, gray, dark gray, and black representing revisited 2 to 5 times, 6 to 10 times, 11 to 15, and over 16 respectively. (a) Run time is 30 minutes (b) Run time is 60 minutes.(c) Run time is 90 minutes. (d) Run time is 120 minutes. 102

6.15 (a) Initial configuration of a team of 5 robots moving in a wedge-shaped formation. (b) Reformed team moving in new direction after encountering a flat wall obstacle. (c) Average error in the position of the robot team during reconfiguration in the scenario of encountering a flat wall obstacle. . 103

6.16 (a) Initial configuration of a team of 5 robots moving in a wedge-shaped formation. (b) Reformed team moving in new direction after encountering a non-uniform wall obstacle. (c) Average error in the position of the robot team during reconfiguration in the scenario of encountering a a non-uniform wall obstacle. 104

6.17 (a) Initial configuration of a team of 5 robots moving in wedge-shaped formation. (b) Reformed team moving in new direction after encountering a narrow obstacle that causes the team to split. (c) Mean error in the desired position of a robot team during the reconfiguration. 106

6.18 (a) Initial configuration of two teams of 3 robots each moving in wedge-shaped formations. (b) Reformed team moving in new direction after encountering each other and merging into a new team. (c) Mean error in the desired position of a robot team during the reconfiguration. 107

List of Tables

2.1 An example of three player game. 17

3.1 Camera Parameters 23

4.1 Physical experimental results showing percentage of coverage of the environment by different number of robots with and without obstacles in a indoor lab environment. 55

4.2 The Kruskal-Wallis statistics tests of the percentage of environment covered with 5 robots in a 1000×1000 cm^2 square environment for different node counting strategies over 10,000 simulation time measured in ticks. 56

4.3 The Kruskal-Wallis statistics tests of the percentage of environment covered with different number of robots in a 1000×1000 cm^2 square environment for $2,500$ time steps. 57

4.4 The Kruskal-Wallis statistics tests of the number of time steps required to complete the coverage of a 1000×1000 cm^2 square environment for different numbers or robots. 58

4.5 The Kruskal-Wallis statistics tests of the percentage of environment covered with different coverage map radii of 5 robots after 10,000 time steps in 1000×1000 cm^2 square environment. 58

4.6 The Kruskal-Wallis statistics tests of the percentage of environment covered after 2500 time steps when the communication range ξ of robots changes, χ=50 cm, number of robots=5, and environment size=1000×1000 cm^2. 59

5.1 Percentage of of a $5 \times 5m^2$ environment covered for different environment shapes and different coverage strategies. 67

6.1 Average and standard deviation number of coalitions with different number of robots and different values of Q_f by using Vig's Multi-Robot Coalition Formation Algorithm and Team Formation using DYN-REFORM Algorithm. 81

Chapter 1

INTRODUCTION

Multi-robot systems can be a competitive solution compared to single robot because they create much more robustness due to their redundancy and in cooperation the individual simplicity which is easy to achieve. Each autonomous robot can work in parallel for fast task execution, e.g., in a coverage or search and rescue task. Especially, for the mini-robot, such as the insect like mobile robot, the size constraints on the robotic platform make it different for a single robot to accomplish the task within an acceptable time. However, besides the mechanical and electronic design, such as small enough motors and wheels and the highly integrated circuit module, there are plenty of challenges for high level algorithm design including, limited peer-to-peer wireless communication, limited micro chip computation, and limited energy budget. Moreover, to address the social behaviors, such as cooperation, coordination, and negotiation in these multi-mini robot systems, is even harder. Also, as these are online systems, the systems need to resist innately with the unexpected noise on crude sensors and actuators. Thus, algorithmic design should not only consider the theoretical optimization of the performance of the whole swarm, but also take into account the practical implementation feasibility.

1.1 Area coverage

Coverage robotics, or area coverage means moving a mobile platform in order to cover a surface by a robot's bottom or sensor range [42]. Terrain Coverage has a variety of industrial, academic, and military applications such as unmanned search & rescue, aerial reconnaissance, robotic demining, automatic lawn mowing, and inspection of engineering structure. In these, a robot is given abounded prior known or unknown working area, most possibly containing obstacles. The robot is assumed that each time it can visit a point within the working area. Distributed terrain coverage is focus on using multi-robot systems to fulfill efficient coverage task. Algorithms of distributed terrain coverage can be implemented on diversified mobile robotic platform ranging from unmanned ground vehicles to unmanned underwater and aerial vehicles.

In recent years, there is increasing interest for multiple robots terrain coverage. Comparing with single robot, multiple robots have potentially better performance on four properties: completeness, efficiency, redundancy, and robustness. First completeness, the algorithm for multiple robots can guarantee to completely visit every point in the working area. Second efficiency, by increasing the number of robots, the time they take to cover the area can be reduced efficiently. Third redundancy, any points of the working area is visited more than once. Theoretically, the optimal algorithm should be non-redundant, in that any portion of the working area is covered only once. Last not the least robustness, the multi-robot algorithm can handle robot failures, which means the coverage task can be fulfilled by part of the robots.

In multi-robot coverage, non-redundancy and efficiency are independent optimization criteria: Non-backtracking algorithms may be inefficient, and efficient algorithms may use backtracking [32]. In other words, the optimal single robot coverage algorithm cannot simply be used or produced the same result in multiple robot cases. Moreover, the initial position of robots in the working area and the different shapes of the same size of working area affect the completion time of the coverage.

1.2 Modeling Multi-Robot Systems

In this thesis, I propose a two level hierarchal architecture to solve the area coverage problem. In the upper most layer, my approach focuses on how to building a robust and reliable multi-robot coordination system to accomplish these applications. Conventional approaches try to use off line or online path planning to solve this problem. Some of the existing approaches use behavior-based algorithms, which are inspired by social insects in nature such as bees and ants who can follow trails laid by homogeneous creatures. My module is based on the weighted voting game theory. Each robot has its recent coverage history, and can calculate its coverage capability by the its coverage history. By using the coverage capability as the weight, robots can form a optimal coalition when they meet. In the lower most layer, each robot can do individual and foundational decision making processes such as obstacle avoidance or heading north.

I derive the minimal winning coalition of weighted voting game theory with the robot domain knowledge to fulfill these advance decision making processes: leaving or joining a team based on maximizing the worth of the whole system. Also, to achieve the efficient coverage by a single team, I extend the Reyolds' flocking model to dynamically change the shape of the team. This is the first module combining the robot team formation with coalitional game theory for the mobile mini-robot system.

1.3 Coalitional Game Theory for Multi-Robot Decision-Making

Collaboration and cooperative decision-making are important issues in multi-robot system design. Especially, robots dynamically form a team and fulfill a certain task, such as the area coverage. In practical situations, robots take a joint decision leading to a certain outcome, which may have a different impact on each of them. Also, as different robot has unique capabilities, the achievement of the team is based on the contribution of each robot. Weighted voting games, which provide a model of decision-making in many political voting procedures, can be applied in the multi-robot cases. In such a game, each player has a weight, and a coalition of players wins the game if the sum of the weights of its participants exceeds a certain threshold. As weights can affect the outcome, one of the key issues to use this model in multi-robot area coverage is how to compute such weight of each robot, instead of pre-given which is unreal in multi-robot systems. Meanwhile, finding a stable and unique coalition which is the well-known problem of coalitional games, is another practical challenge.

In this dissertation, I illustrate the method of calculating the coverage capability based on revisited history of the robot to present the weight in multi-robot system design. Also, I extend the weighted voting game module with robot domain knowledge to get the unique coalition in the core, which I call best minimal winning coalition (BMWC). I theoretically prove the feasibility of this model, and give algorithms to find veto players and the minimal winning coalitions. Also I designed a greedy method and a heuristic method to find BMWC in $O(n\ log\ n)$ time and $O(n^2)$ time respectively.

1.4 Research Objectives

The previous researchers mainly focus on the path planing to solve terrain coverage problem or extend their algorithms from the single robot to multiple robots. The objective of this dissertation is to research the terrain coverage problem with multiple mobile robots based on the framework of weighted voting games. More specifically, the objectives of the research dissertation are:

- To design a novel method of computing the weights with robot coverage capabilities.
- To get the unique coalition in the core by extending the weighted voting game module with robot domain knowledge.
- To build a feasible and practical multi-robot system for distributed terrain coverage.
- To theoretically prove the module and analysis the related algorithms.
- To evaluate the system in the software simulator, as well as the physical robots.

1.5 Organization of the Dissertation

The organization of the dissertation can be briefly described as follows:

The research methodology and related work review are described in Chapter 2. First, the literal reviews of area coverage approaches, as well as flocking robot team control methods are proposed. Next, I introduce the mathematical concepts in coalitional game theory, including the core, Shapley value and the weighted voting games. Finally, I present the computational complexity of coalitional games.

In Chapter 3, I propose a indoor scenario with e-puck robots. Also, I introduce a localization method by using over head camera to mimic the GPS. I use a computer vision software called robotrealm as a tool to get the location of each robot by image reorganization processing.

In Chapter 4, Initially, I concentrate on the problem of distributed coverage of an unknown environment using a swarm of mobile robots in the presence of robot faults and failures. Each robot is limited in its communication range and memory capacity. Moreover, the communication between robots is susceptible to sensor noise, and each robot can fail transiently or permanently. The global objective in the system is to have every cell in the grid-environment visited by at least one robot while doing two things: reducing the coverage time, and reducing the redundancy in the area covered by the robots. Our analytical and experimental results show that the local robot algorithms translate to efficient, scalable and fault-tolerant coverage of the environment under different operational constraints, even in the presence of robot faults and failures.

In Chapter 5, I investigate whether the coverage achieved by robots in the previously described technique can be improved if robots can be coordinated to move together in small teams while covering the environment. A popular nature-inspired, emergent technique for controlling the movement of multiple robots organized as teams is provided by the flocking or herding behavior of birds and animals. I described techniques that enable limited capability mini-robots to cover an unknown environment in a distributed manner by forming multiple small-sized teams and to dynamically adapt their formation within a team on encountering an obstacle. I test our techniques on the Webots simulation platform using accurate models of e-puck robots as well as on physical e-puck robots. Our experimental results showed that our team-based coverage techniques for distributed area coverage can perform comparably while lowering storage and communication overhead with respect to coverage strategies where mini-robots are coordinated individually.

In Chapter 6, One of the problems that I encounter while implementing flocking on robot teams, was that conventional flocking techniques did not provide any mechanism for robots to dynamically change teams depending on the operational conditions. To solve this problem I start investigating mechanisms from coalitional game theory that provide techniques for a set of individuals to form stable teams. In this research, I use a technique from coalitional game theory called a *weighted voting game (WVG)* that allows each robot to dynamically identify other team members and form teams so that the efficiency of the area coverage operation is improved. I propose and evaluate an initial technique and two refinement of computing the weights of a weighted voting game based on each robot's coverage capability and finding the best minimal winning coalition(BMWC). I compare the performance of the weighted voting game based team reformation strategy with the baseline strategy (fixed team size) where teams are not dynamically reformed within the Webots robot simulation platform using up to 40 robots. The results show that the weighted voting game-based strategy yields better coverage than the baseline strategy with different types of obstacles occupying the environment.

Finally, Chapter 7 provides a discussion on main contributions of this dissertation and summarizes the future research directions.

Chapter 2

BACKGROUND

2.1 Multi-robot coverage

2.1.1 Multi-robot coverage with individual robot coordination

Area coverage using distributed multi-robot systems has been an active area of research for over a decade and an excellent overview of this area is given in [16]. Several techniques for multi-robot coverage such as using Boustrophedon decomposition [53], using occupancy grid maps [9], using probabilistic Bayesian models of the coverage map, information gain-based heuristics and graph segmentation techniques[62, 72], negotiation and learning-based approaches[35, 1] have also been proposed. Here, we focus on the recent work on distributed area coverage using limited capability mini-robots. Previous research in the area of distributed area coverage by mini-robots can be divided into three major categories. In the first category, heuristics inspired by ant algorithms are used to guide the motion for ant-like robots performing terrain coverage. In [70], Wagner *et al.* consider a grid-based environment and describe virtual ant-like robots that deposit virtual pheromone in the cells of the grid that they visit. Robots then decide on their next movement using a hill-climbing approach (ANT-WALK-1) or a multi-level depth first search technique with backtracking (ANT-WALK-2). Koenig *et al.*[37] also use a grid-based environment and describe ant-inspired mechanisms such as node counting and LRTA* to enable robots select the next action. However, these approaches assume either that robots can leave behind physical trails within the environment (e.g., with a marker pen attached to each robot for tracing its path[63]) which can be sensed by other robots, or, that all robots deposit virtual pheromone on a centralized pheromone map that is shared by all robots[37]. These mechanisms also do not consider limitations on the communication ranges of robots because sharing the pheromone by depositing markers in the environment or via a central location evidently obviates the need for direct robot-to-robot network communication. In [2, 68], Altshuler *et al.* and Wagner *et al.*, describe ant algorithms inspired by results from computational geometry for robotic area coverage in a floor cleaning application, where the contamination can spread dynamically. Each robot follows an outward spiral pattern and ensures that its cleaning action does not disconnect the dirty subgraph in the environment. However, these techniques have not been to reported to be implemented on physical robots and they appear to have considerable memory requirements on each robot as each robot has to maintain a local copy of the dirty subgraph of the entire environment so that it can calculate the boundary of the subgraph while selecting a cell to clean.

In the second category, the environment is considered as a graph and the coverage problem is treated as constructing the least-cost spanning tree that traverses the graph using either a single robot[27] or multiple robots [32, 74]. In each of these algorithms, the robots need to store and work with a representation of the entire environment within their memory so that they can determine if a graph vertex they just arrived at has been visited previously. Recently, Rutishauser, Correll and Martinoli[56] have addressed this problem using a team of networked mini-robots to cover the surface of a jet turbine where the blades are marked with color markers to aid in inspection and robot localization. However, this algorithm also requires a space complexity equal to the number of vertices plus the number of edges of the graph into which the environment is decomposed.

The third category consists of dispersion-based mechanisms between robots to achieve area coverage. Howard *et al.* [33] present a potential-field-based approach for distributed deployment of mobile sensor nodes in an indoor hospital environment. Batalin and Sukhatme's[5] algorithms are based on local, mutually dispersive interaction between robots when they were within sensing range of each other. Morlok and Gini[45] describe different simplistic coverage strategies for mobile mini-robots that use line of sight communication to avoid each other and obstacles. Experimental results with simulated robots in different environments show that robots perform best by combining a random walk strategy (called *fiducial*) while avoiding each other. In an extension of this work by Ludwig and Gini[40] robots use wireless intensity signals to infer the location of each other and move in a direction to disperse away from each other and improve area coverage. O'Hara and Balch [46] use sensor-less robots without localization and mapping but deploy specialized anchor nodes called attractors at specific locations in the environment to guide the path of robots. In a similar approach, Gasparri *et al.* [28] deploy sensor nodes that efficiently guide the path of robots using a Hamiltonian path-based algorithm. Parker [48, 49, 50] has described several distributed techniques for multi-robot task allocation for applications such as box pushing, mobile target tracking, etc. where robots exert vector force fields to repel or attract each other. Spears *et al.* [59] have applied techniques inspired by the motion of particles in physics to robotic terrain coverage. Some researchers [29] describe theoretical analyzes of the multi-robot terrain coverage problem but memory constraints or communication ranges of robots are not considered as limitations in these techniques and no information about these parameters is provided in these papers.

In contrast to these approaches, our paper targets the middle ground lying between pure dispersion-based approaches, and, memory and computationally involved approaches that work with or need to store a complete map or representation of the environment. Our techniques use the limited on-board memory of the robots to store partial coverage maps of the environment. The robots then intermittently communicate and fuse their coverage maps with each other and use emergent techniques based on the information in the fused maps to guide their motion and achieve complete coverage of the environment.

2.1.2 Team based multi-robot coverage

Most research on formation of robot-teams using distributed techniques has been inspired by Reynolds' seminal work on the mobility of flocks[52]. Reynolds prescribes three fundamental operations for each team member to realize distributed flocking - *separation, alignment* and *cohesion*. Following Reynolds' model, in [10, 69] they describe mechanisms for robot-team motion while maintaining specific formations where individual robots determine their motion strategies from the movement of a team leader or neighbor(s). Balch describes three reactive behavior-based strategies for robot teams to move in formation, viz., unit-center- referenced, neighbor-referenced, or leader-referenced [7]. In contrast to these approaches, Fredslund describes techniques for robot team formation without using global knowledge such as robot locations, or the positions/headings of other robots, while using little communication between robots [24]. Hanada provides a model for dynamic team formation in multi-robot swarms inspired by motion models of schools of tuna fish [31]. However, robots in their system rely extensively on accurate visual sensing because each robot is required to know the the dimensions of an obstacle(e.g., the width of a passage) to be able to separate into sub-teams. Turgut proposes a behavior-based robot swarm that creates self-organized flocking by using heading alignment and proximal control with on-board digital compass [65]. In most of these flocking-based approaches, achieving performance efficiency is not a principal objective for robot motion, and consequently, none of these flocking-based approaches enable robots in a team to dynamically select and change teams to improve the system performance.

Therefore, we need to find a suitable solution to fulfill this decision making process of robots. In this research, we have used coalitional game theory to mimic human group decision-making for robot teams. The basic idea of coalitional games is to study how a single agent or person can improve its benefit while the utility to the whole team can be increased as well.

2.2 Coalitional Games

To make self-interested agents combine to form effective teams, there is a theoretical model called coalitional game theory or cooperative game theory. A coalition game assigns to each group of agents, called a coalition, a set of possible payoffs [34, 58]. Instead of considering choices of individual agents within a coalition, researchers study how they coordinate and cooperate with each other to pursue a coalition or coalitional winning condition.

Definition 2.1. A coalition game (CG) with transferable utility is given by (N, v), where $N = \{1, 2, \ldots, n\}$ is a set of agents or players; and $v : 2^N \mapsto \mathbb{R}$ is a utility function that maps each group of agents $S \subseteq N$ to a real-valued payoff. Intuitively, $v(S)$ is the profit that the members of coalition S can attain by working together [58].

The assumption of transferable utility is that the payoffs to a coalition may be freely redistributed among its members. In other words, players are allowed to communicate before or during the game. Each coalition can be assigned a single value as its payoff. The grand coalition is the name given to the coalition of all the agents in N. There are several important classes of coalitional games which include formal properties.

2.3 Solutions of Coalitional Games

The solution of coalitional game is how to find possible division of the payoff which the benefit or money a agent can achieve when it joins the team, to the grand coalition among the players. We will give some basic definitions about payoff division.

Definition 2.2. Feasible payoff: Given a coalitional game $G = (N, v)$, the feasible payoff set is defined as $\{x \in \mathbb{R}^N | \sum_{i \in N} x_i \leq v(N)\}$ [58].

Definition 2.3. Efficient: We say that an payoff x is *strongly efficient* or *efficient* in a feasible payoff set F iff $x \in F$ and there is no other vector $y \in F$ such that $y_i \geq x_i$ for every $i \in N$ and $y_j > x_j$ for at least one $j \in N$ [43].

Definition 2.4. Pre-imputation: Given a coalitional game $G = (N, v)$, the pre-imputation set, denoted ρ, is defined as $\{x \in \mathbb{R}^N | \sum_{i \in N} x_i = v(N)\}$ [58].

Pre-imputation is the *efficient* feasible payoff set.

Definition 2.5. Imputation: Given a coalitional game $G = (N, v)$, the imputation set, denoted ρ, is defined as $\{x \in \rho | \forall i \in N, x_i \geq v(i)\}$ [58].

As the payoff of each player in an imputation set is no less than the amount it can achieve by forming a singleton coalition, each player is *individually rational*. Imputation sets are payoff vectors that are not only *efficient* but also *individually rational*.

Table 2.1: An example of three player game.

S	$V(S)$
$\{1\}$	1
$\{2\}$	2
$\{3\}$	2
$\{1,2\}$	4
$\{1,3\}$	3
$\{2,3\}$	4
$\{1,2,3\}$	6

Definition 2.6. Core: A payoff vector x is in the core of a coalitional game (N,v) if and only if $\forall S \subseteq N, \sum_{i \in S} x_i \geq v(S)$ [58].

Based on the definition of core, the sum is over only those players in S, ignoring those components of x belonging to players in $N \backslash S$. If a payoff x is in the core, the sum of payoffs in x to any group of players $S \subset N$ is bigger than what they could share if they form their own coalition. Since S covers singleton coalitions, this definition implies that payoff vectors in the core are imputations. For example, a three player game with transferable utility is in Table 2.1. The payoff vector $\{2, 2, 2\}$ is in the core, for the total value is equal to the value of the grand coaltion. On the contrary, the payoff vector $\{1, 2, 2\}$ is not in the core, because the value of coalition $v(\{1, 2\}) = 4$ is bigger than the assigned payoff 3 which are the value summed by player 1 and player 2 in the payoff vector.

2.4 Voting Games

There is a set of players N in the voting or selection like situation. Each player has right to vote based on his/her capability. We call this as the voting power, and generally it is called weight. A quota is established as threshold to win the game. This kind of games are called voting games. In this group decision-making process, if a bill or a candidate is passed, only if there are enough representatives support for it or him. In other words, the sum of the advocator's weights are more than the quota.

Definition 2.7. Simple game: A coalition game $G = (N, v)$ is simple if for all $S \subset N, v(S) \in \{0, 1\}$ [58].

A simple game is a compact representation of a coalitional game.

Definition 2.8. Weighted Voting Game (WVG): In a simple game $G = (N, v)$, weights w_i be assigned to each player $i \in N$. Let $W = \sum_{i \in N} w_i$, and $q \in [0, W]$. The value of a coalition is 1 if $\sum_{i \in N} w_i > q$; 0 otherwise.

A coalition $S \subseteq N$ is called a winning coalition in a WVG G if $v(S) = 1$. The family of all the winning coalitions is denoted by $WIN(G)$ when there is no ambiguity [41].

Definition 2.9. Minimal winning coalition: A coalition $S \subseteq N$ is a minimum winning coalition iff $v(S) = 1$ and $v(S \setminus \{i\}) = 0 : i \in S$. In other words, a winning coalition from which remaining any one player makes it into a losing coalition is called a minimal winning coalition.

Definition 2.10. Dictator: A player is a dictator iff $\exists i \in N \ v(\{i\}) = 1$. The dictator can determine if a weighted voting game wins or not.

Definition 2.11. Veto player: A player is a veto player, iff $\exists i \in S$ and $\forall S \subseteq N$, then $v(\{S\}) = 1$ and $v(S \setminus \{i\}) = 0$. A player that appears in all winning coalitions, without him other players can not reach the quota.

Definition 2.12. Dummy player: A player is a veto player, iff $v(S) = 1$ and $v(S \setminus \{i\}) = 1 : i \in S$. In other words, if we remove the dummy players from a winning coalition, the winning coalition is still winning.

These terms can be given meaning within the framework of voting games. For example, a dictator is a person who can "win" without the assistance of other players. So if a voting game has a minimal winning coalition with a single player, the player in that coalition is a dictator. However, unless there is a rule preventing this, there may be more than one dictator in a game. Several minimal winning coalitions might consist of a single player. Veto players are the players include in all winning coalitions. Dummy players are those players if they are removed from a winning coalition, it is still a winning coalition. I will show an example to explain those terms. Let us consider the game $G = (N, v)$, which has three players $N = \{n_1, n_2, n_3\}$ with $w_1 = 6$, $w_2 = 3$, $w_3 = 2$ respectively. If we set the quota q at the "majority" level, $q = 6$, then we have the following collection of minimal winning coalitions: $\{n_1\}$. So the dictator of the game is n_1. Players n_2 and n_3 are dummies in this game because they are not members of any minimal winning coalition. If we set quota q less than ”majority” level, for example, $q = 5$, the minimal winning coalitions of this weighted voting game are $\{n_1\}$ and $\{n_2, n_3\}$.

2.5 Computational Complexity of Coalitional Games

From a computational point of view, the key issues of coalitional games are how these games could be concisely represented, and how to efficiently compute the solution concepts. Weighted voting games are widely used representations of coalitional games in practice. For example, the voting system of the European Union is a combination of weighted voting games [6]. With an exponentially large number of players, Deng and Papadimitriou [21] show that the computational complexity of the core of coalitional game is infeasible. In other words, it is a NP-hard problem. Also, it is NP-hard to compute the Shapley value of a given player [21]. If the input size is polynomially large, we can check whether the core is empty or not in polynomial time. In [4, 22], Elkind gives a dynamic programming method to find if the core exits or not in a weighted threshold game in polynomial time. By fixed the type of the player, Aziz gives a general algorithm to find the coalition structure in polynomial time [3].

Chapter 3

E-PUCK ROBOT: DESCRIPTION AND LOCALIZATION

This chapter describes the robots used for empirical evaluation in the rate of the thesis. The experimental setup of the robots focuses on a robot's localization and multi-robot coordination aspects. Particular emphasis of our experiments is put on sensor and actuator noise and limited or unreliable communication, because the chosen platform e-puck provides only a limited amount of computational power and memory , has limited 'range R' proximity and has a high wheel-slip.

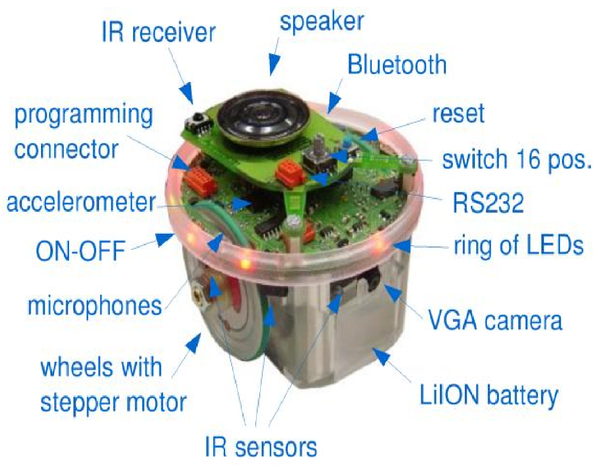


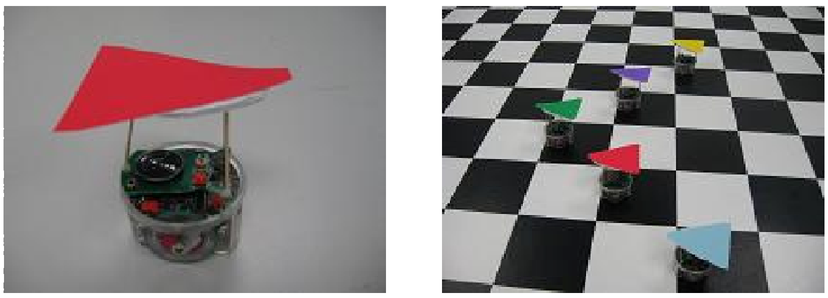
Figure 3.1: The area coverage platform: Epuck robot [30].

3.1 E-puck Robot

The e-puck robot is developed by the Autonomous System Lab at EPFL, as shown in Figure 3.1. The e-puck robot has a diameter of 7 cm, each wheel has a diameter of 4.1 cm. The epuck is operated by a dsPIC 30 microprocessor clocked at 30 MHz, and capable of 15 MIPS. It has 8 KB of on-board RAM, and 144 KB Flash memory. The sensors that are available on the e-puck are: (1) Eight infra-red distance sensors to detect of obstacles in a range of 7cm, (2) Bluetooth capability for wireless communication, (3) 8 LEDs arranged in a ring around the robot, one body LED and one front LED, etc. The maximum speed of the robot is 13 cm/s, and gets about 2 hours of running time in an environment with a smooth floor.

Table 3.1: Camera Parameters

Details	Value
Max Digital Video Resolution	1600 x 1200
Video Capture	1600 x 1200 @ 30 fps
Pixel	2.0 megapixels
Len	Carl Zeiss optics color len
Connection	USB



(a) (b)

Figure 3.2: (a) an e-puck with a red marker, (b) 5 e-pucks moving in the 'V' shape team.

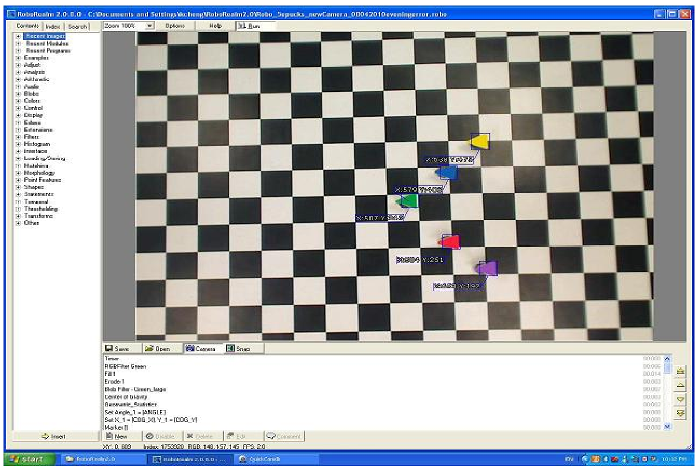
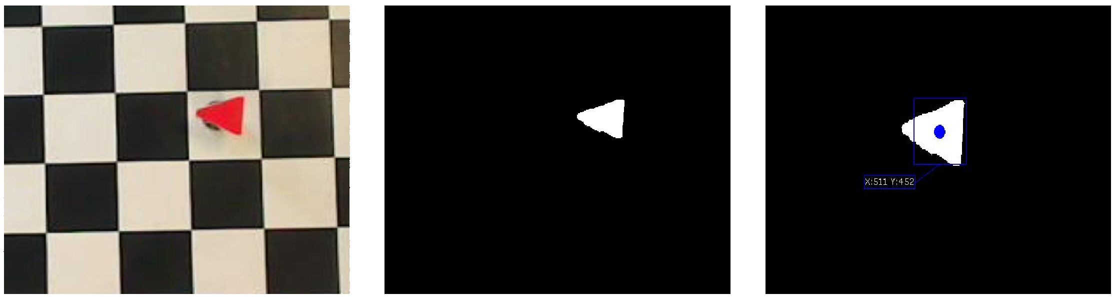


Figure 3.3: Screen shot of five e-pucks within our experiment arena in the image processing software user console.



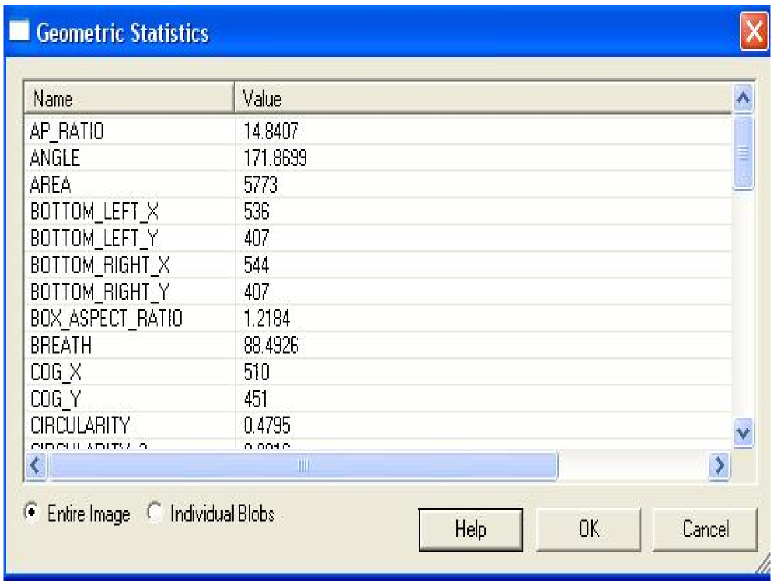
(a) (b) (c)

Figure 3.4: Screen shots of roborealm console: (a)the raw image with red triangle, (b) the image after RGB filter processing, (c) the final analyzed image with a coordinate location.

3.2 Overhead Camera-based Localization

To realize the localization for the robots, we have tested our technique in an indoor lab environment in a square shape measuring 2.5×2.5 m^2 with 5 or 6 epuck robots. To improve the recognition results and reduce noise, we use a floor with clean and smooth surface. We can easily distinguish the color markers from white and black chessboard like floor which are close to minimum and maximum RGB values: white is 255, and black is 0. Moreover, we built a marker for each e-puck using a blue, red, green, yellow, cyan, or magenta triangle marker to denote its location and heading clearly within the image captured by the overhead camera. Figure 3.2 (a) shows an e-puck with a red marker on the top. A five e-puck 'V' shape team is shown in Figure3.2(b).

We have used an overhead camera-based localization system that mimics the functionality of a GPS. Our camera is a Logitech QuickCam Pro 9000, whose main factors are shown in Table 3.1. The image of the environment captured by the camera measures 1273×1059 pixels within the 1600×1200 total range. Therefore, the average size of each pixel is



Name	Value
AP_RATIO	14.8407
ANGLE	171.8699
AREA	5773
BOTTOM_LEFT_X	536
BOTTOM_LEFT_Y	407
BOTTOM_RIGHT_X	544
BOTTOM_RIGHT_Y	407
BOX_ASPECT_RATIO	1.2184
BREATH	88.4926
COG_X	510
COG_Y	451
CIRCULARITY	0.4795

[✓ Entire Image] Individual Blobs

Figure 3.5: Screen shot of a geometric statistics window.

0.249×0.235 cm^2. Also, the camera is capable of 24 frames/sec of RGB capture. We used Roborealm [55] version 2.0.8 to perform image processing operations. The software analyzes the picture shot by the camera every second, as shown . First the color filter function finds the specific colored marker from the background using three value readings of RGB. Next, the erode function consolidates the boundary of the marker. Finally, the geometric statistics function calculates the location of the centroid of the marker. Based on the triangle shape of the marker, the heading of the marker is calculated as well. The screen shot in the image processing software user console is shown in Figure 3.3. The Roborealm software has built-in RGB functions to figure out the locations for each maker in the picture. For example, Figure 3.4 (a) shows the e-puck robot under the red triangle marker. To detect the triangle we use the RGBFilter module set to filter on red. To smooth out the triangle border and remove some of the additional noise we add the Erode module to produce a

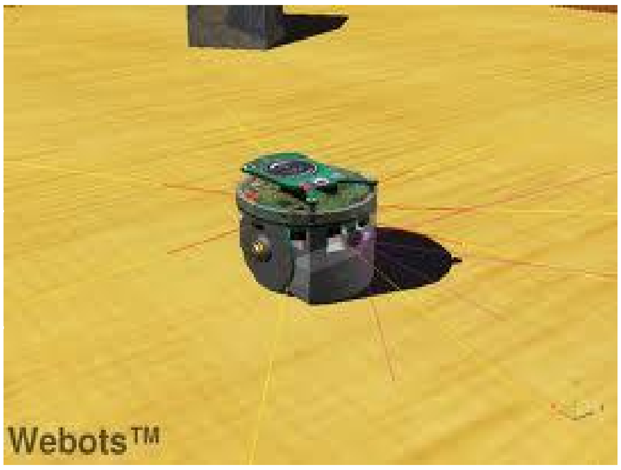


Figure 3.6: A epuck in Webots simulator.

cleaner image. We also add in the Blob Filter and set it to eliminate all but the largest blob just in case some other red artifact remains after the erosion. The image completes the red triangle (robot) extraction, shown in Figure 3.4 (b). Once the red triangle has been segmented from the background we add the Geometric module to run an analysis against the triangle. The value in particular that we are looking for is the ANGLE parameter which gives an indication of orientation of the blob, as shown in Figure 3.5. Finally, the marker is targeted in the Roborealm console, as shown in 3.4 (c). The solid circle shows the centroid of the red triangle, and its location is [511, 452] in the pixel unit.

3.3 Realistic Simulation

The scenario described in the former section has been fully implemented in the realistic robot simulator called Webots [47]. Webots is a powerful robotic simulation platform that allows realistic modeling of robots and environments including the parameters of different sensors on robots and the physics of the environment. Each robot in our simulated system is modeled as an e-puck robot with accurate models of the features and characteristics of the physical e-puck robot, as shown in Figure 3.6.

Chapter 4

MULTI-ROBOT COVERAGE: INDIVIDUALLY COORDINATED ROBOTS USING SWARMING

4.1 Multi-Robot Distributed Coverage Problem

We consider a scenario where R mobile robots are deployed into an initially unknown two-dimensional environment.Without loss of generality we assume the environment to be a square with each side being D units long. The area of the environment to be covered is then D^2 sq. units. Let O be the area within the environment that is occupied by obstacles. Each robot is equipped with a square coverage tool with a width $d << D$. Let a_r^t denote the action performed by a robot $r \in R$ during timestep t that results in the robot's motion and sweeps its coverage tool over an area c_r^t. Informally, we can denote a unit coverage function $f : a_r^t \rightarrow c_r^t$, that transforms robot r's action(motion) a_r^t into the area c_r^t covered due to that action. The objective of the distributed terrain coverage problem is to find a sequence of actions for each robot that ensures the following criteria: a) *Complete Coverage Criterion.* Every portion of the environment should pass under the coverage tool of at least one robot b) *No Overlap Criterion.* There should be no overlap between the regions covered by different robots c) *Distributed Behavior Criterion.* Each robot should determine its actions autonomously, in a completely distributed manner, so that the system can be scalable and robust. Formally, the multi-robot distributed terrain coverage problem can be defined as the following:

Definition 4.1. *Multi-robot Distributed Terrain Coverage Problem.* Given a set of R robots in an initially unknown environment find a set of actions $a_r^1, a_r^2, ...a_r^T$ to be performed autonomously by each robot $r \in R$ such that $\cup_{r\in R} \cup_{t=1...T} c_r^t = D^2 - O$ (complete coverage criterion) and $\cap_{r\in R} \cap_{t=1...T} c_r^t = \{\phi\}$ (no overlap criterion), where $f(a_r^t) = c_r^t$ gives the unit coverage function that translates the action of robot r during timestep t into the corresponding area c_r^t covered by that robot's coverage tool.

4.1.1 Distributed Coverage with Constrained Mini-robots

To solve the constrained distributed multi-robot coverage problem, we have defined two maps that are stored within the memory of each robot to record the coverage history and locations of nearby robots, as described below:

1. **Communication Map.** The communication map ξ_r^t for robot r at time t contains the locations of other robots that are within the communication range ξ of robot r. That is, $\xi_r^t = \{l_{r'}^t\}$ where, $r' \in R - \{r\}$ and $l_{r'}^t$ denotes the location of a robot that is within communication range of robot r.

2. **Coverage Map.** The coverage map χ_r^t for robot r at time t contains locations extending over a radius χ from the robot's current location. We assume that the coverage map radius of a robot is smaller than its communication range, i.e., $\chi < \xi$. Following the node counting technique in [37], each location in a robot's coverage map is associated with a value that represents the number of times the location has been visited by robots. To build/update its coverage map, when a robot r visits a location l at time t it adds the location to its own coverage history cov_r. Because of the finite size, t_h, of the coverage history, the robot has to discard the location it visited $t_h - 1$ time steps before, to accommodate the new location within cov_r. The coverage history update rule is then given by:

$$cov_r \leftarrow cov_r \cup \{l_r^{t_{curr}}\} - \{l_r^{t_{curr}-t_h-1}\}.$$

At each time step, each robot exchanges its own local coverage history with all other robots within its coverage map χ_r^t. Each robot then fuses the information in its own coverage history with the coverage information received from other robots to generate the coverage map for its immediate neighborhood. For fusing the coverage history of multiple robots, each robot simply adds the node count value of overlapping cells in the coverage map of the robots, similar to the node counting technique used in [37]. The coverage map for a robot r at time step t is then given by: $\chi_r^t = \{\chi_l^t\}$, where $\chi_l^t = \sum_{r' \in \chi_r^t} c_{l,r'}^t$, and,

$$c_{l,r'}^t = \begin{cases} 1 & \text{if } l \in cov_{r'}, \\ 0 & \text{otherwise.} \end{cases}$$

A possible source of inaccuracy in the node counting technique is that the same cell could be counted multiple times when a pair of robots exchange coverage information over successive time steps. To prevent this, each robot considers the current location of other robots, sent as part of the communication map at each time step that are also within its coverage map. It then considers the robot locations within its coverage map it had obtained during the previous time step. For the current robot locations that are among the 8-neighbor cells of any of the robot locations in the previous time step, the robot infers that the robot at that current location was already within its coverage map in the previous time step. In such a case, the robot does not fuse the entire information from the other robot's coverage map, but only updates the node count for the current location of the other robot. On the other hand, if the other robot was not within robot r's coverage map in the previous time step, it fuses that robot's coverage map with its own using Equation 4.1 above.

The value associated with a location within a robot's coverage map could be considered similar to the cumulative deposit of a virtual chemical substance such as *pheromone* by different robots at that location. The combination of pheromone values in the coverage maps across different robots enables each robot to maintain a consistent pheromone landscape across robots which are within the coverage map/range of each other. The pheromone landscape stored in the local coverage map of each robot can then be used by a robot to select an action that enables it to move towards locations that have less pheromone within its coverage map.

This pheromone-guided dispersion of robots is successful in preventing a robot from revisiting locations already visited by itself and in dispersing robots away from each other to enable them cover disjoint regions. However, our simulation experiments show that such a dispersion approach has two deficiencies that reduce the efficiency of the global objective of achieving complete coverage of the environment. First, it 'pushes' robots towards the periphery of the environment and prevents any robot from covering the center of the environment. Secondly, the technique of the fusing the coverage map information from multiple robots relies on the presence of other robots within a robot's communication range. If robots continuously disperse away from each other, they would ultimately be beyond each others' communication range and would not be able to exchange coverage maps with each other to avoid redundant coverage. To address these deficiencies, we describe an information gain based technique along with the pheromone-guided dispersion in the next section.

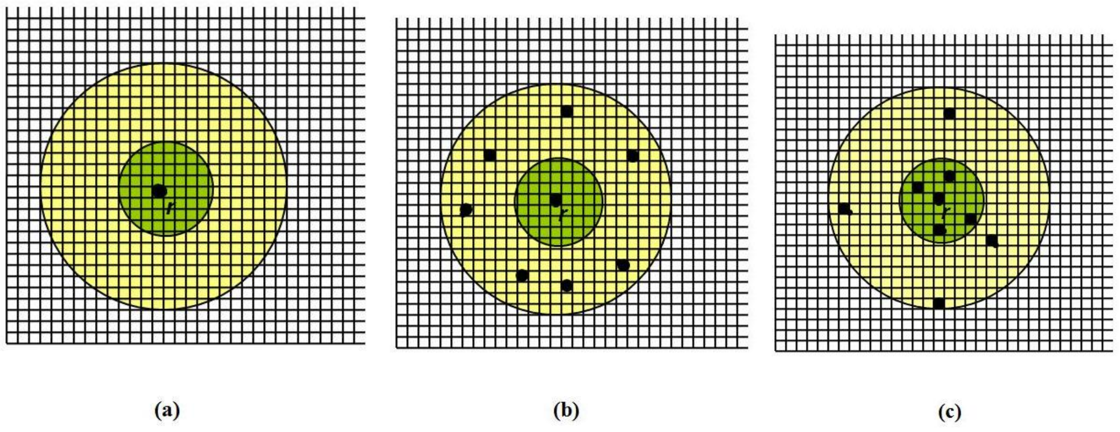


Figure 4.1: Different configurations of robots in the neighborhood of a robot r. The outer circle represents robot r's communication range ξ and the inner circle represents robot r's coverage map radius χ. (a) There are no other robots within robot r's communication and coverage maps; (b)There are some robots in robot r's communication map, but no robot in its coverage map; (c) There are some robots within robot r's communication and coverage maps.

4.2 Area Coverage Algorithms

Our multi-robot area coverage technique is based on the insight that memory constrained robots can reduce the redundancy in the area covered by each other, if they selectively move closer to each other to exchange their coverage information maps instead of always moving away from each other. However, the coverage information for a robot can improve after taking an action, only if that action brings the robot within communication range of at least one more robot. On the other hand, always moving closer to other robots to improve the coverage map can limit the dispersion of the robots across the environment and adversely affect the coverage. Clearly, at each step a robot faces a tradeoff between coverage redundancy (measured by the amount of local coverage information it can gain by moving closer to other robots) and dispersion (measured by the amount of uncovered

```
function selectActionECM returns Action selectedAction {
inputs: Map $\xi_r^t$;
    Action $lastAction$;
    Location $l_r^t$;
variables: Set $R_\xi$;
    Action[] $AC^i$, $AC^j$;
    Set¡Action[]¿ $AC$;
    double[] $f_r$, $F_r$; // Probability density and distribution
        functions for selecting robot $r$'s action
1. $R_\xi \leftarrow$ Set of locations of robots
        in $\xi_r^t$ including $r$;
2. if $R_\xi = \{\phi\}$ doRandomWalk();
3. For every $r \in R_\xi$, find the set of actions $\{ac_r\} \subset Ac$ s. t.:
4.     $l \leftarrow move(r, ac_r, l_r^t)$;
5. $\overline{AC} \leftarrow \times_{r \in R_\xi}\{ac_r\}$
6. For every $AC^i \in \overline{AC}$
7.     $f_r[ac] \leftarrow \frac{h_{MD}(Ac^i)}{\sum_{Ac^j \in AC} h_{MD}(Ac^j)}$  such that $(ac_r) \in AC^i$;
//Calculate the distrib. function $F_r$ from the p.d.f. $f_r$
// and select an action probabilistically according to $F_r$.
8. $F_r[ac] \leftarrow \sum_{i \in Ac} f_r[i]$;
9. if (rand() $\geq F_r[ac] \mid \exists ac \in Ac$)
10    selectedAction $\leftarrow$ ac;
11. return selectedAction;
}
function $h_{MD}$ returns int cumManhattanDist{
inputs:inputs: JointAction $Ac^i$;
for every pair of $(r_i, ac_{r_i}), (r_j, ac_{r_j}) \in \overline{ac}^i$
  $(l_i) \leftarrow move(r_i, ac_{r_i}, l_{r_i}^t)$;
  $(l_j) \leftarrow move(r_j, ac_{r_j}, l_{r_j}^t)$;
  cumManhattanDist $\leftarrow$ cumManhattanDist
      $+ dist((x_i, y_i), (x_j, y_j))$
return cumManhattanDist;
}
```

Figure 4.2: Algorithm used by a robot r to select its next action if there are some robots within its communication map but no other robots in its coverage map.

territory it can visit by moving away from other robots). Considering these issues, we design three local movement (action selection) rules for each robot r based on the number of other robots in its neighborhood, as described below:

1. **No other robots within r's communication map or coverage map.** In this scenario shown in Figure 4.1(a), robot r has no other robots either within its communication map or within its coverage map. Robot r then performs a random walk realized by

moving in a straight line for a random number of time steps (between 30 and 100) and then selecting one of its possible actions randomly to make a random turn. If the robot encounters an obstacle, it makes a random turn and continues with its random walk.

2. **Some robots in r's communication map, but no robots in r's coverage map.** This scenario is shown in Figure 4.1(b). Because there are no other robots within the coverage map of robot r, robot r cannot increment its coverage information by fusing coverage information from other robots. In such a scenario, robot r selects an action that makes it disperse away from the robots that are within its communication map. The algorithm for acheiving this is described in the *selectActionECM* method in Figure 4.2. First, robot r obtains the locations of robots that are within its communication map in the set R_ξ (line 1). If this set is empty, robot r infers that it is in a scenario similar to case 1 above and performs a random walk (line2). Otherwise, robot r determines the set of possible joint actions with other robots whose locations are in R_ξ (lines 3-5). Each robot then calculates the sum of the Manhattan distances between itself and other robots for each joint action (lines 4-7). The $move(r, ac_r, l_r^t)$ method used in line 4 in Figure 4.2 gives the location that robot r would reach in the next time step if it executed action $ac_r \in Ac$ from its current location l_r^t at time t. The normalized values of these combined Manhattan distances are then used to probabilistically select the next possible action at each robot (lines 9-11).

Using this technique, the expected action of a robot r corresponds to the action with the maximum value of the Manhattan distance heuristic and is given by: $E(ac_r) = arg_{ac^i_r} \max P_{Ac^i} \mid ac^i_r \in Ac_i$. The Manhattan distance based heuristic therefore ensures that robots have the highest probability of selecting the joint action that maximizes their combined Manhattan distances with each other. This results in robots dispersing away from each other in successive steps to explore the environment. In certain scenarios, this algorithm can cause a robot to have the maximum heuristic value for an action that takes it towards an obstacle. However, the robot does not continuously reattempt the same action of moving towards the obstacle (i.e., does not get perennially stuck trying to move into an obstacle) because the probabilistic nature of the action selection ultimately allows thee robot to select an action that does not correspond to the action giving the maximum combined Manhattan distances, with a non-zero probability.

3. **Some robots within robot r's coverage map** This scenario is illustrated in Figure 4.1(c). Evidently a robot r faces a tradeoff while selecting an action between moving closer to some of the other robots to gain more coverage information (and attempt to reduce redundant coverage) and moving away from some of the other robots to achieve better dispersion in the environment. We include another factor in this decision, by considering the potential information gain from robots that would move into the coverage range of robot r (from within its communication range) as result of each of its possible actions/movements. The algorithm for achieving this functionality is shown in the *selectAction* method in Figure 4.3. Robot r first determines the set of

```
function selectAction returns Action selectedAction{
inputs: Map $\chi_r^t$, $\xi_r^t$;
      Action $lastAction$;
      Location $l_r^t$;
variables: Set $R_\xi$, $R_{\chi,ac}$;
      Action[] $Ac^i$;
      double[] $f_r$ $F_r$; // Probability density and distribution
         functions for selecting robot $r$'s action

$R_\xi \leftarrow$ Set of locations of robots in $\xi_r^t$ including self;
if $R_\xi = \{\phi\}$ doRandomWalk();
$R_{\chi,ac} \leftarrow$ Set of robots in $\chi_r^t$ after taking action $ac \in Ac$
         from $loc_r^t$;
if ($R_{\chi,ac} = \{\phi\} \mid \forall ac \in Ac - \{lastAction\}$)
   selectActionECM($\xi_r^t$,lastAction,$l_r^t$);
else for every $ac \in Ac$
   For every $l \in 8-$ neighbor $of l_r^t$
      $\overline{\chi}_l^t = \frac{\chi_l^t}{\sum_{l \in 8-neighbors\,of\,l_r^t} \chi_l^t}$;
   $R_{dec,ac} \leftarrow$ Set of robots in $\chi_t^r$ with which distance
         decreases after taking action $ac \in Ac$;
   $R_{inc,ac} \leftarrow$ Set of robots in $\chi_t^r$ with which distance
         increases after taking action $ac \in Ac$;
   $R_{ig,ac} \leftarrow$ Set of new robots that enter into $r$'s coverage map
         after taking action $ac \in Ac$;
   // Function IE returns the information entropy in coverage maps of robots
   $f_r[ac] \leftarrow \frac{\lambda_1|R_{dec,ac}|+\lambda_2|R_{inc,ac}|+\lambda_3|R_{ig,ac}|\times IE(R_{ig,ac})}{|R_{dec,ac}|+|R_{inc,ac}|+|R_{ig,ac}|} \times (1-\overline{\chi}_l^t)$,
      where $l = move(r, ac, l_r^t)$;
//calculate the distribution function $F_r$ from the p.d.f. $f_r$
$F_r[ac] \leftarrow \sum_{i \in Ac} f_r[i]$;
if (rand() $\geq F_r[ac] \mid \exists ac \in Ac$)
   selectedAction $\leftarrow$ ac;
return selectedAction;
}
```

Figure 4.3: Algorithm used by a robot to select its next action when there are robots both in its communication map and coverage map.

other robots $R_{\chi,ac}$, that would be within its coverage map by taking action $ac \in Ac$. If the coverage map of robot r is empty for all possible actions, robot r cannot gain any coverage information from other robots by taking any of its actions. It then infers that it is in a scenario similar to case 2 above and uses the *selectActionECM* algorithm given in Figure 4.2 to disperse away from other robots. On the other hand, if the set $R_{\chi,ac}$ is non-empty for some subset of actions of robot r, robot r first calculates the normalized pheromone or node count values for each of the 8-neighbors of its current location l_r^t. Robot r then considers the number of robots with which its distance decreases($R_{dec,ac}$), the number of robots with which its distance increases($R_{inc,ac}$), and the number of robots that enter into r's coverage map ($R_{ig,ac}$) because of action ac. The information gain associated with action ac is calculated as the information entropy (IE) from the robots in the set $R_{ig,ac}$. This information entropy from the set of robots in $R_{ig,ac}$ is given by $IE(R_{ig,ac}) = -P_i \times log_2 P_i$, where $P_i = \frac{1}{\chi_r^t}$, and χ_r^t is the size of the coverage map of a robot. Robot r then calculates the density function for its different actions according to the product of the linear weighted sum of the number of robots with which its distance increases, the number of robots with which its distance decreases and the information entropy from the robots that move into its coverage map, and, the inverse of the normalized pheromone value of the location it would reach by taking the action ac. It then probabilistically selects one of its possible actions using the distribution function calculated from the above density function.

4.2.1 Node Counting Strategies

As mentioned in Section 4.1, we consider a node counting technique similar to [37] in our algorithm. In [37], a robot associates an integer value called a cell's *count* with every cell in the map of the environment. A robot increments the count value of a cell every time it visits that cell, to update its coverage information of the environment. However, this node counting technique requires every robot to maintain the count value of every cell within the map of the environment. While [37] uses a central location that has no memory limitations to store and fuse the coverage information from all robots, we cannot use such a centralized storage technique in our system because each robot stores the coverage information only within its local memory. The limited storage available on each robot in our system can lead to the following two problems when we use node counting techniques for recording coverage information:

1. **Finite History Sizes.** Each robot has to discard a part of its coverage map containing its own coverage history as well as fused coverage information obtained from other robots with every movement/action it makes, corresponding to the cells that exit its coverage map. Discarding older coverage information by each robot could potentially result in repeated coverage of the same region several times and could adversely affect the efficiency of the area coverage.

2. **Repeated, but not continuous encounters between the same pair of robots.** As mentioned in Section 4.1, robots prevent repeated fusion of the same coverage information by maintaining a history of robot locations from the last time step and discarding duplicate coverage information sent by robots whose coverage information had already been fused earlier. However, if a pair of robots intermittently leave and re-enter the coverage map/range of each other, they are not able to remember having encountered each other before, and, consequently fuse the duplicate coverage information in their coverage maps. Evidently, this results in inflated values of the node count for the locations on the coverage maps of these robots and adversely

affects coverage.

We posit that the coverage related problems with the node counting technique due to limited memory on-board the robots can be mitigated if the coverage information is made volatile. We achieve this by using the concept of pheromone evaporation commonly used in emergent systems[8]. To realize the pheromone evaporation, each robot r decrements the pheromone values stored within its coverage map at each time step using the equation $\chi_l^t \rightarrow \beta \times \chi_l^t$, where β is a constant. The pheromone decrement over time results in a pheromone gradient along the trail or path followed by a robot with higher pheromone values at locations recently visited by the robot and lower pheromone values at locations visited earlier. This pheromone gradient information from the coverage history of a robot is exchanged with other robots when the coverage maps of the robots are exchanged at each time step. A robot could then use this pheromone gradient information from its fused coverage map to infer the direction of movement of other robots in its vicinity and select an action that balances the information gain and dispersion from other robots. We have used four different strategies for pheromone deposit and update in the coverage map of robots to observe the effect of different degrees of information volatility and pheromone gradient in the coverage map of a robot. These strategies are summarized in Figure 4.4.

4.3 Agents with Communication Faults

The local heuristic h_{MD} used by an agent in our algorithm relies extensively on the correct location coordinates of other agents located within its coverage and communication maps. However, most sensors, especially GPS sensors that are commonly used to obtain location coordinates, are characterized by noisy readings. Therefore, it makes sense to analyze the operation of the agents' local algorithm as well as the global system behavior in the presence of sensor noise. To model faulty communication of coordiniates in our system, we assume that sensor noise is distributed as a uniform, zero mean distribution. Let δ denote the maximum number of units of error along each coordinate that can be reported in a sensor reading. As before, we consider $\mid R \mid$ agents in a two-dimensional environment. Because we consider the sensor noise as zero-mean, uniform distribution a mean value analysis of the effect of the noise on h_{MD} is likely to return a zero value. Therefore, we quantify the effect of sensor noise in terms of the standard deviation in h_{MD}.

Proposition. The standard deviation in h_{MD} due to the presence of sensor noise with a maximum of δ units in each dimension, distributed uniformly with a zero mean has an upper bound of $O(\delta)$.

Proof. For simplifying our analysis, we assume that all agents are within communication range of each other. Then, the number of possible sensor readings that can be received by agent $r \in R$ is given by $(2\delta + 1)^{2|R|}$. Out of this set of readings, the ones that have the same

Pheromone Update Strategy	Possible Agent Movement	Next action select criterion
Increase only	To any adjacent cell except most recently visited, increment visited count in cell by 1	if $(x, y) \leftarrow move(r, ac, loc^t_r)$ $\mu^{t+1}_{x,y} \leftarrow \mu^t_{x,y} + 1;$ Select ac corresponding to min $\mu^{t+1}_{x,y}$
Decrease	Same as increase only, but visited value in a cell decreases(evaporates) exponentially with time	if $(x, y) \leftarrow move(r, ac, loc^t_r)$ $\mu^{t+1}_{x,y} \leftarrow \mu^t_{x,y} + 1;$ Select ac corresponding to min $\mu^{t+1}_{x,y}$ and $\mu^{t+1}_{x,y} < V_{Thr}$
Increase or Decrease with trails	Each agent considers pheromone gradient from other agents. Uses gradient information to select next action	$\{(x, y)_{r'}\} \leftarrow$ pher. gradient from agent r'. Select ac in direction of decreasing gradient

Figure 4.4: Selection condition for agent action for different pheromone update strategies.

amount of error with opposite signs along the x(latitude) and y(longitude) coordinates give zero error in h_{MD}, because the Manhattan distance heuristic adds the distances along the two coordinate axes. Therefore, the total number of values of h_{MD} that have zero error is given by $\delta \mid R \mid$. Correspondingly, the number of values with e units of error in h_{MD} is given by $[(2\delta + 1)- \mid e \mid]^{|R|}$, where $1 \leq e \leq 2\delta$. Since errors are distributed uniformly with zero mean, the standard deviation in the error in calculating h_{MD} is given by:

$$SD_{err} = \sqrt{\frac{2}{(2\delta+1)^R} \sum_{e=1}^{e=2\delta} [((2\delta + 1)- \mid e \mid)^R \times e]^2}$$

or, $SD_{err=} = O(\sqrt{\frac{2(2\delta)^{2|R|}}{(2\delta+1)^{|R|}}}) = O(\sqrt{\delta^2}) = O(\delta)$.

In our system, we have designed the agent rules at the local level, while the global behavior of the system is manifest by the interactions between the agents. In the next section, we analyze the global behavior of our system in terms of the two metrics coverage time and coverage redundancy under different operational constraints, including faults and failures of the agents.

4.4 Experimental Results in Webots Simulator

The different parameters used for the experimental setup are listed in Figure 4.5. Each simulation was allowed to run for 10000 simulator ticks. A simulator tick corresponds to the time required to perform one execution step for each entity in the simulation environment.

In our first set of experiments shown in Figure 4.6, we observe the coverage of the environment with different coverage strategies. Each agent has a coverage map radius of 5 meter (denoted by $MR5$ in the figure caption). One of the strategies tested allows agents to select a random action at each step. For the strategy labeled *Binary*, the value in each cell is considered as a binary number instead of a real value. In the strategies labeled *Inc*, the pheromone (node count) information does not decrease over time. The caption *Trail* denotes that the agents use trail gradient information in their coverage map to select their next action, as described in Figure 4.4. As shown in Figure 4.6, the strategies that use the trail information obtain the higest coverage because the agents decide their actions by combining the coverage information from their own coverage maps, as well as the possible

Symbol	Parameter	Value
λ_1	Weight of Decrease Manhattan Distance	$\{$-1, 0.5, 1 $\}$
λ_2	Weight of Increase Manhattan Distance	$\{$-1, 0.5, 1 $\}$
λ_3	Weight of Information Gain	0.05
ω	Percentage of Obstacles	$\{$5, 10, 25 $\}$
χ	Size of the whole map	$\{10 \times 10,$ $40 \times 40,$ $70 \times 70,$ 100×100 $\}$
ξ	Communication range of an agent	$\{11, 21, 41$ $\}$
μ	Radius for coverage map of an agent	$\{5, 7, 11\}$
A	Action of an agent	$\{'u',' l',' d',' r'\}$
α	Communication rate of an agent	8 ticks
β	Decay rate of pheromone	$\frac{1}{40}$ per tick
P_a	Probability of the next action	$\{0.8, 0.1\}$
P_f	Failure probability constant of an agent	$\{0.01 - 0.1\}$
P_r	Recovery probability constant of an agent	$\{0.01 - 0.1\}$
N_r	Number of agents	$\{5, 10, 15, 20\}$

Figure 4.5: Parameters used in the experimental settings for the simulations.

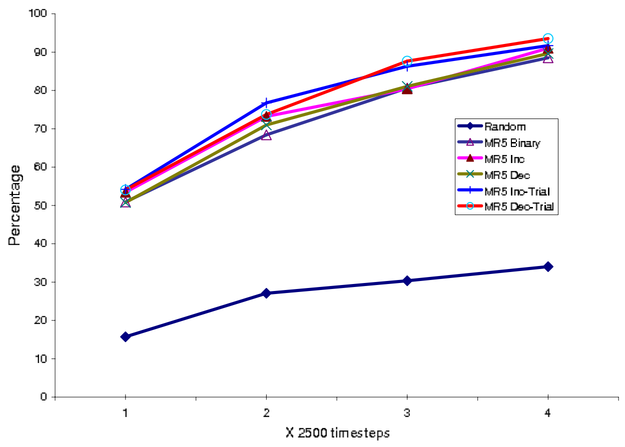


Figure 4.6: Percentage of area covered with 5 agents in a 100×100 square environment for different coverage strategies.

directions of movement of nearby agents estimated from the pheromone gradient in their coverage map. The strategies that use node counting without considering trail information perform slightly worse because each agent does not consider the possible locations of other agents while taking its action. The strategy with binary node counting performs poorly because binary node counting can capture less coverage information than real valued node counting. Finally, and quite expectedly, the random strategy performs very poorly because of the total absence of coordination among the agents.

In our second set of experiments, we analyzed the effect of changing the number of agents in the environment. Quite evidently, as the number of agents increases the coverage quality improves as long as the environment does not change. To further quantify the effect of the number of agents on the overall solution quality, we analyzed the redundancy in coverage in the environment by observing the number of cells that were revisited a specific number of times by agents. The results are shown in Figure 4.7. Quite interestingly, we

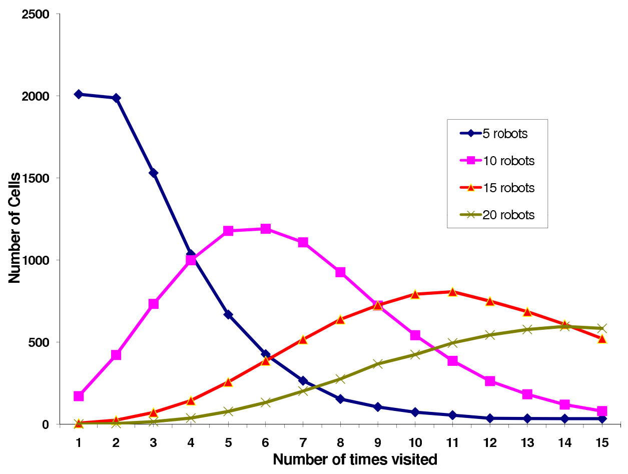


Figure 4.7: Redundancy in area covered as the number of agents in the environment changes.

observe that as the number of agents increases from 5 to 20, the peak of the redundacy curve shifts rightwards. Although this result simply indicates that as the number of agents increases while keeping the environment unchanged, the redundancy in coverage increases, these curves offer the useful insight of determining the optimal number of agents that result in the least redundancy for a given environment.

In our third set of experiments, we once again observe the variation in coverage redundancy while changing the size of the coverage map radius of each agent from 5 meter to 11 meter.[1] As shown in Figure 4.8, as the size of the coverage map increases, each agent has more coverage information about its immediate neighorhood and is able to select its actions more efficiently to reduce its coverage redundancy.

In our fourth set of experiments, we observe the variation in percentage of the coverage while changing the size of the communication range from 11 meter to 41 meter. As shown

[1]At 11 meter, the coverage map and communication map radii of each agent are identical.

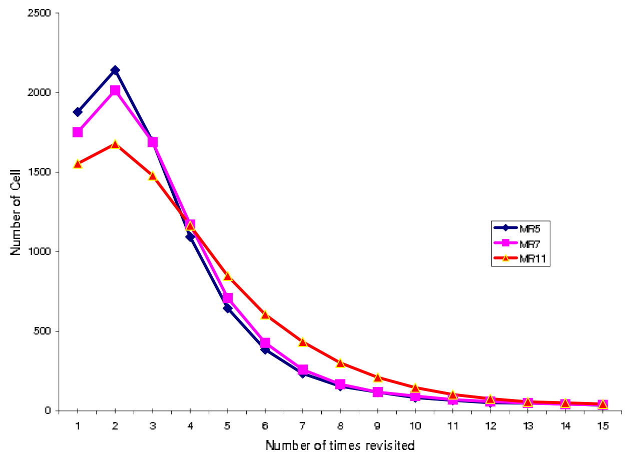


Figure 4.8: Redundancy in area covered as the coverage map radius MR of agents changes.

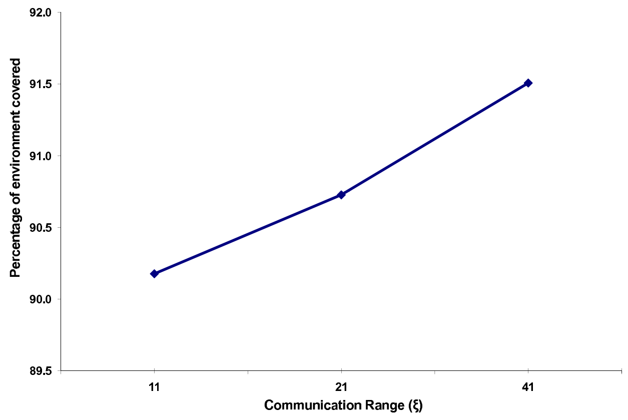


Figure 4.9: Percentage of the coverage with the communication range radius CR of agents changes.

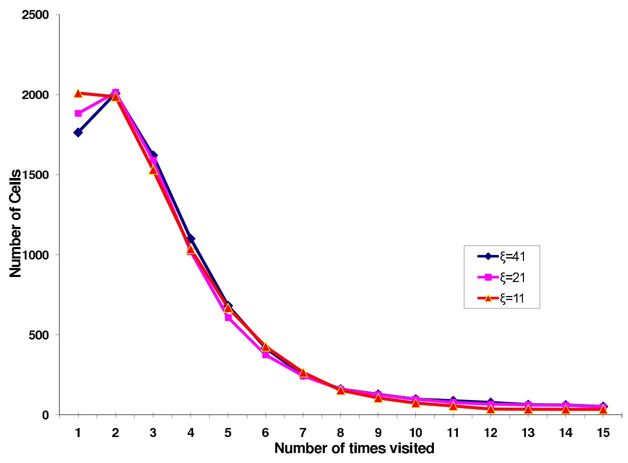


Figure 4.10: Redundancy in area covered as the communication range radius CR of agents changes.

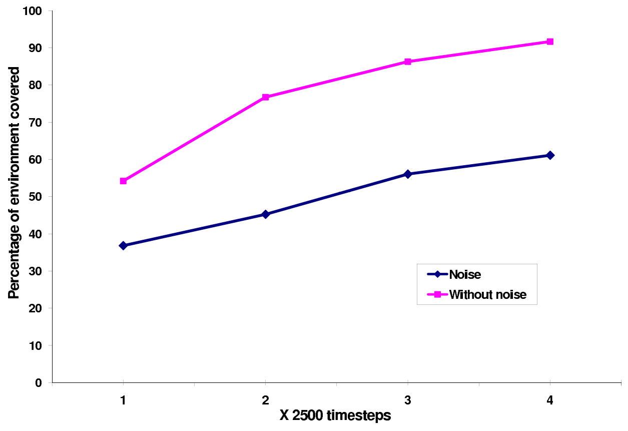


Figure 4.11: Percentage of the coverage with and without the GPS sensor noise.

in Figure 4.9, as the size of the communication range increases, each agent is in larger range to exchange both location and coverage map with other agents. Quite interestingly, we observe that as the CR increases from 11 to 41, the three redundancy curves nearly overlap each other in Figure 4.10. Although larger CR should have better coverage rate, it has no effect to the coverage redundancy with other parameter unchanged.

In our fifth set of experiments, we observe the variation in percentage of the coverage

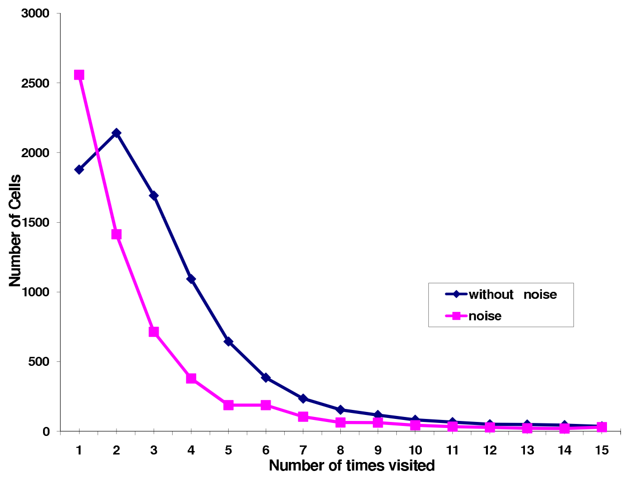


Figure 4.12: Redundancy in area with and without the GPS sensor noise.

while changing the size of the maze (the simulation environment) from $10\ meter^2$ to $100\ meter^2$. Unsurprisingly, larger area of the searching map, worse percentage of the coverage, shown as Figure 4.13.

In our sixth set of experiments shown in Figure 4.14, we randomly designed the obstacles with different shapes and locations from 5% to 25% in the simulator. As shown in Figure 4.15, as the maze size is unchanged, more obstacles got more percentage of the coverage. Figure 4.16 illustrates the coverage redundancy in area, and the results rightly match the results of the percentage of the coverage in Figure 4.15: more obstacles, less coverage redundancy. Our algorithms are not influenced by the obstacles quite much.

In our seventh set of experiments, we did the simulation tests by changing the weights of the information gain. The weights are used to tradeoff between the coverage information it can gain from other robots and the quality of coverage achieved by the robots. As there are 5 robots in our test scenario, the value of the information gain from a robot is from 0 to 20.

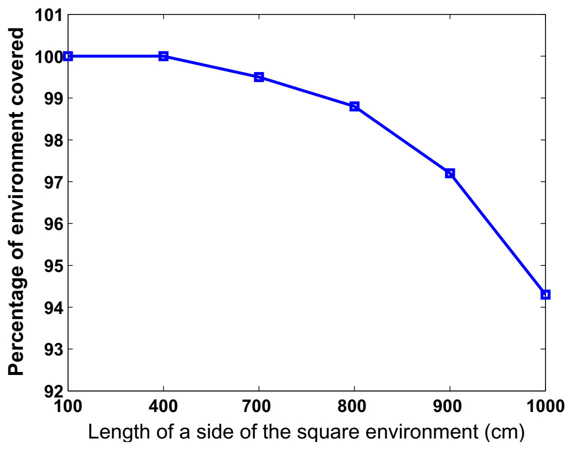


Figure 4.13: Percentage of the coverage with different size of mazes.

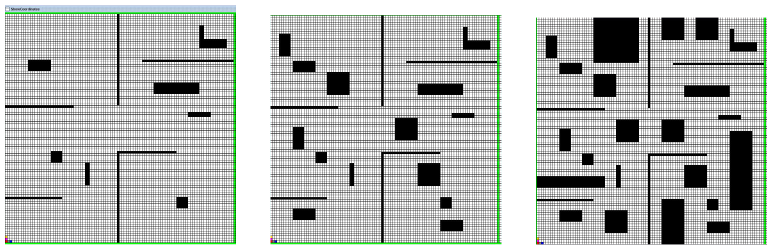


Figure 4.14: 5, 10, 25 percentage of obstacles.

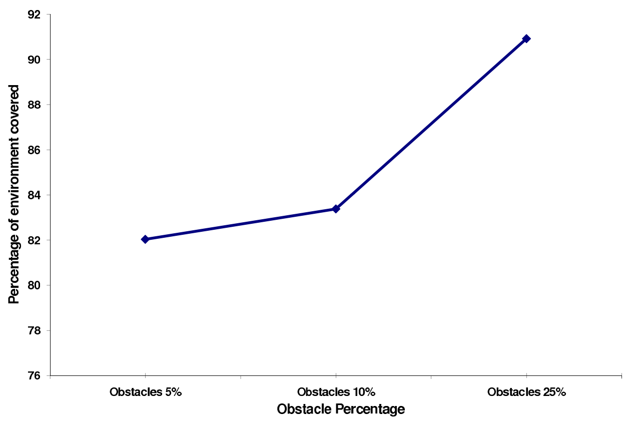


Figure 4.15: Percentage of the coverage with obstacles.

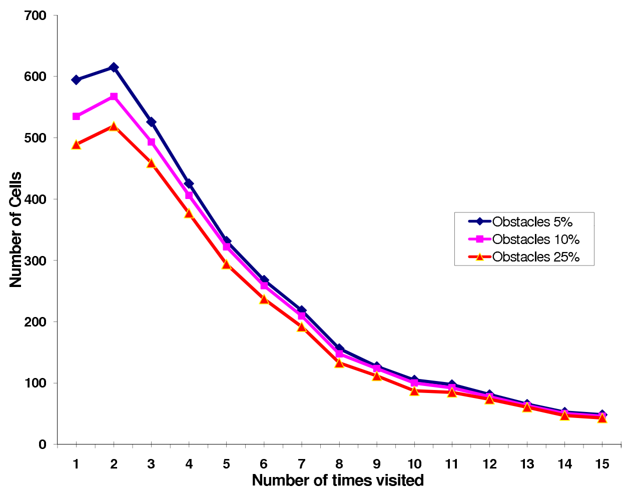


Figure 4.16: Redundancy in area with obstacles.

The weights represent the reward of the distance vary between two robots by a negative or positive number. Figure 4.17 illustrates percentage of the coverage with different weights of the information gain. The curve which represents the weight value always the same as +20 whenever the two robots go close or far shows the worst percentage of the coverage, and the other three curves show litter difference of the percentage of the coverage. Figure 4.18 illustrates the redundancy in area with different weights of the information gain. Except the curve A20A20, the three curves of the redundancy are also very similar. Empirically, we should use positive weight value to make the agents disburse and balance with the information gain.

In our next set of experiments, we analyze the effect of agent communication faults on the coverage time and coverage redundancy in the system. For the communication fault model, we have followed the sensor noise model described in [71] which is given by $y(n) = x(n)+a(0.0015\tilde{N}(0,1)+0.0012cos(2\pi n/r_u)+\phi_0)$. This models approximates sensor noise

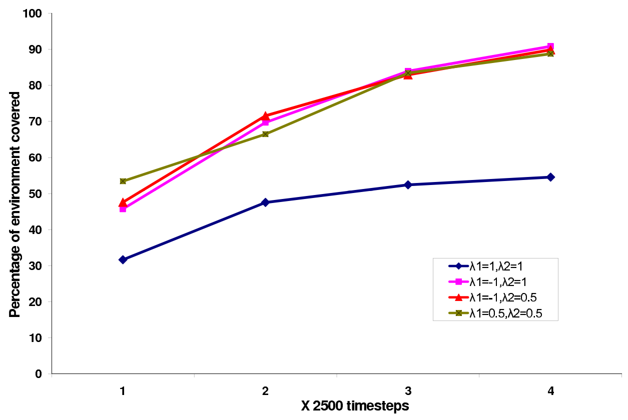


Figure 4.17: Percentage of the coverage with different weight values

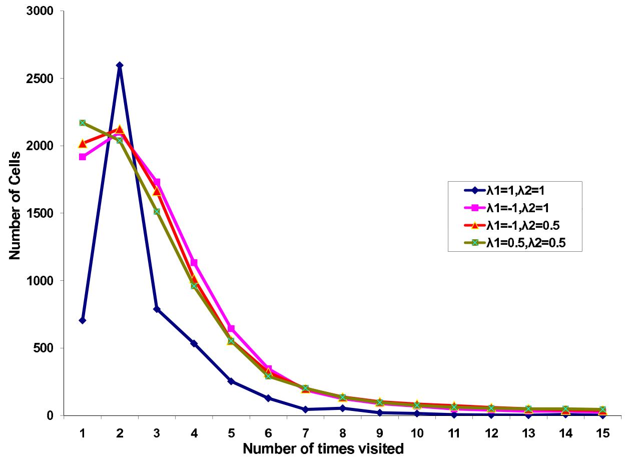


Figure 4.18: Redundancy in area with different weight values.

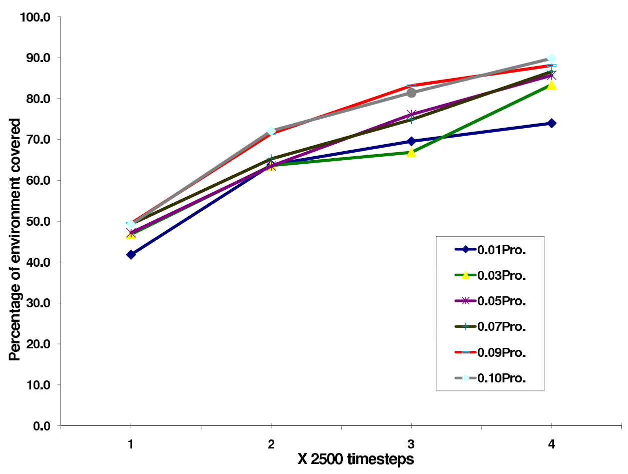


Figure 4.19: Percentage of the coverage if agents malfunction with different recover probabilities.

as a combination of a high frequency, zero mean, white Gaussian noise and a low frequency cosine wave with random period and random intial phase. For our simulation purposes, we have assumed, $a = 100$, $r_u = 1600$ and $\phi_0 = 0$. Figure 4.11 shows the coverage of the environment using with and without the sensor noise. In the presence of sensor noise, agents obtain about 35% less coverage than with noiseless communication, over the same number of timesteps. As observed in our mathematical analysis of the sensor noise model, the effect of sensor noise decreases the coverage linearly. A possible explanation of this behavior is offered by the coverage redundancy graphs with and without sensor noise shown in Figure 4.12. With sensor noise, the redundancy in coverage decreases. Therefore, although fewer cells get covered with sensor noise, many of the uncovered cells are those that were being covered redundantly when there was no noise.

In our last set of experiments, we measured the robustness of our system when some of agents failed and recovered at different time steps. During each time step, each agent

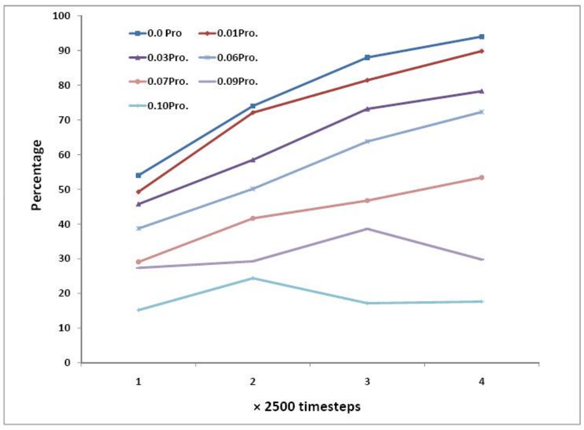


Figure 4.20: Percentage of the coverage if agents malfunction with different failure probabilities.

might fail with a given failure probability and subsequently recover with a given recovery probability. Fig. 4.19 illustrates the percentage of the coverage with 5 agents while keeping the failure probability constant at 0.01. The recovery probability of each agent is varied between 0.01 and 0.10. With increased recovery probability, the percentage of the coverage increased. Although some curves intersected possibly due to sensor noise, higher recovery probabilities got better coverage. Fig. 4.20 illustrates the results of a similar experiment where we kept the recovery probability constant at 0.10 and varied the failure probability between 0.01 and 0.10. Interestingly, we observe that the agent failure probability has a significantly more detrimental effect on the coverage than the agent recovery probability. Also, in Figure 4.20, the two non-monotonically increasing curves near the failure probability of 0.10 show that that a high agent failure probability not only results in worse coverage, but fails to keep up with the volatality of coverage information (pheromone evaporation) in the system, causing coverage to reduce over time and rendering the system unstable. In both these cases, the redundancy in coverage decreased as the respective probabilites

Table 4.1: Physical experimental results showing percentage of coverage of the environment by different number of robots with and without obstacles in a indoor lab environment.

	No obstacle		
Number of robots	Braitenberg	Fiducial	Transient Node Counting
3	40.3	43.7	44.5
4	45.6	49.2	50.2
5	47.7	56.5	58.3
	4.5% obstacles		
3	43.2	47.9	48.7
4	46.5	53.3	55.6
5	49.3	59.2	62.1
	12% obstacles		
3	44.7	52.7	56.4
4	48.5	58.7	60.1
5	53.2	65.2	68.7

increased, although a more distinct reduction in redundancy was obtained over the various recovery probabilities. [2]

4.5 Physical Robot Experimental Results

As we can not get signal intensity from physical epuck robots, we simulate the fiducial algorithm by the distance of each pair of robots in the communication range. In other words, when two or more epucks in the communication range, by using the dispersion method each one can change its heading to a quadrant where less epucks plan to cover in

[2]Additional experiments of our system have shown that linearly increasing the size of the environment while keeping other parameters fixed increases the coverage time linearly, and, introducing obstacles in the environment improves coverage time and redundancy. Also, weights $\lambda_1 = 0.1$, $\lambda_2 = -1.0$ and $\lambda_3 = 1.0$, in algorithm *selectAction* (Figure 4.3) performs better than other weights in terms of coverage time and redundancy.

Strategies	Mean Rank
Braitenberg Motion	2.00
Binary Node Counting	7.60
Incremental Node Counting	11.60
Transient Node Counting	9.60
Incremental Node Counting with trail	12.00
Transient Note Counting with trail	14.00
	Test Score
Chi-Square	9.561
df	5
p-value	0.089

Table 4.2: The Kruskal-Wallis statistics tests of the percentage of environment covered with 5 robots in a 1000×1000 cm^2 square environment for different node counting strategies over 10,000 simulation time measured in ticks.

that quadrant. We observe that our transient node counting algorithm performs a little bit better by about 3-5 percent than the fiducial algorithm. Because with our algorithm epucks consider and share local coverage maps of each other, as well as simply dispersion.

We analysis our test data by Kruskal-Wallis statistics tests in SPSS, as our experiments were taken randomly and independently of each other. H_0: There is no difference of the percentage of the environment with different node counting strategies; H_a: on the contrary, not all of the results are the same. In Table 4.2, since p-value $= 0.089 \leq 0.1 = \alpha$, we reject the null hypothesis. Thus, at $\alpha = 0.1$ level of significance, there exists enough evidence to conclude that there is a difference among the six different node counting strategies based on the test scores.

We analysis our test data by Kruskal-Wallis statistics tests in SPSS, as our experiments were taken randomly and independently of each other. H_0: There is no difference of the

Number of Robots	Mean Rank
5 robots	2.00
10 robots	5.30
15 robots	8.00
20 robots	11.60
40 robots	15.00
60 robots	16.60
80 robots	18.30
	Test Score
Chi-Square	17.535
df	6
p-value	0.008

Table 4.3: The Kruskal-Wallis statistics tests of the percentage of environment covered with different number of robots in a 1000×1000 cm^2 square environment for $2,500$ time steps.

percentage of the environment when the number of robots changes; H_a: on the contrary, not all of the results are the same. In Table 4.3, since p-value $= 0.008 \leq 0.01 = \alpha$, we reject the null hypothesis. Thus, at $\alpha = 0.01$ level of significance, there exists enough evidence to conclude that there is a significant difference among the percentage of environment covered with different number of robots based on the test scores.

We analysis our test data by Kruskal-Wallis statistics tests in SPSS, as our experiments were taken randomly and independently of each other. H_0: There is no difference of the number of time steps required to complete the coverage of the environment when the number of robots changes; H_a: on the contrary, not all of the results are the same. In Table 4.4, since p-value $= 0.007 \leq 0.01 = \alpha$, we reject the null hypothesis. Thus, at $\alpha = 0.01$ level of significance, there exists enough evidence to conclude that there is a significant difference among the number of time steps required to complete the coverage of the environment

Number of Robots	Mean Rank
5 robots	19.60
10 robots	16.60
15 robots	14.00
20 robots	10.00
40 robots	8.00
60 robots	6.60
80 robots	2.00
	Test Score
Chi-Square	17.610
df	6
p-value	0.007

Table 4.4: The Kruskal-Wallis statistics tests of the number of time steps required to complete the coverage of a 1000×1000 cm^2 square environment for different numbers or robots.

Coverage Map Radii	Mean Rank
50 cm	2.00
70 cm	6.00
110 cm	7.00
	Test Score
Chi-Square	5.600
df	2
p-value	0.061

Table 4.5: The Kruskal-Wallis statistics tests of the percentage of environment covered with different coverage map radii of 5 robots after 10,000 time steps in 1000×1000 cm^2 square environment.

with different number of robots based on the test scores.

We analysis our test data by Kruskal-Wallis statistics tests in SPSS, as our experiments were taken randomly and independently of each other. H_0: There is no difference of the percentage of environment when the coverage map radii of robots changes; H_a: on the contrary, not all of the results are the same. In Table 4.5, since p-value $= 0.061 \leq 0.1 = \alpha$, we reject the null hypothesis. Thus, at $\alpha = 0.1$ level of significance, there exists enough

Communication Range	Mean Rank
110 cm	13.20
210 cm	14.70
410 cm	18.60
	Test Score
Chi-Square	2.005
df	2
p-value	0.367

Table 4.6: The Kruskal-Wallis statistics tests of the percentage of environment covered after 2500 time steps when the communication range ξ of robots changes, χ=50 cm, number of robots=5, and environment size=1000×1000 cm^2.

evidence to conclude that there is a difference among the three different coverage map radii based on the test scores.

We analysis our test data by Kruskal-Wallis statistics tests in SPSS, as our experiments were taken randomly and independently of each other. H_0: There is no difference of the percentage of environment when the communication range ξ of robots changes; H_a: on the contrary, not all of the results are the same. In Table 4.6, since p-value $= 0.367 > 0.1 = \alpha$, we accept the null hypothesis. Thus, at $\alpha = 0.1$ level of significance, there exists enough evidence to conclude that there is not much difference among the three different communication ranges based on the test scores.

Chapter 5

MULTI-ROBOT COVERAGE: TEAM-BASED ROBOTS USING FLOCKING

In Chapter 4, we proposed a swarm based individual robot coordination for area coverage. Although robots can share their recent coverage histories with each other, the information from the environment is still very limited. In this chapter, we propose to organize robot into small teams while performing coverage. The motivation behind our team-based coverage algorithm is to improve the coverage of the environment in terms of time and redundancy by using robot teams organized in a coverage-maximizing formation. We will introduce a team flocking method for distributed area coverage in this chapter. We also quantify the effect of various parameters of the system such as the size of the robot teams, as well as environment related features like the size and shape of the environment and the presence of obstacles and walls on the performance of the area coverage operation. Results show the flocking based mechanism can improve the performance of the system.

5.1 Single Team Flocking

In our single-team flocking technique, the leader robot communicates the direction it is moving as the prescribed direction of motion for each follower robot in the team. Each follower robot then attempts to move in the prescribed direction. If any follower robot fails to move in this direction, it stops and communicates to the leader robot that its motion failed. Depending on the position of the follower robot in the team and its attempted direction of motion, the team leader then selects a new direction of motion that would possibly allow the affected follower robot to avoid the obstruction in its path. The team leader then broadcasts this newly selected direction as the prescribed direction for the next time step to all the follower robots in the team. In some scenarios, due to communication noise, a follower robot might fail to receive the communication containing the prescribed direction of motion from the leader robot. Then the follower robot just continues to move in the same direction it moved during its previous time step. The pseudo-code algorithm used by a team of robots in the leader-referenced motion strategy is described in Figure 5.1.

```
function LeaderReferencedMotion {
  $ac^{t-1}$ ← action(movement direction) performed during
        last time step $t - 1$;
  if (I am not the leader)
    $Ac_{leader}$ ← movement direction received from leader;
    if ($Ac_{leader} \neq$ NULL)
      $ac^t \leftarrow Ac_{leader}$;
    else $ac^t \leftarrow ac^{t-1}$
    execute $ac^t$;
    if ($ac^t$ fails)
      execute STOP;
    sendMessage ($MotionFailed$, leader);
  else //I am the leader
    $ac^t \leftarrow ac^{t-1}$
    execute $ac^t$;
    if ($ac^t$ fails)
      execute STOP;
      broadcastMessage ($electNewLeader$);
    if (received $MotionFailed$ message from follower robot)
      newAction ← An new direction of motion that will
        allow the follower robot to avoid the obstacle
        in the next step
      $ac^t$ ← newAction;
      broadcastMessage($nextAction$, $ac^t$);
}
```

Figure 5.1: Algorithm used by a robot to realize the leader-referenced formation control.

5.2 Single-Team Coverage Technique

A robot team using the formation control mechanism described above uses a very simple, memoryless technique to cover the environment. In this technique, each leader robot uses a random-walk coverage strategy to cover the environment - the leader (and the following team members) follows a linear motion until it encounters an obstacle on its forward-facing distance sensors. The leader robot then uses the team reconfiguration mechanism to read-just the formation of the team and a new leader is selected to continue the coverage. To keep the coverage technique very simple, we have made it memoryless, that is, each robot does not keep any record of the regions it has already covered.

5.2.1 Formation Maintenance

When a team of robots moves in formation, the noise in the wheel rotation and sensor readings can cause one or more of the team members to lose their desired positions which destroys the configuration in the team. To address this problem, we have used a formation maintenance protocol that is invoved at intervals of t_{FM} timesteps by a team to enable each follower robot to retain its position in the team. In this protocol, the team leader first calculates the desired positions (DP_i) of every follower robot i relative to its own position. Each follower robot also sends its actual relative position AP_i to the team leader. The team leader then compares DP_i with the actual position AP_i of each follower robot i. If the distance between DP_i and AP_i is greater than δ, for any follower robot i, the team leader sends a message to robot i with its desired position DP_i, and, sends a message to all other team members to stop. After robot i has reached DP_i, it sends a message to the team leader. The team leader then sends a message to all the follower robots to continue moving in the previous direction before the formation maintenance protocol was invoked. The calculation of DP_i for follower robot i is given below. In these formulae, the actual position AP_i is represented by (x_0, y_0), the desired position DP_i is represented by (x_i, y_i), a is the direction of motion of the team, u is the angular separtion in the team and d_{sep} is the linear separation between adjacent robots in a team.

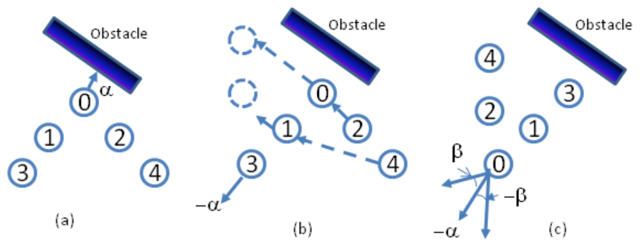


Figure 5.2: (a) The leader robot (id=0) in a team of five robots encounters an obstacle. (b) A new leader is selected (id=3) and the old leader robot (id=0) calculates the positions for every robot in the new configuration under the new leader. (c) Each robot assumes its new position and the new leader robot selects its heading from randomly between $-\alpha \pm \beta$.

(a) Case 1: $0 \leq a < \pi$

$$x_i = \begin{cases} x_0 - \frac{i}{2} \times d_{sep} \times \cos(a - u) & \text{if } i \text{ is odd} \\ x_0 + \frac{i}{2} \times d_{sep} \times \cos(a - u) & \text{if } i \text{ is even} \end{cases}$$

$y_i = y_0 - \frac{i}{2} \times d_{sep} \times \sin(a - u)$

(b) Case 2: $\pi < a \leq 2\pi$

$$x_i = \begin{cases} x_0 + \frac{i}{2} \times d_{sep} \times \cos(a - u) & \text{if } i \text{ is odd} \\ x_0 - \frac{i}{2} \times d_{sep} \times \cos(a - u) & \text{if } i \text{ is even} \end{cases}$$

$y_i = y_0 + \frac{i}{2} \times d_{sep} \times \sin(a - u)$

5.2.2 Team Reconfiguration

A leader robot that encounters an obstacle ahead of it will fail to move in its direction of motion. When the team leader encounters an obstacle it stops and communicates to the follower robots to stop moving. Then, the leader robot selects a new leader and directs the follower robots to change the configuration of the team to a new configuration so that the team can continue covering the environment. To enable team reconfiguration, the leader robot uses the team position identifiers. The positional adjustments of the robots after its team leader encounters an obstacle is given by:

$$p_i = \begin{cases} n & \text{if i = 0} \\ n - i - 1 & \text{if i is odd number} \\ n - i + 1 & \text{if i is even number} \end{cases}$$

where n is the number of robots in team minus 1, i is the old position in the team, and p_i is the new position identifier in the team. After getting in the new formation, the new leader selects a new heading given by a random value in the range of $\pm\beta$ from the reverse direction of the old heading. This results in the team performing a $\pi \pm \beta$ turn to avoid the obstacle encountered during its movement. In general, if the obstacle is encountered by the old leader using its forward-facing distance sensors on its righthand (lefthand) side going clockwise from current heading, then the follower robot that is farthest from the leader on its righthand (lefthand) side is selected as the new leader. If the old leader robot approaches the obstacle orthogonally resulting in comparable readings on both pairs of the forward-facing (left and right) distance sensors, then one of the two follower robots that is farthest from the old leader robot is selected at random to become the new leader. A scenario illustrating the team reconfiguration is shown in Figure 5.2.

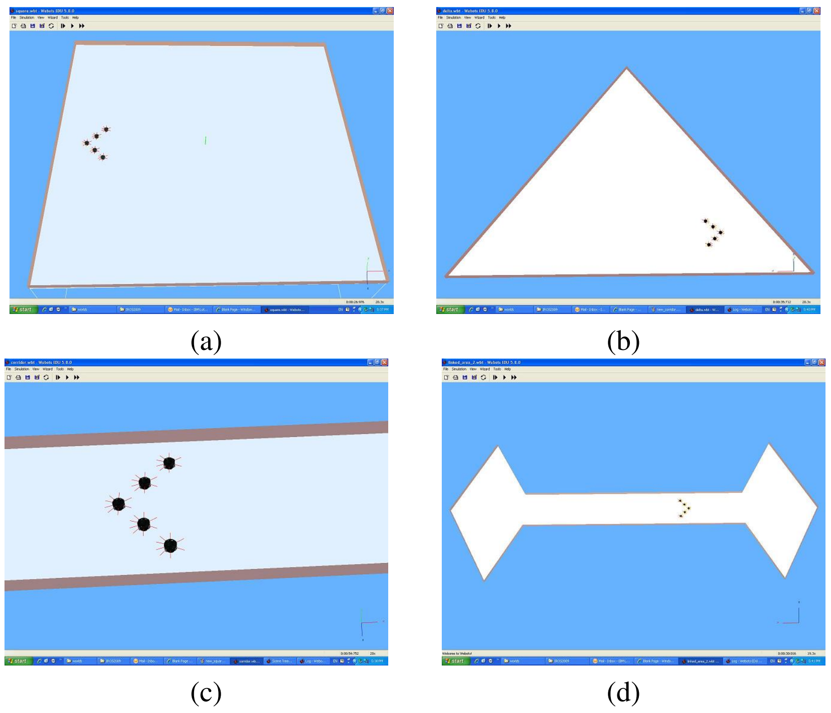


Figure 5.3: Four different environments of five robots with V shape in Webots simulation platform: (a) a square, (b) a triangle, (c) a corridor, (d) two diamonds connected by a corridor.

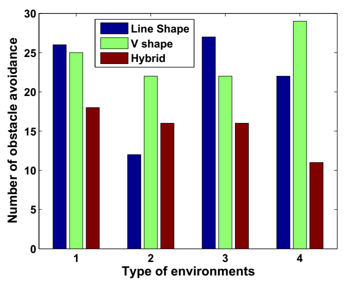
5.3 Experiment in Webots Simulator

To test the performance of our single team flocking technique, we used four different environments for our simulations - square, triangle, corridor, and two diamonds connected with a narrow corridor, as shown in Figure 5.3 . All these environments have the same area of 25 meter^2. In each environment, we placed 3, 5 or 10 robots that were organized into 1 or 2 teams. By combining these parameters, we tested our coverage strategies in 16 different experiment scenarios. Each experiment scenario was allowed to run over a duration of 2 hours. Each result was averaged over 10 simulation runs. For evaluating our results, we have compared the performance of four different coverage strategies: **(1) Individual**

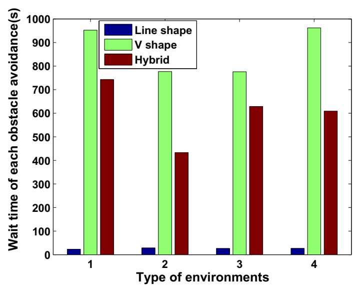
Table 5.1: Percentage of of a $5 \times 5m^2$ environment covered for different environment shapes and different coverage strategies.

	3 robots in 1 team			
Coverage Strategy	Square	Triangle	Corridor	Two diamonds
individual coor.	64.74	61.98	45.05	39.9
line flocking	51.05	48	45.76	45.88
’V’ flocking	40.33	41.92	46.06	31.33
hybrid flocking	42.96	42.48	47.45	41.46
	5 robots in 1 team			
individual coor.	83.36	78.85	55.16	52.05
line flocking	58.17	56.14	34.92	44.09
’V’ flocking	62.83	55.44	40.37	40.26
hybrid flocking	59.91	57.41	37.08	43.42
	10 robots in 2 teams of 5 robots each)			
individual coor.	93.27	84.53	73.37	75.88
line flocking	82.8	78.2	66.57	74.38
’V’ flocking	76.6	81.23	57.38	73.63
hybrid flocking	81.77	79.3	58.66	62.17

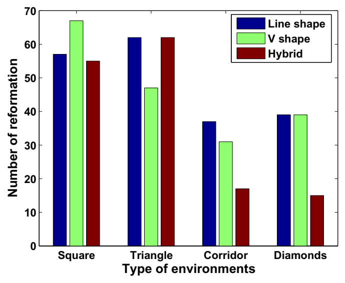
coordinated or formation-less strategy. This strategy is similar to the node counting and pheromone-based coverage strategy described in [37]. The individually coordinated coverage strategy is used by independently moving robots (that is, robots not moving as a team in formation). This coverage strategy involves storage and computation overhead as each robot records a finite history of the region covered by it and exchanges and fuses this information with robots within its communication range. We have used a slightly modified version of this algorithm suitable for e-puck robots described in [11]. **(2) Line-shape Formation**. In this coverage strategy, robots form teams with a straight line configuration. **(3) ’V’ shape Formation**. Here, robots form a V-shaped configuration using the techniques described in Section 5.1, and, **(4) Hybrid Formation.** In this technique, robots dynami-



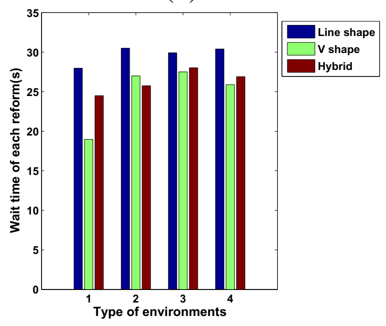
(a)



(b)



(c)



(d)

Figure 5.4: (a) Number of times an obstacle is encountered by the team, (b) Average time required for team reconfiguration after encountering an obstacle, (c) Number of times the team has to perform formation maintenance while not encountering an obstacle, and, (d) Average time required for formation maintenance.

cally alternate between the line formation and V-formation while regaining configuration after the team's shape transformation. To compare the relative performance of these strategies for the different environments and different numbers of robots, each simulation was allowed to run for a period of almost 2 hours that corresponds to the maximum battery life of an e-puck robot. Our objective in the experiments we performed was to investigate how much of the environment could be covered within the battery life of each robot-team. Although not shown here, each environment was completely covered when the simulations were allowed to run for 2.5 hours (with 10 robots) to 4 hours (with 3 robots). For each simulation, all robots were deployed from the center of each environment.

The results of our simulations for the area coverage of the different scenarios is summarized in Table 5.1. Each value in Table 5.1 represents the percentage of the environment that was covered by the robots at the end of 2 hours of simulation time. We observe that the three flocking-based techniques perform comparably with the individual coordinated strategy. The extra overhead of the flocking-based techniques can be attributed to the extra time required by the flocking-based methods for formation maintenance and for reconfiguring the team after encountering an obstacle. However, as we increase the number of robots, the coverage achieved using the three flocking-based techniques improves with respect to the individually coordinated strategies. We also observe that for more complex environments such as the corridor and the non-convex environment with two diamond-shapes connected by a corridor, the line formation obtains the best coverage among the flocking-based coverage techniques, followed by the hybrid formation. This can be attributed to the fact that in a line-formation, the robot teams are able to traverse narrow passages such as corridors more efficiently. On the other hand, in a V-formation the angular dispersion of $2 \times u$ between the two sets of follower robots impedes the movement of the team in narrow spaces. Finally, because in the hybrid formation the robots alternate between a line and V-shape formation, therefore, the hybrid formation achieves a coverage between the line and V-shape formations. We also observe that when we increase the number of teams from one team of 5 robots to two teams of five robots the coverage does not improve linearly. This is because the robots cannot remember the regions covered by themselves or by other teams in the past. Therefore, they end up re-covering regions already covered in the past.

To further analyze the performance of the flocking-based techniques we have compared the average time spent by the different formations in the flocking-based techniques for reconfiguring after avoiding an obstacle and for performing formation maintenance. The results of these experiments are reported for one 5-robot team for the different environments in Figures 5.4 (a)-(d). In Figure 5.4(a), we observe that the number of obstacles encountered by the robots using different formation control strategies depends on the shape of the environment. However, as shown in Figure 5.4(b), the line-formation requires very little time for reconfiguration because of its simpler configuration shape. On the other hand, the V-shape requires the longest reconfiguration time to accurately determine the positions of the different followers robots along the two arms of the V-shape. Finally, the hybrid formation requires slightly lesser reconfiguration time than the V-shape because sometimes the obstacle might be encountered when the robots are in a line formation. Figures 5.4(c) and (d), show the corresponding number of formation maintenance operations and the average time required to perform a formation maintenance operation. The formation maintenance operations reported here include both the selection of a new action by the leader to alleviate a failed action by a follower robot as well as the position adjustment operations performed by the follower robots described in Section 5.1. We observe in Figure 5.4(c) that the number of reformations for the square and triangle environments is larger. This can be attributed to the fact that the free space in these two environments is larger and the robot team can travel in the same direction without encountering any obstacle. During continuous motion in the same direction, the follower robots get out of their desired positions required for the configuration due to noise in the wheel motion. Therefore, more time the maintenance formation is invoked more often in these two environments. However, in the corridor and diamonds environment, the robots are in a narrow passage and they frequently encounter obstacles which causes them to adjust their positions through team reconfiguration instead of performing formation maintenance. The average time required by the line-shape formation for performing formation maintenance is also the highest, as shown in Figure 5.4(d). This can be attributed to the fact that when robots use the line formation they are moving in a single flank. Therefore, multiple robots are likely to require formation maintenance when one of the follower robots fails to perform an action. Consequently, the leader robot expends more time to find an action that will alleviate the failure to perform an action for multiple follower robots.

Chapter 6

MULTI-ROBOT COVERAGE: COALITION GAME-BASED ROBOT TEAM FORMATION

6.1 System Modeling

A robot uses the coverage information from its coverage map to calculate its coverage capability. The coverage capability of robot i is denoted as $C_i = a \times \theta_i - b \times \eta_i + C_0$. θ_i is the coverage rate of robot i in recent time period T, and η_i is the redundancy rate. The coverage rate is $\theta_i = \frac{V_{cov}}{V_{map}}$, where V_{cov} is the area robot i covered in the last T timesteps, and V_{map} is the area of its whole coverage map. For example, if robot i can record a $100 \times 100cm^2$

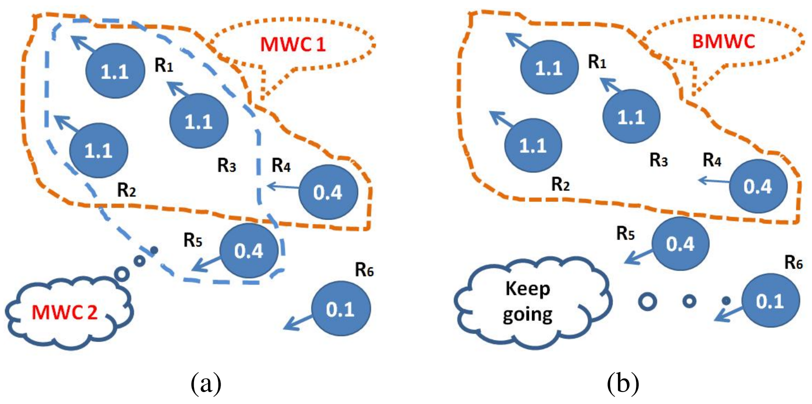


Figure 6.1: (a) 6 robots calculate two minimal winning coalitions in the communication range , (b) 4 robots form a best minimal winning coalition and head to a new team direction.

local area in its coverage map, and in T timesteps it has covered a $50 \times 50cm^2$ area, we can calculate the value θ_i is 0.25. The redundancy rate is $\eta_i = \frac{V_{red}}{V_{map}}$, where V_{red} is the area of the revisited region within the coverage map, and V_{map} is the area of the whole coverage map. a and b are normalizing constants, and C_0 is the initial value of coverage ability. Thus, we can get C_i in the range $[0, 1.96]$, when $a = 2$, $b = 1$, $C_0 = -0.04$. The coverage capability C_i of robot i is upper and lower bounded.

6.1.1 Best Coalition Formation

Coalition structures represent the different combinations of teams which a robot can form within other robots within its communication range. Every time, there are two or more teams of robots in each other's communication range, we can search the coalition structures in the solution space. Our goal is to find an efficient partition, which increases the payoff of the robot team without reducing the payoff of any single robot. For example, suppose there are 4 robots that are within each other's communication range. These robots can form $2^4 = 16$ possible coalitions. To get a winning coalition, we can set a robot's own weight as a linear function of the robot coverage capability defined by $w_i = \alpha \times C_i + \beta$, where C_i is the robot's coverage capability introduced in former section, and α and β are the adjustment constants to make w_i remain within a certain interval. For example, $w_i \in$ [0,1] with $\alpha = \frac{1}{2}$ and $\beta = 0.02$. The weight w_i associated with a robot gets updated with its coverage map and coverage capability after every T timesteps. We set the winning threshold of the WVG to $q = c \times n$, where n is the number of robots and c is an adjustment constant. In each round of the voting game, the weight w_i of each robot and the quota q are both fixed values. For example, we consider a weighted voting game with n $=$ 4 robots and individual robot weights at $w_1 = 1.3, w_2 = 1.1, w_3 = 1.2, w_4 = 0.2$, in a concise form $G = \{3 : 1.3, 1.1, 1.2, 0.2\}$. We can get the minimal winning coalition $\{r_1, r_2, r_3\}$, when we set $q =$ 3, with $c = 0.75$. A problem with applying the winning condition from general WVGs to our context is that sometimes there could be more than one minimal winning coalition. Suppose there are 6 robots with the weight set {1.1, 1.1, 1.1, 0.4, 0.4, 0.1}, and quota $q =$ 3.5, as shown in Figure 6.1 (a). We can get two winning coalition $\{r_1, r_2, r_3, r_4\}$ and $\{r_1, r_2, r_3, r_5\}$.

As the minimal winning coalition is not unique, we design a function $\xi = argmin_{S \in I}(g \times \prod_{k \in S} x_k + \frac{\sum_{i,j \in S}(e \times d_{i,j} + f \times \varphi_{i,j})}{|S|})$. We call the coalition which has the optimized value ξ the best minimal winning coalition(BMWC). The value x_k is a prime number, which can be a unique number such as the robot's ID. $d_{i,j}$ is the distance between two robots i and j. $\varphi_{i,j}$

is the angle between the two robots i and j. e , f ,and g are adjustment constants, which make $g \times \prod_{k \in S} x_k \ll \frac{\sum_{i,j \in S}(e \times d_{i,j} + f \times \varphi_{i,j})}{|S|}$. I is the set of minimal winning coalitions, and S is a minimal winning coalition in I. The definition of ξ considers the angle and distance between robots in the minimal winning coalitions as well as the intrinsic value of, such as an id of each robot to come up with a unique value for each coalition. An example of determining the BMWC is shown in Figure 6.1 (b). If there is no winning coalition, we just keep the status of each robot unchanged. As veto players are present in all winning coalitions, we can check the exist of veto players of a voting game to determine the non-emptiness of the core. Also we can find that BMWCs have the unique and stable properties of the voting games.

As the theorem in Shoham's book [58], " In a simple game the core is empty iff there is no veto player. If there are veto players, the core consists of all payoff vectors in which the nonveto players get zero.", we know that to determine the existence of the nonempty core is based on the existence of the veto players in a simple game. A weighted voting game is also a simple game. Thus, if we find a winning coalition with veto players in a weighted voting game, we find a nonempty core. As the minimal winning coalitions are in the set of all winning coalitions, if a minimal winning coalition is formed by veto players, it is in the nonempty core. The best minimal winning coalition is in the nonempty core as well, because it is one of the minimal winning coalitions with veto players.

However, we can not guarantee the core is always nonempty in a weighted voting game. In other words, some times there does not exist veto players in a weighted voting game.

In such a case, the core of the game is empty, which means no new strategies for a robot team or teams. This is also useful in the real multi-robot coverage case. For example, if the weights of all the team members are the same (in the real cases, there is a bit difference of each robot's weight) and the quota value is less than 1, there are no veto players. The leader robot does not need to calculate the best minimal winning coalition of the team.

To guarantee there exists veto players in a weighted voting game, the quota value need to be set in a proper range. The upper bound of the quota value should be equal to the sum of the weights W. The lower bound can be found, when there is only one player as the veto player. Based on the definition of veto players, we get the lower bound of quota greater than $max_{i \in N}(w_i)$. Thus, to guarantee there exists nonempty core of the weighted voting game, the quote value should be set in the interval $(W - max_{i \in N}(w_i), W]$. Also, if $q > (1 - \frac{1}{|N|})W$, then we are guarantee to find the veto player set, when all player weights are equal.

Theorem 6.1. *The best minimal winning coalition is unique.*

Proof. $d_{i,j}$ is less than the length L of the communication range and bigger than 0. $\varphi_{i,j}$ is in the interval $[0, 2\pi]$. $\frac{\sum_{i,j \in S}(e \times d_{i,j} + f \times \varphi_{i,j})}{|S|}$ is in the interval $(0, e \times L + 2\pi \times f]$. Suppose, two winning coalitions S and $S^{'}$ have the same value $\xi = \xi^{'}$, where exists one robot $n \in S^{'}$ and n is not contained by S. As both coalitions are minimal winning coalitions, they have the same number of elements $|S| = |S^{'}|$. From the definition of ξ, we have $\frac{\sum_{i,j \in S}(e \times d_{i,j} + f \times \varphi_{i,j})}{|S|})$ = $\frac{\sum_{i,j \in S^{'}}(e \times d_{i,j} + f \times \varphi_{i,j})}{|S^{'}|})$, which means the geometric shape of the two coalitions are same.

In other words, they have the same perimeter and the sum of all angles. Thus, we know $\prod_{k\in S} x_k \neq \prod_{k\in S'} x_k$. However, as $|S| = |S^{'}|$, if we get $\xi = \xi^{'}$, x_k can not be prime numbers. It is contradictive for the define of the x_k. Therefore, S and $S^{'}$ are the same coalition, which infers that the best minimal winning coalition is unique. □

Theorem 6.2. *The ξ_b value of best minimal winning coalition is smaller than any ξ of minimal winning coalitions.*

Proof. Suppose, there exists a value ξ_{small} which is the minimal ξ of minimal winning coalitions. There also exists a positive real number ε, where $\varepsilon = \xi_b - \xi_{small}$. Based on the BMWC, $\xi_b = argmin_{S\in I}(g \times \prod_{k\in S} x_k + \frac{\sum_{i,j\in S}(e\times d_{i,j}+f\times\varphi_{i,j})}{|S|})$, there exists a distance between two robots or the robot id that is a negative real number. However, it is not true for physical robots. Thus, ξ_b is the smallest value of all ξ value of minimal winning coalitions. □

6.1.2 The other coalition formation algorithm

Vig and Adams [66] have proposed a multi-robot coalition formation algorithm to deal with actual multi-robot systems. He extended the Shehory and Kraus' [57] algorithm which is used for task allocation using software agent coalition formation. They declared that agents could freely change combinations in different coalitions. However, it is infeasible for physical robots. They represented this constraint as a Constraint Satisfaction Problem (CSP). For example, the sensors and actuators must reside on the same robot or on different robots. To check the feasibility of each coalition, they designed the constraint graph by using arc-consistency. Moreover, they considered the balance property in the task format and contrived a formula to calculate the Balance Coefficient (BC). Although their algorithm does not improve the time complexity of Shehory and Kraus' algorithm, it does give a real and possible solution for the multi-robot task format.

As this paper focuses on the multi-robot coverage domain, we have fulfilled Vig and Adams's multi-robot coalition formation algorithm in Webots robot simulator which is the same simulation platform for our DYN-REFORM Algorithm. The define of task here is the coverage capability of the robot team, which can be set as a threshold. In stead of multiple task allocation, this is a single task team formation problem. We have used single robot coverage capability proposed in the former section as the inputs of both algorithms. Most input coverage capabilities of robots have high values as moving in a field without obstacles. For the sake of wheel slide noises, these values have $5 - 10\%$ deviation. A little portion of the input coverage capabilities of robots have low values, for they can be stuck by walls or obstacles. Outputs are the coalitions of robots which have the coverage capability to fulfill the team coverage task.

We ran the algorithms respectively 10 times using 5 to 30 robots and the value Q_f from 0.5 to 0.9. As shown in Figure 6.2 with different number of robots and $Q_f = 0.7$ (changing the value of Q_f, the result curves are almost the same) in the logarithm scale, the run time of both algorithms are less than one millisecond and close to each other with number of robot less than 10. However, as the number of robots increasing, the run time of multi-robot

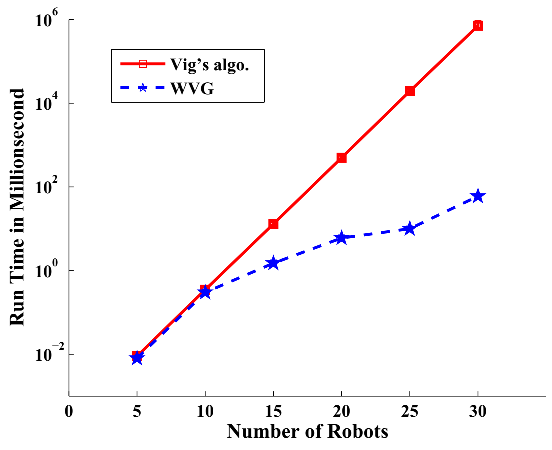


Figure 6.2: Run time in millisecond (in the logarithm scale) with different number of robots and $Q_f = 0.7$ by using Vig's Multi-Robot Coalition Formation Algorithm and Team Formation using DYN-REFORM Algorithm.

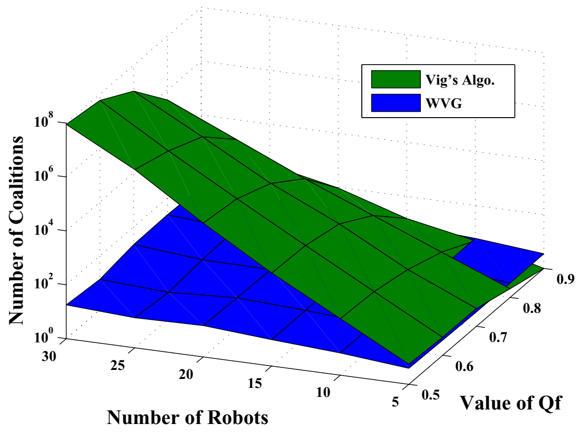


Figure 6.3: Number of coalitions (in the logarithm scale) with different number of robots and different values of Q_f by using Vig's Multi-Robot Coalition Formation Algorithm and Team Formation using DYN-REFORM Algorithm.

coalition formation algorithm increases from a couple of milliseconds to about 10 minutes. Although we can put bounds to the task value and the number of robots, the multi-robot coalition formation algorithm runs in exponential time. While, the time complexity of our DYN-REFORM Algorithm is square complex of input number n. With less than 30 robots, we can find the BMWC within 100 milliseconds as shown in Figure 6.2. Also, we found that for both algorithms, we change the value of Q_f, the run times are almost the same. In other words, the run time of neither algorithm is related with the value of Q_f, but related with the number of robots.

As shown in Figure 6.3, the number of coalitions in the logarithm scale, the possible coalitions of Vig's multi-robot coalition formation algorithm increase exponentially in the memory by raising the number of robots. Also, in Table 6.1, with the same number of robots, by increasing the value of the Q_f, the number of coalitions decreases for Vig's algorithm, because smaller value of quota or task value can achieve more coalitions in the solution set. On the contrary, the number of minimal winning coalitions of DYN-REFORM Algorithm increase slightly in the logarithm scale, as shown in Figure 6.3. Also, as shown in Table 6.1, with the same number of robots, although we change the values of Q_f, the number of coalitions are changed very tiny for our Team Formation using DYN-REFORM Algorithm. In stead of storing all the possible coalitions, our method firstly finds the veto player set which includes most robots with normal coverage capabilities and creates the MWC set based on the veto player set with small portion of the left robots. For example, suppose there are three robots with resource set $\{3, 3, 5\}$ for robot 1, robot

Table 6.1: Average and standard deviation number of coalitions with different number of robots and different values of Q_f by using Vig's Multi-Robot Coalition Formation Algorithm and Team Formation using DYN-REFORM Algorithm.

		Value of Q_f									
		Vig's Algo.					DYN-REFORM Algo.				
		0.5	0.6	0.7	0.8	0.9	0.5	0.6	0.7	0.8	0.9
5 robots	Avg.	5.6	4.7	4.6	2.1	1	3.9	3.6	3.4	3.4	3.4
	STDEV	2.2	1.1	0.47	0.63	0	0.57	0.52	0.52	0.69	0.69
10 robots	Avg.	124.7	106.4	59	22	4.9	6.7	6.6	6	5.9	5.5
	STDEV	2.1	4.4	2.3	1.6	1.2	1.6	1.8	2.3	1.9	2.4
15 robots	Avg.	2844.5	2500.2	1326.7	230.7	15	10.2	9.7	9.5	7.6	8.3
	STDEV	896	35.8	28.7	10.9	0	2.4	2.9	2.7	2.2	3.1
20 robots	Avg.	92361.7	62636.6	19439.4	2457.6	93.1	14.5	13.1	12.2	10.4	13.9
	STDEV	949	563	308	49.9	8.5	4.5	4.7	5	4.7	4.5
25 robots	Avg.	4111592	1716496	479418	32851	307	13	9.6	13.1	15.3	17.2
	STDEV	98424	29451	931	224	2.1	9	2.8	6	5.1	6.5
30 robots	Avg.	89210570	58273576	10860190	428995	2442	17.9	13.4	21.9	23.7	21.1
	STDEV	1809877	858013	35727	4838	75	10.3	8.5	5.9	4.3	6.1

2, robot 3 respectively, and the value of task is 6. By using Vig's multi-robot coalition formation algorithm, there are 4 solution sets: $\{3, 3, 5\}$, $\{3, 3, \emptyset\}$, $\{3, \emptyset, 5\}$, $\{\emptyset, 3, 5\}$. By comparing with the value of the Balance Coefficient, the finial coalition is $\{3, 3, \emptyset\}$ which BC is 1 while others are less than 1. The MWC sets of the DYN-REFORM Algorithm are: $\{3, \emptyset, 5\}$, $\{\emptyset, 3, 5\}$. By comparing the intrinsic factors of each robot, such as location or robot ID, the finial coalition is $\{3, \emptyset, 5\}$. Thus, this example shows the possible solution sets of the DYN-REFORM Algorithm is half of the sets of multi-robot coalition formation algorithm. As shown in Figure 6.3, by increasing the number of the robots, the solution sets of the multi-robot coalition formation algorithm exponentially larger than the sets of the DYN-REFORM Algorithm.

Finally, we quantify the performance of the two algorithms combined with the flocking control model with $5, 10, 15$ and 20 robots within a 2×2 m^2 environment where 10% of the total area of the environment is occupied by obstacles. Each experiment was run

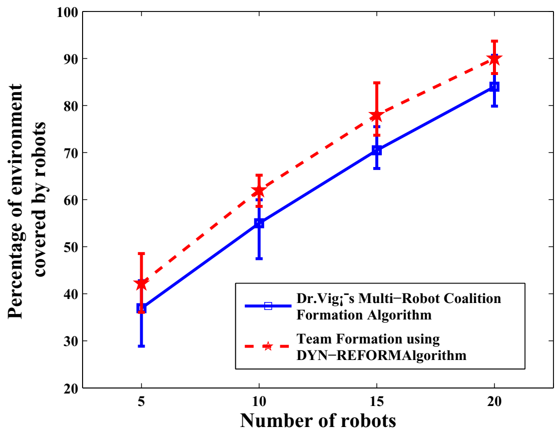


Figure 6.4: Percentage of environment covered by different numbers of robot for using Dr.Vig's Multi-Robot Coalition Formation Algorithm and Team Formation using DYN-REFORM Algorithm. within a 2×2 m^2 environment where 10% of the total area of the environment space is occupied by obstacles. The run time is 30 minutes.

for a period of 30 minutes. We observe that The DYN-REFORM Algorithm gets more percentage of coverage than the multi-robot coalition formation algorithm. By calculating the value of balance coefficient, the multi-robot coalition formation algorithm generates the grand coalition with the higher BC value than the other coalitions. In other words, each time all the robots join into the same flocking team. Not surprisingly, the coverage of the fixed size flocking team can be improved by using DYN-REFORM Algorithm. Also, it is a shortcoming by using balance coefficient value to find the team with equal capability of each member for the single task allocation.

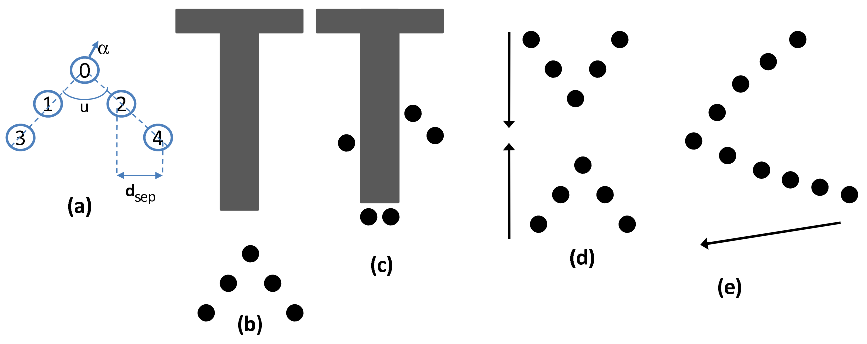


Figure 6.5: (a) A robot team showing the position identifiers of each robot. The angular separation in the team is u, the separation between adjacent robots is d_{sep} and α is the heading of the team. (b)-(c) A scenario where a single team in formation encounters a T-shaped obstacle and needs to split. (d)-(e) A scenario where two teams in close proximity of each other encounter each other and need to merge.

6.2 Dynamic Team Reconfiguration and Weighted Voting Games

We have used a flocking-based, leader-referenced formation control algorithm[52], to maintain a specific formation among the robots in a team while in motion, as shown in Figure 6.5(a) and described in [13]. Each robot in a team is given a local identifier with '0' as the leader robot's identifier and odd and even numbered identifiers for the follower robots on either side of the leader. The shape of the team can be controlled by varying the angle u to transform it, for example, from a wedge shape to a line shape. Each robot has a specified separation d_{sep} that it must maintain from its neighbors. The leader robot has a predetermined direction α that it wants to move the team in. At the end of each time step, the leader robot determines the position that it should reach at the end of the next time step so that it can continue its desired motion. The leader robot also calculates the positions for each of the follower robots for the next time step relative to their current positions, so that the formation of the team is maintained. The leader robot then communicates the desired position information to each of the follower robots. Finally, the leader and follower robots start moving towards their respective designated positions.

A principal problem with flocking-based formation control is that it can fail if the leader or a follower robot encounters an obstacle that impedes its motion to the desired position for the next time step. The problem of obstacle avoidance while moving in formation is accentuated if robots are not be able to perceive the obstacle boundary before planning their motion for the next time step, as shown in Figures 6.5(b) and (d). In such inefficient scenarios, it would be beneficial to reform the members of a team into new teams by splitting the original team, as shown in Figure 6.5(c), or, merge some or all of the members of two robot teams into a single team, as shown in Figure 6.5(e). We approach this problem of merging and splitting robot teams as the robot team reconfiguration problem. Reconfiguring a robot team can be viewed as a problem of finding the partition that is the best for all the participating robots. A single, hard-coded splitting or merging rule cannot be guaranteed to be the best partition in all scenarios, and, therefore, the partitioning of the robot teams has to be done dynamically. We have used concepts from the branch of micro-economics that deals with cooperative or coalitional games [43] to determine rules for solving the robot-team partition problem. Coalition games are particularly attractive for our problem because they can ensure that the solution is stable, or in other words, it is acceptable to all the participants. Additionally, the players, or robots in our case, do not have to be explicitly informed if a split or a merge is the best thing to do as the rules of the coalition game calculate the set of robots that are incentivized to remain together, based on the performance of each robot in the recent past. For our problem, we have used a suitable and succinct representation of a coalition game called a weighted voting game(WVG). The main parameters of a WVG are the following:

R	Set of players or robots interested in forming a coalition
$\mathcal{R}$	Set of possible partitions among the members of the set R, $\mathcal{R} = 2^R$. Each member of the set $\mathcal{R}$ is called a coalition of robots and denoted by $C_j : j = 1, \ldots, \mid \mathcal{R} \mid$
w_i	weight of robot $r \in R$ that is calculated from the past performance (e.g., distance and time over which a robot has not deviated from its designated position in a formation over some finite time window in the recent past)
Q	quota or threshold of the WVG. A coalition C_j of robots becomes a winning coalition if the sum of the weights of the players exceeds the quota,
v	value function that denotes whether a coalition $C \in \mathcal{R}$ is a winning coalition or not, i.e., $v(C) = 1$ if $\sum_{i \in C} w_i \geq Q$, and $v(C) = 0$, otherwise.

A few additional concepts in WVGs are useful for the formulation of our problem. A **veto player** is a player such that if the player is excluded from a coalition, that coalition cannot be a winning coalition anymore. As an example. consider a WVG with 4 players A, B, C and D with weights $4, 2, 1,$ and 1 respectively. For this WVG, let quota $Q = 5$, that is, any coalition must have a combined weight of at least 5 to be a winning coalition. The set

```
**function** FindVetoPlayers **returns** *set* $V$
**inputs:** *set* $R$, *double* $Q$, $w_i$ ;
**variables:** *double* $W$, $W_{-i}$;

$V = \{\emptyset\}$;
$W = \sum_{i \in R} w_i$;
for each $i \in R$
    $W_{-i} = \sum_{j \in R \setminus \{i\}} w_j$;
    if ($W_{-i} < Q$)
        $V = V \cup \{i\}$;
return $V$;
```

(a)

```
**function** BMWCHeuristics **returns** *Set* $BMWC$
**inputs:** *Set* $V$; *ArraySet* $MWC$;
**variables:***Set* S; *double[]* $\xi$;
$centroid \leftarrow \frac{\sum_{v \in V}(Location(P_v))}{|V|}$;
$avgBearing \leftarrow \frac{\sum_{v \in V}(Angle(P_v))}{|V|}$;
for $i = 1$ to $| MWC |$
    $S = MWC[i] \setminus V$;
    for each robot $r \in MWC[i]$
        $\xi[i] = (\sum_r Distance(r, centroid)+$
                $Angle(r, avgBearing) + ID(r))/(| S |)$;
$j = \arg\min_i \xi[1.. | MWC |]$;
return $MWC_j$;
```

(b)

Figure 6.6: (a) Algorithm to find veto players in a WVG. (b) Algorithm to find the best minimum winning coalition (BMWC) from a set of MWCs.

of winning coalitions for this WVG are $\{A, B\}, \{A, C\}, \{A, D\}, \{A, B, C\}, \{A, B, D\}$, $\{A, C, D\}$ and $\{A, B, C, D\}$. This makes A a veto player because it is present in all the winning coalitions. WVGs can have more than one veto player or no veto players. For example, if we change Q from 5 to 7 both A and B become veto players, while if we change Q from 5 to 2, none of the players is a veto player anymore. We use the notation V to denote the set of veto players. The minimum set of players that can get enough combined weight among themselves to get to the quota is called the **minimum winning coalition(MWC)**. In the example above with $Q = 5$, there are three MWCs - $\{A, B\}, \{A, C\}, \{A, D\}$. MWCs are important because they imply that players in a MWC will not deviate from the coalition they are in because they cannot improve the benefit that they receive by forming a different coalition or a sub-coalition. This makes MWCs stable coalitions which are guaranteed not to break off after the coalition is formed.

The problem solved in a WVG is to identify a set of players that form a minimum winning coalition. This can be achieved in three steps as given below:

1. Identify all the veto players in R, because the veto players, if any, must be there in every winning coalition. The algorithm for identifying the veto players is based on the definition of veto players as players whose exclusion causes any coalition of the remaining players to lose. In other words, the combined weight of players excluding the veto player would fall below the quota. Our *FindVetoPlayers* algorithm shown in Figure 6.6(a), uses this concept to calculate set of veto players V. It has linear time complexity as it has to inspect each player from the set of players R for checking if it is a veto player or not.

2. Identify all the MWCs, by adding the minimum number of non-veto players to each set of veto players identified in step 1 above. Let w_v denote the combined weight of the veto players found in step 1. Then, $Q' = Q - w_v$, denotes the deficit in combined weight that should come from the non-veto players to reach the quota and form an MWC. Our objective then becomes to determine the set of players from the set $R \backslash V$ that can together reach a combined weight of Q'. This problem is a simplified version of the subset sum problem [17], with the relaxation that we need to find the smallest subset of players that is able to reach a combined weight of at least Q', (instead of exactly Q' of the subset sum problem). We have used a greedy method to solve this problem that has a quadratic time complexity in the worst case. The output of this algorithm is the set of MWCs.

3. Select one MWC called the best minimum winning coalition (BMWC) from the set of MWCs identified in step 2 above that appears to be most amenable to robot team formation. We measure the eligibility of an MWC towards forming a robot team using a heuristic-based fitness function ξ. For designing this heuristic function, we first consider the pose of the veto players because the veto players must be included in the final winning coalition. We calculate the centroid of the locations of the veto players and the average of the bearing between them. Then, for each of the non-veto players in each MWC, we calculate the distance and relative bearing with the centroid and average bearing of the veto players. If there are still any ties remaining, we use a prime number calculated from the robot id to make the value of ξ unique. The minimum of the ξ values for each MWC gives the best MWC. This algorithm is shown in Figure 6.6(b) and it has polynomial running time because it takes $O(\mid R\mid^2)$ steps in the worst case to calculate the value of ξ for each non-veto robot in each MWC. Integrating all the three steps, the worst case time complexity of running a WVG among R robots is $O(\mid R\mid) + O(\mid R\mid^2) + O(\mid R\mid^2) = O(\mid R\mid^2)$.

DYN-REFORM Algorithm. The DYN-REFORM algorithm realizes dynamic team reconfiguration by integrating the WVG algorithm and the flocking-based formation control algorithm. This integration is important and challenging because the WVG works only with a performance value or weight for each robot while the formation control algorithm relies on operational conditions such as presence of obstacles, proximity of robots, etc. Before running the WVG, the DYN-REFORM algorithm provides methods to determine

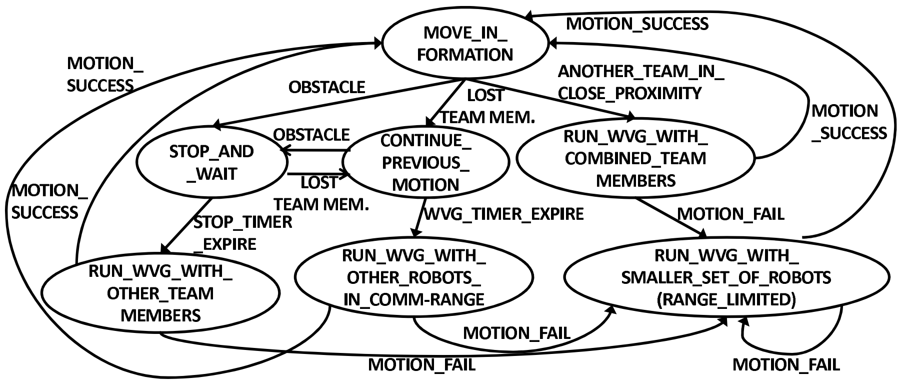

MOVE_IN_ FORMATION
MOTION_SUCCESS
ANOTHER_TEAM_IN_ CLOSE_PROXIMITY
MOTION_ SUCCESS
OBSTACLE
LOST TEAM MEM.
MOTION _SUCCESS
STOP_AND _WAIT
OBSTACLE
CONTINUE_ PREVIOUS_ MOTION
RUN_WVG_WITH_ COMBINED_TEAM MEMBERS
LOST TEAM MEM.
MOTION_ SUCCESS
STOP_TIMER _EXPIRE
WVG_TIMER_EXPIRE
MOTION_FAIL
RUN_WVG_WITH_ OTHER_TEAM MEMBERS
RUN_WVG_WITH_ OTHER_ROBOTS_ IN_COMM-RANGE
RUN_WVG_WITH_ SMALLER_SET_OF_ROBOTS (RANGE_LIMITED)
MOTION_FAIL
MOTION_FAIL
MOTION_FAIL

Figure 6.7: State transition diagram for a robot participating in the dynamic reconfiguration algorithm.

the set or subset of robots from a team that will participate in the WVG. After a coalition has been computed by the WVG, the DYN-REFORM algorithm provides mechanisms to handle situations where the computed coalition might not be realizable by the formation control algorithm (e.g., because of occlusions that prevent robots in a coalition from reaching their designated positions in the new team). It also specifies the operation of the robots that are excluded from the winning coalition after running the WVG.

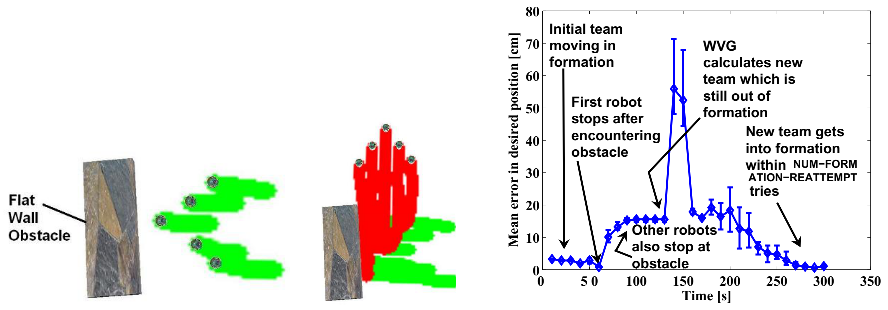
The operation of the DYN-REFORM algorithm is summarized by the state transition diagram of a robot shown in Figure 6.7. First, we consider the case when a team moving in formation encounters an obstacle and could possibly have to split. When any member of the team encounters an obstacle, it enters into a STOP_AND_WAIT state for a certain time period given by the value of a timer called STOP-TIMER. When this timer expires, the robot runs a WVG. Other team members that encountered an obstacle and stopped before the STOP-TIMER expires, possibly in the vicinity of the robot that is running in the WVG, are included as participants in the WVG. In certain scenarios, only some members of a robot team might encounter an obstacle while other robots in the same team do not. The team members that do not encounter the obstacle continue their previous motion (CONTINUE_PREVIOUS_MOTION state) by moving in a straight line. Because the original team has lost some of its members due to an obstacle, it would make sense for the robots continuing their motion to try and reconfigure with other robots in the system. To achieve this, these robots schedule to run a WVG at some time in the future by starting a timer called the WVG-TIMER. When the WVG-TIMER expires on a robot, it runs a WVG including the robots that are within its communication range.

After the WVG has determined the robots comprising the best minimum winning coalition (BMWC), the robot that has the highest weight in the BMWC is selected as their leader robot. The leader robot then selects a new position and heading, usually in the direction opposite to which the obstacle that caused the reconfiguration was sensed. It then starts running the flocking-based formation algorithm to get the follower robots in their desired positions and start moving together as a team in formation. In certain cases, for example, when the vicinity of the robots forming a coalition is occupied by obstacles, the coalition of robots calculated by a WVG might not be amenable to get into formation and move together as a team. When this happens the robots that are not able to get into the desired position reattempt to get into their desired positions for NUM-FORMATION-REATTEMPT iterations. At the end of the reattempts, the robots that managed to get into their desired position for the new team, exclude the unsuccessful robots from the team and continue their motion. The unsuccessful robots attempt another WVG among themselves. If these robots are unsuccessful to form a team after NUM-WVG-REATTEMPT successive WVGs (and included formation reattempts), they continue to move individually using Braitenberg motion. Also, after running a WVG, if there are some robots that are not included as part of the best minimum winning coalition, they continue to move individually using Braitenberg motion until they encounter another team and possibly get assimilated with that team after running a WVG.

When a robot moving individually or a robot team gets within close proximity of another robot team, they run a WVG with the combined team members as the participating robots. Finally, to avoid identical yet repetitive calculation of the BMWC in a WVG by all the participating robots, we have selected the robot with the lowest local identifier to run the WVG. This robot receives the weights from all the participating robots and after running the WVG reports the outcome of the WVG to all the participants.

6.3 Experimental Results in Simulator

The performance function used to compute the weight value for each robot for participating in the WVG uses the mean error in the robot's desired position and the area of previously uncovered region covered by the robot, over the last 25 time steps. Based on this function, for the DYN-REFORM algorithm, the duration of the WVG-TIMER is set to 25 time steps so that the robots do not continue in inefficient configurations for long durations. The STOP-TIMER is set to 15 time steps so that robots that do not encounter an obstacle and do not need to stop get a sufficient time window to move away from the stopped robots and possibly form a new team. The number of reattempts by a newly formed team to get into formation after running a WVG (NUM-FORMATION-REATTEMPT) was set to 5 to balance between splitting teams very frequently and inefficiently trying to form a team between robots when it is physically not possible (e.g., due to an obstacle between two sets of robots trying to form a single team). To prevent the formation of excessively large teams that have a high communication and computation overhead in the DYN-REFORM algorithm, we took two steps. First, we set the quota for a WVG at $q_w \times$ (sum of the weights of participating robots), where $q_w \in [0, 1]$ is a real number called the *quota ratio.* The quota ratio determines the size of the partitions or teams that are formed after running the WVG and a larger value of q_w gives preference to larger teams. For most of our experiments, we set $q_w = 0.9$, which guarantees that robots with very poor performance, and, consequently low weights get excluded from the new team after reconfiguration. Secondly, we limited the maximum team size in our experiments to 7 robots. If the winning coalition calculated by the WVG has more than 7 robots, the excess robots that have the lowest weights in the coalition are removed from the team and move individually without forming a team, until they merge with another team at a later stage. For quantifying the efficacy of team formation, we used the deviation in positions of the team members from a perfect formation. To measure this, we calculated the mean error in the positions of the follower robots at intervals of 10 sec. over the duration of the experiment. All results were averaged over 10 simulation runs.



(a) (b) (c)

Figure 6.8: (a) Initial configuration of a team of 5 robots moving in a wedge-shaped formation. (b) Reformed team moving in new direction after encountering a flat wall obstacle. (c) Mean error in the desired position of the robot team during the reconfiguration.

6.3.1 Team Reformation Experiments

For our first set of experiments, we verified the performance of the DYN-REFORM algorithm. We considered three types of obstacles - a flat wall obstacle, a non-uniform wall obstacle and a perpendicular, narrow wall obstacle, as shown in Figures 6.8(a), 6.9(a) and 6.10(a). For this set of experiments, we have traced the trail of the robots within Webots to show their motion before, during and after reconfiguration. For the flat wall obstacle, the leader robot encounters the wall first, enters into the STOP_AND_WAIT state and starts the STOP-TIMER, according to the DYN-REFORM algorithm. The follower robots successively encounter the wall and also enter the STOP_AND_WAIT state and start their individual STOP-TIMERs. The leader robot's STOP-TIMER expires first and it runs the WVG including the other robots that are stopped in its vicinity as the WVG's participants. As an example of the weight and quota values used in this WVG, one of the reported runs for this experiment had weights of the five robots as $0.72, 1.0, 0.96, 1.16$ and 1.24 respectively and quota $Q = 0.9 \times \sum_i w_i = 4.57$. The leader robot of the team had the worst performance recently because it encountered the obstacle first and stopped, giving it a weight 0.72 in the WVG. The fringe robots that stopped last have the highest weights of 1.16 and 1.24 respectively. The BMWC contains all the five robots in this scenario. which means that all the robots should stay together in the new team. The new leader robot then selects a position in a direction opposite to the direction in which it encountered the wall, a pose or heading for the new team, and communicates the desired position of the follower robots to get in formation in the new team. After the formation succeeds, the new team starts moving as shown in Figure 6.8(b). Figure 6.8(c) shows the mean error in the position of the robots during the entire reconfiguration process. Initially, between $0 - 50$ seconds, the team is moving in formation before encountering the obstacle, and this is shown by a low mean error of about 3 cm in the position of the robots. When the robots successively start encountering the obstacle, between $50 - 100$ seconds, the error between their actual positions and their desired positions to remain in formation increases. This happens because when the robots successively stop at the obstacle they form straight line along the boundary of the wall, while their current formation, in which they had been before encountering the obstacle, requires them to form a wedge shape. While the WVG runs, the robots are stopped and their mean error in position remains unchanged, as seen between $100 - 125$ seconds. After determining the best minimum winning coalition, the robots get a new formation and the mean error in the position of the robots decreases back to the low value of around $3 - 4$ cm over $125 - 250$ seconds. We notice that the mean error suddenly spikes to about 60 cm around 150 seconds when the new team calculated by the WVG attempts reformation.

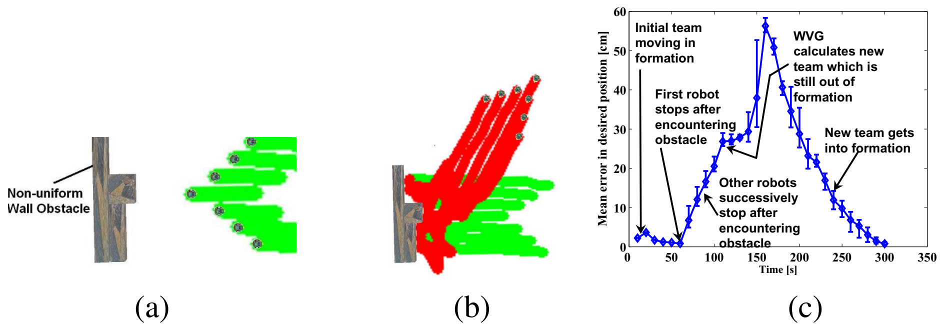


Figure 6.9: (a) Initial configuration of a team of 7 robots moving in a wedge-shaped formation. (b) Reformed team moving in new direction after encountering a non-uniform wall obstacle. (c) Mean error in the desired position of a robot team during the reconfiguration.

This happened because all the robots stopped at the flat wall forming a horizontal line and they are in close proximity of each other. The robots themselves occlude each others paths when they try to get into a new formation. However, within 20 seconds, which was within NUM-FORMATION-REATTEMPT$= 5$ iterations, the formation control algorithm is able to resolve this problem and the robots are able to regain formation.

The scenario with robots encountering the non-uniform wall obstacle, shown in Figures 6.9(a) and (b), is very similar to the flat wall case. The main difference is that because of the non-uniform surface of the wall, the robots along the fringes of the former team persist longer in the CONTINUE_PREVIOUS_MOTION state than in the flat wall case. This behavior is also seen in the mean error of the robots during reconfiguration, shown in Figure 6.9(c). The mean error in the position of the robots w.r.t. their positions in the previous formation becomes larger than the flat wall case because the robots move farther from their erstwhile desired positions in formation into the clefts of the wall. However, in this case, we do not see any significant spikes during after the WVG when the new team is

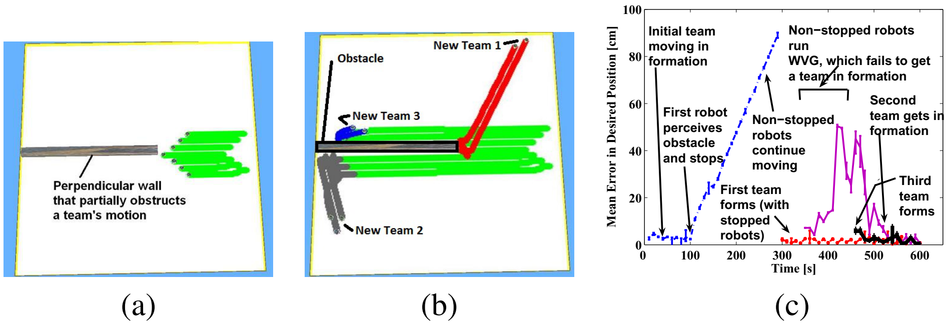
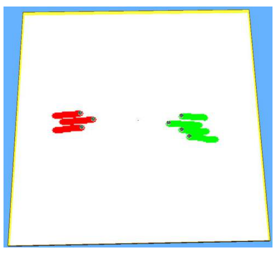
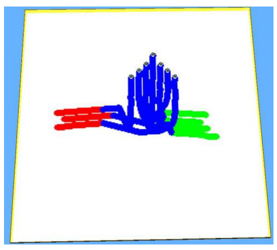


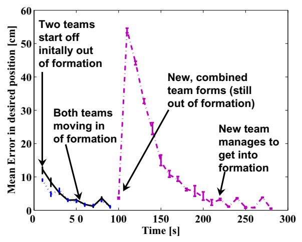
Figure 6.10: (a) Initial configuration of a team of 7 robots moving in wedge-shaped formation. (b) Reformed team moving in new direction after encountering a narrow obstacle that causes the team to split. (c) Mean error in the desired position of a robot team during the reconfiguration.

getting into its formation because the robots have dispersed further from each other because of the non-uniform surface of the wall. Therefore, they do not occlude each other's path while getting into formation.

Figures 6.10(a) and (b) show a scenario where a robot-team encounters a wall perpendicular to its heading that obstructs the team partially. In this scenario, only two robots at the center of the 7 robot team encounter the obstacle, enter into the STOP_AND_WAIT state and start their STOP-TIMERs. The rest of the team members do not encounter the obstacle and enter into the CONTINUE_PREVIOUS_MOTION state while starting the WVG-TIMER. We notice that the obstacle forces two sets of team members to continue their motion along the two sides - above and below the obstacle. When the STOP-TIMER expires for the two stopped robots, they run a WVG. Since there are no other stopped robots in their vicinity, the stopped robots form a new team among themselves and continue moving in a new direction. When the WVG-TIMER expires for the robots that continued their


(a)


(b)



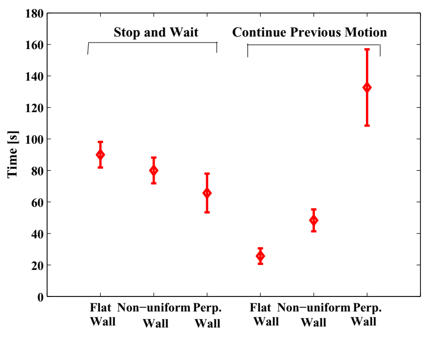
(c)

Figure 6.11: (a) Initial configuration of two teams of 3 and 4 robots moving in wedge-shaped formations. (b) Reformed team moving in new direction after encountering each other and merging into a new team. (c) Mean error in the desired position of a robot team during the reconfiguration.

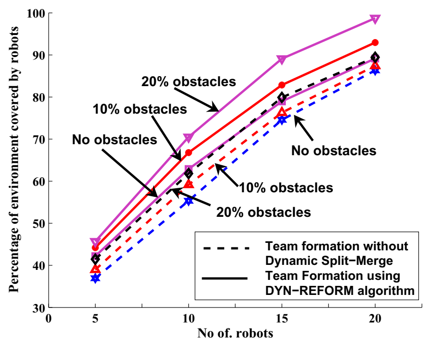
motion, they run a WVG. These robots are in the vicinity of each other and the BMWC contains all these robots. However, the BMWC contains two sets of robots that are located on opposite sides of the obstacle and can never get into a single formation. Therefore, although the WVG succeeds, the new team fails to form within NUM-FORMATION-REATTEMPT($=5$) reattempts. The DYN-REFORM algorithm then causes the robots that were unable to get in formation to to run another WVG and form a new team. Each team gets into its desired formation and continues its movement. The graphs in Figure 6.10(c) show that the mean error in the desired position of the robots from their erstwhile team keeps increasing because some of the robots do not encounter the obstacle and therefore, do not stop. When the WVGs run, three teams are formed at different times based on the STOP-TIMER expiry (first team) and WVG-TIMER expiry followed by WVG reattempt (second and third team), as shown in Figure 6.10(c).

Figure 6.11(a)-(c) show the scenario of two robot teams moving towards each other and merging using a WVG, and the mean error in robot positions during the reconfiguration process for this scenario. Before encountering each other, the two teams are moving in formation and consequently, the mean error in the desired position of the robots is low. When the teams encounter each other (team leaders separated by a distance of 70 cm or less) they stop and run a WVG. Examples of quota and weight values from one experiment run show that the weights of the robots in the two teams are $\{1.28, 1.2, 1.2, 1.2, \}$ and $\{1.32, 1.12, 1.16\}$ respectively, while the quota $Q = 0.9 \times \sum_i w_i = 7.6$. The WVG outputs the combined set of all robots in both teams as the BMWC, implying that the two teams have to merge into a new team. As in the case of reformation with the flat wall obstacle, we notice a large spike in the mean error of the positions of the robots around 120 seconds in Figure 6.11(c) because the robots from the combined teams get in each others' way while forming the new team. However, finally, they manage to get into their desired position before NUM-FORMATION-REATTEMPTS and move together as one team.

Figure 6.12(a) shows the mean times spent by the robots in the STOP_AND_WAIT state and the CONTINUE_PREVIOUS_MOTION state for the three different obstacle types. The time in the STOP_AND_WAIT state is the highest for the flat wall because all robots stop at the flat wall, while it is the lowest for the perpendicular wall where only two robots stop and the rest of the team continues its motion. A complementary trend happens for the time spent in the CONTINUE_PREVIOUS_MOTION state with a very low value when all team members stop at the flat wall, and a higher value in the perpendicular wall case when some team members never encounter the wall and continue moving until their WVG-TIMER expires and they run a WVG.



(a)



(b)

Figure 6.12: (a) Time spent by the robots in the STOP_AND_WAIT state and CONTINUE_PREVIOUS_MOTION state of the DYN-REFORM algorithm for the three different types of obstacles. (b) Percentage of the environment covered by a set of 5 robots initially configured as a team without and with DYN-REFORM algorithm. The environment is 2×2 m^2 with 0%, 10% or 20% of the area of the environment occupied by obstacles. Each experiment was run for a period of 30 mins. Error bars were omitted for legibility.

Figure 6.12(b) shows the improvement in coverage achieved using the DYN-REFORM algorithm by a set of 5 robots, initially configured as a team. The robots are placed within a 2×2 m^2 walled environment with 0%, 10% or 20% of the total area of the environment occupied by obstacles. The coverage algorithm used by a robot in a team maintains only the recent coverage information (over the last 25 time steps) and passes it to the team leader. A leader robot exchanges this recent coverage history of its team with other leader robots within its communication range to avoid covering regions that have been recently covered by other robot teams. We observe that the robots using the DYN-REFORM, because of their capability to dynamically reconfigure at obstacles and avoid inefficient configurations, are able to improve coverage by about 5% when there are no obstacles, and about $7 - 10\%$ when there are obstacles in the environment. [1]

[1]To determine the effect of changing the amount of historical coverage information stored in the memory of a robot (which determines the robot's 'weight' in the WVG), on the coverage performance, we considered

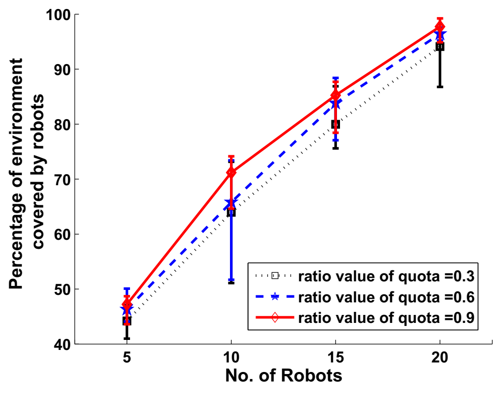


Figure 6.13: Percentage of environment covered by different numbers of robot for different values of the quota for the WVG within a 2×2 m^2 environment where 10% of the total area of the env. space is occupied by obstacles.

Next, we quantify the effect of varying the quota parameter, Q, that determines the size of the partitions or teams formed by the WVG. Recall that for a WVG, $Q = q_w \times \sum_{i \in R'} w_i$ where R' is the set of players or robots participating in the WVG, w_i is the weight of the i-th robot in this set and $q_w \in [0, 1]$ is the quota ratio. In our experiments for observing the effect of various quota values, we vary the value of q_w. A large value of q_w closer to 1 will enable the formation of larger teams, while smaller values of q_w closer to 0 will enable the formation of smaller teams or no teams at all (robots move individually). We report results from experiments with $5, 10, 15$ and 20 robots within a 2×2 m^2 environment where 10% of the total area of the environment is occupied by obstacles. Each experiment was run for a period of 30 mins. We observe that as the value of q_w increases towards 1.0, the performance of the coverage improves. This indicates that higher quota values,

different coverage history sizes corresponding to storing the coordinates of the last 10, 15, 25, 50 and 100 locations. Based on the setting that gave the best coverage performance, we used a coverage history size of storing the last 25 locations.

which results in larger partitions or larger teams among robots that are participating in the WVG result in improved coverage. Similar effect of varying the value of q_w were obtained when there were no obstacles in the environment or 20% of the environment was occupied by obstacles. Overall, the results of this experiment suggest that robots that are in proximity of each other and running a WVG among themselves produce better results for coverage if they remain together as a single team instead of fracturing into smaller teams or moving individually. In other words, for the scenarios tested in our experiments, dynamically forming teams of robots results in improved area coverage, as compared to a setting where the robots cover the environment individually.

Finally, we exported the region visited data from Webots and drew the coverage map in gray scale as shown in Figure 6.14 (a) - (d). There are 5 robots in a team in a 4×4 m^2 square environment. If the robots visited the regions more times, they are colored darker. Figure 6.14(a), (b), (c) and (d) show the coverage map in every 30 minute within 2 hours. We observe the wedged trail of the robot team. There are some sparse white cells in the path, because the balance distance between two adjacent robots in team is larger than the diameter of the coverage area per robot, as well as the robots have wheel slide noises. Also we observe that most cells are covered in Figure 6.14(d) (93.5% coverage rate), and a few of them are colored in black which means revisited over 15 times.

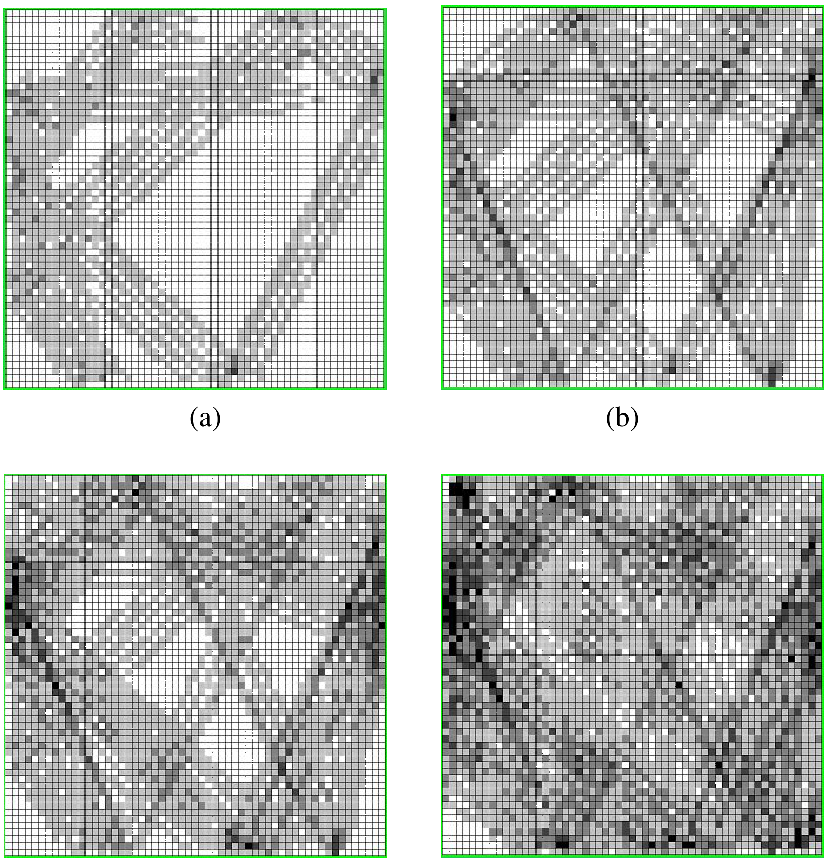
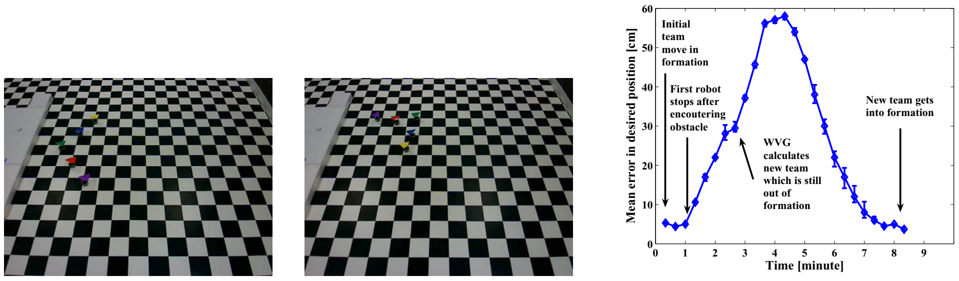

(a) (b) (c) (d)

Figure 6.14: 5 robots in a team visited in a 4 $\times$ 4 m^2 square environment. The region is colored by light gray, gray, dark gray, and black representing revisited 2 to 5 times, 6 to 10 times, 11 to 15, and over 16 respectively. (a) Run time is 30 minutes (b) Run time is 60 minutes.(c) Run time is 90 minutes. (d) Run time is 120 minutes.



(a) (b) (c)

Figure 6.15: (a) Initial configuration of a team of 5 robots moving in a wedge-shaped formation. (b) Reformed team moving in new direction after encountering a flat wall obstacle. (c) Average error in the position of the robot team during reconfiguration in the scenario of encountering a flat wall obstacle.

6.4 Physical Experimental Results

For the first set of experiment, we deployed five e-puck robots in a wedged shape initially, and moved from right to left as shown in Figure 6.15 (a). After a minute, the leader robot in the middle first encountered the flat wall and stopped. After that, the other four robots found the wall in front of them and stopped respectively. As we mentioned in the former section, the leader robot triggered the STOP-TIMER. In the physical robot, the stop time is longer than the time set in the simulator. When the leader robot's STOP-TIMER, it ran the WVG including the other robots that were stopped in its vicinity as the WVG's participants. First, it went backward about 30 seconds. Then, it turned to the new direction opposite to the wall and stopped. Third, it sent reform team request to the team member in the range. When the other robot received the request and the desired position in the team, they turned to go from current location to the desired location asynchronously. As there are noises from the picture catched by the camera and the wheel slip of the robot, each robot would

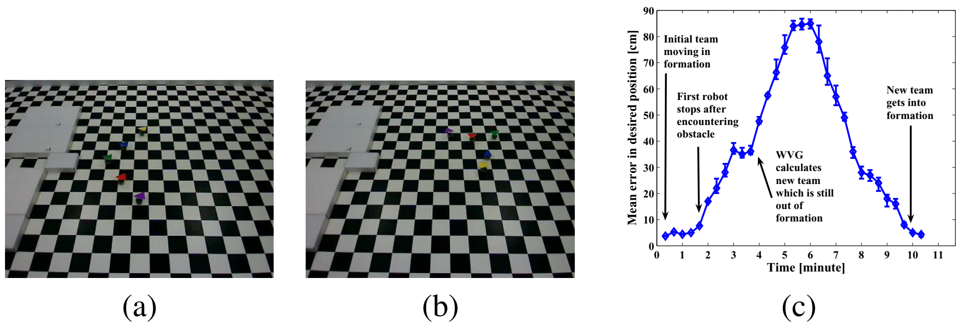


Figure 6.16: (a) Initial configuration of a team of 5 robots moving in a wedge-shaped formation. (b) Reformed team moving in new direction after encountering a non-uniform wall obstacle. (c) Average error in the position of the robot team during reconfiguration in the scenario of encountering a a non-uniform wall obstacle.

try several times to reach to the desired location. Figure 6.15 (b) shows after running 8 minutes, the 5 e-puck robots reformed the wedged shape and headed to the up right region of the environment. We collected the data of the mean error from current location to the desired position of each robot in the team by every 20 seconds as shown in Figure 6.15 (c). Within the first minute, as the team moved in a wedged shape, the mean error in the desired position is roughly 5 cm. For the leader robot encountered the wall and stopped, the mean error of the desired position increased sharply to about 30 cm. As the leader robot went back to reform the team shape, the mean error in the desired position keeps going to 55-58 cm as a peak area from 4 to 5 minutes. After that, the mean error in the desired position goes back to approximately 5 cm. This means the team have finished reforming the shape and heading to a new direction after 8 minutes run time.

For the next set of experiment, we deployed five e-puck robots in a wedged shape initially as the former experiment while there was a non-uniform wall in their way, as shown in

Figure 6.16 (a). After a minute, the leader robot in the middle first encountered the flat wall and stopped. After that, the other four robots found different walls in front of them and stopped respectively. Also, the leader robot triggered the STOP-TIMER. When the leader robot's STOP-TIMER, it ran the WVG including the other robots that were stopped in its vicinity as the WVG's participants. First, it went backward about 30 seconds. Then, it turned to the new direction opposite to the wall and stopped. Third, it sent reform team request to the team member in the range. When the other robot received the request and the desired position in the team, they turned to go from current location to the desired location asynchronously. As there are noise from the picture catched by the camera, the wheel slip of the robot and even the other side of the wall , each robot would tried more time than the former experiment to reach to the desired location. Figure 6.16 (b) shows after running 10 minutes, the 5 e-puck robots reformed the wedged shape and headed to the up right of the environment. We observed the mean error from current location to the desired position of each robot in the team by every 20 seconds as shown in Figure 6.16 (c). Within the first minute, as the team moved in a wedged shape, the mean error in the desired position is roughly 5 cm. For the leader robot encountered the wall and stopped, the mean error of the desired position increased sharply to about 40 cm, for the robots were not in a vertical line. As the leader robot went back to reform the team shape, the mean error in the desired position keeps going to 80-85 cm as a peak area from 5 to 7 minutes. After that, the mean error in the desired position goes back to approximately 5 cm. This means the team have finished reforming the shape and heading to a new direction after 10 minutes.

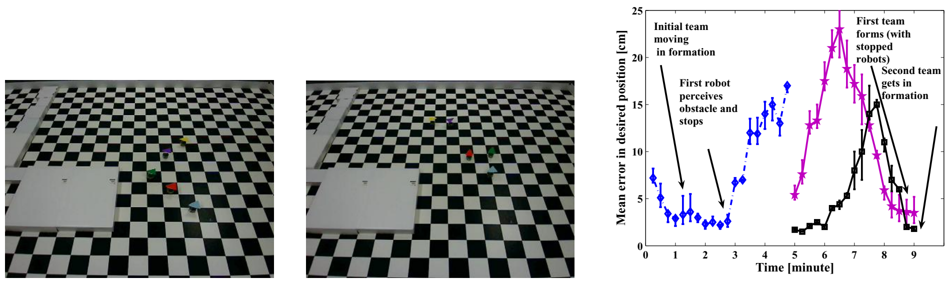


Figure 6.17: (a) Initial configuration of a team of 5 robots moving in wedge-shaped formation. (b) Reformed team moving in new direction after encountering a narrow obstacle that causes the team to split. (c) Mean error in the desired position of a robot team during the reconfiguration.

For the third set of experiment, Figures 6.17(a) and (b) show a scenario where a robot-team encounters a wall perpendicular to its heading that obstructs the team partially. In this scenario, only three robots at the bottom of the 5 robot team encounter the obstacle, enter into the STOP_AND_WAIT state and start their STOP-TIMERs. The rest two of the team members do not encounter the obstacle and enter into the CONTINUE_PREVIOUS_MOTION state while starting the WVG-TIMER. Since there are no other stopped robots in their vicinity, the stopped robots form a new team among themselves and continue moving in a new direction. When the WVG-TIMER expires for the robots that continued their motion, they run a WVG. These robots are in the vicinity of each other and the BMWC contains all these robots. The graphs in Figure 6.17(c) show that the mean error in the desired position of the robots from their erstwhile team keeps increasing because some of the robots do not encounter the obstacle and therefore, do not stop. When the WVGs run, two teams are formed at different times based on the STOP-TIMER expiry (first team) and WVG-TIMER

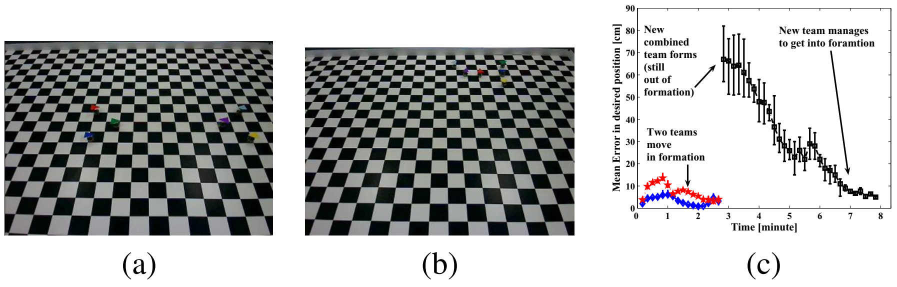


Figure 6.18: (a) Initial configuration of two teams of 3 robots each moving in wedge-shaped formations. (b) Reformed team moving in new direction after encountering each other and merging into a new team. (c) Mean error in the desired position of a robot team during the reconfiguration.

expiry followed by WVG reattempt (second team), as shown in Figure 6.10(c).

For the last set of experiment, figure 6.18(a)-(c) show the scenario of two robot teams moving towards each other and merging using a WVG, and the mean error in robot positions during the reconfiguration process for this scenario. Before encountering each other, the two teams are moving in formation and consequently, the mean error in the desired position of the robots is as low as 3 cm. When the teams encounter each other (team leaders separated by a distance approximately the same as the simulator) they stop and run a WVG. The WVG outputs the combined set of all robots in both teams as the BMWC, implying that the two teams have to merge into a new team. As in the case of reformation with the flat wall obstacle, we notice a large spike in the mean error of the positions of the robots around 3 minutes in Figure 6.18(c) because the robots from the combined teams get in each others' way while forming the new team. The peak point of the mean error in the physical experiment is $10 - 15$ cm larger than the one in the simulation. However, finally, they manage to get into their desired position before NUM-FORMATION-REATTEMPTS and move together as one team.

Chapter 7

SUMMARY OF CONTRIBUTIONS AND FUTURE DIRECTIONS

This chapter summarizes the work presented in previous chapters and shows this dissertation's contributions to mobile robotics field. This chapter also outlines potential avenues for future research.

7.1 Summary of Contributions

The main contribution of this research to multi-robot area coverage domain:

(1) This dissertation introduces a swarm based multi-robot individual coordination ap-

proach. Robots are also susceptible to sensor noise while communicating with other robots, and can be subject to transient or permanent failures. The objective of the robots is to cover the entire environment while reducing the coverage time and the redundancy in the area covered by different robots. First, we describe our distributed coverage algorithm where each robot uses a local heuristic based on Manhattan distances and the information gained from other robots at each step to decide on its next action(movement). We then describe and analyze the fault model of our robots and show that the local heuristic used by the robots deteriorates linearly as the communication noise increases. We observe that the system is robust against inaccurate localization due to noisy sensor data and failures of a few robots in the system

(2) Our terrain coverage algorithm is inspired by the flocking behavior of birds and animals observed in nature. We identify certain deficits with the basic model of flocking and propose techniques that allow robots to dynamically form teams and adapt the team's configuration and size based on the operational conditions in the environment. The robot team can adaptively change the shape of the team in the complex environment.

(3) A heuristic-based coalition formation algorithm is proposed. To model the decision process as voting game, we simplify the issue of multi-robot team formation. Instead of exponentially calculating all possible coalitions, this square algorithm significantly reduces the solution space based on veto player set. We propose and evaluate this technique of com-

puting the weights of a weighted voting game based on each robots coverage capability and finding the best minimal winning coalition(BMWC). We theoretically prove the feasibility of our model, and give algorithms to find the BMWC as well.

This dissertation is on developing a distributed multi-robot system to perform complete coverage of an initially unknown environment in an efficient manner. I have conducted research in the design, analysis, evaluation, and testing of a multi-robot coalition formation system performing area coverage both on the Webots robotic simulation platform, as well as with e-puck mini robots.

7.2 Future Directions

This dissertation provides swarm based multi-robot coordination combined with coalitional team formation which uses weighted voting game mechanism. However, much work needs to be done to improve this WVG based multi-robot system for area coverage. A future problem we are investigating is to dynamically adapt the value of the threshold in setting the WVGs, so that the team size can be automatically changed in for cluttered or open environments perceived by robots. We plan to use a Q-learning algorithm to learn the value of the quota parameter and a policy reuse mechanism to adapt the learning process to changes in the underlying environment.

We envisage that this technique also supports heterogeneous robots with different memory and communication limitations as well. we are investigating to use teams of heterogeneous robots with diverse sensor capabilities and how to integrate the robot heterogeneity into the WVG framework. For example, team leader can be the robot with high computation capability. Some of the team members have detection sensor, and some of the team members have localization sensor. How to represent different types of these capability as weights is the next topic in this field.

As we fulfilled our system in the indoor lab environment, we use a over-head camera to mimic the satellite and color marks to identify each robot. We also use a commercial computer vision software to process the image and calculate the location for each robot. However, the localization for our system is still a little bit ideal. Instead of global localization, a further direction can be a technique combining the global localization with local position. Although the algorithms presented in this dissertation have been fulfilled on e-puck robots, theoretically they can be used in other type of robots such Corobot: a four-wheeled robot that has an on board mini-ITX computer, Explorer: designed for outdoor application with GPS, laser, and an optional 4 DOF arm, and Seekur: the first commercially available robot to demonstrate MDARS-like capabilities for general use by airports, utility plants, corrections facilities and Homeland Security which need more efforts in implementation. These high capability robots are design for outdoor applications.

Another potential area of research is the handling of localization and communication failures by the multi-robot coalition formation systems and the formulation of techniques to make the robot team formation process more robust to these failures. Currently, the parameters for these systems were adjusted based on trial in Webots simulator and e-puck physical robots. Future work involves dynamically adjusting these parameters with other robot models and experience.

Bibliography

[1] M. Ahmadi and P. Stone. “A multi robot system for continuous area sweeping tasks”, Proc. Intl. Conf. on Robotics and Automation, 2006, pp. 1724-1729.

[2] Y. Altshuler, A.M. Bruckstein and I.A. Wagner. “Swarm Robotics for a Dynamic Cleaning Problem”, Proc. IEEE Swarm Intelligence Symposium, 2005, pp. 209-216.

[3] H. Aziz and B. de Keijzer. “Complexity of coalition structure generation.”, AAMAS 2011, The Tenth International Conference on Autonomous Agents and Multiagent Systems, 2011, Taiwan, pp. 191-198.

[4] Yoram Bachrach, Edith Elkind. “Divide and Conquer: False-Name Manipulations in Weighted Voting Games”, The Seventh International Joint Conference on Autonomous Agents and Multiagent Systems, 2008, pp. 409-416.

[5] YM. Batalin and G. Sukhatme. “A local approach to multi-robot coverage”, the 6th International Symposium on Distributed Autonomous Robotic Systems, 2002, pp. 373-382.

[6] J. M. Bilbao, J. R. Fernandez, N. Jiminez, and J. J. Lopez. “Voting power in the

European Union enlargement”, European Journal of Operational Research, 2002, pp. 181-196.

[7] T. Balch and R. Arkin. “Behavior-based formation control of multi-robot teams”, IEEE Transactions on Robotics and Automation, 1998, Vol 14, No. 6, pp. 926-939.

[8] E. Bonabeau, M. Dorigo and G. Theraulaz, ”Swarm Intelligence: From Natural to Artificial Systems,” Oxford University Press, New York, 1999.

[9] W. Burgard, M. Moors, D. Fox, R. Simmons and S. Thrun. “Collaborative multi-robot exploration”, IEEE Transaction on Robotics, 2005, Vol 143, No. 3, pp. 376- 386.

[10] Q. Chen and J. Luh. “Coordination and control of a group of small mobile robots”, Proc. Intl. Conf. on Robotics and Automation, 1994, pp. 2315-2320.

[11] K. Cheng, and P. Dasgupta. “Dynamic Area Coverage using Faulty Multi-agent Swarms”, Proc. IEEE/WIC/ACM International Conference on Intelligent Agent Technology, 2007, pp. 17-23.

[12] K. Cheng and P. Dasgupta. “coalition game based distributed coverage of unknown environments using robot swarms”, (short paper) AAMAS 2008, 2008, pp. 1191-1194.

[13] K. Cheng, P. Dasgupta, and Y. Wang. “Distributed Area Coverage Using Robot Flocks”, World Congress on Nature and Biologically Inspired Computing, 2009, pp. 678-683.

[14] K. Cheng, and P. Dasgupta. "Weighted Voting Game Based Multi-robot Team Formation for Distributed Area Coverage", 3rd International Symposium on Practical Cognitive Agents and Robots, 2010, pp. 9-15.

[15] K. Cheng, and P. Dasgupta. "Multi-Agent Coalition Formation for Distributed Area Coverage: Analysis and Evaluation", Second International Workshop on Collaborative Agents – REsearch and Developmen, 2010, pp. 334-337.

[16] H. Choset. "Coverage of robotics - A survey of recent results", Annals of Mathematics and Arrifical Intelligence, 2001, Vol 31 pp. 113-126.

[17] T. Cormen, C. Leiseron, R. Rivest and C. Stein. "Intro. to Algorithms", McGraw Hill, 2001.

[18] N. Correll and A. Martinoli. "Robust Distributed Coverage using a Swarm of Miniature Robots", Proc. Intl. Conf. on Robotics and Automation, 2007, pp. 379-384.

[19] P. Dasgupta, and K. Cheng. "Robust Multi-robot Team Formations using Weighted Voting Games", 10th International Symposium on Distributed Autonomous Robotics Systems, 2010, pp. 334-337.

[20] P. Dasgupta, K. Cheng, and L. Fan. "Flocking-based Distributed Terrain Coverage with Mobile Mini-robots", Swarm Intelligence Symposium, 2009, pp. 96-103.

[21] X. Deng and C. H. Papadimitriou. "On the complexity of cooperative solution concepts", Math. of Oper. Res., 1994, Vol 19, No. 2 pp. 257-266.

[22] E.Elkind, L.A.Goldberg, P.W.Goldberg, and M. Wooldridge. "Comutational Complexity of Weighted Threshold Games", In Proc. of AAAI, 2007, pp. 718-723.

[23] E.Elkind, L.A.Goldberg, P.W.Goldberg, and M. Wooldridge. "On the Dimensionality of Voting Games", In Proc. of AAAI, 2008, pp. 69-74.

[24] J. Fredslund and M. Mataric. "A general algorithm for robot formations using local sensing and minimal communication", IEEE Trans. on Robotics and Automation, 2002, Vol 18, No. 5, pp. 837-846.

[25] R. Falconi, S. Gowal, A. Martinoli. "Graph Based Distributed Control of Non-Holonomic Vehicles Endowed with Local Positioning Information Engaged in Escorting Missions", Proc. Intl. Conf. on Robotics and Automation, 2010, pp. 3207-3214.

[26] Michael R. Garey, David S. Johnson. "Computers and Intractability A Guide to the Theory of NP-Completeness", W.H. Freeman and Company, 1997, pp. 3207-3214.

[27] Y. Gabriely and E. Rimon. "Spanning tree based coverage of continuous areas by a mobile robots", Annals of Mathematics and Artificial Intelligence, 2001, Vol 31, pp. 77-98.

[28] A. Gasparri, B. Krishnamachari and G. Sukhatme. "A framework for multi-robot node coverage in sensor networks", Annals of Math and AI, 2008, Vol 52, pp. 281-305.

[29] S. Ge and C. Fua. "On Redundancy, Efficiency, and Robustness in Coverage for Multiple Robots", Proc. Intl. Conf. on Robotics and Automation, 2005, pp. 727-732.

[30] Generation Robots web site, online document, accessed at 2011: http://www.generationrobots.com/e-puck-programmable-robot-with-battery,us,4,E-puck-robot.cfm

[31] Y. Hanada, G. Lee and N. Chong. "Adaptive flocking of a swarm of robots based on local interactions", Proc. of 2007 Swarm Intelligence Symposium, 2007, pp. 340-347.

[32] N. Hazon, G. Kaminka. "A framework for multi-robot node coverage in sensor networks", Robotics and Autonomous Systems, 2008, Vol 56, No.12 pp. 1102-1114.

[33] A. Howard, M. Mataric, and G. Sukhatme. "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem", Proc. Intl. Symp. on Distributed Autonomous Robotic Systemss, 2002, pp. 299-308.

[34] S.Ieong, Y.Shoham. "Bayesian Coalitional Games", In Proc. of AAAI, 2008, pp. 95-100.

[35] M. Jager and B. Nebel. "Dynamic decentralized area partitioning for cooperating cleaning robots", Proc. Intl. Conf. on Robotics and Automation, 2002, pp. 3577-3582.

[36] M. Ji and M. Egerstedt. "A Graph-Theoretic Characterization of Controllability for Multi-Agent Systems", Proc. American Control Conference, 2007, pp. 4588-4593.

[37] S. Koenig, B, Szymanski and Y. Liu. "Efficient and inefficient ant coverage methods", Annals of Mathematics and Artificial Intelligence, 2001, Vol 31, No. 1-4 pp. 41-76.

[38] G. Kaminka, R. Schechter, and V. Sadov. “Using Sensor Morphology for Multirobot Formations”, IEEE Transactions on Robotics, 2008, Vol 24, No. 2, pp. 271-282.

[39] J. Malkevitch. “Voting Games”, Providence, RI: American Mathmatic Society, 2010, accessed at:http://www.ams.org/featurecolumn/archive/weighted1.html.

[40] L. Ludwig and M. Gini. “Robotic swarm dispersion using wireless intensity signals”, Proc. 8th Intl. Symp. on Distributed Autonomous Robotic Systems, 2006, pp. 135-144.

[41] T.Matsui, Y.Matsui. “A Survey of Algorithms for Calculating Power Indices of Weighted Majority Games”, Journal of the Operations Research Society of Japan, 2000, Vol 43, No. 1, pp. 71-86.

[42] M. Mazo Jr., K. H. Johansson. “Path-Planning for Robust Area Coverage: Evaluation of Five Coordination Strategies”, AUCLA Electrical Engineering, 2010, accessed at:http://www.ee.ucla.edu/ mmazo/docs/PathPlanCov.pdf.

[43] R. B. Myerson. “Game Theory”, Harvard Universiy Press, 1997.

[44] O. Michel. “Webots TM: Professional mobile robot simulation”, International Journal of Advanced Robotics Systems, 2004, Vol 1, No. 1, pp. 39-42.

[45] R. Morlok and M. Gini. “Dispersing robots in an unknown environment”, Proc. 7th Intl. Symp. on Distributed Autonomous Robotic Systems, 2004, pp. 291-301.

[46] K. O'Hara and T. Balch. "Pervasive sensor-less networks for cooperative multi-robot tasks", Proc. 7th International Symposium on Distributed Autonomous Robotic Systems, 2004, pp. 192-201.

[47] M. Oliver. "Professional Mobile Robot Simulation", International Journal of Advanced Robotic Systems, 2004, pp. 39-42.

[48] L. Parker. "ALLIANCE: An architecture for fault tolerant multi-robot cooperation", IEEE Transactions on Robotics and Automation, 1998, Vol 14, No.2, pp. 220-240.

[49] L. Parker. "Adaptive heterogeneous multi-robot teams", Neurocomputing, 1999, Vol 28, pp. 75-92.

[50] L. Parker. "Distributed algorithms for multi-robot observation of multiple moving targets", Autonomous Robots, 1999, Vol 12, No. 3, pp. 231-255.

[51] L. Parker. "J. Pearce, B. Powers, C. Hess, P. Rybski, S. Stoeter, and N. Papanikolopoulos", Robotics and Autonomous Systems, 2006, Vol 45, No. 4, pp. 307-321.

[52] C. W. Reynolds. "Flocks, Herds, and Schools: A Distributed Behavioral Model", Computer Graphics, 1987, Vol 21, No. 4, pp. 25-34.

[53] I. Rekleitis, A. New, E. Rankin and H. Choset. "Efficient Boustrophedon Multi-Robot Coverage: an algorithmic approach", Annals of Mathematics and Artificial Intelligence, 1987, Vol 52, No. 2-4, pp. 109-142.

[54] I. Rekleitis, V. Lee-Shue, A. New and H. Choset. “Limited communication, multi-robot team based coverage”, Proc. Intl. Conf. on Robotics and Automation, 2004, pp. 3462-3468.

[55] roborealm. “Online document”, roborealm, 2010, accessed at: http://www.roborealm.com

[56] S. Rutishauser, N. Correll, and A. Martinoli. “Collaborative Coverage using a Swarm of Networked Miniature Robots”, Robotics and Autonomous Systems, 2009, Vol 57, No. 5, pp. 517-525.

[57] O. Shehory, and S. Kraus. “Methods for task allocation via agent coalition formation”, Artif. Intell. J., 1998, Vol 101, No. 1-2, pp. 165-200.

[58] Y. Shoham and K. Leyton-Brown. “Multiagent Systems: Algorithmic, Game Theoretic and Logical Foundations”, Cambridge University Press, 2009.

[59] D. Spears, W. Kerr, and W. Spears. “Physics-based Robot Swarms for Coverage Problems”, Proc Intl. Journal on Intelligent Control and Systems, 2006, Vol 11, No. 3, pp. 124-140.

[60] B. Smith, M. Egerstedt, and A. Howard. “Automatic Generation of Persistent Formations for Multi-Agent Networks Under Range Constraints”, Mobile Networks and Applications Journal, 2009, Vol 14, pp. 114-127.

[61] J. Svennebring, S. Koening. “Building Terrain Covering Ant Robots: A Feasibility Study”, Autonomous Robots, 2004, Vol 16, pp. 313-332.

[62] C. Stachniss and W. Burgard. “”, Proc. Intl. Joint Conf. on Artificial Intelligence, 2003, pp. 1127-1134.

[63] J. Svennebring and S. Koenig. “Trail-laying robots for robust terrain coverage”, Proc. Intl. Conf. on Robotics and Automation, 2003, pp. 75-82.

[64] A. Taylor, W. Zwicker. “Simple Game: Desirablility Relations, Trading, Pseudo weightings”, Princeton Univerity Press, 1999.

[65] A. Turgut, H. Celikkanat, F. Gokce, E. Sahin. “Self-organized Flocking with a Mobile Robot Swarm”, Proc. International Conference on Autonomous Agents and Multi-Agent Systems, 2008, pp. 39-46.

[66] L. Vig, and J. Adams. “Multi-Robot Coalition Formation”, IEEE Transactions on Robotics, 2006, Vol 22, No. 4, pp. 637-649.

[67] I. Wagner, M. Lindenbaum and A. Bruckstein. “Distributed covering by ant-robots using evaporating traces”, IEEE Transactions of Robotics and Automation, 1999, Vol 15, No. 5, pp. 918-933.

[68] I. Wagner, M. Lindenbaum and A. Bruckstein. “Cooperative Cleaners: A Study in Ant Robotics”, International Journal of Robotics Research, 2008, Vol 27, pp. 127-151.

[69] P. Wang. “Navigation strategies for multiple autonomous mobile robots moving in formation”, IEEE/RSJ Intl. Workshop on IROS, 1989, pp. 486-493.

[70] I. Wagner, M. Lindenbaum and A. Bruckstein, ”Distributed covering by ant-robots using evaporating traces,” IEEE Transactions of Robotics and Automation, Vol. 15, no. 5, IEEE Press, Piscataway, NJ, USA, 1999, pp. 918-933.

[71] J. Wang, S. Chao, and A. Agogino,“Sensor noise model development of a longitudinal positioning system for AVCS,” Proc. American Control Conference, IEEE Press, Piscataway, NJ, USA, 1999, pp.3760-3764

[72] K. Wurm, C. Stachniss and W. Burgard. “Coordinated multi-robot exploration using a segmentation of the environment”, Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008, pp. 1160-1165.

[73] B. Yamayuchi. “Frotier based exploration using multiple robots”, Proc. 2nd Intl. Conf. on Autonomous Agents, 1998, pp. 47-53.

[74] X. Zheng, S. Jain, S. Koenig and D. Kempe. “Multi-robot forest coverage”, Proc. of IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems, 2005, pp. 3852-3857.

Appendix A

Refereed Publications by Ke Cheng

1. **K. Cheng**, and P. Dasgupta, “Multi-Agent Coalition Formation for Distributed Area Coverage”, (accepted) Lecture Notes in Computer Science series, volume 6066, 2011, pp. 1-13.

2. **K. Cheng**, P. Dasgupta, and B. Banerjee “Adaptive Multi-Robot Team Reconfiguration using a Policy-Reuse Reinforcement Learning Approach”, The Autonomous Robots and Multi-Robot Systems workshop (ARMS 2011), Taibei, Taiwan, 2011.

3. P. Dasgupta, T. Whipple, and **K. Cheng**,“Distributed Area Coverage of Unknown Environments Using Mini-robots”, International Journal of Swarm Intelligence Research (IJSIR), 2011, Vol. 2(1), pp. 45-70.

4. P. Dasgupta, and **K. Cheng**, “Robust Multi-robot Team Formations using Weighted Voting Games”, 10th International Symposium on Distributed Autonomous Robotics

Systems (DARS 2010), EPFL, Switzerland, 2010.

5. **K. Cheng**, and P. Dasgupta, “Multi-Agent Coalition Formation for Distributed Area Coverage: Analysis and Evaluation”, the Second Collaborative Agents: Research and Development (CARE Workshop 2010), Toronto, Canada, 2010.

6. **K. Cheng**, and P. Dasgupta, “Weighted Voting Game Based Multi-robot Team Formation for Distributed Area Coverage”, 3rd Practical and Cognitive Agents and Robots (PCAR) Workshop, (co-located with AAMAS 2010), Toronto, Canada, 2010.

7. **K. Cheng**, S. Srinivasan, and A Tripathi, “Adaptive ARQ in Wireless Sensor Networks”, 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT 2010), Chengdu, China, 2010.

8. **K. Cheng**, P. Dasgupta, and Y. Wang, “Distributed Area Coverage Using Robot Flocks,” World Congress on Nature and Biologically Inspired Computing (NaBIC’09), 2009.

9. P. Dasgupta, and **K. Cheng**, “Distributed Coverage of Unknown Environments using Multi-robot Swarms with Memory and Communication Constraints,” UNO Tech Report (cst-2009-1), 2009.

10. P. Dasgupta, **K. Cheng**, and L. Fan, “Flocking-based Distributed Terrain Coverage with Mobile Mini-robots,” Proc. IEEE Swarm Intelligence Symposium (SIS’09), Nasville, TN, March 2009, pp. 96-103.

11. L. Fan, P. Dasgupta and **Ke Cheng**, "Swarming-based Mobile Target Following Using Limited-Capability Mobile Mini-robots," Proc. IEEE Swarm Intelligence Symposium (SIS'09), Nashville, TN, March 2009, pp. 168-175.

12. **K. Cheng**, and P. Dasgupta, "Coalition game based distributed coverage of unknown environments using robot swarms," International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'08), Estoril, Portugal, 2008.

13. J. Murphy, S. Petter, **K. Cheng**, and R. Briggs, "Hitting the Collaboration Target: Computer- Guided thinklet Selection", Proc. International Conference on Design Science Research in Information Systems and Technology, Atlanta, Georgia, 2008.

14. **K. Cheng**, and P. Dasgupta, "Dynamic Area Coverage using Faulty Multi-agent Swarms" Proc. IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2007), Fremont, CA, 2007.

15. P. Dasgupta, M. Hoeing, **K. Cheng**, et al., "Dynamic Pricing Algorithms for Task Allocation in Multi-agent Swarms," Proc. First International Workshop on Coordination and Control in Massively Multi-agent Systems (CCMMS'07), Honolulu, HI, May 2007, pp. 1-15.