8-2006

ASyMTRe: Building Coalitions for Heterogeneous Multi-Robot Teams

Fang Tang
University of Tennessee - Knoxville

To the Graduate Council:

I am submitting herewith a dissertation written by Fang Tang entitled "ASyMTRe: Building Coalitions for Heterogeneous Multi-Robot Teams." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Computer Science.

Lynne E. Parker, Major Professor

We have read this dissertation and recommend its acceptance:

Bradley T. Vander Zanden, Bruce J. MacLennan, J. Douglas Birdwell

Accepted for the Council:

Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:
I am submitting herewith a dissertation written by Fang Tang entitled "ASyMTRe: Building Coalitions for Heterogeneous Multi-Robot Teams". I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Computer Science.

Lynne E. Parker
Major Professor

We have read this dissertation
and recommend its acceptance:

Bradley T. Vander Zanden

Bruce J. MacLennan

J. Douglas Birdwell

Accepted for the Council:

Anne Mayhew
Vice Chancellor and
Dean of Graduate Studies

(Original signatures are on file with official student records.)

ASyMTRe: Building Coalitions for Heterogeneous Multi-Robot Teams

A Dissertation presented for the
Doctor of Philosophy
Degree
The University of Tennessee, Knoxville

Thesis Advisor: Lynne E. Parker

Fang Tang
August 2006

Dedication

For my amazing family who gave me tremendous encouragement and support in my work for all these years, with love and gratitude. May this first Ph.D. dissertation in our family be a lasting testimony of God's love.

Acknowledgements

This dissertation could not have been written without the inspiration, guidance, and support from many people. First and foremost, I would like to express my deepest thanks to my thesis advisor, Lynne Parker, for guiding me into the interesting world of robotics with herself as a great role model, for giving me the inspiration and guidance that are indispensable to the completion of this dissertation, and for offering me helpful suggestions when they are necessary. I would also like to thank the rest of my thesis committee, Brad Vander Zanden, Bruce MacLennan, and Doug Birdwell for their invaluable guidance and suggestions that help to broaden and improve this work.

I really enjoyed being a part of the Distributed Intelligence Laboratory (DILab). Working with Arno, Blitz, Cappy, and many other robots left me many joyful moments. I would like to give special thanks to Michael Bailey, who is always there to help when the robots are unwilling to behave, and gives me tremendous help in improving my English writing skills. I enjoyed discussing various robotics topics with Yifan Tang, Yuanyuan Li, Balajee Kannan, and Xingyan Li and would like to thank them for their constant inspiration and encouragement, and for their helpful suggestions on my presentation. I also enjoyed working with the members in the SDR project (Michael, Balajee, Xiaoquan Fu, Chris Reardon, Yifan, and Ben Birch), for their advice and collaborative efforts.

I am fortunate to receive many helpful reviews that helped to improve this work in many different ways. I also saved a lot of time using Player/Stage for robot control code development. I would like to thank the anonymous reviewers and the developers of the Player/Stage project for their great effort in supporting the robotics society.

Many administrative assistants and staffs have provided me great help in countless occasions. I would like to give thanks to Dorsey Bottoms, Donna Bodenheimer, and our lab staffs. I would also like to thank Alan Hollis for reviewing the dissertation.

While studying here in Knoxville, thousands of miles away from home, my husband and I have been fortunate to meet many friends. We are thankful to be members of the Knoxville Chinese Christian Church and are offered great support from many brothers and sisters. I enjoyed our Bible study group with Yifan, Lydia, Marci, Rachel, Jen, and Xinyan, for their support and encouragement. I also enjoyed the friendly environment of our Computer Science Department, the friendship with Yifan, Yuanyuan, Xingyan, Yun Zhang, Huadong Liu, Siyuan Liu, Yihua Bai, Jun Xu, Shaotao Liu, Chang Cheng, Miao Chen, Min Zhou, Yuanlei Zhang, and many others.

Lastly and importantly, I express my gratitude for my parents Xiaozhong Tang and Ling Ling, and the rest of my family, who encouraged me to pursue a Ph.D. degree and supported me unconditionally, with great patience and love. Most of all, I give my heartfelt

thanks to my husband Qichao – who encouraged and supported me in pursuing my dreams, shared both the joyful and the difficult times with me, and accompanied me always, with love.

Abstract

This dissertation addresses the problem of synthesizing task solution strategies for a heterogeneous robot team to accomplish multi-robot tasks by sharing sensory and computational resources. The approach I developed is called ASyMTRe, which stands for Automated Synthesis of Multi-robot Task solutions through software Reconfiguration, pronounced "Asymmetry". When dealing with heterogeneous teams, it is challenging to determine how the capabilities of each team member can be appropriately utilized to facilitate the accomplishment of the team-level goal. The ASyMTRe approach provides a way for the robots to reason about how to solve a task depending on their individual capabilities. The fundamental idea of ASyMTRe is the change of abstraction of robot capabilities from the traditional *task/sensor* perspective to a more fine-grained *schema* perspective. Inspired by the information invariants theory, the mapping of schemas to information types enables robots to connect schemas within or across robots to form coalitions in which robots share sensory or computational information with their coalition members.

The contributions of the ASyMTRe approach include: (1) enabling robots to automatically generate task solutions based on sensor-sharing across robot coalition members, in configurations not previously explicitly defined by the human designer; (2) providing a way for robots to develop coalitions to address multi-robot tasks; (3) enabling flexible software code reuse from one multi-robot application to another through the task-independent schema abstraction that is viewed as a generator of semantic information content which can be used in many ways by various diverse tasks; and (4) providing a way for robots to reconfigure solutions online when the team encounters unexpected sensor or robot failures. ASyMTRe has two different versions of implementation on multi-robot teams: the centralized ASyMTRe configuration algorithm and the distributed ASyMTRe-D negotiation protocol, depending on the amount of information shared among team members. In addition, the ASyMTRe approach has been integrated with an auction-based task allocator such that the resulting system generates robot coalitions to accomplish multi-robot tasks at a low level, and these coalitions compete for task assignment at a high level.

The ASyMTRe approach has been successfully implemented and tested in three different application scenarios: multi-robot navigation, cooperative box pushing, and site clearing. These experiments have validated the ASyMTRe approach and demonstrated its solution generation process, robustness, code reusability, and applicability to a wide range of multi-robot applications.

Contents

1 Introduction **1**
- 1.1 The Problem . . . 1
- 1.2 Illustrating Examples . . . 2
- 1.3 The Approach . . . 3
- 1.4 Preview of Results . . . 4
- 1.5 Organization of the Dissertation . . . 6

2 Related Work **7**
- 2.1 Task Allocation . . . 7
  - 2.1.1 Single-robot task assignment . . . 7
  - 2.1.2 Multi-robot task assignment . . . 9
- 2.2 Coalition Formation . . . 9
- 2.3 Teamwork . . . 11
- 2.4 Information Invariants . . . 12
- 2.5 Schema Theory . . . 13
- 2.6 Multi-Robot Box Pushing . . . 13
- 2.7 Summary . . . 14

3 Foundation of ASyMTRe **16**
- 3.1 Schema Theory . . . 16
- 3.2 Information Types . . . 17
- 3.3 Knowledge Base . . . 18
- 3.4 Potential Solutions . . . 18
- 3.5 Solution Quality . . . 20
- 3.6 Summary . . . 21

4 Centralized ASyMTRe **22**
- 4.1 Potential Configuration Space (PCS) . . . 23
- 4.2 Instantiation on Robot Teams . . . 24
- 4.3 Centralized ASyMTRe Configuration Algorithm . . . 25
- 4.4 Algorithm Analysis . . . 25
  - 4.4.1 Soundness . . . 27
  - 4.4.2 Completeness and optimality . . . 27
  - 4.4.3 Formal analysis of centralized ASyMTRe . . . 28
- 4.5 Summary of Results . . . 31

5 Distributed ASyMTRe-D **33**
5.1 Overview of Approach . . . 33
5.2 Distributed ASyMTRe-D Negotiation Protocol . . . 34
5.3 Summary . . . 38

6 Layering Coalition Formation With Task Allocation **39**
6.1 Overview of the Problem . . . 39
6.2 Motivating Example: Site Clearing Task . . . 40
6.3 Formalism of the Problem . . . 42
6.4 The Approach . . . 43
6.4.1 Low-level coalition formation . . . 45
6.4.2 Higher-level task allocation through auction . . . 45
6.5 Summary . . . 47

7 Experimental Validation **48**
7.1 Platforms . . . 48
7.1.1 Player and Stage . . . 48
7.1.2 Pioneer 3DX . . . 49
7.2 Implementation Details . . . 49
7.3 Multi-Robot Navigation Task . . . 50
7.3.1 Task description . . . 50
7.3.2 Experimental setup . . . 52
7.3.3 Vision-based fiducial marker detection . . . 54
7.3.4 Multi-robot navigation using centralized ASyMTRe . . . 56
7.3.5 Multi-robot transportation using distributed ASyMTRe-D . . . 57
7.4 Cooperative Multi-Robot Box Pushing . . . 61
7.4.1 Task description . . . 61
7.4.2 Box pushing protocols . . . 61
7.4.3 Discussion . . . 62
7.4.4 Experimental setup . . . 65
7.4.5 Physical box pushing using distributed ASyMTRe-D . . . 67
7.5 Comparison between Centralized ASyMTRe and Distributed ASyMTRe-D . . . 69
7.6 Site Clearing Task . . . 71
7.6.1 Task description . . . 71
7.6.2 Code reusability . . . 72
7.6.3 Simulation setup . . . 73
7.6.4 Physical robot experiments setup and results . . . 77
7.7 Summary . . . 77

8 Possible Future Extension: Peer-to-Peer Human-Robot Teaming **80**
8.1 Introduction . . . 80
8.2 The Problem . . . 82
8.3 Challenging Issues in Human-Robot Teaming . . . 82
8.3.1 How to represent human/robot capabilities? . . . 82
8.3.2 How do humans and robots work as peers? . . . 83

8.3.3 What and how do humans and robots communicate? 84
8.3.4 How to evaluate team performance? 85
8.3.5 How to adapt to changes? 86
8.4 The Approach 86
8.4.1 Agent capabilities 87
8.4.2 Information sharing 88
8.4.3 Human-robot interaction 89
8.4.4 Metrics and evaluation 89
8.5 Summary 90

9 Summary and Conclusions **91**
9.1 Summary of Contributions 91
9.2 Future Work 92

Bibliography **94**

Vita **102**

List of Tables

3.1 Connection constraints for schemas. 18

4.1 Factors that influence the analytical results. 28

5.1 The list of messages that are used in the protocol. 34
5.2 Ways to handle possible communication failures. 35

7.1 Environmental sensors on Pioneer 3DX mobile robots. 50
7.2 Schema class definition. 51
7.3 Robot class definition. 51
7.4 Eight types of robot with different sensing capabilities. 52
7.5 Input and output information types in the navigation task. 53
7.6 Input and output information types for corresponding schemas and their sensing costs and success probabilities. 53
7.7 The average time and the standard deviation of the three set of experiments over 10 trials per set. 59
7.8 Input and output information types in the cooperative multi-robot box pushing task. 66
7.9 Input and output information types for corresponding schemas and their sensing/time costs and success probabilities. 66
7.10 Comparison between centralized ASyMTRe and distributed ASyMTRe-D. . 71
7.11 Schemas reused in the site clearing task. 73
7.12 Robot capabilities in the site clearing task. 73
7.13 Average completion time in the box pushing task 76
7.14 Summary of experiments. 79

List of Figures

3.1 A typical sense-think-act style behavior-based architecture on a single robot. 17
3.2 Two types of inputs to a schema. 18
3.3 Examples of schema connections for centralized ASyMTRe. 19
3.4 Examples of schema connections for distributed ASyMTRe-D. 20

4.1 An example of reducing OCS to PCS, the potential solutions, and their instantiation on robots. 23
4.2 A special case that might produce incomplete results. The numbers indicate utilities. 27
4.3 The number of potential solutions is decided by the number of inputs ("OR" condition) and the number of redundant schemas. 30
4.4 The number of potential solutions is decided by the number of inputs ("AND" condition) and the number of redundant schemas. 30
4.5 Scalability of the centralized ASyMTRe configuration algorithm. 31
4.6 The quality of the solution increases over time. 32

6.1 The site clearing application. 41
6.2 A partial-order plan for the site clearing application. 42
6.3 The relationships between tasks, coalitions and robots. 44
6.4 An example task tree for the Remove Obstacle task. 45

7.1 Pioneer 3DX mobile robot. 49
7.2 Two ways to connect the schemas in the navigation task. 54
7.3 Cylindrical marker design to provide unique visual ID, relative position, and orientation information. 55
7.4 Physical robot implementation of the navigation task. 56
7.5 The simulation results of a more complex multi-robot navigation task, where robots are given successive goal positions. 58
7.6 ASyMTRe-D handles partial robot failures in the navigation task. 60
7.7 ASyMTRe-D handles simulated robot failures in the navigation task. 60
7.8 Three different ways to connect schemas on a robot to accomplish the box pushing task. 67
7.9 A series of snapshots taken during the box pushing task. 68
7.10 The comparison between ASyMTRe and ASyMTRe-D in time and utility. . 70
7.11 The comparison of solution qualities between ASyMTRe and ASyMTRe-D. 70
7.12 A series of states of the system that illustrated the task execution process. . 74

7.13 Using the same instance as in Figure 7.12, this figure shows the major events from the auctioneer's point of view. 76
7.14 Using the same instance as in Figure 7.12, this figure shows the state of each robot during the task execution. 76
7.15 The site setup for the site clearing task. 77
7.16 Major events from the auctioneer's point of view in the physical experiment. 78
7.17 A series of snapshots taken during one run of the site clearing task. 78
7.18 Results of the site clearing task. 78

8.1 The human-robot teaming approach. 87
8.2 An example of how the human commands and sensing data are processed in the behavior control architecture. 88

Chapter 1

Introduction

1.1 The Problem

This dissertation addresses the problem of synthesizing task solutions for a heterogeneous robot team to accomplish multi-robot tasks by sharing sensory and computational resources. Typically, a multi-robot task requires a *strongly cooperative* solution [Brown and Jennings, 1995], meaning that the task is not trivially serializable, so that it cannot be decomposed into subtasks that can be completed by individual robots operating independently; instead, it requires robots to act in concert to achieve the task. Sometimes, this type of task is also called a *tightly-coupled* or *tightly-coordinated* task. Robots that join together to solve this type of multi-robot task are referred to as *coalitions* by some researchers [Gerkey and Mataric, 2004]. In this dissertation, robots also form coalitions for accomplishing these strongly cooperative multi-robot tasks. Even though I am not using the traditional definition of coalition by calculating payoffs as in game theory [Luce and Raiffa, 1957], the use of robot coalition here shares the same motivation behind coalition formation as mentioned in [Shehory and Kraus, 1995]; that is, robots in a coalition should work together to share resources and cooperate on task execution due to their decision that they would benefit more from working together as a coalition than they would working individually. In the context of this dissertation, I use the word "coalition" to explicitly mean coalitions for tightly-coupled multi-robot tasks.

Researchers generally agree that multi-robot systems have several advantages over single robot systems [Arai et al., 2002, Cao et al., 1997]. The most common motivations for developing multi-robot system solutions are that: (1) the task complexity is too high for a single robot to accomplish; (2) the task is inherently distributed; (3) building several resource-bounded robots is much easier than having a single powerful robot; (4) multiple robots can solve problems faster using parallelism; and (5) the introduction of multiple robots increases robustness through redundancy. Although multi-robot systems are essential in today's applications, there are many challenges in developing such systems. The issues that must be addressed in developing multi-robot solutions are dependent upon the task requirements and the sensory and effector capabilities of the available robots. The earliest research in multi-robot systems focused on swarm intelligence approaches using homogeneous robot teams, inspired by insect societies (e.g., [Beni and Wang, 1989, Mataric, 1992]). In these approaches, individual robots typically perform the same subtask in the

same environment, resulting in global group behaviors that emerge from the local interaction of individual robots. The fundamental research challenge in these systems is designing the local control laws so as to generate the desired global team behavior. Other types of robot systems involve heterogeneous robots, which have differing sensor and effector capabilities. In these teams, the mapping of tasks to robots is much more important to the efficiency of the system, since robots vary in the quality of their solutions to tasks. Traditionally, this problem has been called the multi-robot task allocation (MRTA) problem. Gerkey [Gerkey and Mataric, 2004] has developed a taxonomy for describing these problems, distinguishing robots as either *single-task* (*ST*) or *multi-task* (*MT*), tasks as either *single-robot* (*SR*) or *multi-robot* (*MR*), and assignment types as either *instantaneous* (*IA*) or *time-extended* (*TA*). The vast majority of prior work [Botelho and Alami, 1999, Dias, 2004, Gerkey and Mataric, 2002, Parker, 1998b, Werger and Mataric, 2000, Zlot and Stentz, 2006] on multi-robot task allocation has addressed single-task robots executing single-robot tasks using either instantaneous assignment (denoted ST-SR-IA) or time-extended assignment (denoted ST-SR-TA). Some recent work also addresses the allocation of multi-robot tasks [Jones et al., 2006, Kalra et al., 2005, Lin and Zheng, 2005]. This dissertation addresses the ST-MR problem in the MRTA taxonomy – namely single-task robots performing multi-robot tasks. I am particularly interested in the development of heterogeneous robot coalitions to accomplish multi-robot tasks.

My approach to accomplishing a multi-robot task is for robots to form coalitions as necessary by coupling coalition members' sensing and computational capabilities. More generally, multi-robot coalition formation deals with the issue of how to organize multiple robots into subgroups to accomplish a task that efficiently uses the available robot sensing and computational capabilities. Coalitions are typically considered to be temporary organizations of entities that bring together diverse capabilities for solving a particular task that cannot be handled by single robots. Coalitions are similar to the ideas of teams, except that they have a shorter duration and can change frequently over time. I am interested in developing automated techniques for coalition formation, especially when the specific task solution is highly dependent upon the available capabilities of the heterogeneous robots, and thus cannot be specified in advance. This is especially challenging in heterogeneous robot systems, in which sensory and computational resources are distributed across different robots. For such a group to accomplish the task as a whole, it must determine how to couple the appropriate capabilities from each robot, resulting in automatically formed coalitions that serve specific purposes. Multiple coalitions may be generated to compete for tasks at the higher level. Traditional task allocation techniques are also integrated on top of the coalition formation system to allocate one or more multi-robot tasks to the generated coalitions. With the success of the automated coalition formation techniques for multi-robot teams, I also plan to extend it for human-robot teaming.

1.2 Illustrating Examples

To motivate the need for robot coalitions, I now present several examples. In 2002, our University of Tennessee (UT) Distributed Intelligence Laboratory (DILab) began working on a project called Software for Distributed Robotics (SDR), aimed at the deployment of a team of 100+ heterogeneous mobile robots in an indoor surveillance and reconnaissance task. As

outlined in [Parker, 2003], the types of behaviors needed to deploy a mobile sensor network are shown to be highly dependent upon the total robot team composition/capabilities. For homogeneous robots with the ability to detect obstacles and other team members, a potential-field-based dispersion algorithm is appropriate to achieve mobile sensor network deployment (e.g., [Howard et al., 2002]). For teams involving a highly capable "mother ship" robot and many simple sensor nodes, deployment using a marsupial delivery method is appropriate, where the highly capable robot carries the simple sensor nodes and transfers them to a desired location. For teams with a few robots that can perform laser localization, obstacle avoidance, and visual detection of teammates, plus many simpler robots that do not have these capabilities, an assistive navigation approach is appropriate, in which the more capable robots assist in the navigation of the simpler robots (e.g., [Parker et al., 2004]).

To further elaborate on the type of applications I am particularly interested in, consider a simple multi-robot navigation application where a team of robots needs to navigate from a set of starting positions to a set of goal positions. Various approaches can be used to perform the task. If every robot knows its current position relative to its goal position, a straightforward approach would be to have each robot navigate independently, for example, using laser range scanner-based localization. However, if some robots do not have the sensing capability to localize, an alternative approach would be for the more capable robots to guide the less capable robots towards their goals by providing them with positioning information, for example, using a camera to estimate the relative position of another robot. Alternative approaches could be imagined in other team compositions. The important point here is that the resulting robot behaviors for accomplishing the task could be very different depending upon the combination of sensors that is available for solving the task.

From the above examples, we can see that it is challenging to create robot behavior code that is flexible enough to solve tasks in a variety of ways. Ideally, I would like to design the behavior code for an individual robot to be independent of any particular robot team composition, towards the goal of flexible, reusable robot behaviors.

1.3 The Approach

To address the problem and challenges, I present the ASyMTRe approach, which stands for Automated Synthesis of Multi-robot Task solutions through software Reconfiguration, pronounced "Asymmetry" (resembling the heterogeneity of the team). This approach is aimed at increasing the autonomous task solution capabilities of heterogeneous multi-robot systems by changing the fundamental abstraction that is used to represent robot competences from the typical "task" abstraction to a biologically-inspired "schema" [Arkin, 1987, Lyons and Arbib, 1989] abstraction, and providing a mechanism for the automatic reconfiguration of these schemas to address the multi-robot task at hand. In the old abstraction, the human writes software in advance for robots to accomplish subtasks or roles, which is sensor, effector, and task dependent. In the new abstraction, robot capabilities are viewed as low-level building blocks, and the human writes software for building blocks, which are sensor and effector dependent, but task independent. Thus the low-level building blocks can be configured in many ways to solve different tasks or the same task in different ways based on the team capabilities.

Built upon the theory of *information invariants* [Donald et al., 1993, Donald, 1995, Donald et al., 1997] and biologically inspired *schema theory* [Lyons and Arbib, 1989, Arkin, 1987, Arkin, 1998], the ASyMTRe approach maps robot capabilities (schemas) to the required flow of information through the multi-robot system, automatically configuring the connections of schemas within or across robots to synthesize valid and efficient multi-robot behaviors for accomplishing the team objectives. The connection across robots enables robots to form coalitions based on the information sharing requirement. To ensure an efficient solution, I defined a utility function that takes into account the cost and performance of all candidate solutions. The goal is to minimize the cost while optimizing the performance for the team. When applying the ASyMTRe approach, the robot team is able to generate task solutions based on sensor sharing through coalition members. Ultimately, the ASyMTRe approach is layered with prior task allocation approaches, with ASyMTRe serving as a lower-level solution generator for *how* to solve single multi-robot tasks by forming coalitions, with the higher-level, more traditional task allocation strategies using these lower-level solutions (coalitions) for achieving multiple multi-robot tasks by allocating tasks to coalitions. As a possible future extension, the ASyMTRe approach could be extended and applied to human-robot teams to serve as the basic foundation for forming human-robot coalitions.

1.4 Preview of Results

The major contribution of this dissertation is the development of ASyMTRe - a novel approach that autonomously configures solutions for heterogeneous robots to accomplish multi-robot tasks. It is the first approach that views robot capabilities at a more fine-grained schema level, maps the schemas to the flow of information required by a task, and automatically forms robot coalitions as task solutions. The generated solutions are in the form of schema connections built within and across networked robots. Thus, coalitions are formed through the connections. The team members within the same coalition can share sensory and computational information with each other in order to couple their different capabilities efficiently for solving a multi-robot task. The development of ASyMTRe involves the design and construction of the following sub-systems:

- The centralized ASyMTRe configuration algorithm, which I first introduced in [Tang and Parker, 2005a]. This anytime algorithm generates multi-robot coalitions using complete information of the robot team, with solution quality increasing as more time is available for the reasoning process. I have formally proved that the configuration algorithm is sound, and it is complete and optimal given enough time.

- The distributed ASyMTRe-D negotiation protocol, which I first introduced in [Tang and Parker, 2005c]. This protocol uses inter-robot communication to enable distributed formation of coalitions. This protocol is fully distributed such that team members do not need to share their sensing and computational information with each other, while the final decision is optimized locally. The ASyMTRe-D approach offers a tradeoff of increased robustness versus solution quality compared to the centralized ASyMTRe approach.

- Coalition formation layered with task allocation. To extend the capability of ASyMTRe for solving multiple tasks, I also developed an auction-based task allocation approach [Botelho and Alami, 1999, Dias, 2004, Gerkey and Mataric, 2002] and integrated it with the ASyMTRe approach. The resulting system is a single framework that enables the robot team to form coalitions at the lower level, with the coalitions competing for task assignments at the higher level.

- A basis for coalition formation in human-robot teams. The ASyMTRe approach can be extended to support human-robot teaming. Particularly, it is suitable for a system that supports both traditional supervisory control and peer-to-peer human-robot interaction. This work remains an interesting future work for implementation.

I have implemented the centralized ASyMTRe, distributed ASyMTRe-D and the auction-based task allocation in simulation and on physical robot teams performing a variety of applications, including: multi-robot navigation, cooperative multi-robot box pushing, and site clearing applications. The experimental results have validated the ASyMTRe approach and addressed the following key issues:

- Autonomous coalition formation based on the available team capabilities and the task requirements.

- Tightly-coupled cooperation among coalition members through the sharing of sensory or computational information.

- Robustness of the system with ASyMTRe reconfiguring solutions upon failures.

- Code reusability for the multi-robot team by viewing robot capabilities in terms of task-independent schemas.

- Applicability of the ASyMTRe approach to a large class of multi-robot applications.

- Integration with the higher-level task allocation approach so that the resulting system successfully handles multiple multi-robot tasks by forming coalitions to compete for each task.

Many researchers have addressed the allocation of single-robot tasks [Parker, 1998b, Werger and Mataric, 2000, Botelho and Alami, 1999, Gerkey and Mataric, 2002, Dias, 2004, Zlot and Stentz, 2006]. Some recent work also addresses the allocation of multi-robot tasks [Jones et al., 2006, Kalra et al., 2005, Lin and Zheng, 2005]. The ASyMTRe approach is different in that I am addressing the multi-robot tasks through the dynamic configuration of low-level behavioral building blocks (schemas) instead of predefined plans or roles. This dissertation emphasizes the change of abstraction of robot capabilities from the traditional task or sensor perspective to a more fine-grained schema perspective, and thus enables the team to configure solutions that are flexible for different team capabilities and task requirements.

1.5 Organization of the Dissertation

The organization of this dissertation is as follows:

Chapter 2 Related Work. This chapter presents a review of the related work in: task allocation, coalition formation, teamwork, schema theory, information invariants theory, and multi-robot box pushing.

Chapter 3 Foundation of ASyMTRe. This chapter describes the foundation of the ASyMTRe approach: the change of abstraction of robot capabilities from the traditional sensor/task-dependent representation to task-independent schema representation; and the information types corresponding with each schema that enables the connections of schemas within and across robots.

Chapter 4 Centralized ASyMTRe. This chapter presents the centralized ASyMTRe configuration algorithm that forms robot coalitions based on the robot team's complete information and formally evaluates the soundness, completeness and optimality of the centralized algorithm.

Chapter 5 Distributed ASyMTRe-D. This chapter presents the distributed ASyMTRe-D negotiation protocol, a fully distributed system that enables robots to form coalitions based on their local knowledge only.

Chapter 6 Layering Coalition Formation with Task Allocation. This chapter describes the higher-level auction-based approach that I propose for allocating tasks to coalitions.

Chapter 7 Experimental Validation. This chapter presents the experiments that I have implemented for validating the ASyMTRe approach and its integration with the task allocation approach. Experiments are performed both in simulation and on physical robots on three different applications.

Chapter 8 Peer-to-Peer Human-Robot Teaming. This chapter describes the potential to extend the ASyMTRe approach for humans and robots to form coalitions as needed to accomplish a task. Peer-to-peer human-robot interaction could be realized through information sharing.

Chapter 9 Summary and Conclusions. This chapter summarizes the main contributions of this dissertation and describes possible future work.

Chapter 2

Related Work

In recent years, research interest in multi-robot cooperation has been focusing on the problem of allocating tasks to robots in an efficient way. The research in multi-agent societies has also provided many useful mechanisms for cooperating agent capabilities, which are also starting to be applied to multi-robot systems [Vig and Adams, 2005]. In this chapter, I review several topics related to the ASyMTRe work I present in this dissertation: multi-robot task allocation, multi-agent coalition formation, and teamwork. Additionally, I also briefly review the foundations of this work: the information theoretic work of Donald et al. [Donald et al., 1993, Donald, 1995, Donald et al., 1997] and the basic building blocks – schema theory [Lyons and Arbib, 1989, Arkin, 1987, Arkin, 1998]. Finally, since the ASyMTRe approach has been applied to a cooperative box pushing task, I also present the related work in multi-robot box pushing.

2.1 Task Allocation

Task allocation is one of the essential problems in heterogeneous robot teams. It is the problem of determining a suitable mapping between robots and tasks, such that the task pool can be completed efficiently. Since it has been proven that developing the optimal mapping between robots and tasks is NP-hard [Parker, 1998b], existing approaches for multi-robot task allocation (MRTA) use heuristic greedy strategies to achieve the mapping. Usually, a task is decomposed into independent subtasks, hierarchical task trees, or roles either by a general autonomous planner, or simply by the designer. Independent subtasks can be achieved concurrently, while subtasks in task trees are achieved according to their interdependence. Roles are defined by action strategies for achieving part of the application. In these settings, many mechanisms have been developed by selecting the current best mapping between subtasks/roles and team members considering team member's capabilities and perhaps performance. To determine which mapping is the best, some types of *utility* estimation are introduced.

2.1.1 Single-robot task assignment

According to the formal analysis of the MRTA problem [Gerkey and Mataric, 2004], most of the existing work today addresses the assignments of single-robot tasks to single-task

robots. Parker [Parker, 1998a] describes a behavior-based control architecture ALLIANCE that allows a team of robots to select appropriate actions to contribute to a mission based on the mission requirements, the activities of other teammates, and the robots' internal states. Adaptive action selection is achieved by *motivational behaviors*. This motivational behavior is a combination of *impatience* and *acquiescence* that measure its own and its teammates' fitnesses of performing a certain subtask. If the robot that is currently performing a subtask is inefficient, its own acquiescence level drops quickly, and other teammates' impatience levels increase rapidly. When the acquiescence level drops below a threshold, the robot will give up the task and other teammates will take over depending on their impatience levels. Fault tolerance is achieved through this architecture. BLE [Werger and Mataric, 2000] is another behavior-based approach to multi-robot coordination. Different from ALLIANCE, it is based on a subsumption style behavior control architecture. It allows robots to efficiently execute tasks by continuously broadcasting locally-computed eligibilities and only selecting the robot with the best eligibility to perform the task. In this case, task allocation is achieved through behavior inhibition.

Another large body of research related to task allocation comes from the Distributed Artificial Intelligence (DAI) community, which typically uses a distributed negotiation-based mechanism to determine the mapping between tasks and robots. After Smith [Smith, 1980] first introduced the Contract Net Protocol (CNP), many approaches addressing multi-robot cooperation through negotiation were developed. For example, there are auction-based approaches such as M+ [Botelho and Alami, 1999], MURDOCH [Gerkey and Mataric, 2002] and First-price auctions [Dias, 2004, Zlot and Stentz, 2006]. In these works, a task is divided into subtasks for the robots to bid and negotiate to carry out the subtasks. Each robot can estimate the utility of executing a subtask, which measures the quality and cost factors of the resulting actions. The goal is to greedily assign subtasks to the robot that can perform the task with the highest utility. Among them, the work of Gerkey [Gerkey and Mataric, 2002] employs a resource centric, *publish/subscribe* communication model to carry out auctions, which has the advantage of anonymous communication. In MURDOCH, a task is represented by the required resources, such as the environmental sensors. The methods for how to use such a sensor to generate satisfactory results is preprogrammed into the robot. The TraderBots approach [Dias, 2004] applies market economy techniques for generating efficient and robust multi-robot coordination in dynamic environments, which has been further studied in [Kalra et al., 2005,Zlot and Stentz, 2006]. In a market economy, robots act based on self interests. A robot receives revenue and incurs cost when trying to accomplish a task. The goal is for robots to trade tasks through auctions/negotiations such that the team profit (revenue minus cost) is optimized. In addition to the above auction methods applied to the robotic systems, many researchers have also studied the auction mechanism and applied it to other areas such as information and decision systems [Bertsekas, 1988], which could also provide many useful insights to our problem.

From the role perspective, there are applications that divide a task into specific roles that are defined by the human designer based upon the knowledge of the application and the types of available robots. Roles can then be dynamically assigned to robots in a similar way as in the auction-based approaches (e.g. [Jennings and Kirkwood-Watts, 1998,Simmons et al., 2000]).

2.1.2 Multi-robot task assignment

Some recent work in task allocation [Jones et al., 2006, Kalra et al., 2005, Lin and Zheng, 2005] begins to address multi-robot task allocation, where team members need to tightly cooperate with each other to accomplish the task.

In [Jones et al., 2006], the authors present an approach that enables robots to form heterogeneous teams for executing coordinated tasks. This approach and the ASyMTRe approach share the same motivation in that heterogeneous robots have different individual capabilities, thus different combinations of individual capabilities may solve a tightly-coordinated task in different ways. When the team information is limited prior to a task, the team should rely on dynamic solution strategies. In [Jones et al., 2006], team coordination is achieved through Plays, which provides a fixed team plan that describes a sequence of actions for each role to achieve the team goal(s). At the high level, this approach uses the TraderBots approach for each robot to select the proper play to accomplish a certain role and bid for the task.

The Hoplites approach [Kalra et al., 2005] focuses on the selection of an appropriate joint plan for the team to execute by incorporating joint revenue and cost in the bid. However, this approach does not specify a particular algorithm for searching joint action spaces and recommends algorithms that sacrifice optimality for speed. The low level solution strategies are predefined for a robot to accomplish a selected plan. The work in [Lin and Zheng, 2005] solves multi-robot tasks by matching task required capabilities with robot capabilities and enabling robots to auction for multi-robot tasks through combinatorial bids, with the assumption that robots joining together as a team have predefined solution strategies for accomplishing a task cooperatively.

The ASyMTRe approach also addresses multi-robot tasks with heterogeneous robots. Unlike the above approaches for single-robot tasks, however, I do not assume that a task can be subdivided into independent subtasks or roles, and thus can be allocated to single robots without the consideration of cooperation between robots. Instead, the ASyMTRe approach addresses tasks that require more than one robot working together. My approach is also different from the allocation of multi-robot tasks in that I do not assume that robots have predefined behavioral-level solution strategies on how to accomplish a certain task, unlike the fixed plays, plans, or roles that are provided to the robots in [Jones et al., 2006, Kalra et al., 2005, Lin and Zheng, 2005, Simmons et al., 2000]. By abstracting the task at the schema level rather than at the sensor level or task level, I believe that more flexible solution strategies can be generated that determine *how* to solve multi-robot tasks on the behavioral level, which are not dependent upon predefined solution strategies in the form of task decompositions, roles, plays, or plans that are specified in advance by the human designer.

2.2 Coalition Formation

Multi-robot coalition formation for multi-robot tasks deals with the issue of how to organize robots into subgroups to accomplish multi-robot tasks, using a strongly cooperative solution approach. The motivation behind coalition formation for multi-robot tasks is to enable team members to work together as a group to accomplish tasks that cannot be handled

by individual robots working independently (i.e., tasks that are not trivially serializable, as defined by Brown and Jennings [Brown and Jennings, 1995]). Since robots have different sensor, effector and computational capabilities, a team of resource-bounded robots may not individually possess all of the required capabilities to accomplish a task. However, they could work with other robots as a coalition to effectively accomplish the task objectives.

Research in multi-agent systems provides a variety of approaches to coordinate agent behaviors. In one of these approaches, agents are organized into coalitions to achieve a higher-level goal. In the survey of multi-agent organization [Horling and Lesser, 2004], the authors present various paradigms to organize multi-agent teams, among which the hierarchical, coalition-based, and team-based organizations are the most popular methods. The motivation behind coalition formation and my approach are similar in that the coalitions/subteams are formed with a purpose to accomplish the team-level goals. In particular, the work of Shehory [Shehory, 1998] inspired some aspects of my work. Shehory's work describes a method of allocating a set of interdependent tasks to a group of agents by forming coalitions. It assumes that tasks have a precedence order, and there is an efficient task allocation by the agents themselves, which is achieved through coalition formation. This problem is similar to the set-partitioning problem and is indeed NP-hard. However, by applying limitations on the permitted coalitions (e.g., the coalition size k), their greedy distributed set-partitioning algorithm has a low ratio bound $O(n^k)$. Their algorithms are also anytime algorithms, which return a solution that is better than their initial solution or other preceding solutions. They claimed that this anytime property is important for dynamic environments.

There are also other approaches that address the coalition formation issue in multi-agent teams. Sandholm et al. [Sandholm et al., 1999] present an approach to find coalitions via a partial search and the generated results are guaranteed to be within a bound from the optimum. Although their algorithm reduces the search space dramatically, it is still exponential in the number of agents, and thus is not applicable for teams with large size. When there are a large number of agents, self-organization helps to improve the performance and to generate coalitions dynamically based upon interactions and communication among local agents; the work such as [Low et al., 2004, Sims et al., 2003] falls in this category. Unlike the work in [Sims et al., 2003], which forms coalitions through negotiation, the work in [Lerman and Shehory, 2000] claims that the negotiation-style approach has a high requirement for computational and communicational capabilities of the whole system, and thus is impractical for large systems. Instead, they present a physics-based model to achieve coalition formations through local interactions among self-interested agents. Their work assumes that agents have basic information about some other agents via middle agents [Sycara et al., 1997], and thus minimal communication is required between agents. In fact, the work of Parker [Parker, 1993] has pointed out the tradeoffs between local control and global control, and stated that the designer must determine an appropriate balance between them to achieve cooperation without an excessive communication requirement. This idea is further explored in this dissertation in that there is also a tradeoff between the amount of shared information and the solution quality.

My work is different from the prior work in coalition formation in several ways. First, the representations of agents' or robots' capabilities are different. The ASyMTRe approach addresses the autonomous synthesis of cooperative behaviors at the low level of sensors and

schemas, thus it requires a more complicated method of describing each robot's capabilities since sensors and schemas are situated in the robots and they are not transferable [Vig and Adams, 2005]. My representation of the robot capabilities is essentially the same as in STRIPS planning [Fikes and Nilsson, 1971], although the problem is abstracted in a very different manner. In common STRIPS planning, the robot capabilities are usually represented by predicates such as *Has*(*laser*) and *Action*(*Goto*), which requires solution strategies to be previously incorporated into the STRIPS rules. I represent the capabilities at the level of schema, which allows the planner to be independent of pre-compiled solution strategies, enabling ASyMTRe to generate *how* to solve tasks in a much more general manner. Second, my work provides a method for robots to share distributed sensory and/or computational information with each other, and it results in directly executable cooperative robot control code.

A few researchers have addressed the formation of coalitions for *multi-robot tasks* [Parker and Tang, 2006, Vig and Adams, 2005]. The earlier versions of ASyMTRe [Parker and Tang, 2006] only generate solution strategies for one multi-robot task. In [Tang and Parker, 2006b], the ASyMTRe approach has been layered with a traditional auction-based task allocator and the resulting system is able to efficiently allocate multiple multi-robot tasks to automatically formed robot coalitions. The work in [Vig and Adams, 2005] also addresses multi-robot tasks. It identifies several challenges of applying multi-agent coalition formation techniques to multi-robot teams and aims to bridge the discrepancies between multi-agent and multi-robot teams. The proposed approach is a variation of the task allocation algorithm via coalition formation presented in [Shehory, 1998], with a new task format that takes in account the robot-situated capabilities (non-exchangeable), a balanced resource distribution among coalition members, etc. The fundamental difference between the integrated ASyMTRe approach and the above approach is the way I define robot capabilities. With the abstraction of robot capabilities in terms of fine-grained schemas instead of traditional sensor/task point of view, robot teams have more flexibility in their solution strategies by forming a variety of coalitions for solving the same task in different ways.

2.3 Teamwork

Similar to coalitions, agents form teams to work together to accomplish a common goal [Tambe, 1997]. The goal is to maximize the utility of the team as a whole, rather than that of the individual members [Horling and Lesser, 2004]. In [Hexmoor and Beavers, 2001], the authors state the differences between coalitions and teams: agents in a team work together to achieve a goal even if they may engage in activities that adversely affect their utilities. Coalitions are temporary groups that try to maximize their utilities. Coalitions are stable and efficient with a low cost of forming a coalition and a distributed load over coalescent agents. Teams are suitable for agents that must reason about how to achieve a goal, how to carry out the plan and accommodate environmental and agent changes concurrently, which are also the main challenges for team-based organization.

Several teamwork models have been developed to provide mechanisms for agents to negotiate with each other to agree upon a plan to achieve the team level goal. A common technique is the use of joint intentions (e.g. [Cohen et al., 1991]) to develop shared plans to achieve the team level goal. In this approach, a belief-goal-commitment model is presented

with formal definitions of events, belief, goal and mutual belief. There are also other models that reason about the proper teamwork, such as joint responsibility [Jennings, 1995] and SharedPlan [Grosz, 1996]. Based on some of the above theoretic work [Cohen et al., 1991,Grosz, 1996], Tambe built STEAM (Shell for Team) [Tambe, 1997], a general teamwork model that takes into account the flexibility in a dynamic environment and reusability for different task domains. Jennings' work [Jennings, 1995] is also an extension of the joint intention model, which is called joint responsibility. It defines pre-conditions which must be satisfied to start cooperation and generates plans for agents to behave during cooperation and in faulty cases.

Under the theories of joint intentions, agents negotiate with each other to agree upon a shared plan that all will follow. The agents typically select certain roles and interact based upon the roles that the agent team members are filling. The selection of roles depends on the requirements of the goal and the agents' capabilities. Usually, the requirements of the goal are also defined as a set of capabilities, thus an agent with such capabilities can perform the task. The above approaches provide powerful high-level models for problem solving and role assignment taking into account the team capabilities. However, they do not address the issue of how agents can autonomously determine their proper contributions to the solution based upon their sensing, effector, and behavior capabilities. These teaming approaches do not involve the lower level of decision-making. The ASyMTRe approach is aimed at automatically determining these low level solution strategies.

2.4 Information Invariants

My approach to dynamically configuring solution strategies on heterogeneous team members is inspired by the earlier work of Donald et al. [Donald, 1995] on information invariants. According to Donald, information invariants can be viewed as a mapping from tasks or sensors to some measure of information. This measure characterizes the intrinsic information required to perform the task, which is used to estimate the complexity of the task. Donald also explored properties of *situated sensor systems*, describing methods for transforming sensori-computational systems. Here, a *sensori-computational system* (SCS) is a module that computes a function of its inputs and its current pose or positions [Donald et al., 1993]. Donald showed that reductions from one sensori-computational system to another can occur based upon equivalences between communication, internal state, external state, computation, and sensors. Thus, different solution strategies are available for solving the same task.

While the main focus of the work of Donald was to measure the information complexity of robot tasks, I can use information invariants to characterize task requirements and team member capabilities, making it an abstraction through which heterogeneous team members can interact with each other. The main contribution of the ASyMTRe approach is that it automates the solution generation, whereas the original information invariants approach was just used to analyze existing systems. My ultimate goal is to develop solution strategies based upon the individual capabilities of each team member. By providing team members with their capabilities information, they are able to design solution strategies according to the flow of information required by the task.

2.5 Schema Theory

The basic building block of my work is a collection of schemas. The most famous foundation works on the schema theory are from Arbib [Lyons and Arbib, 1989] and Arkin [Arkin, 1987, Arkin, 1998]. In [Lyons and Arbib, 1989], a formal model of computation based on the characteristics of the robot domains is constructed, called robot schemas (RS). The schema description includes: a list of input and output ports, a local variable list, and a behavior, which defines how the input is processed to generate the output. Only the ports of the same data type can be connected together. At the lower level, each schema can be instantiated with the proper variables. A network of schemas can then be built if the output ports of one schema are connected to the inputs of another schema. At the higher level, a nested network is established to represent the collaboration among robots. In this model, task plans are defined as the sensory schemas, motor schemas and the possible connections between schemas. This work provides a solid foundation to my work. Compared with this work, the ASyMTRe approach dynamically connects the schemas at run time instead of using manually pre-defined connections. Furthermore, a schema is defined as a black box in our work. I specify the input and output information types of a schema, but I do not need to consider the "behavior" – the output data generation process. Information types define the specific sensing or computational data of a schema or a sensor, for example, *global position* of a robot. They are fundamentally different from data types. A task is only represented as the set of motor schemas that need to be activated. Schemas are situated in each robot and they are not connected to each other at the beginning of a task but are instantiated after the solution strategies are generated. Thus, the ASyMTRe approach provides an automated method for the configuration of schemas to accomplish a task rather than requiring a pre-defined solution.

In [Arkin, 1987, Arkin, 1998], Arkin presents a schema-based control for a mobile robot, where each of the behaviors (motor schemas) computes its reaction to its perceptual stimuli. These computations are summed and normalized to generate the motor commands for each robot. The robots can then react instantly to the current status of the real world. The schema theory is further extended to include perceptual schema, which processes the data from environmental sensors and extracts useful output information to other schemas, and defines a behavior to be a schema that is composed of a motor schema and a perceptual schema. The motor schema embodies the physical activity and the perceptual schema embodies the sensing. The same ideas are used in this dissertation. Additionally, a new schema is introduced to transfer information between distributed schemas, called *communication schema*.

2.6 Multi-Robot Box Pushing

Multi-robot box pushing is a typical cooperative application studied by many researchers. These applications typically assume that the box is too heavy or too long to be handled by one robot pushing at the middle point. Thus, a robot needs to push both ends of the box in turns, or more robots are required to cooperatively push the box towards a goal or along a certain path. The work of Kube et al. [Kube and Zhang, 1993, Kube and Zhang, 1996] addresses this issue using a team of homogeneous robots to collectively

push a box, which is inspired from the decentralized control mechanism exhibited by social insects. Another work [Sen et al., 1994] incorporates reinforcement learning into the box pushing problem to enable two agents to push the box without explicitly sharing information with each other. The action selection is influenced by the changes in the environment. To analyze the information requirement of the box pushing task, Donald et al. [Donald et al., 1994] present three box pushing protocols. The first protocol depends on the robot's capability of sensing the applied force on the box and communicating this information to each other to compute the net torque, which is used to decide the next step of pushing. The second protocol computes the relative displacements along the line of pushing instead of forces. These two protocols need explicit inter-robot communication. The third protocol requires no communication if the two robots can sense the relative orientation of the box. Parker [Parker, 1994b] also successfully demonstrates a fault-tolerant box pushing example for heterogeneous teams using the ALLIANCE architecture. The work of Mataric [Mataric et al., 1995] explores how communication can improve the performance of box pushing by resource-bounded robots, in this case, two Genghis-II six-legged robots. A recent application of box pushing [Gerkey and Mataric, 2002] divides the task into sub-tasks, and dynamically allocates the subtasks to robots. In this example, box pushing is achieved through two pusher robots and a watcher robot that directs the pusher robots towards the goal.

Although the above applications provide a variety of ways for robots to cooperate on a box pushing task, each box pushing protocol is a special design to allow a particular team composition to accomplish the task. By applying the ASyMTRe approach to the robot team, the robots can automatically configure solutions through different combinations of schema connections within or across robots. Eventually, the robot team can flexibly switch between various protocols based on the current team capabilities.

2.7 Summary

The ASyMTRe approach that I developed enables heterogeneous robots to form coalitions as solution strategies based upon their individual capabilities, and eventually, accomplish multi-robot tasks by working cooperatively. By forming coalitions, robots can share information with coalition members and thus accomplish tasks that are difficult for certain resource-bounded robots to accomplish individually. This work is closely related to the body of literature on multi-robot task allocation, and multi-agent coalition formation and teamwork. Prior research on multi-robot teams have focused on providing suitable assignments of tasks to robot team members [Botelho and Alami, 1999, Dias, 2004, Gerkey and Mataric, 2002, Parker, 1998b, Werger and Mataric, 2000, Zlot and Stentz, 2006]. These architectures typically assume the task solutions are provided in advance and are independent of the particular team composition/capabilities. Some recent work in multi-robot task allocation [Jones et al., 2006, Kalra et al., 2005, Lin and Zheng, 2005] also depends on predefined plans, joint plans or mapping of tasks to capabilities. The multi-agent community also provides a variety of approaches to coordinate agent behaviors, where agents are organized into coalitions to achieve a higher-level goal [Klusch and Gerber, 2002, Lerman and Shehory, 2000, Sandholm and Lesser, 1997, Sandholm et al., 1999, Shehory, 1998, Sims et al., 2003]. Additionally, many teamwork architectures have been developed to organize agents into teams to accomplish tasks [Beavers and Hexmoor, 2001, Cohen et al., 1991, Hexmoor

and Beavers, 2001, Jennings, 1995, Levesque et al., 1990, Parker, 1993, Tambe, 1997, Tidhar et al., 1998].

This work is different from the above work in that I do not assume that a multi-robot task can be decomposed into independent subtasks or roles that can be accomplished by individual robots. I also do not assume that robots have pre-defined plans or scripts on how to accomplish a task. My work goes beyond the traditional mapping of capabilities to tasks by abstracting the problem at the *schema* level, rather than the task or sensor level, and by mapping schemas to information types, and eventually enabling a variety of solution strategies to be configured online based on different team capabilities and the flow of information required to accomplish a task. In the following chapters, I present the ASyMTRe approach in detail.

Chapter 3

Foundation of ASyMTRe

This chapter describes the foundation of the ASyMTRe approach, as reported in [Tang and Parker, 2005a, Tang and Parker, 2005b] and [Tang and Parker, 2005c]. This foundation provides a theoretic basis for the development of the centralized ASyMTRe configuration algorithm and the distributed ASyMTRe-D negotiation protocol.

3.1 Schema Theory

ASyMTRe is based on a distributed extension to schema theory [Lyons and Arbib, 1989, Arkin, 1987, Arkin, 1998]. As such, the basic building blocks of ASyMTRe are collections of environmental sensors, perceptual schemas (PS), motor schemas (MS), and a simple new component, called *communication schemas* (CS), which transfer information between various schemas distributed across robots. The introduction of communication schemas helps to differentiate the connections built within a robot from the connections across robots. The input to an environmental sensor is a specific physical sensor signal. Its output represents a set of features of the sensory data (e.g., range and intensity), which is connected to the input of a perceptual schema. Perceptual schemas process output from environmental sensors to provide information (e.g., distance, angle) to motor schemas, which then generate an output control vector (e.g., velocity) corresponding to the way the robot should move in response to the perceived stimuli. When multiple motor schemas are operating in parallel, the output vectors are summed to generate the resultant robot motion. All schemas are assumed to be pre-programmed into the robots at design time, and represent the fundamental individual capabilities of the robots. In most prior work, the connections between schemas are pre-defined (see Figure 3.1) and each robot employs a sense-think-act behavior-based architecture. In my case, the connections between schemas are not fixed at design time, but are configured at run time. ASyMTRe automatically determines the proper connections between sensors and schemas, across multiple robots, to ensure that the team-level goals are achieved. ASyMTRe enables the team to design new solution strategies at run-time based upon the team composition.

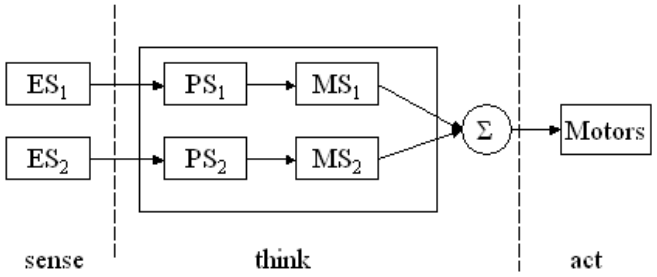


Figure 3.1: A typical sense-think-act style behavior-based architecture on a single robot.

3.2 Information Types

ASyMTRe is inspired by the information theoretic work of Donald et al. [Donald, 1995], which showed the equivalences between different combinations of sensing, communication, and action based upon information content. ASyMTRe allows robots to reason about how to solve a multi-robot task based upon the fundamental information needed to accomplish the task. The information needed to activate a certain schema remains the same regardless of the way that the robot may obtain or generate it. Thus, inputs and outputs of all schemas can be labeled with a set of information types that are unique to the task. Note that I use the term *information types* as distinct from *data types*. Thus, semantics of the information is built into these information types, and does not just refer to a data type (such as boolean or integer). For example, the input information types of a *go_to_goal* schema could be {*current_position, goal_position*}, and its output types could be the specific motor commands. I define the inputs and outputs of the schemas to be the set of information types $F = \{F_1, F_2, ...\}$. For schema S_i, I^{S_i} and $O^{S_i} \subset F$, represent the set of inputs and outputs of S_i, respectively. As in [Lyons and Arbib, 1989], I assume that each schema has multiple inputs and outputs. As shown in Figure 3.2, there are two types of inputs to a schema. The solid-line arrows going into a schema represent an "OR" condition, meaning that it is sufficient for the schema to only have one of the specified inputs. The dashed-line arrows represent an "AND" condition, where all the indicated inputs are needed to produce a result. For example in Figure 3.2, MS_1 can calculate output if it receives either F_1 or F_2. However, MS_2 can produce output only if it receives both F_1 and F_2. An output of a schema can be connected to an input of another schema if and only if their information labels match. Using the mapping from schemas to information types, schemas can be configured within and across robots to build proper connections for accomplishing the task. Once the interconnections between schemas are established, the robot team members have executable code to accomplish their task. Since the solution strategies are configured at the schema level, rather than the sensor level, ASyMTRe can determine *how* to solve tasks in a much more general manner.

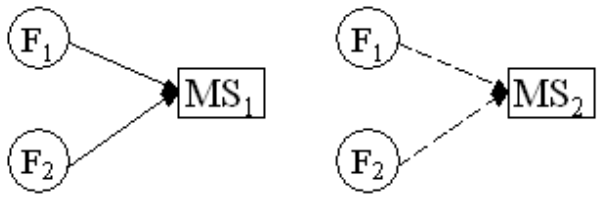


Figure 3.2: Two types of inputs to a schema. The solid-line arrow represents an "OR" condition. The dashed-line arrow represents an "AND" condition.

Table 3.1: Connection constraints for schemas.

Sensor/Schema	**Input Sources**	**Output Feeds into:**
ES	Sensor Signals	PS, MS
PS	ES, PS or CS	PS, CS or MS
CS	PS, or CS	PS, CS, or MS
MS	PS, CS, or ES	Actuators

3.3 Knowledge Base

Each robot has a knowledge base that describes its sensing and behavioral capabilities. The knowledge base is represented as (T, R_i, U_i), where $T = \{MS_1, MS_2, \cdots\}$ is the set of motor schemas that define the team-level task to be achieved, along with application-specific parameters as needed, such as the weight of the box. R_i provides basic information about robot R_i, and U_i provides utility information to be defined later. A robot, R_i, is represented by $R_i = (ES^i, S^i)$. ES^i is a set of environmental sensors that are installed on R_i, where $O^{ES^i_j} \subset F$ is the output of ES^i_j. S^i is the set of schemas that are pre-programmed into R_i at design time. Each schema is represented by $\{S^i_j, I^{S^i_j}, O^{S^i_j}\}$, with each schema's input and output information. A schema can be activated if and only if its input can be obtained from the output of schemas or sensors on the local robot or can be directly transferred from other robots. Additionally, a set of *Connection Constraints* specifies the restrictions on correct connections between various schemas, as shown in Table 3.1.

3.4 Potential Solutions

A *potential solution* is one possible connection of schemas within or across robots such that the task can be accomplished. A potential solution must satisfy the following: for all $MS_j \in T$, the inputs of MS_j are satisfied, along with all the inputs from the schemas that feed into MS_j. It is important to note that a potential solution is one way to connect schemas for a single robot to activate the motor schemas specified in the task, while not a solution for the entire team-level task. When connecting schemas based on their inputs and outputs, loops might be generated within a potential solution. However, the depth of a loop can be controlled by applying constraints on the connection process, such as limiting the number of times the same type of information type is communicated between two robots.

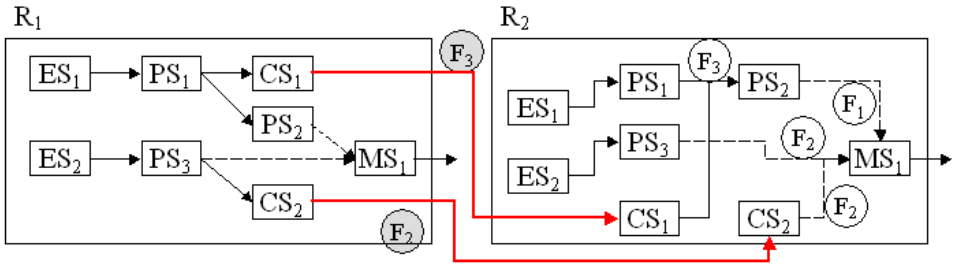


Figure 3.3: Examples of schema connections for centralized ASyMTRe.

Additionally, the utility calculation introduced in the next section also helps to eliminate the solution with a loop when comparing with other potential solutions without loops.

In ASyMTRe, a potential solution has two different representations because of the different degrees of robot capabilities sharing. At the beginning of task, the centralized reasoner, with all the information of each robot team member, generates an exhaustive list of potential solutions for each team member to accomplish a portion of the team-level task. Here, a potential solution is represented by a set of schemas that need to be activated on a local robot and on other robots in order to accomplish the task. It is defined in the following form:

$$PoS_i = (S_x, [S_y]). \tag{3.1}$$

where PoS_i is the ith potential solution in the list of all potential solutions, S_x is the set of x schemas that need to be activated on the local robot, and S_y is the set of y schemas that need to be activated on other robots. For example, in Figure 3.3, if I assume that $T = \{MS_1\}$ and $R = \{R_1, R_2\}$, the list of potential solutions for R_2 to accomplish its portion of the task is:

- $\{PS_1, PS_2, PS_3, MS_1\}$, provided that a robot has both ES_1 and ES_2.
- $\{PS_1, PS_2, CS_2, [PS_3], [CS_2], MS_1\}$, provided that a robot only has ES_1 and another robot can provide F_2 through activating PS_3 and CS_2 on that robot.
- $\{[PS_1], [CS_1], CS_1, PS_2, PS_3, MS_1\}$, provided that a robot only has ES_2 and another robot can provide F_3 through activating PS_1 and CS_1 on that robot.
- $\{[PS_1], [CS_1], CS_1, PS_2, [PS_3], [CS_2], CS_2, MS_1\}$, provided that another robot can provide both F_2 and F_3 by activating the proper schemas.

In distributed ASyMTRe-D, a potential solution is represented by the schemas that need to be activated on the local robot and a set of information types that must be obtained from other robots to fulfill a task. A distributed reasoner running on every robot, with the knowledge of the local robot only, generates an exhaustive list of potential solutions for the local robot to accomplish the task. Without the capabilities information on other robots, a potential solution in distributed ASyMTRe-D is defined in the following form:

$$PoS_i = (S_x, F_y). \tag{3.2}$$

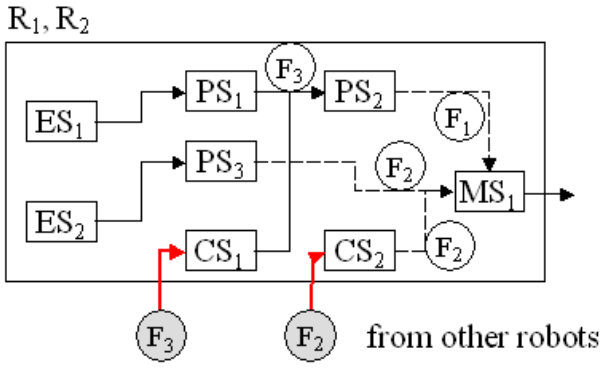


Figure 3.4: Examples of schema connections for distributed ASyMTRe-D.

where PoS_i is the ith potential solution in the list of all potential solutions (either locally or globally), S_x is the set of x schemas that need to be activated on the local robot, and F_y is the set of y information types that need to be transferred from other robot(s). For example, in Figure 3.4, if I assume that $T = \{MS_1\}$ and $R = \{R_1, R_2\}$, the list of potential solutions for both robots is:

- $\{PS_1, PS_2, PS_3, MS_1\}$, provided that a robot has both ES_1 and ES_2.
- $\{PS_1, PS_2, MS_1, CS_2, F_2\}$, provided that a robot only has ES_1 and another robot can provide F_2.
- $\{PS_2, PS_3, MS_1, CS_1, F_3\}$, provided that a robot only has ES_2 and another robot can provide F_3.
- $\{PS_2, MS_1, CS_1, CS_2, F_2, F_3\}$, provided that another robot can provide both F_2 and F_3.

3.5 Solution Quality

With multiple potential solutions available, I introduce *utility* to measure their qualities, which is decided by each robot's current sensing capabilities and the estimated utility of the particular solution it chooses. I would like each robot to select a solution that is the most efficient, or least costly. I define a *sensori-computational system* (SCS) [Donald, 1995], which is a module that computes a function of its sensory inputs and produces outputs. It is represented by $SCS_j^i = (S_j^i, ES_j^i, O^{S_j^i})$, where S_j^i is the jth PS/CS on R_i, ES_j^i is the sensory input, and $O^{S_j^i}$ is the output. Each SCS_j^i is assigned a cost C_j^i and a success probability P_j^i, where C_j^i $(0 \leq C_j^i \leq 1)$ represents the sensing or computational cost of using ES_j^i and P_j^i $(0 \leq P_j^i \leq 1)$ represents the success rate of S_j^i to generate a satisfactory result*. Sensing cost is determined by the sensory and computational requirements of the solution.

*In fact, the utility of a solution can also consider other aspects, such as the quality of information, frequency of the output, computational complexity, time, etc. The utility calculation could be extended to include different aspects as needed.

Perceptual processes with a significant amount of sensor processing, such as laser scan matching or image processing, are given higher sensing costs. Perceptual processes with a relatively low processing requirement, such as DGPS, are assigned lower sensing costs. Success probability is an estimated value based upon experience. Perceptual processes that are easily influenced by environmental factors, such as image processing under different lighting conditions, are given lower success probabilities. Otherwise, they are given higher success probabilities. I calculate the utility[†] of activating SCS_j^i or producing $O^{S_j^i}$ (or F_k) by:

$$U(SCS_j^i) = U(O^{S_j^i}) = w \cdot P_j^i - (1-w) \cdot (C_j^i / \max_j(C_j^i)). \tag{3.3}$$

$$U^i = \sum_p U(SCS_p^i) + \sum_q U(SCS_q^j) + \sum_k U(F_k) \tag{3.4}$$

$$U = \sum_i U^i \tag{3.5}$$

where w is a weight factor from 0 to 1. The utility of a potential solution PoS for R_i in a centralized system (3.1) is the sum of utilities of all the SCS_p^i that need to be activated on the local robot R_i plus the sum of utilities of all the SCS_q^j that need to be activated on the networked robot R_j. The quality of a potential solution PoS for R_i in a distributed system (3.2) is the sum of the utilities for all the SCS_p^i that need to be activated on the local robot R_i plus the utilities of the information types F_k that are obtained from other robots. The goal is to maximize the utility U^i of the selected potential solution for each robot and eventually maximize the overall utility U of the team.

3.6 Summary

In this chapter, I have described the theoretical basis of the ASyMTRe approach: schema theory and information invariants. By viewing robot capabilities in terms of schemas that are preprogrammed on robots and identifying the input and output information types for each schema, the ASyMTRe approach can generate more flexible solution strategies based on the current team capabilities and the task requirements. In the following chapters, I describe the implementation of the centralized ASyMTRe configuration algorithm and the distributed ASyMTRe-D negotiation protocol, which are based on this foundation work.

[†]This utility calculation assumes that the sensing cost and success probabilities are independent. Future work includes generalizing the formula to integrate the cost and probability when they are dependent.

Chapter 4

Centralized ASyMTRe

This chapter describes a centralized configuration algorithm that I have designed and implemented for robot team members to autonomously connect the inputs and outputs of their available schemas, resulting in dynamic task solution strategies that are a function of the current team's capabilities. Since none of the schemas are connected initially, the configuration process must generate such a mapping that leads to all required connections being made for each robot to accomplish its task, given the robot team composition and the task requirements. This chapter is organized as follows. Sections 4.1 and 4.2 describe a way to reduce the potential solution space in the centralized approach. Section 4.3 presents the centralized configuration algorithm in detail, followed by a thorough analysis in Section 4.4 on the algorithm performance both theoretically and empirically, demonstrating that it is applicable to a large class of challenging multi-robot problems. Section 4.5 gives a brief summary of the centralized approach.

This centralized configuration process can be run on a single robot (or base station). To do that, the base station needs to collect all the information of the robot team members at the beginning, and communicate the solution strategies back to team members after the configuration process. During task execution, if there is a sensor failure, this information can be communicated to the base station and a new solution will be configured based on the current team capabilities. This method suffers from the problem of single point failure. Duplication of this process on every robot can increase the robustness. However, the duplication increases the communication overhead and also requires a complicated maintenance of the knowledge base on every robot. As discussed in Chapter 5, the distributed ASyMTRe-D overcomes the single point of failure problem by distributing the reasoning process on every robot. In a real application, the designer needs to take into account the tradeoff between robustness and communication effort (see Chapter 5 for details). This work was presented at IEEE International Conference on Robotics and Automation (ICRA'05) [Tang and Parker, 2005a] and IEEE International Conference on Advanced Robotics (ICAR'05) [Tang and Parker, 2005b].

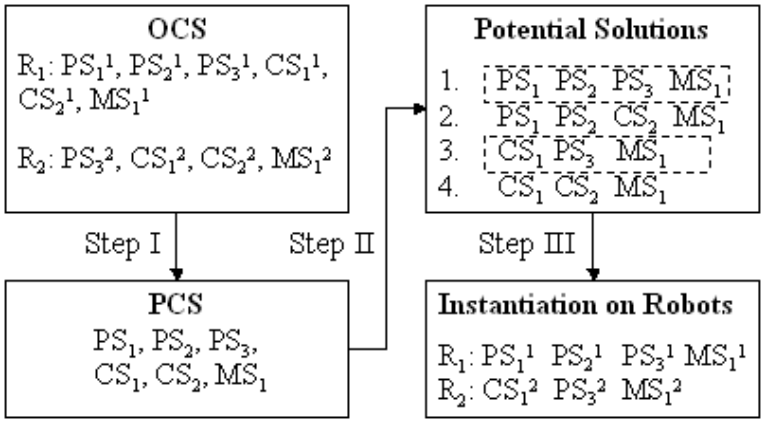


Figure 4.1: Step I: Reduce the OCS to the PCS by creating one entry per schema equivalence class. Step II: Generate the list of potential solutions based on the schemas in the PCS Step III: Instantiate the selected potential solutions on specific robots. Assume $T = \{\text{MS}_1^1, \text{MS}_1^2\}$, $\text{func}(\text{PS}_k^i) = \text{func}(\text{PS}_k^j)$, $\text{func}(\text{CS}_k^i) = \text{func}(\text{CS}_k^j)$, and $\text{func}(\text{MS}_k^i) = \text{func}(\text{MS}_k^j)$.

4.1 Potential Configuration Space (PCS)

When a group of robots is brought together to accomplish a task, each with its unique sensors and corresponding schemas, one way to solve the problem is to perform an exhaustive search of all the possible combinations of schemas within or across robots. I refer to this complete space as the Original Configuration Space (OCS). However, the number of possible connections in the OCS is exponentially large in the number of robots. In the general case, the robots will be developed separately, and it is highly possible that the schemas with the same functionality are represented differently on different robots. By definition, schemas S_i and S_j are of the same functionality, *func(S_i) = func(S_j)*, and are thus in the same equivalence class, if and only if $I^{S_i} = I^{S_j}$ and $O^{S_i} = O^{S_j}$. To reduce the size of the search space, I generate a reduced configuration space, called the Potential Configuration Space (PCS), by including only one entry for each equivalence class of schema. The extent of the size reduction that I achieve depends upon the specific team composition, and the degree of overlap in their capabilities. Step I in Figure 4.1 shows an example of reducing the configuration space. Assume, without loss of generality, that the team is composed of n robots, each of which has h schemas, and each schema requires k inputs. The OCS is of size is $O((nhk)^{nh})$. After the reduction, the PCS is of size $O((h'k)^{h'})$, where h' is the number of equivalence classes, and $h' < nh$. This analysis assumes that every output of any schema can be a potential input to any schema. In practice, the sizes of the PCS and OCS are even smaller because of the connection constraints shown in Table 3.1. The conversion from the OCS to the PCS not only reduces the size of the entire search space, but also discards the duplicated solutions. As an example, consider a case of n homogeneous robots that have the same p schemas. Then, the OCS would have np schemas, but the PCS would only have p schemas, since the duplicates would be removed from the PCS. Thus, in practice, the reduction can have a significant impact on the search space size.

Theorem 1: *If a solution exists in the OCS, it must exist in the PCS.*

Proof: Assume to the contrary that one of the solutions present in the OCS does not exist in the PCS. Recall that a solution is a combination of schemas. This could only occur if at least one schema S_i in the solution does not exist in the PCS, which means there exists an S_i in the OCS, but there does not exist an S_j in the PCS such that $func(S_i) = func(S_j)$. According to the definition of the PCS, there will be one entry for each equivalence class of schema. Thus, there cannot be a solution in the OCS that does not exist in the PCS. □

4.2 Instantiation on Robot Teams

After the PCS is generated, the centralized ASyMTRe configuration algorithm searches through the PCS to generate a list of *potential solutions*, which are the exhaustive combinations of schemas that could be connected for a single robot to accomplish its part of the task. For example, if the task T requires every robot to activate its MS_1, then a potential solution could be one of the combinations of schemas such that the input to MS_1 is satisfied, along with the input of other schemas that feed into MS_1. During the configuration process, the algorithm searches through the list of potential solutions to assign the best solution for each robot. As shown in Theorem 1, the algorithm will not miss any solutions by searching in the PCS. Once solutions are found in the PCS, they still need be mapped back to the OCS and instantiated on robots by translating the schemas in the solutions into robot-specific schemas. Steps II and III in Figure 4.1 demonstrate an example of the instantiation process. It is important to note that the list of potential solutions are for each robot, not for the entire task. As shown in the figure, the solutions are independent for each $MS \in T$.

The configuration process involves greedily searching through the list of potential solutions and selecting the solution with the maximum local utility from each robot's perspective. Suppose the algorithm eventually assigns a solution to every robot R_i with a utility U_i. The goal is to maximize the total utility $\sum_i U_i$. Assuming robots work in a non-super-additive environment [Shehory, 1998], I also impose a *maximum cooperation size* constraint on the algorithm*, to reduce the complexity of the robots executing the resulting solution†. In considering potential solutions, the exhaustive, brute force approach would involve enumerating all possible orderings of robots and selecting the one that maximizes $\sum_i U_i$. However, this approach could be impractical since the complexity is $O(n!)$, where n is the number of robots. Therefore, I impose an additional heuristic on the search process, which is the ordering in which robots are considered by the algorithm. At the beginning, I heuristically define two special orderings with which the robots begin the search, to increase the likelihood of generating a quality result. The first ordering sorts robots according to increasing robot sensing capabilities. Less capable robots whose motor schema must be activated as part of the solution are considered first, because it is generally difficult to find a solution when the sensing resources are limited. The second ordering sorts robots by the

*The maximum cooperation size represents the maximum number of robots that are allowed to form a coalition. The larger the coalition size, the more interference exists among cooperative robots. Thus a smaller size is usually preferred.

†Due to the similarity between my configuration algorithm and the coalition formation algorithm presented in [Shehory, 1998], I plan to analyze the bounds on the solution quality in future work. It has been proved in [Shehory, 1998] that similar algorithms are of low logarithmic ratio bounds to the optimal solution.

number of robots that they can cooperate with, as calculated during the first configuration process. This number represents the number of robots that can provide the information that is needed by the local robot. Thus, robots with fewer other robots to cooperate with will have the chance to find their collaborators earlier in the search process. These two special orderings are selected based on my understanding of the solution space that less capable robots may need more information sharing from the other team members, and thus should configure their solutions earlier in the process when resources are abundant. However, there is no direct proof or data that characterizes the solution qualities generated through the two special orderings.

An *anytime algorithm* [Zilberstein, 1996] can be applied here, since it can provide a satisfactory answer within a short time and its quality of results can be improved given more time. In this problem domain, the algorithm first generates all orderings of robots with which the algorithm can configure solutions. After the algorithm heuristically assigns solutions according to the two special orderings, it begins to test each ordering sequentially, and reports the solution if its utility is higher than the previous one. If more time is available, another ordering of robots can be selected and the configuration process is repeated until the deadline is reached. The algorithm reports the solution with the highest utility it has found so far or reports no solution if there does not exist a solution. Since there is a finite number of robots and a finite solution space, the algorithm will ultimately find the optimal solution if there exists one, given sufficient search time.

4.3 Centralized ASyMTRe Configuration Algorithm

All of the previously described aspects of the ASyMTRe approach are combined to yield the centralized ASyMTRe configuration algorithm shown in Algorithm 1. With every ordering, the algorithm starts to configure solutions on robots. Each potential solution is tested and the utility of the solution is maximized from every robot's perspective. If a robot does not have the required sensing resources to accomplish a goal, the algorithm checks the other robots to see if they can provide the required information. Among the robots that provide the information with the highest utility, the robot with the least sensing capability is selected. Therefore, robots with more sensor resources are saved for future configuration. After every robot has been assigned a solution, the algorithm calculates the team utility and replaces the previous solution if the utility of the current solution is higher. At this point, if the algorithm is given more time, it continues by selecting another ordering of robots and repeats the configuration process. This process will continue until the deadline is reached. The algorithm will report the current best solution it has computed so far, or it will report "failure" in two cases: (1) cannot find a solution given the deadline, or (2) there does not exist a solution to the problem.

4.4 Algorithm Analysis

I now analyze the soundness, completeness, and optimality of the ASyMTRe configuration algorithm. Here, *soundness* is the proof of the correctness of the generated solutions with respect to the environmental setting. *Completeness* is the guarantee that a solution will be

Algorithm 1 The centralized ASyMTRe configuration algorithm.

(R, T, U): the robot team composition, task, and utility
n: the number of robots in the team
m: the number of configurations to accomplish the task
k: a constant, which specifies the number of iterations

1. Generate all sequential orderings of the robots.
2. Generate the PCS based on equivalence classes. $[O(n)]$
3. Generate a list of potential solutions of size m by connecting schemas in the PCS to satisfy the task's requirements.
4. Sort the robot team members according to their increasing sensing capabilities. $[O(n\log(n))]$
5. For each robot R_i, according to the current ordering: $[O(n)]$
   - For each potential solution j to accomplish the task: $[O(m)]$
     - If R_i can accomplish the task by itself, assign solution j to R_i. $[O(1)]$
     - Else check the other $n-1$ robots to see if one can provide the needed information. $[O(n)]$
     - If the estimated utility U_i of R_i using solution j is greater than the utility of its previous solution, and the constraints on cooperation size are satisfied, update the solution strategy on R_i. $[O(1)]$
   - Continue the above process until:
     - All the robot team members can accomplish the task.
     - Or, after k number of trials.
6. Calculate the team utility U. If U is greater than the utility of previous solutions, update them.
7. If the other special ordering (based on the increasing number of potential robots that a robot can cooperate with) has not been checked yet, sort the team by this ordering and repeat Step 5. Otherwise, if more time is available, select another ordering of robots sequentially and repeat Step 5.
8. If a solution exists, report the current best solution. Otherwise, report "Failure".

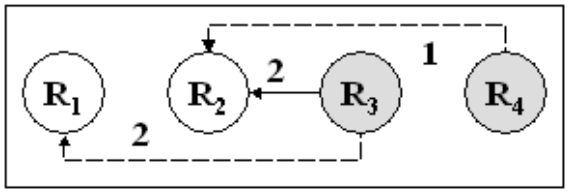


Figure 4.2: A special case that might produce incomplete results. The numbers indicate utilities.

found if it exists. *Optimality* is ensuring that the system will select an optimal solution. An optimal solution is a solution that maximizes the summed team utility U as calculated by Equation 3.5.

4.4.1 Soundness

Theorem 2: *The ASyMTRe configuration algorithm is correct.*
Proof: Assume to the contrary that the algorithm is not correct. This could occur in two ways: (1) At least one of the connections made is not a valid connection; (2) Not all needed connections are made to compose a solution. I argue that these two cases will not occur.

First, an output of a schema S_i can be connected to an input of a schema S_j if and only if $O^{S_i} = I_j^{S_k}$, which means the output of S_i has the same information type as one of the input required by S_j. Since all the input and output information types for every schema are known, and the connection constraints, which specify the types of schema connections that the system allows, is maintained, it is not possible to generate an invalid connection.

Second, a potential solution is a combination of schemas such that the inputs for each $MS_i \in T$ are satisfied, along with all the inputs from the schemas that feed into MS_i. Since the algorithm complies with this standard, all needed connections will be made to become a potential solution. Thus a solution generated by ASyMTRe will be a correct solution. □

4.4.2 Completeness and optimality

Since the heuristics cause the search to begin in a greedy manner, searching the most promising candidates first, the search process (based on the current ordering) suffers from the well-known problems of greedy algorithms. Namely, in this domain, the ordering in which the robots are considered may cause the system to miss solutions, even though sufficient sensor resources exist to accomplish the task. Such an example is shown in Figure 4.2. In this example, I assume that R_3 can provide the information needed by both R_1 and R_2, and R_4 can provide the information needed by R_2. The utility for each connection is different and the maximum cooperation size is two. Consider the situation where the ordering under consideration by the algorithm is $R_2 \prec R_1$. The algorithm certainly chooses R_3 to cooperate with R_2 because R_3 can help R_2 with a higher utility than R_4 can. Therefore, R_1 cannot find any robot to cooperate with. To avoid this situation, the algorithm reorders the robots by the number of potential robots that can cooperate with them. These numbers can be collected during the previous search process. In the previous example, the new ordering is $R_1 \prec R_2$ since R_1 has only one potential robot to cooperate with, while R_2 has two. With

Table 4.1: Factors that influence the analytical results.

Labels	Factors	Ranges
n	Size of the robot team	1 to 100
p	Number of inputs to a schema	1 to 10
q	Number of redundant schemas	1 to 5
m	Number of potential solutions	1 to 128

this ordering, the solution is to have R_3 cooperate with R_1 and R_4 cooperate with R_2 with a total team utility of 3.

Theorem 3: *The ASyMTRe configuration algorithm is complete and optimal given enough time.*

Proof: I prove completeness and optimality by showing that the algorithm can perform a brute force search of the entire solution space given enough time. Step 5 (in Algorithm 1) shows the major configuration process in which, given an ordering of robots[‡], the algorithm searches through the list of potential solutions to find solutions for every team member. If the algorithm is given enough time, it will ultimately test all possible orderings of robots, which is $O(n!)$, and reports the solution with the highest utility. Since the solution space is eventually explored in its entirety by the algorithm, it is complete and optimal, given enough time. □

Despite this proof, I note that in reality, practical constraints require a fast response, especially when dealing with failures that require quick reconfiguration. Since the problem itself is NP-hard and the worst case time complexity is $O(n!)$, it is not possible to generate an optimal solution in practice when n is large. However, as is shown in the experiments, the algorithm can always return a good solution for the team within a few seconds, which demonstrates this characteristic of ASyMTRe.

4.4.3 Formal analysis of centralized ASyMTRe

In prior sections, I have demonstrated the theoretic analysis of ASyMTRe to automatically generate different cooperation solutions to the same task, as a function of the robot team capabilities. Here, I illustrate the computational performance of the ASyMTRe configuration algorithm by reporting the results of a large number of "generic" experiments in simulation. The results illustrate the computational tractability and scalability of ASyMTRe in practice.

Experimental setup

A variety of characteristics in an experimental setup will influence the computational results of ASyMTRe, such as the robot team composition and the available schemas. Table 4.1 lists all the major factors that are considered in the applications and their ranges. The "number of redundant schemas" represents the number of schemas that can provide the same type

[‡]Note that the particular ordering is selected sequentially from the list of orderings generated at the beginning of the task, rather than randomly.

of information within the robot team. For example, there can be two different perceptual schemas that use either a laser scanner or a camera to estimate the relative position of an object. During the experiments, I randomly generate values for different factors and apply the ASyMTRe configuration algorithm, collecting data on the amount of time needed to generate a solution and the number of potential solutions.

Potential solutions

Since the computational complexity is partially determined by the number of potential solutions, I first explore how the characteristics of the schemas influence this size. The configuration algorithm was run on 100 problem instances with different schemas setups by varying the number of inputs to a motor schema and the number of redundant schemas that provide the inputs to the motor schema. Figure 4.3 and Figure 4.4 plot the number of potential solutions as I increase the redundancy and the number of inputs for OR-type inputs and AND-type inputs respectively. Note that the number of potential solutions is also dependent on the team size, especially when new schemas are introduced into the PCS. However, the increase introduced by the team size is relatively small compared with the number of inputs and the number of redundant schemas. Thus, I used a specific team size ($n = 10$) in these experiments. The curves should be similar to this case for the other team sizes.

The results are consistent with my analysis. When a schema has multiple choices of inputs (OR), the number of potential solutions increases linearly with the increasing redundancy. If an MS needs one of the p inputs and each input can be provided by q schemas, then the number of potential solutions is $O(pq)$. When a schema needs the combination of multiple inputs (AND), the size increases exponentially with the increasing redundancy. Here, if an MS needs p inputs and each input can be provided by q schemas, then the number of potential solutions is $O(q^p)$. However, this result assumes that solution involves a large coalition of robots, all of which are sharing sensor data. In practice, this is impractical because of the constraints on the size of the coalition. If I add the constraints that limit the size of the coalition, the size of the PCS can be greatly reduced. However, the robot's decision making process is less flexible.

These results show that there is a tradeoff between the complexity and the flexibility of a solution. Increasing the flexibility by allowing a large coalition may increase the solvability of a task but may also increase the complexity of the solution, and thus the computational requirement for finding that solution. Usually, when the flexibility is not crucial to the task, using the constraints would make the solution generation process quicker.

Time complexity and scalability

Since the ASyMTRe configuration algorithm is an anytime algorithm, I can measure the complexity of the algorithm by observing the search procedure only; that is, how long it takes to configure solutions on a team of robots given one ordering of the robots. The time complexity of the search process is $O(mn^2)$, where m is the number of potential solutions and n is the number of robots. This algorithm was run over 1,000 problem instances with different m and n values within the ranges specified in Table 4.1. The actual running time

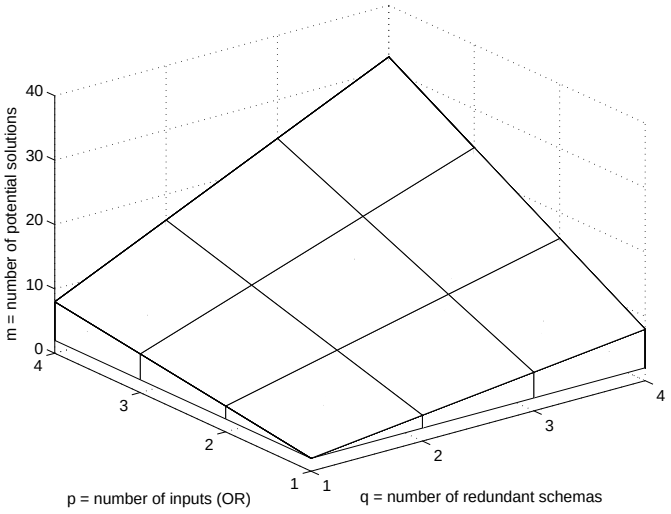


Figure 4.3: The number of potential solutions is decided by the number of inputs ("OR" condition) and the number of redundant schemas. Here, $n = 10$.

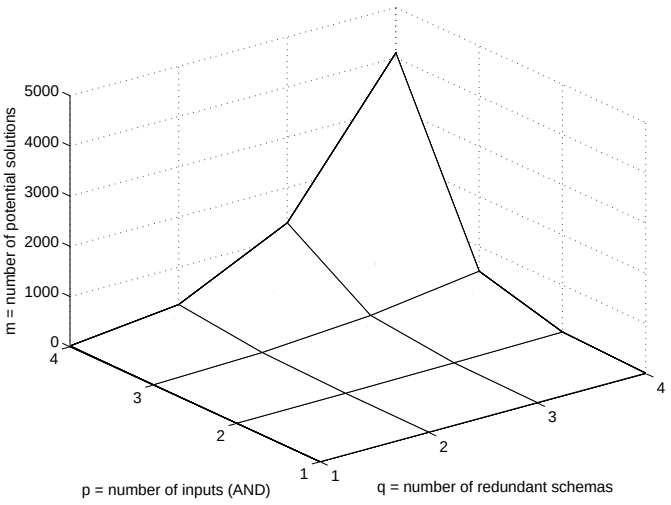


Figure 4.4: The number of potential solutions is decided by the number of inputs ("AND" condition) and the number of redundant schemas. Here, $n = 10$.

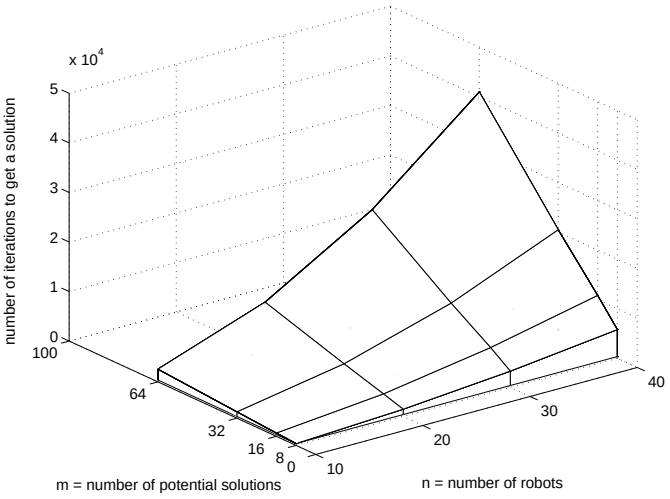


Figure 4.5: Scalability of the centralized ASyMTRe configuration algorithm.

to find a first solution for all of the experiments ranges from 1 to 2 seconds. Experiments were run on a Linux machine with 512MB RAM and an Intel Pentium 4 2.4GHz processor.

These experiments illustrate that the running time for finding a first solution does not vary dramatically with different settings (not considering the time to generate all orderings of robots). To see how the number of potential solutions (m) and the number of robots (n) influence the complexity of the approach, I also measured the number of iterations taken for a robot team to find a solution. As suggested in [Gerkey and Mataric, 2004], it is assumed that the running time is determined by some dominant operation, which in our domain, is the comparison of utilities. Figure 4.5 shows the experimental results for the scalability of the ASyMTRe configuration algorithm, in which n varies from 10 to 40, and m varies from 8 to 64. These empirical results confirm that the running time to complete a single search process is $O(mn^2)$.

To demonstrate the anytime aspect of the ASyMTRe configuration algorithm, I collected data on how the quality of the solution can be improved when given more time. Figure 4.6 plots the relationship between time and solution quality for one of the typical runs. Here, each search process is associated with an ordering of the robots, with the special ordering (according to increasing sensing capabilities) as the starting point. The quality of the solution, which is determined by the summed team utility, increases slowly as I increase the number of iterations of the search process. For this particular run, the optimal solution was obtained after 40 iterations, which is about 40 seconds. This graph illustrates that the algorithm can report a good solution at any time and the quality of the solution is determined by the amount of time that the program is given.

4.5 Summary of Results

The characteristics of the centralized ASyMTRe configuration algorithm are essential to the types of problems to which it can be applied. The ASyMTRe approach coalesces robots into teams to solve a single multi-robot task by coordinating the sharing of sensors and

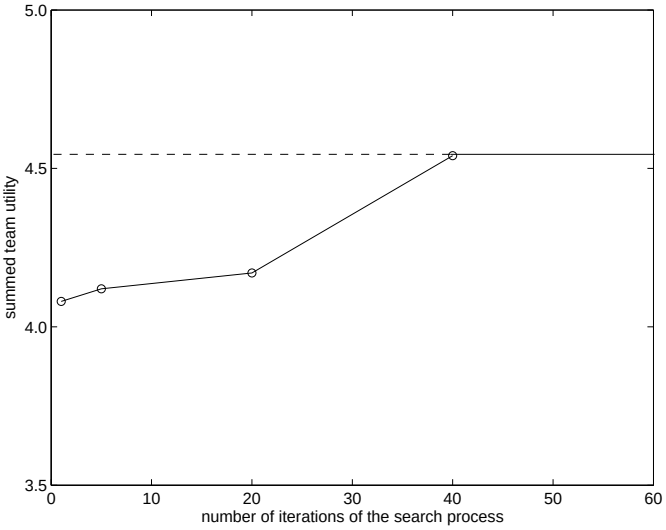


Figure 4.6: The quality of the solution increases over time. The dashed-line indicates the maximum team utility.

effector capabilities, and is suitable for applications where robots cooperate with each other to accomplish a team-level goal. The cooperation enables some robots to accomplish the task that is impossible for them to achieve by themselves. Since the algorithm runs in a reasonable time (1 to 2 seconds) to provide a satisfactory solution and the running time does not increase dramatically as the size of the application increases, it is applicable to most of today's multi-robot applications, where the team size is normally less than 100 robots. This feature is also important in applications where robot or sensor failures are common. Whenever there is a failure, the algorithm can be called again to synthesize new solutions, taking into account the current robot team's sensing capabilities. Although I have proven that the centralized approach is sound, complete, and optimal (given enough processing time), it is well-known that centralized approaches suffer from lack of robustness. One approach to achieving robustness is to duplicate the configuration process on every robot. However, this approach would require that every robot be aware of the capabilities of all the other team members. Thus, if there is any change to the robot team composition or an individual robot's capabilities, all the other robots must update their corresponding knowledge base. This type of system would not work efficiently when robot failure or sensor failure is common, or when robots join or leave the team dynamically. To increase the robustness and flexibility of the system, I have developed a distributed, negotiation-based ASyMTRe, which is introduced in Chapter 5.

Chapter 5

Distributed ASyMTRe-D

5.1 Overview of Approach

Distributed ASyMTRe-D shares the same theoretic foundation with centralized ASyMTRe, which defines the mapping of schemas to information types to enable robots to share sensory and computational data across robots. In distributed ASyMTRe-D, I incorporate a negotiation protocol that is inspired by the well-known Contract Net Protocol (CNP) [Smith, 1980] (similar to the approach taken in other multi-robot negotiation approaches, such as [Dias and Stentz, 2000, Gerkey and Mataric, 2002]), in order to distribute the reasoning process across multiple robots and eliminate the need for a centralized reasoner. The robots can be viewed as a set of information sources, where some of the team members do not have sufficient information to solve the task by themselves. To accomplish the task for the team as a whole, more capable robots can provide useful information to less capable (resource-bounded) robots. The sharing of information, and thus the cooperation among robots, can be achieved through a distributed negotiation process, in which robots negotiate with each other to make decisions on how to accomplish a task. Each robot decides what information it needs and then requests the information from other robots. A robot is also expected to provide the information needed by other robots if the request is within that robot's capability. The solution is evaluated based upon each robot's local information, and the final decision is determined by mutual selection. The negotiation process is totally distributed, with no centralized control or centralized data storage.

Such a distributed system offers a reliable, extensible, and flexible mechanism to make ASyMTRe suitable for applications where robot and sensor failures are common, or where a robot team composition is dynamic (robots may join or leave frequently). The negotiation process is triggered at the beginning of each task to generate initial solution strategies, and is called to reconfigure solutions to accommodate changes in robot teams or tasks. It is important to note, however, that the distributed approach trades off solution quality for team robustness. The intent of this approach is not to develop a new negotiation protocol, but instead to develop a method for the robot team to vary their reasoning between fully centralized and fully distributed decision-making, according to the desired balance between solution quality and robustness. This work was published and presented at IEEE International Conference on Intelligent Robots and Systems (IROS'05) [Tang and Parker, 2005c]. My ultimate objective is to enable the distributed solution generation process converge to

Table 5.1: The list of messages that are used in the protocol.

Type	**Format**
Simple Request	*('F1', from, numinfo,* F_1, F_2, $\cdots$, $F_{numinfo}$*)*
Complex Request	*('F2', from, numinfo,* F_1, F_2, $\cdots$, $F_{numinfo}$*)*
Simple Reply	*('H1', from, to)*
Complex Reply	*('H2', from, to, utility)*
Confirmation	*('C', from, to)*
Cancellation	*('A', from, to)*

an optimal result given enough time (similar to the anytime centralized configuration algorithm). The idea is that the human operator can specify a deadline, and robots can continue the negotiation process and increase the level of information-sharing among robots until the deadline approaches. When enough time is given, the robots could completely share their capability information with each other and reason the solution strategies with a complete knowledge of the team, and eventually find the optimal solution.

5.2 Distributed ASyMTRe-D Negotiation Protocol

I now present the distributed ASyMTRe-D negotiation protocol. Table 5.1 lists all of the message types that are used in the distributed negotiation protocol. This negotiation protocol involves the following major steps:

- **Make request**. Depending on the requirements of each potential solution, a robot broadcasts requests for the set of information types that it needs to obtain from other robots. Simple requests are sent out at the beginning of a task to estimate the potential number of robots (pn) that can provide the required information. Each robot will then wait for a period of time that is proportional to its pn value before sending out the complex requests. Therefore, the robots with fewer potential helpers have higher priorities to make a request, since these robots will likely have fewer chances for success.

- **Serve request and submit help**. After evaluating the required information, each robot replies based on a first-come-first-serve (FCFS) order. Simple replies are sent out without the estimation of utilities to enable the requesting robot to collect information about its pn. Otherwise, the robots will estimate the utility of providing the required information by Equation 3.5 in Chapter 3. Since a requesting robot selects the potential solution with the highest utility, some capable robots are more likely to be chosen than others. In practice, I impose a *max-to-help* (k) constraint on each robot, which limits the number of robots that a robot can help with. It can reduce the complexity of the robots executing the solution due to motion constraints and balance the burden among capable robots.

- **Rank and confirm help**. Complex replies are ranked by decreasing utilities, which are then combined with local schemas to generate the utility of every potential solution. Each robot selects the solution with the highest utility and sends a confirmation

Table 5.2: Ways to handle possible communication failures.

Message Loss	Countermeasures	Name	Time	Result
Request/Reply	Finite waiting time	Timeout 2	5s	Repeats request
	Repetitive requests	Timeout 1	15s	Reports failure
Confirmation	Finite waiting time	Timeout 3	2s	Repeats request

message. When there are multiple solutions with the same utility, the selection also follows the FCFS rule. If no robot responds to the request after timeout, the robot will repeat the negotiation process until it reports "failure" after a period of time. The confirmation message will be broadcast to all robots, so that the other robots that are also willing to help can be released from their commitment and serve more requests.

Algorithms 2 and 3 present the two main threads in the distributed ASyMTRe-D negotiation protocol. A robot runs both threads at the same time, with one thread handling outgoing requests, and the other thread handling incoming requests from other robots. Step 3 in algorithm 2 is used to make sure that robots with a fewer number of potential helpers can send out their request earlier in the process. In doing this, more time is needed for negotiation. To decrease the total negotiation time, this step can be skipped.

To ensure a general and robust negotiation process, some additional mechanisms are built into the distributed protocol. First, the protocol employs a variety of timeout values in the negotiation process. As shown in Table 5.2, a requesting robot waits for a finite time (timeout 2) for any replies, and if there is no reply, it sends out requests again. The requesting robot waits for a finite time (timeout 3) for a confirmation message from the helping robot. If no confirmation is received, the robot continues sending requests. This process continues for a period of time (timeout 1) before the robot reports "failure", which is either due to no robots being available to help, or to the requests or replies being lost. The values shown in Table 5.2 are used in the experiments and can be tuned accordingly to decrease the negotiation time. Timeout 2 decides the time that the robot wants to wait for potential helping messages from other robots. Thus, it needs to be set to a proper value to guarantee that requests are processed and replied by others. Timeout 3 represents the patience of the robot while waiting for confirmation. Since the confirmation message only comes from a single robot, this timeout can be shorter. Timeout 1 is determined partially by timeout 2 and 3 (timeout 2 + timeout 3 $\leq$ timeout 1), and the number of times that the requesting robot is allowed to send repetitive requests. Another mechanism to ensure robustness is the use of broadcast messages, rather than point-to-point messages, because it is efficient in transferring data and does not require the system to know specific destination information.

The distributed ASyMTRe-D negotiation protocol acts as a greedy planner, since each robot selects the locally best solution to accomplish its part of the task. However, it may not yield a globally best solution, suffering from the usual problems of greedy algorithms. Namely, in our domain, the ordering in which the robots send out requests may cause the system to miss solutions, even though sufficient resources exist to accomplish the task. In

Algorithm 2 The distributed ASyMTRe-D negotiation protocol (request thread).

Step 1: Generate a list of potential solutions:
for all ($MS_i \in T$) **do**
  if I^{MS_i} is not NULL, check the output information type(s) of local schemas to see whether it can provide the information type(s); if yes, append the schema(s) to the solution, otherwise, append the information type(s) to the solution.
  Repeat the above step until the input of every schema is satisfied.
end for
Step 2:
for all (Potential solutions PoS_i) **do**
  if PoS_i does not have any requirement for information types, calculate the utility of the current solution and append it to the solution list.
  Otherwise, append it to needy potential solution list.
end for
Step 3:
For each PoS_i in the needy potential solution list, **make a simple request** of the list of information types that are needed.
Potential number of helper (pn) is initialized to be 0.
while (not Timeout 2) **do**
  Listen to any incoming help messages and increase pn.
end while
Sleep for ($C \times pn$) seconds; here C is a constant.
Step 4:
while (not Timeout 1) **do**
  For each PoS_i in the needy potential solution list, **make a complex request** of the list of information types that are needed.
  while (not Timeout 2) **do**
    Listen to any incoming help message.
    Rank help according to the utility of the PoS_i.
  end while
  if (Solution list is empty) **then**
    Continue.
  else
    Select the potential solution with the maximum utility.
    Confirm help to the team member who provides helping information.
  end if
  while (not Timeout 3) **do**
    Listen to any incoming message.
    if it is a "Confirmation" message to me, exit loop.
    if it is a "Cancellation" message to me, Continue.
  end while
end while

Algorithm 3 The distributed ASyMTRe-D negotiation protocol (help thread).

while (not Timeout 1) **do**
  Listen to any incoming request messages.
  if (it is a message that requests help) **then**
    if (I can provide the information types that are requested) **then**
      if it is a simple request: **submit help** without utility calculation.
      if it is a complex request: **submit help** with the utility of providing the information.
    else
      Continue.
    end if
  else if (it is a confirmation message) **then**
    if (I am currently helping less than k number of robots) **then**
      Record the current solution.
      Confirm help to the requesting robot.
    else
      Send "Cancellation" message to the requesting robot.
      Exit the loop.
    end if
  end if
end while

Chapter 4, I have given an example of this problem and presented the centralized approach that takes into account all the orderings of robots (if given enough time), therefore generating the best solution for the team as a whole. Clearly, this represents the tradeoff between the robustness of a distributed solution and the solution quality of a centralized solution. Experimental results in Chapter 7 show, nevertheless, that distributed ASyMTRe-D results in a good solution with a reasonable setting of the max-to-help (k) parameter.

5.3 Summary

This chapter has presented a fully distributed ASyMTRe-D negotiation protocol for forming coalitions based on the flow of information that is needed to accomplish a task. The motivation of building a distributed system is to increase the robustness of the reasoning process by eliminating the single point of failure in a centralized system. In addition, other techniques, such as message broadcasting, acknowledgment, and timeouts, have also been applied to ensure a robust distributed negotiation protocol. With the increasing robustness, there is also a tradeoff in the solution quality since robots make decisions based on their local knowledge rather than the knowledge of the entire team. In Chapter 7, I compare and analyze the various aspects of the centralized ASyMTRe approach and the distributed ASyMTRe-D approach.

Chapter 6

Layering Coalition Formation With Task Allocation

In the previous chapters, I have presented the foundation for the ASyMTRe approach and its implementation in both centralized and distributed versions. Although the ASyMTRe approach enables the robots to form coalitions for solving a single multi-robot task, approaches are lacking that combine the techniques of coalition formation with task allocation into a single system. This chapter defines an approach that enables the allocation of both single-robot or multi-robot tasks to a robot team. This approach layers the ASyMTRe coalition formation system that I have developed with an auction-based mechanism for achieving the allocation of single-robot and/or independent subtasks.

The rest of the chapter is organized as follows. Section 6.1 gives an overview of the problem and Section 6.2 describes a motivating example of site clearing. Section 6.3 gives the formalism of the problem. Section 6.4 presents the approach of layering lower-level coalition formation with higher-level task allocation. This work is accepted to appear in the proceedings of the AAAI Summer Workshop on auction mechanisms for robot coordination [Tang and Parker, 2006b].

6.1 Overview of the Problem

Traditionally, task allocation approaches in multi-robot teams have dealt with the assignment of single-robot tasks, which are tasks (or collections of tasks or subtasks) that can be accomplished independently by a single robot. Another important type of task in multi-robot teams is the multi-robot task, which requires a *strongly cooperative* solution and is not trivially serializable, so that it cannot be decomposed into subtasks that can be completed by individual robots operating independently; instead, it requires robots to act in concert to achieve the task. Sometimes, this type of task is also called *tightly-coupled* or *tightly-coordinated*. In ASyMTRe, robots form coalitions for accomplishing these strongly cooperative multi-robot tasks. The motivation behind coalition formation is that robots in a coalition should work together to share resources and cooperate on task execution due to their decision that they would benefit more from working together as a coalition than they would working individually.

In addition to the majority of work that addresses single-robot task allocation, some recent work also addresses the allocation of multi-robot tasks [Jones et al., 2006, Kalra et al., 2005, Lin and Zheng, 2005]. The approach I present here is different from the above approach in that I am addressing the multi-robot tasks through the dynamic configuration of low-level behavioral building blocks instead of predefined plans or roles. In particular, the Hoplites approach [Kalra et al., 2005] focuses on the selection of an appropriate joint plan for the team to execute by incorporating joint revenue and cost in the bid. The work in [Jones et al., 2006] achieves multi-robot task allocation through matching roles with robot capabilities. The work in [Lin and Zheng, 2005] also matches task required capabilities with robot capabilities and accomplishes multi-robot tasks through combinatorial bids. My approach of task allocation on the higher level is similar to the above approaches, but is different in the way that coalitions (subgroups) are formed to accomplish a single multi-robot task. My approach forms a coalition through configuring the sensors and preprogrammed schemas on every team member so that they share sensory or computational information with each other in order to accomplish the task. The coalitions are generated "on the fly" instead of using predefined plans, roles, etc. With this capability, the robot team is able to generate flexible and versatile solutions for solving a task based on the current team capabilities.

Although ASyMTRe provides a way of generating robot coalitions, it can only handle a single multi-robot task at a time. For missions of multiple tasks, I would like to achieve task allocation amongst coalitions and/or individual robots, thus combining the benefits of low-level coalitions with those of higher-level, more traditional, task allocation. My idea is to layer ASyMTRe for low-level coalition formation for solving a single multi-robot task, with a higher level, traditional task allocator for solving a set of tasks. The resulting system provides a flexible mechanism for a broad range of realistic multi-robot applications, with the ability to generate both strongly cooperative and weakly cooperative solution strategies, as appropriate.

6.2 Motivating Example: Site Clearing Task

To motivate the need for the combination of coalitions and more traditional task allocators, I introduce a representative application, called the *site clearing* application. The site clearing application is a simplified version of the site preparation task [Parker et al., 2000], which has been identified by NASA as an important prerequisite for human missions to Mars. The site clearing application, illustrated in Figure 6.1, requires a specific area to be cleared of obstacles, which I simplify to be boxes with different weights or sizes. The objective of the application is to clear the site in as little time as possible while minimizing the cost to the robots (e.g., energy consumption or computational requirements). For the purposes of this discussion, I assume that a map is available to enable the robot team to determine the positions of the obstacles in the area. I assume that the obstacles to be removed from the site can either be pushed outside the area, or can be pushed to a common collection point, as indicated by a beacon. I further assume that a partial-order planner exists to determine the ordering constraints of removing the obstacles, in case certain obstacles need to be removed before other obstacles can be cleared.

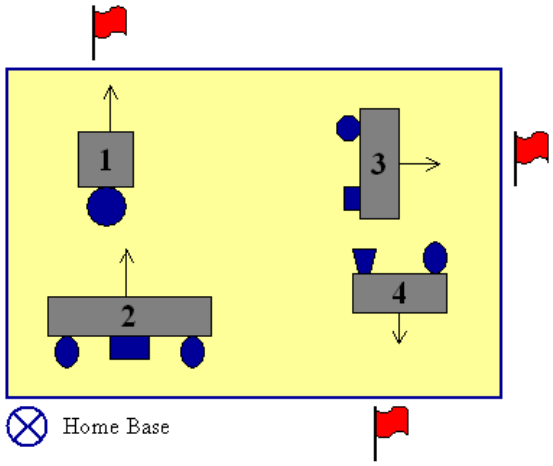


Figure 6.1: The site clearing application. Red flags represent collection points. Different shapes represent the heterogeneity of the robots.

There are a number of constraints that make the site clearing problem challenging, such as:

- *Limited team size*: The number of obstacles is greater than the number of robots, thus requiring robots to iteratively move obstacles for several rounds to clear the site.
- *Varying weights/sizes of obstacles*: Robots have different weight/size requirements for the kind of obstacle they can manipulate. Thus, the weight or size of an obstacle determines the number of robots required to transport it. Robots working on the same obstacle need to cooperate with each other.
- *Heterogeneous robots*: Robots may differ in their capabilities, thus requiring the allocation approach to appropriately map robots to tasks.
- *Resource-bounded robots*: Robot team members may be resource-bounded, and thus unable to transport an obstacle independently, or navigate in the site independently. Robot coalitions may therefore be needed to share sensory, perceptual, computational, or effector resources to enable the team as a whole to accomplish the required task(s). Although sometimes these interactions can be trivially serializable (e.g., as in the box pushing example of [Parker, 1994a]), in the general case of resource-bounded robots, they cannot. Thus, this constraint illustrates the need for expanding current task allocation approaches to include coalition formation for multi-robot tasks.
- *Uncertainty*: The uncertainty of the environment and robot team capabilities (due to sensor or robot failures) requires that team solutions should be based on current team capabilities instead of predefined solutions.

The site clearing application can be decomposed into a series of tasks with ordering constraints. Each task is aimed at removing one obstacle from the site, which I call "Remove Obstacle". For example, the task shown in Figure 6.1 can be accomplished through the

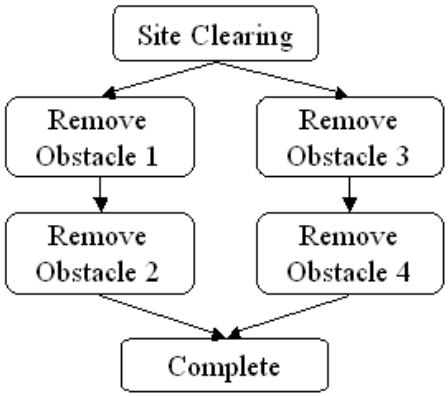


Figure 6.2: A partial-order plan for the site clearing application.

partial-order plan in Figure 6.2. Since only some tasks have ordering constraints, the system can allocate a subset of the tasks to the robots for concurrent execution. Thus, when making a task allocation decision, robots are considering more than one task at a time. In addition, because of the application challenges mentioned earlier, a "Remove Obstacle" task may require multiple robots to form a coalition to accomplish the task in a manner that efficiently uses the available robot capabilities. Additionally, when multiple coalitions are available, the system must determine which coalition is the best fit to the current task.

Note that from my perspective, an individual task (such as those defined in Figure 6.2) cannot be categorized in advance as a multi-robot task or a single-robot task. Instead, whether or not the task requires single or multiple robots depends upon the capabilities of the robot team members. Some robots may be able to perform a given task on their own (thus making the task a single-robot task), while other robots may require help from teammates to accomplish that same task (thus making that same task a multi-robot task). The ASyMTRe approach is able to find combinations of robot capabilities that can accomplish the task in either the single-robot case or the multi-robot case, depending upon the team capabilities.

6.3 Formalism of the Problem

The multi-robot task I address can be formally defined as follows:

- $R = \{R_1, R_2, \cdots, R_n\}$ is a collection of n robots, where each robot R_i is represented by its available environmental sensors (ES), and its corresponding perceptual (PS), motor (MS), and communication schemas (CS). For a complete definition of R, please refer to Chapter 3.
- T is the team-level task to be accomplished, which is denoted as $T = \{t_1, t_2, t_3, \cdots\}$.
  - A set of *ordering constraints* defines a proper partial order of tasks. $t_i \prec t_j$ means that task t_i must be executed sometime before task t_j.

  - A set of *open preconditions*. A precondition is open if it is not achieved by some task in the plan.
  - A subset T^i of T can be allocated to robots concurrently if the tasks in T^i do not have ordering constraints and their preconditions are not open.
  - Each task t_i is further defined as a set of motor schemas that need to be activated in certain ways in order to accomplish this task.

- To accomplish a subset of tasks T^i, a collection of m coalitions, denoted $C^i = \{C^i_1, C^i_2, \cdots, C^i_m\}$, needs to be generated based on the task requirements of T^i and the robot team capabilities.

- With multiple solutions available, I define a *cost* function for each robot, specifying the cost of the robot performing a given task, and then estimate the cost of a coalition performing the given task. I consider two types of cost:
  - A robot-inherent cost measures the inherent cost (e.g., in terms of energy consumption or computational requirements) of using particular capabilities on the robot (such as a laser or a mapping algorithm). I denote robot R_i's inherent cost by *robot_cost*(R_i).
  - A task-specific cost measures cost according to task-related metrics, such as time, distance, success probability, etc. I denote the cost of R_i performing task t_j by *task_cost*(R_i, t_j).
  - The *cost* function of R_i performing t_j is represented by *cost*(R_i, t_j), which is a weighted combination of both the robot-inherent cost and task-specific cost, normally in the form of a linear function. Other type of costs can also be easily incorporated when necessary.
  - The cost of a coalition C_i performing a task t_j is the sum of individual costs of robots that are in the coalition, which is denoted as:

$$cost(C_i, t_j) = \sum_{R_k \in C_i} cost(R_k, t_j) \tag{6.1}$$

The problem I address here is: Given (T, R), assign a set of tasks T^i to coalitions of R such that the sum of the coalition costs $\sum_{t_k \in T^i, C_j \in C^i} cost(C_j, t_k)$ is minimized.

6.4 The Approach

To allocate multi-robot tasks to a team of robots, I propose an approach encompassing four main steps as shown in Algorithm 4. Figure 6.3 describes a general procedure that first decomposes a team-level task to a set of tasks with ordering constraints. At the lower level, coalitions from the team of robots are formed to address the given tasks. The coalitions then compete for the assignment of tasks using a traditional task allocation approach. Note that these coalitions are not distinct, but may have the same team members. When allocating individual tasks t_i to coalitions, the allocator needs to make the appropriate assignments such that the winning coalitions do not overlap.

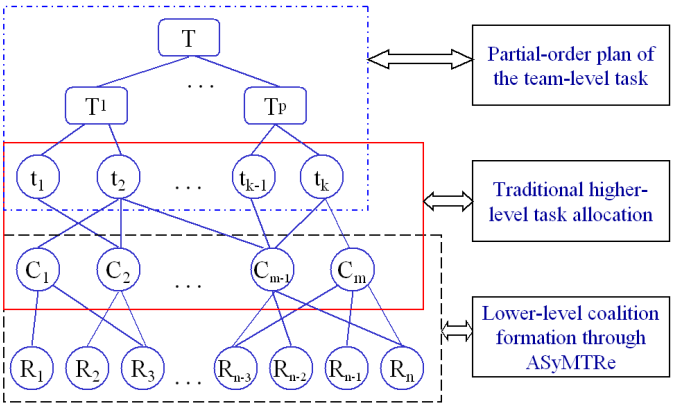


Figure 6.3: The relationships between tasks, coalitions and robots.

Algorithm 4 Allocating multi-robot tasks to a team of robots.

Input: (T, R)

1: Find the set of tasks T^i up to a constant number, such that both the ordering constraints and the preconditions of tasks are satisfied. Note that the maximum number of tasks allowed for allocation is limited to a constant number b to decrease the computational complexity of the allocation of multiple tasks at once.

2: Configure solutions for each task t_j in T^i by forming a set of coalitions C^i, based on t_j's objective and the current team capabilities.

3: Allocate tasks in T^i to coalitions in C^i, such that:

- The task-specific cost and the robot-inherent cost are minimized for the set of tasks.
- A coalition can win at most one task at a time. Assuming $C' \subseteq C^i$ is the set of coalitions selected to perform the tasks in T^i, then the following condition must be satisfied: $\forall_{C'_i, C'_j \in C', i \neq j}, C'_i \cap C'_j = \emptyset$.

4: Monitor the execution of tasks. If there are still some tasks waiting to be allocated, start the allocation process (go to step 1) when robots are within Δt time to complete their current tasks. Otherwise, exit.

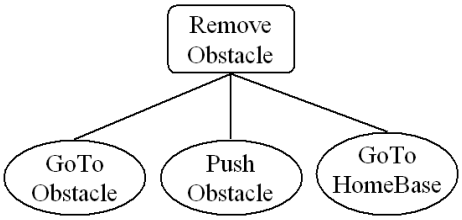


Figure 6.4: An example task tree for the Remove Obstacle task.

6.4.1 Low-level coalition formation

In Algorithm 4, step 2 is accomplished through the ASyMTRe approach, which serves as the low-level solution generator. The distributed ASyMTRe-D negotiation protocol that is described in Chapter 5 is used here, which provides beneficial mechanisms for multiple robots to: (1) synthesize task solutions using different combinations of robot sensors and effectors, (2) share information across distributed robots and form coalitions as needed to assist each other in accomplishing the task, and (3) reconfigure new task solutions to accommodate changes in team composition and task specification, or to compensate for faults during task execution.

Previously, a task in ASyMTRe is defined as a set of motor schemas that need to be activated to accomplish this task. Multiple motor schemas are related through AND and OR logical operators. However, these relationships are not rich enough for multiple tasks, since, for some applications, two motor schemas may need to be executed sequentially. Therefore, to better characterize the relationships between motor schemas, I use *task trees* to represent tasks, similar to the tree generated by TDL [Simmons and Apfelbaum, 1998]. The root of the task tree is the most abstract task description. Each successive level of the tree represents a refinement of the tasks in the immediate upper level. The tree will be refined until all the leaf nodes can be represented by motor schemas that are preprogrammed on the robots. The task tree embeds parent/child relationships and synchronization constraints between nodes, including: *sequential*, meaning that the tasks associated with the nodes need to be executed in a sequential order (such as from the leftmost child node to the rightmost child node); and *concurrent*, meaning that the tasks associated with the nodes can be executed at the same time, or roughly the same time. An example task tree for the "Remove Obstacle" task is shown in Figure 6.4, which involves the following sequential tasks: (1) navigating to the obstacle, (2) pushing the obstacle to the goal, and (3) navigating to the closest home base and waiting for new tasks. Given the task tree, each robot can then use the distributed ASyMTRe-D negotiation protocol to decide how to form coalitions to accomplish a task, while maintaining the synchronization constraints during task execution.

6.4.2 Higher-level task allocation through auction

Although ASyMTRe provides the mechanism for a heterogeneous robot team to accomplish a task by forming coalitions, it can only handle one multi-robot task at a time. I therefore propose the use of an auction mechanism to provide a higher-level task allocation approach on top of ASyMTRe for handling multiple tasks. Note that the intent of this approach is not

to develop a new auction mechanism, but instead to layer existing auction mechanisms with the ASyMTRe approach for allocating multi-robot tasks to robot coalitions. The following higher-level auction process is similar to [Jones et al., 2006], although the technique for coalition formation is different. Additionally, I allow the allocation of multiple tasks at a time instead of one.

The auction process is described as follows:

1. *Task announcement*: Initially, the human operator introduces the site clearing task T to the system. Each task t_i in T is embedded with task-specific information, such as the size and the position of the obstacle to be removed. The human operator has an interface "Auctioneer" that interacts with the other robots in the system (similar to OpTrader in [Dias, 2004]). This auctioneer holds the partial-order plan for T, selects a subset of tasks T^i that satisfies the ordering constraints and the preconditions, and makes an auction call of T^i to all robots.

2. *Coalition formation*: Robots that receive T^i start negotiating with others to generate solutions for accomplishing tasks in T^i. For each task t_j in T^i:

   (a) Each robot tries to find a list of coalitions (up to a constant number c) that it can join to accomplish t_j. The revised ASyMTRe-D negotiation protocol returns the top c coalitions per task. The size of a coalition is limited to a max coalition size d assuming robots work in a non-super-additive environment [Shehory, 1998].

   (b) Coalitions are not arbitrarily formed, but are selected based on the combination of the robot-inherent cost and the task-specific cost (please refer to Section 6.3 for details of cost estimation.).

3. *Bid submission*: Once coalitions are formed for each task t_j, a randomly selected coalition leader submits a bid to the auctioneer, including information such as the list of coalition members, the cost of this coalition performing t_j, the leader of the coalition, etc.

4. *Winner determination*: Once bids for all tasks in T^i are collected or a timeout has expired, the auctioneer then determines the winner coalition for each task. The goal for the auctioneer is to find a coalition C_j for each task t_j, such that the total cost of performing the tasks in T^i is minimized and there is no overlapping of coalition members assigned to the tasks. If no such coalition C_j exists for task t_j and C_k for t_k such that $C_j \cap C_k \neq \emptyset$, then one of the tasks (either t_j or t_k) is auctioned again in the next round. The problem of determining the winner is equivalent to the combinatorial auction where multiple tasks are offered and each coalition can bid a subset of tasks. Existing combinatorial auction clearing algorithms (such as [Sandholm et al., 2005]) can be applied here with a constraint that the assigned coalitions do not overlap for different tasks.

5. *Award acceptance*: Once winner coalitions are determined, the auctioneer awards each task to the leader of the selected coalition. The leader robot then contacts the other coalition members to get ready for the task. Once responses from other coalition members are received, the leader robot accepts the award by sending a task acceptance

message to the auctioneer and the coalition members commit themselves to the task until the task is complete. Otherwise, the award is rejected and the task needs to be auctioned again.

6.5 Summary

This chapter has described my plan for building multi-robot coalitions to perform multi-robot tasks. The lower-level ASyMTRe approach automatically forms coalitions according to the task objective and the team capabilities. The higher-level auction-based task allocation provides the mechanism for the team to allocate sets of tasks, holding auctions to assign tasks to the best-fitting individual robots or coalitions.

As presented later in Chapter 7, a higher-level auction-based task allocator has been implemented and tested both in simulation and on physical robot teams. Different from the proposed approach, the allocator I implemented in the experiments allocates one task at a time (instantaneous assignment) instead of multiple tasks at a time (time-extended assignment). The time-extended task assignment remains as a future work. I also believe that the ASyMTRe approach for coalition formation can be merged with other, non-auction-based approaches to task allocation, such as the motivation-based approach of ALLIANCE [Parker, 1998b]. It would be interesting to investigate the combination of ASyMTRe and ALLIANCE, as an alternative approach for achieving the merging of coalitions for multi-robot tasks with traditional task allocation techniques.

Chapter 7

Experimental Validation

I have designed and implemented both the centralized and the distributed ASyMTRe approaches and validated them with a series of experiments in simulation and on physical robots. These experiments are designed to: (1) illustrate the solution generation process of both approaches by showing that valid and efficient connections are built between schemas, (2) demonstrate the robustness and code reusability of multi-robot applications by applying ASyMTRe, and (3) evaluate the performance of both approaches in time/computational complexity and scalability. In addition, I have developed a high-level auction-based task allocation approach on top of ASyMTRe-D. With this additional layer, a multi-robot team can handle multiple single-robot or multi-robot tasks with instantaneous assignment.

This chapter is organized as follows. I first introduce the experimental platform in Section 7.1 and programming details in Section 7.2. Then, I use a multi-robot navigation task in Section 7.3 and a cooperative multi-robot box pushing task in Section 7.4 to illustrate the solution generation (coalition formation) process of both the centralized and the distributed ASyMTRe-D approaches. The performance evaluation and comparison of both approaches are presented in Section 7.5. Finally, in Section 7.6, I demonstrate the capability of a multi-robot team for handling multiple tasks through a site clearing task.

7.1 Platforms

The ActivMedia Pioneer 3DX mobile robots (Figure 7.1) have served as the hardware platforms in my experiments. Simulation experiments are designed for the Pioneer robots using Player, a networked server for robot control, and Stage, a multi-robot simulator [Gerkey et al., 2001]. Physical experiments are implemented on the Pioneer robots using Player.

7.1.1 Player and Stage

The simulation experiments are performed using Player and the Stage simulation environment [Gerkey et al., 2001]. Player is a network server which provides an interface to a collection of sensors and actuators constituting a robot. Stage is a simulator that supports multiple robots in a two-dimensional bitmapped environment. It is capable of simulating a population of robots and sensors controlled by Player. When used with Stage, Player

Figure 7.1: Pioneer 3DX mobile robot.

provides simulated data in the place of real sensor data. Client programs communicate with Player over a TCP socket, reading data from sensors, writing commands to actuators, and configuring devices on the fly. Since the client side of Player supports C++, the control program for my experiments is implemented in C++. The platform for Player is the Pioneer family of mobile robots. Thus, it can also be used in physical experiments in a similar way as in simulation.

7.1.2 Pioneer 3DX

Pioneer 3DX is a mobile robot with a two-wheel differential drive. It is a standard commercial robot that contains all of the basic components for sensing and navigation in a real-world environment. The CPU of the Pioneer robot is an Intel Pentium III processor with 850MHz speed. The sensors on the Pioneer robots are shown in Table 7.1. In the experiments, I vary the sensing capability of the Pioneer robots so that they play different roles of both capable and resource-bounded robots in the applications. The different sensing capabilities of the Pioneer robots represent the heterogeneity of the multi-robot team.

7.2 Implementation Details

The centralized ASyMTRe approach, the distributed ASyMTRe-D approach, and the high level task allocator are all implemented using C. Multiple threads are used to increase the efficiencies of the programs. The first demonstration of the ASyMTRe approach on physical robot teams was presented in the Master's thesis by [Chandra, 2004] in our DILab. The demonstration in her work still used manually configured schema connections for solutions rather than the automatically generated connections used in the experiments here. Similar to her work, the robot control code in this dissertation is also implemented in C++ for both simulation and physical experiments, and it runs on Player (Version 1.6.5). I also follow the same definitions of sensors, schemas, and robots, and the same flow control as presented in [Chandra, 2004]. Additionally, a set of connection functions are defined to

Table 7.1: Environmental sensors on Pioneer 3DX mobile robots.

ES	Abbreviation	Description
ES_1	laser	Range-finding laser (SICK LMS-200) with .5 degree resolution
ES_2	camera	Pan-Tilt-Zoom Cannon VC-C4 communication camera
ES_3	sonar	8 sonars on the front and 8 sonars on the back
ES_4	communication	Orinoco wireless card

facilitate the flexible connections between schemas and/or sensors. The following contents briefly describe the implementation details.

Environmental sensors and schemas are defined as classes. Each sensor or schema class includes the declaration of a set of input/output information types, any internal variables, and a *compute_output*() or *compute_vector*() function, which calculates output information using the input information. An example of a schema implementation is shown in Table 7.2.

Each robot is also defined as a class, which includes the robot's available sensors, schemas, and internal function definitions. There are three main function calls in the robot class. The solutions are configured by calling the ASyMTRe procedure in a function called *configure*(), which generates the proper connections of schemas to accomplish a task. All the schema connections are then initialized in a function called *init_connection*(). When there are sensor or sonar failures, solutions are configured again in the function *reconfigure*() by modifying the corresponding robot configuration files and calling the ASyMTRe procedure. An example of the robot class implementation is shown in Table 7.3.

In addition to the three main functions, the robot class also includes a series of functions that build connections between two schemas. The connections between schemas are generated by the ASyMTRe approach. Connecting two schemas simply means assigning one schema's outputs to another schema's inputs. For example, if the ASyMTRe procedure produces an output such as "(ES1, PS1)", then the function *connect_es1_ps1*() is called in the *init_connection*() to assign the output from ES1 to the input of PS1.

7.3 Multi-Robot Navigation Task

7.3.1 Task description

In this application, a group of robots must navigate from their starting positions to a set of goal positions (one per robot) defined in a global coordinate reference frame. I assume that all the robots are programmed with the motor schema *go-to-goal*, which moves the robot from its current position to a goal position, defined in a global coordinate reference frame. To successfully use this motor schema, a robot must know its own current position relative to its goal. If every robot can localize itself, obviously the solution is to have every robot navigate independently. However, in some situations, there may be resource-bounded robots that do not have the required sensing, effector, and algorithmic capabilities to localize (e.g., see [Parker et al., 2004]); they need help from more capable robots to provide the information needed to accomplish the navigation task. The purpose of using ASyMTRe here is to enable a close, dynamic cooperation amongst heterogeneous team

Table 7.2: Schema class definition.

Sensor/Schema Class	**Example**

class sensor/schema_name {
public:
Data_Type input_1;
...
Data_Type input_n;
Data_Type output_1;
...
Data_Type output_m;
 	
Data_Type variable;
 	
void compute_output();
}
	class PS4_marker : public PS {
	public:
	long input_id;
	int ***input_pix;
	int input_rows;
	int input_cols;
	Marker output_marker;
	
	void compute_output();
	}

Table 7.3: Robot class definition.

Robot Class	**Example**

class Robot_name : public Robot {
ES1 es1;
...
PS1 ps1;
...
MS1 ms1;
...
 	
public:
void configure();
void reconfigure();
int init_connection();
 	
void connect_esi_ps_j;
...
}
	class Pioneer : public Robot {
	ES1_laser es1_laser;
	PS1_self_pos ps1_self_pos;
	MS1_goto ms1_goto;
	MS2_avoid ms1_avoid;
	
	public:
	void configure();
	void reconfigure();
	void init_connection();
	
	void connect_es1_ps1();
	void connect_ps1_ms1();
	void connect_es1_ms2();
	}

Table 7.4: Eight types of robot with different sensing capabilities.

Type	Available Sensor(s)
R_1	`comm`
R_2	`sonar, comm`
R_3	`laser, comm`
R_4	`camera, comm`
R_5	`sonar, laser, comm`
R_6	`sonar, camera, comm`
R_7	`laser, camera, comm`
R_8	`sonar, laser, camera, comm`

members to accomplish tasks that might be impossible for certain resource-bounded robots to achieve.

7.3.2 Experimental setup

The environmental sensors I use in these experiments are: a laser scanner (ES_1) with an environmental map, a camera (ES_2), and sonars (ES_3), as shown in Table 7.1. These sensors provide up to 2^3 different configuration of robot capabilities in Table 7.4. I make the following assumptions of the robots: (1) all robots have communication capabilities because they have (ES_4), (2) a robot with a laser or sonar can estimate its current global position in an environment with a map, (3) a robot with a camera can estimate the relative position of another robot, as long as the other robot is within its field of view.

I have implemented the following schemas; however, these schemas are not connected to each other at the beginning of the task:

- PS_1, which calculates a robot's own global position using laser;
- PS_2, which calculates a robot's own global position using sonar;
- PS_3, which calculates another robot's relative position using camera with a fiducial marker;
- PS_4, which calculates a robot's own global position according to another robot's global position and relative position;
- PS_5, which calculates the global position of another robot according to its own global position and the estimated relative position of the other robot;
- CS_i, which transfers information across robots; and
- MS_1, which calculates motor commands that lead the robot toward the goal.

The task T is defined as MS_1, meaning that MS_1 should be activated on all coalition members. In Table 7.5 and Table 7.6, I define the set of information types F and label the input and output information for each schema used in this application. According to the

Table 7.5: Input and output information types in the navigation task.

Type	**Description**
F_1	Self_Global_Position
F_2	Other_Global_Position
F_3	Other_Relative_Position
F_4	Goal_Position
F_5	Motor_Commands

Table 7.6: Input and output information types for corresponding schemas and their sensing costs and success probabilities.

S_i	I^{S_i}	O^{S_i}	C_i	P_i
PS_1	Laser	F_1	High	High
PS_2	Sonar	F_1	Medium	Medium
PS_3	Camera	F_3	Medium	Medium
PS_4	F_2 and F_3	F_1	None	Medium
PS_5	F_1 and F_3	F_2	None	Medium
CS_1	F_1	F_2	Low	High
CS_2	F_2	F_1	Low	High
MS_1	F_1 and F_4	F_5	None	High

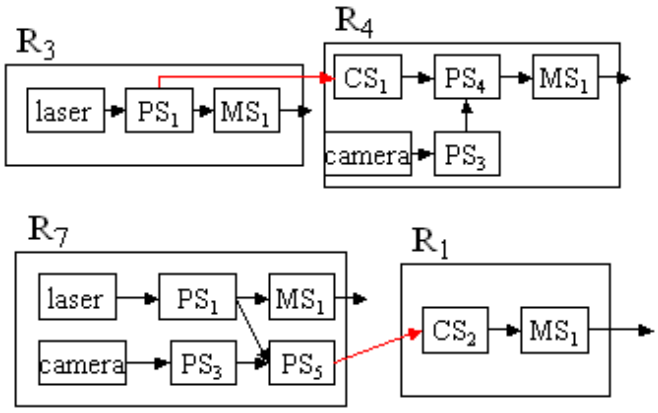


Figure 7.2: Two ways to connect the schemas in the navigation task.

flow of information, the ASyMTRe configuration process should generate all the possible connections that can connect the available schemas and lead each coalition member to achieve its goal. Two specific connections are shown in Figure 7.2. The first solution involves R_3 using its laser to localize and then communicating this information to R_4. R_4 then combines the received information with the detected relative position of R_3 (using R_4's camera) to calculate its own global position. The second solution involves R_7 using its laser to globally localize itself and using the camera to calculate the relative position of R_1. With this information, R_7 can calculate the global position of R_1 and communicate this information to R_1.

With multiple solutions available, the robot team needs to determine which solution it should use. This is decided by each robot's sensing cost and the estimated success rate of the particular solution it chooses. Typically, the sensing cost is determined by the sensory and computational requirements of the solution. Perceptual processes with a significant amount of sensor processing, such as laser scan matching or image processing, are given higher sensing costs. Perceptual processes with a relatively low processing requirement, such as sonar, are assigned lower sensing costs. Success probability is an estimated value based upon learning and experience. Perceptual processes that are easily influenced by environmental factors, such as image processing under different lighting conditions, are given lower success probabilities. Here, we only provide fuzzy estimates for costs and probabilities (see Table 7.6); in actual applications, these estimates will likely be specific numeric values learned from experience.

7.3.3 Vision-based fiducial marker detection

To detect the relative pose information for another robot, I used the algorithm that I developed for our navigation assistance in the SDR project. The following design and algorithms are previously reported in [Parker et al., 2004]. The marker I use is a striped cylindrical color marker as shown in Figure 7.3. The actual height of the marker is 48 cm, and its circumference is 23 cm. The marker is composed of four parts: a START block, an ID block, and Orientation block and an END block. The START block is a combination

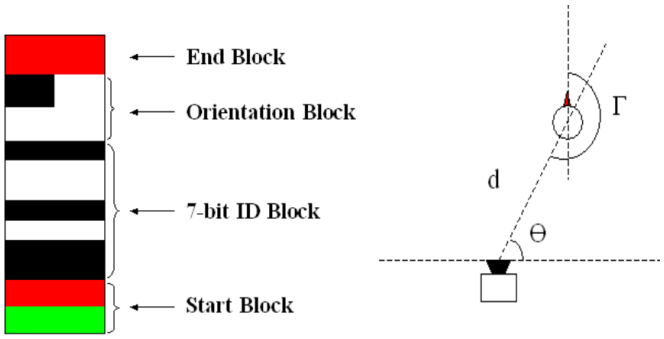


Figure 7.3: Cylindrical marker design to provide unique visual ID, relative position, and orientation information.

of red and green stripes at the bottom of the marker. The END block is a red stripe at the top of the marker. The START and END blocks make the marker unique in a regular environment. Adjacent to the END block is the Orientation block. The relative orientation of a robot is calculated by the width ratio of black and white in the Orientation block. The ID block is composed of 7 black or white stripes, where black represents 1 and white represents 0. This block provides $2^7 = 128$ different IDs and is easily extended to identify more robots if needed.

Once a marker is recognized from the camera image, the marker detection algorithm determines the identity and the relative position of the marker in terms of the following parameters, as shown in Figure 7.3:

- d: the distance between the observing camera and the center of the marker;
- Γ: orientation of the marker – the angle between the heading of the marker and the center of the camera; and
- Θ: the angle between the center of the marker and the plane containing the camera.

Suppose that a marker of height h is located at (x, y) in the image plane of (r, c) pixels, the edges of the marker are (l, r), and the delimitation is located at column k. Then the above parameters are calculated as follows:

$$d = \frac{C_1}{h \times C_2} \tag{7.1}$$

$$\Gamma = 180 \times \frac{k - l}{r - k} \tag{7.2}$$

$$\Theta = FOV + \frac{x}{c} \times (180 - 2 \times FOV) \tag{7.3}$$

where FOV is the field of view of the camera, and C_1 and C_2 are constants defined by the size of the real marker.

Figure 7.4: Physical robot implementation of the navigation task [Chandra, 2004]. The initial setup of the two robots is on the left. The result when the two robots reach their goal points is shown on the right.

7.3.4 Multi-robot navigation using centralized ASyMTRe

The schemas described above were implemented on two Pioneer robots equipped with a SICK laser range scanner and a Cannon pan-tilt-zoom camera. Both robots possess a wireless ad hoc networking capability, enabling them to communicate with each other. Experiments were conducted in a known indoor environment using a map generated using an autonomous laser range mapping algorithm. Laser-based localization used a standard Monte-Carlo localization technique. The implementation of PS_3 makes use of prior work in [Parker et al., 2004] for performing vision-based sensing of the relative position of another robot. This approach makes use of a cylindrical marker designed to provide a unique robot ID, as well as relative position and orientation information suitable for a vision-based analysis. Using these two robots, three sets of experiments based on sensor availability were tested to illustrate the ability of these building blocks to generate fundamentally different cooperative behaviors of the same task through sensor sharing.

Experimental results

Experiment set 1 is a baseline case in which both robots have full use of their laser scanner and camera. Each robot localizes itself using its laser scanner and reaches its own goal independently. Experiment set 2 involves a fully capable robot R_3 with laser, as well as a robot R_4 with only a camera. They connect their schemas according to the first solution shown in Figure 7.2 to accomplish the task. Experiment set 3 involves a robot R_1 with communication capabilities only and a fully-capable robot R_7. They connect their schemas according to the second solution shown in Figure 7.2. A snapshot of these experiments is shown in Figure 7.4.

In extensive experimentation, data on the success rate was collected with an average of 10 trials of each set. Robots in experiment set 1 were 100% successful in reaching goal positions. Experiment set 2 had four failures and set 3 had one failure. The failures were caused by variable lighting conditions that led to a false calculation of the relative robot positions using the vision-based robot marker detection or the robot was not in the field of view of the observer. However, even with these failures, these overall results are better than what would be possible without sensor sharing. In sets 2 and 3, if the robots did not share their sensory resources, one of the robots would never reach its goal position, since it would not have enough information to determine its current position. Thus, the sensor sharing

mechanism extends the ability of the robot group to accomplish tasks that otherwise could not have been achieved. To increase the robustness of the application, once a failure is detected, the ASyMTRe reasoning process can be called to reconfigure solutions for the available robots. I discuss this further in the following section.

7.3.5 Multi-robot transportation using distributed ASyMTRe-D

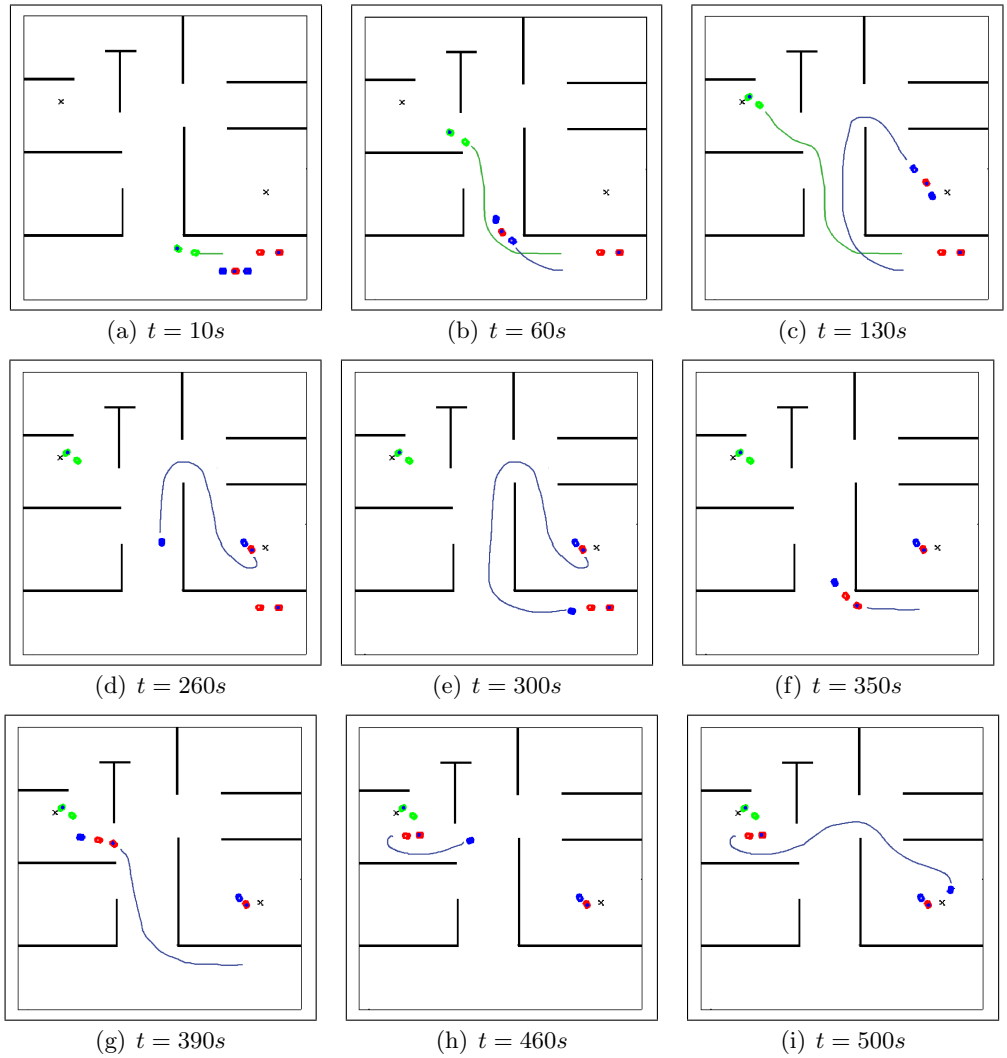
The previous experiments on the centralized ASyMTRe approach have validated the correctness of the ASyMTRe approach for automatically generating schema connections within or across robots. Now, I evaluate the ASyMTRe-D approach through more complex experiments, in order to demonstrate two aspects of the ASyMTRe-D approach: the continuous reasoning capabilities of the robot group over a period of time while performing a more complex task, and the robustness of the ASyMTRe-D approach.

Simulation setup and results

To help illustrate how ASyMTRe-D can be used in more complex tasks requiring new coalitions over time, I implemented the navigation task in simulation, using a larger number of robots. In this task, robots are randomly assigned a series of goal positions to visit*. In some cases, robots are assigned the same goal position. Once a subgroup of robots has been assigned their goal positions, the robots form coalitions as needed using ASyMTRe-D. When all the robots accomplish their current tasks (i.e., visit their assigned goal positions), new positions are assigned and new coalitions are formed as needed. In the experiment reported in Figure 7.5, only two robots out of the group of seven can navigate independently to goal positions. The remaining robots need navigation assistance, as described previously in the earlier version of the navigation task. Note that there is no representation of the fiducial marker for physical robots in the Stage simulation environment. Thus, instead of calculating the relative position of a marker as presented earlier in this section, each robot in this experiment is programmed with a perceptual schema that uses camera to detect a particular color blob and a motor schema that enables it to follow a color blob. The task priority for robots in this application is: (1) help robots in the group that share the same goal position; (2) help robots in any other group; (3) navigate to robot's own goal position.

Figure 7.5 is one of the typical runs of the transportation task in simulation. Here, I generated two random goals for the robot group, assigning the first goal position to a group of three robots, and the other goal position to a group of four robots. As shown in this figure, each of the more capable robots leads less capable robots to their goal positions. One of the more capable robots then returns to form a second coalition of robots for reaching their goal positions. Any introduction of a new position or the failure of a robot to find help will trigger the ASyMTRe-D negotiation process to configure new coalitions. This process continues repeatedly as new position assignments are made.

*Note again that I am not addressing the higher-level assignment of single-robot tasks (i.e., goal positions) to multiple robots. This issue is addressed by other task allocation approaches fitting the ST-SR-IA and ST-SR-TA taxonomic category. Here, I assume that other task allocation approaches would determine the assignments of goal positions to robots. I focus instead on how robots can repeatedly form different coalitions to enable individual robots to reach their assigned positions.



(a) $t = 10s$ (b) $t = 60s$ (c) $t = 130s$

(d) $t = 260s$ (e) $t = 300s$ (f) $t = 350s$

(g) $t = 390s$ (h) $t = 460s$ (i) $t = 500s$

Figure 7.5: The simulation results of a more complex multi-robot navigation task, where robots are given successive goal positions. Figure (a) to (c): the first group (four robots) configures the solutions, two robots start out to go to the position on the left (represented by the cross), while two robots wait at the starting positions for help; the other group configures solutions and starts out moving to the position on the right. Figure (d) to (f): after the third round of configuration that involves the two capable robots and the rest of the robots that still need help, the leader robot in the second group (right) goes back to pick up the two robots since it is the only one that can help them (The two robots can only perceive the color blob carried on this particular leader robot.). Figure (h): the other two robots follow the leader to their goal position. Figure (i) and (j): the leader robot goes back to its original goal.

Table 7.7: The average time and the standard deviation of the three set of experiments over 10 trials per set.

Task	Completion Time	Std. Deviation
No failure introduced	96.1s	1.45s
Partial failure	112.6s	2.63s
Robot death	187.1s	5.45s

Physical experimental setup and results

In order to evaluate the robustness of the ASyMTRe-D approach, I designed three different sets of experiments on a group of physical robots. In addition to the schemas stated in Table 7.6, the robots are also programmed with a motor schema MS_2 that calculates motor commands that lead the robots to follow a certain marker. The team-level goal is to activate either MS_1 (goto) or MS_2 (follow) on the team of robots. According to the different categories of failures in a typical experiment defined by [Dias, 2004], I illustrate how the ASyMTRe-D approach will help the robot coalition recover from partial (sensor) failure, or complete robot death. The robot group is composed of the following types of robots: R_1 with a laser-scanner and an environmental map for the robot to localize, and a camera mounted backward to calculate the global position of another robot (within its field of view) based on marker detection; R_2 with a camera mounted in the front to track the marker within its field of view; R_3 with no sensors. I also assume that all robots have communication capabilities. To accomplish a task, robots of the coalition must navigate from a starting position to a goal position.

I conducted three sets of experiments to illustrate the robustness of the ASyMTRe-D approach. In experiment set 1, R_1 and R_2 form a coalition such that R_2 follows R_1 by tracking the marker mounted on R_1. There are no failures introduced in this set. In experiment set 2, R_1 and R_2 still form a coalition with R_2 tracking R_1. During the execution, the camera on R_2 is covered such that it cannot detect R_1 anymore. This sensor failure triggers the reasoning process to generate new solutions for the two to accomplish the goal. The new solution is for the leading robot R_1 to calculate R_2's global position and communicate this information to R_2. An example of this experiment is shown in Figure 7.6. In experiment set 3, R_1 and R_2 form a coalition at the beginning, then during execution, a simulated robot death is introduced on R_2, which triggers the whole group to reconfigure solutions. The new solution is that R_1 goes back to pick up R_3, and they navigate together to the goal position. An example of this experiment is shown in Figure 7.7.

I performed 10 successful trials for each experiment set, and collected data on the completion time (reported in Table 7.7 and Figures 7.6 and 7.7) and the number of successful trials. In total, I have performed 32 trials, with one failure in set 2 and 3 respectively. Both failures happened because of false marker detection data when the leader robot tried to guide the follower robot. More fault detection mechanisms can be built into the application to increase its robustness, such as monitoring group members to detect their faults [Gerkey and Mataric, 2002]. These experiments show the software reconfigurability of ASyMTRe-D upon failures.

Figure 7.6: ASyMTRe-D handles partial robot failures in the navigation task. In this experiment, the leader robot R_1 uses its laser to navigate and the follower robot R_2 uses its camera to follow the marker on the leader. During task execution, the camera on R_2 is covered (as indicated by the arrow), and the coalition reconfigures to continue the task. The new solution is that R_1 uses its camera to guide R_2 to the goal.

Figure 7.7: ASyMTRe-D handles simulated robot failures in the navigation task. The first figure shows that the leader robot R_1 uses laser to navigate and the follower robot R_2 uses camera to follow the marker on the leader. The second figure shows that there is a simulated failure of R_2 and the coalition reconfigures the solution. The new solution is that R_1 goes back to pick up another follower R_3 and guide it to the goal, which are shown in the rest two figures.

7.4 Cooperative Multi-Robot Box Pushing

The cooperative box pushing problem has been addressed by many researchers in designing multi-robot systems [Kube and Zhang, 1993,Kube and Zhang, 1996,Sen et al., 1994,Donald et al., 1994,Mataric et al., 1995,Parker, 1994b,Gerkey and Mataric, 2002]. The foundation of this work – the theoretic study of information invariants by Donald et al. – also used box pushing examples to measure the information complexity of robot tasks. To illustrate the connection of the ASyMTRe approach to the theory of information invariants, I have developed the box pushing experimentation in a similar manner to [Donald et al., 1993]. Additionally, I have implemented the box pushing example using the method in [Parker, 1998a], where one robot may push on both ends of a box in order to successfully move it to a goal position. The pusher-watcher example in [Gerkey and Mataric, 2002] is also a potential protocol that can be applied to ASyMTRe, although it is not implemented in the experiments. Over 20 physical robot runs of the box pushing task were completed to illustrate the code reusability introduced by ASyMTRe, the flexibility of applying ASyMTRe to various protocols, and the robustness of the system.

7.4.1 Task description

The cooperative multi-robot box pushing task requires multiple team members (≥ 2) to push a long box from a starting position to a goal position, without any requirement for the orientation of the box or the need to deal with obstacles along the pathway. I make the following assumptions:

- The box is long enough for multiple team members to work on, and they are properly arranged on one side of the box at the beginning of the task. Additionally, robots know that they are on the same flat face of the box.
- The length of the box decides the different pushing points on the box. For example, a box that needs two pushing points can by pushed by two robots simultaneously or by one robot pushing both ends of the box in turns.
- The goal position is represented by a red blob. There is an obstacle-free path between the starting position and the goal position that is wide enough for the box and the team members to pass.

7.4.2 Box pushing protocols

I now present the various box pushing protocols that inspired my work on physical experiments of the box pushing task. The schemas supporting these protocols are implemented on the Pioneer robots. In these physical experiments, the number of robots required to push a box is two. The protocols presented in the following descriptions can also be extended as needed to accommodate more robots.

Protocol I

The first protocol is adapted from *Protocol II* stated in [Donald et al., 1994]. In the adapted protocol, two robots push the box towards a red blob by continuously monitoring the

difference $\theta(t)$ between the current pushing direction $p(t)$ and the relative box orientation $n(t)$, updating their speed profiles, and coordinating their motor controls. Two robots are initially placed close to the same face of the box. The robots can use the *align* motor schema to locate the correct positions on the box to push and align themselves with the box such that the orientations of the robots are perpendicular to the pushing face of the box. First, $\theta(0)$ is measured. Then, at every time step of the task, each robot continues to measure $\theta(t)$ (the angle between $p(t)$ and $n(t)$), and compares it with the initial angle $\theta(0)$. An increase in the angle implies a counterclockwise rotation of the box and a decrease in the angle implies a clockwise rotation. The robots adjust their pushing speed profile according to the desired direction of rotation. Additionally, two robots pushing the box concurrently share their status with each other to coordinate their behaviors. The status information includes: *pushed*, meaning that the other side of the box has just been pushed; and *alignment*, meaning that the other robot may lose contact with the box and needs alignment to locate the pushing position on the box. Algorithm 5 shows the details of the box-pushing protocol.

Protocol II

The second protocol is similar to the box pushing control that is implemented in [Parker, 1998a]. Robots continue checking whether they are in contact with the box and push both ends concurrently. In this case, two robots also share their status with each other to decide the next desired move. Compared with protocol I, this protocol does not need the robot to calculate the accurate orientation of a box, and thus can easily be implemented using less accurate sensors. Algorithm 6 shows the details of this box pushing protocol.

Protocol III

The third protocol is a variation of protocol II and is also inspired by [Parker, 1998a]. Instead of having two robots push both ends of a long box concurrently, this protocol enables one robot to push one end of the box, move to the other end, and push the other end of the box. In this way, one robot could accomplish the box pushing task with a longer duration of time. Algorithm 7 shows the details of this box pushing protocol.

7.4.3 Discussion

The above protocols provide three different ways of accomplishing the cooperative box pushing task. However, these protocols are limited to pushing a box with either one or two robots. Ideally, we would like to have the ASyMTRe approach decide the number of robots that are required to push a box. The idea is that when the weight and size of the box are known, and the average pushing force and the size of the robot are known, we are able to reason about the potential pushing positions on the box, and thus the required robot coalition size. To do this, we need to model the physics of both the box and the robot and reason about their relationships, which is outside the scope of this dissertation. For the box pushing experiments and the site clearing experiments presented in this chapter, I assume that the number of robots required to push a box is decided by human operators, while the robot coalitions are automatically formed with the specified size.

Algorithm 5 The box pushing protocol for two robots to push a long box towards a goal position, adapted from [Donald et al., 1994].

```
Estimate θ1(0)
while (goal_not_arrived) do
  Estimate pushing direction p
  estimate θ1(t)
  if (break?) then
    align()
  end if
  if (other_side_been_pushed) then
    if (θ1(t) ≈ θ1(0)) then
      push(p)
    else if (θ1(t) ≫ θ1(0)) then
      if (at the left side of the box) then
        push(p)
      else if (at the right side of the box) then
        no action
      end if
    else if (θ1(t) ≪ θ1(0)) then
      if (at the left side of the box) then
        no action
      else if (at the right side of the box) then
        push(p)
      end if
    end if
    send_msg(this_side_been_pushed)
  else
    wait(t)
  end if
end while
```

push(p): push the box towards direction p
align(): track the box when the contact is lost
send_msg(): send "pushed" message to the teammate
wait(t): wait for the teammate's message

Algorithm 6 The box pushing protocol for two robots to push a long box towards a goal position, adapted from [Parker, 1998a].

```
while (goal_not_arrived) do
  Estimate pushing direction p
  if (break?) then
    align()
  end if
  if (other_side_been_pushed) then
    push(p)
    send_msg(this_side_been_pushed)
  else
    wait(t)
  end if
end while
```

push(p): push the box towards direction p
align(): track the box when the contact is lost
send_msg(): send "pushed" message to the teammate
wait(t): wait for the teammate's message

Algorithm 7 The box pushing protocol for one robot to push a long box towards a goal position, adapted from [Parker, 1998a].

```
while (goal_not_arrived) do
  Estimate pushing direction p
  if (break?) then
    align()
  end if
  push(p)
  move_to_opposite_end()
  align()
end while
```

push(p): push the box towards direction p
align(): track the box when the contact is lost
move_to_opposite_end(): move along the box until the other end is reached

7.4.4 Experimental setup

According to the above box pushing protocols, I have developed a library of perceptual, communication, and motor schemas for a team of robots. The environmental sensors are: a laser scanner (ES_1), a camera (ES_2), and sonars (ES_3), as shown in table 7.1.

The schemas that are preprogrammed on the robots are:

- PS_1, which calculates the orientation of the box relative to the pusher robot using laser range data;
- PS_2, which calculates the pushing direction for the pushers based on camera image data;
- PS_3, which checks whether the pusher robot has lost contact with the box or needs alignment using laser range data;
- PS_4, which selects different combinations of motor schemas according to the box's orientation, the pushing direction, etc;
- PS_5, which checks whether the pusher robot has lost contact with the box or needs alignment using sonar range data;
- MS_1, which generates motor control vectors that lead the robot to push the box towards goal position;
- MS_2, which generates motor control vectors that lead the robot to locate the box and align with it;
- MS_3, which generates motor control vectors that lead the robot to move to the opposite end of a box using sonar range data only;
- MS_4, which combines output from proper motor schemas and generates the ultimate "push" behavior; and
- CS_i, which transfers information across robots.

The task T is defined as MS_1, MS_2 or MS_1, MS_3, meaning that either MS_1 and MS_2 or MS_1 and MS_3 need to be activated on all coalition members. Table 7.8 shows the set of information types used in the cooperative box pushing task. Table 7.9 shows the input/output information types of each sensor and schema and their predefined cost and success probability used in the experiments.

According to the experimental setup of the box pushing task, the ASyMTRe approach can generate different ways of schema connections that enable the robot team to accomplish the box pushing task. Figure 7.8 shows the details of the connections corresponding with the three protocols presented above. The code reusability of the ASyMTRe approach is illustrated in protocol II and III. These two protocols share most of the same schemas, but generate different behaviors based on the different connection of schemas. The implementation of the three box pushing protocols using ASyMTRe-based schema approach also demonstrates the applicability of ASyMTRe to various multi-robot applications.

Table 7.8: Input and output information types in the cooperative multi-robot box pushing task.

Type	**Description**
F_1	Laser range data
F_2	Camera image
F_3	Sonar range data
F_4	Box's relative orientation
F_5	Pushing direction
F_6	Minimum laser index
F_7	Minimum sonar index
F_8	Motor control option
F_9	Preprocessed sonar range data
F_{10}	Motor control vectors for pushing
F_{11}	Motor control vectors for alignment
F_{12}	box's status information: pushed by itself
F_{13}	box's status information: pushed by teammate
F_{14}	Motor control vectors for box pushing towards a goal position

Table 7.9: Input and output information types for corresponding schemas and their sensing/time costs and success probabilities.

S_i	I^{S_i}	O^{S_i}	C_i	P_i
PS_1	F_1	F_4	High	Medium
PS_2	F_2	F_5	Low	High
PS_3	F_1	F_6	High	High
PS_4	F_4 and F_5	F_8	None	High
PS_5	F_3	F_7, F_8 and F_9	Medium	Medium
MS_1	F_5 and F_8	F_{10}	Low	High
MS_2	F_6 and F_8, or F_7 and F_8	F_{11} and F_{12}	Low	High
MS_3	F_7, F_8 and F_9	F_{11}, F_{12} and F_{13}	High	High
MS_4	F_{10}, F_{11} and F_{13}	F_{14}	None	High
CS_1	F_{12}	F_{13}	Low	High

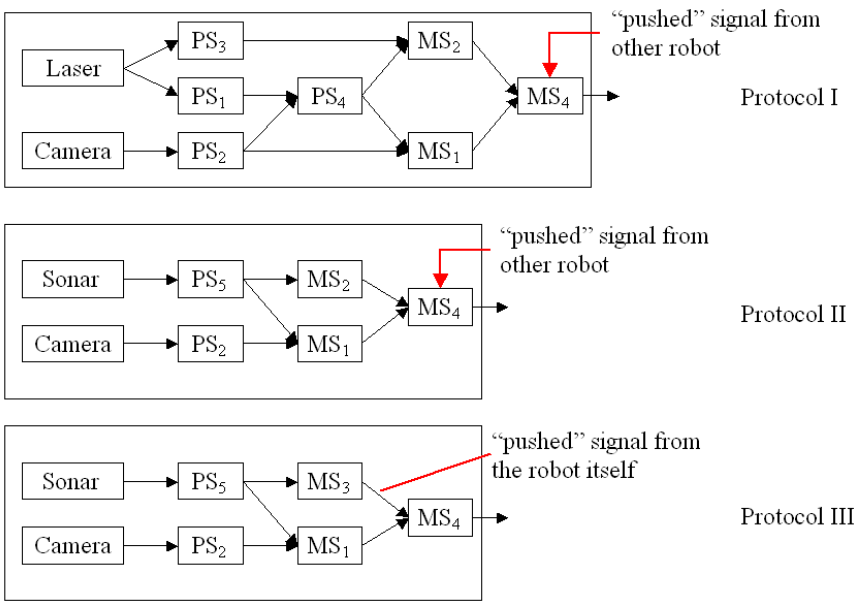


Figure 7.8: Three different ways to connect schemas on a robot to accomplish the box pushing task. Each way corresponds with one protocol described above.

7.4.5 Physical box pushing using distributed ASyMTRe-D

The following physical box pushing experiment is designed to demonstrate the flexibility of the solutions generated by ASyMTRe-D based on the different sensing and computational capabilities of the robot team, and the robustness of the ASyMTRe-D approach upon various sensor failures. Initially, the robot team is composed of two robots, both of which are equipped with a laser-scanner, a camera mounted forwardly, and a ring of sonars. I also assume that both robots have communication capabilities and they are preprogrammed with all the schemas shown in Table 7.9.

I conducted two sets of experiments to illustrate the flexibility and the robustness through using ASyMTRe-D. Figure 7.9 illustrates an instance in experiment set 2 when failures are introduced to the system. In experiment set 1, R_1 and R_2 form a coalition such that both of them use protocol II, using sonars to detect and align with the box and using camera to detect the goal position. There are no failures introduced in this set of experiments. The average time for robots to accomplish the box pushing task in experiment set 1 over 5 runs is 45s with a standard deviation of 7.8s. In experiment set 2, R_1 and R_2 still start out by using protocol II. During the execution of the task, sonar failures are introduced to R_2 that triggers the team to stop and reconfigure solutions based on current team capabilities (R_1 with laser, sonar, and camera, and R_2 with laser and camera). A new solution is generated online through ASyMTRe-D such that R_1 uses protocol II with sonar, but R_1 uses protocol I with laser to continue pushing the box at both ends. Again, a laser failure is introduced to R_2 during its execution which triggers the ASyMTRe-D approach to reconfigure a new solution to accommodate the changes in team capabilities. The new solution is that R_1 uses protocol III to push the box by itself using sonar. The average time

(a) Configuring solutions: both robots push the box with sonars.

(b) Reconfiguring solutions: sonar failure on R_2.

(c) New solution: R_1 uses sonar and R_2 uses laser.

(d) Reconfiguring solutions: laser failure on R_2.

(e) New solution: R_2 backs up, R_1 pushes by itself.

(f) R_1 moves along the box to reach the opposite end.

(g) R_1 pushes the left side.

(h) Goal reached.

Figure 7.9: A series of snapshots taken during the box pushing task. Two sensor failures are handled by the system: sonar and laser failures on R_2.

for robots to accomplish the box pushing task in experiment set 2 over 10 runs is 152.8s with a standard deviation of 9.8s.

7.5 Comparison between Centralized ASyMTRe and Distributed ASyMTRe-D

The above experiments present the example operational results of applying ASyMTRe to various multi-robot applications and the robustness of the ASyMTRe approach through solution reconfiguration. The following experiments explore the performance of both approaches and compare their scalabilities and solution qualities by varying the size of the robot group (n). First, I define a simple case where the group is composed of five robots with various capabilities (see Table 7.4) to accomplish the navigation task. I then increase the group size by duplicating the robots with the same set of capabilities in the simple case. I ran these instances on both centralized ASyMTRe and distributed ASyMTRe-D and collected data on the reasoning time and the average coalition utility (team utility averaged over the number of robots). In centralized ASyMTRe, the total time for generating a solution includes: the time to generate all the orderings of robots, which increases exponentially ($O(n!)$), and the actual reasoning time ($O(mn^2)$). In ASyMTRe-D, the time is the averaged reasoning time ($O(mn)$) for the group to generate a solution. Here, m is the size of the solution space. For these experiments, the negotiation timeout values are set as follows: wait for reply: 0.75s; try repetitive requests: 10s; and wait for confirmation: 4s. As shown in Figure 7.10, the average time to generate a solution increases as the robot group size increases, linearly for ASyMTRe-D, but exponentially for centralized ASyMTRe, but the coalition utility does not vary significantly with varying group sizes. Additionally, in Figure 7.11, the coalition utility is plotted for four different coalition sizes. In centralized ASyMTRe, the coalition utility increases as more time is given because of the anytime aspect of the centralized reasoning algorithm. Additionally, the centralized results always have a higher utility than that of the ASyMTRe-D, since the centralized approach operates with complete information.

When comparing centralized ASyMTRe with ASyMTRe-D (see Table 7.10), we observe that ASyMTRe-D is robust and flexible with little maintenance of the knowledge base since any change in the team capabilities only needs to be updated locally. However, ASyMTRe-D trades off its solution quality because of the local greedy search process. If we run centralized ASyMTRe on a single robot (method 2 in Table 7.10), the best solution can be found given enough time. However, except for the concern of single point failure, this method requires a complete sharing of robots' capabilities at the beginning and sending the solutions back to all robots at the end. The centralized knowledge base also needs to be updated when the team capabilities change. To increase the robustness, we could run centralized ASyMTRe on every robot (method 3 in Table 7.10). However, robots still need to share capability information with each other at the beginning or whenever the team capabilities change. This method requires more work to maintain the knowledge base than the centralized approach on a single base station, since the knowledge base updates must be duplicated on all robots. In conclusion, the centralized ASyMTRe approach is

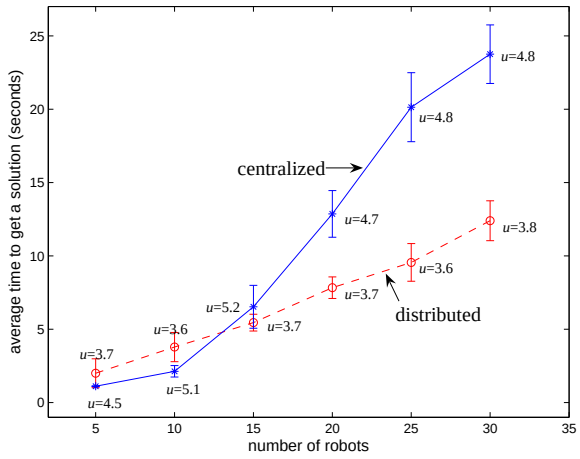


Figure 7.10: The comparison between ASyMTRe and ASyMTRe-D in time and utility. The average time increases linearly for ASyMTRe-D, but exponentially for centralized ASyMTRe with increasing team size. The coalition utility is also shown on each data point given the specific group size and reasoning time. The computation of the time and utility is averaged over 10 samples. The standard deviation of the utility is 0.3 with an average of 3.7 for ASyMTRe-D, and is 0.2 with an average of 4.8 for centralized ASyMTRe.

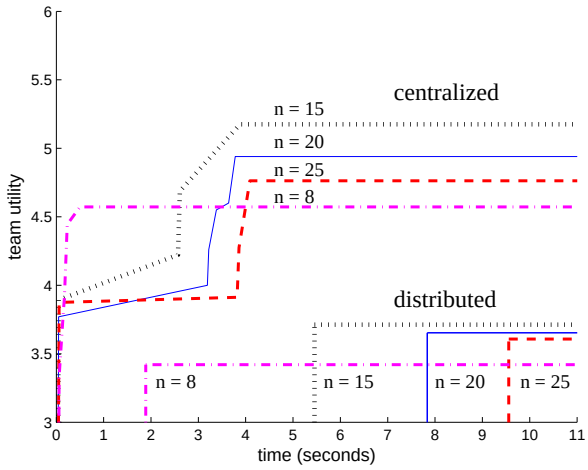


Figure 7.11: The comparison of solution qualities between ASyMTRe and ASyMTRe-D. Given a specific group size (n = 8, 15, 20, 25), the solution quality of the centralized approach increases over time, which is also relatively better than the distributed approach. In these applications, we were able to run the centralized algorithm to its completion only when n is 8, which generates an optimal solution. Note that the time to generate the orderings of robots is not counted for the centralized ASyMTRe approach here.

Table 7.10: Comparison between centralized ASyMTRe and distributed ASyMTRe-D.

Method	**ASyMTRe-D**	**Centralized ASyMTRe on one robot (base station)**	**Centralized ASyMTRe on every robot**
Computational Complexity	$O(mn)$	Optimal solution: $O(mn!)$ First solution: $O(mn^2)$	Optimal solution: $O(mn!)$ First solution: $O(mn^2)$
Solution Quality	Locally optimal	Globally optimal given enough time	Globally optimal given enough time
Robustness	High	Single point failure	Yes, with redundancy
Maintenance	Low	Medium	High

appropriate for robot teams with a smaller size, and the distributed ASyMTRe-D approach is appropriate for robot teams with a larger size.

7.6 Site Clearing Task

The experiments in the above sections have validated that the ASyMTRe approach can enable a robot team to: (1) build proper schema connections for a robot team to accomplish a task, (2) configure/reconfigure task solutions based on current team capabilities and task requirements, and (3) form robot coalitions as needed to accomplish a multi-robot task. However, the ASyMTRe approach only handles one task at a time. To extend the capability of the ASyMTRe to solve multiple tasks, I developed a high-level auction-based task allocation layer on top of the ASyMTRe approach. The purpose here is not to develop a new task allocation method, but to demonstrate that the ASyMTRe approach can serve as a low-level solution generator, together with the traditional task allocation method, to handle multiple tasks (either single-robot or multi-robot task). This section focuses on the development of the site clearing task as a proof-of-concept application that shows the new capability of the system by layering ASyMTRe with task allocation.

7.6.1 Task description

Recall the motivating example from Chapter 6, illustrated in Figure 6.1. The site clearing task requires a specific area to be cleared of obstacles, which we simplify to be boxes with different weights or sizes. The task objective is to clear the site in as little time as possible while minimizing the cost to the robots (e.g., energy consumption or computational requirements). I design this site clearing task to demonstrate the ability of layering lower-level coalition formation with higher-level task allocation techniques to solve multiple single-robot or multi-robot tasks. According to the assumptions listed in Chapter 6, the overall site clearing task can be decomposed into a series of tasks with ordering constraints. Each task is aimed at removing one obstacle from the site, which we call "Remove Obstacle", requiring that the robots as a coalition first navigate to the obstacle and then push the obstacle to the desired collection point. Unlike the task tree I have shown in Figure 6.4, robots do not need to go back to the home base after they have successfully removed an obstacle. Instead, the robots stay near the obstacle and wait for the next task. In these

experiments, I make the following assumptions: the human operator specifies the position of each obstacle and its closest collection point, and each robot knows its current position, navigation speed and pushing speed.

I have developed an auction-based system that performs multi-robot task allocation with instantaneous assignment[†]. In this system, the human operator is represented by an auctioneer whose responsibilities include: randomly introducing the "Remove Obstacle" task to the robot team; conducting an auction on each task in the order of its arrival time; collecting bids and awarding the task to a selected coalition; and monitoring the execution of the task. Each robot is responsible for negotiating with each other to form coalitions when a task is announced, bidding for the task, accepting the award, and performing the task with the other coalition members. In order to minimize the time spent to clear the site, a greedy algorithm is applied to the instantaneous task assignment, meaning that the current task under consideration is allocated to the coalition that could accomplish this task with the least cost. The cost is a weighted combination of the task- and coalition-related cost and the inherent cost of the coalition performing the task. The inherent coalition cost is calculated by Equation 3.5. The task- and coalition-related cost is decided by the task completion time $t_{complete}$:

$$t_{complete} = t_{nav} + t_{push} \tag{7.4}$$

$$t_{nav} = \max_{R_j \in coalition_i} distance(R_j, box)/speed(R_j) \tag{7.5}$$

$$t_{push} = \max_{R_j \in coalition_i} distance(box, goal)/speed(R_j) \tag{7.6}$$

From the above functions, we can see that when robots in a coalition have different speed profiles and different positions, the time for this coalition to accomplish a "Remove Obstacle" task depends on the slowest robot in the coalition. Thus, a robot always informs the other coalition members of its speed and current position to calculate the overall coalition cost.

7.6.2 Code reusability

The above task has been implemented in simulation and on physical teams to demonstrate the potential of layering ASyMTRe with high level task allocation approaches. One benefit that can be achieved by applying ASyMTRe to various applications is code reusability. Recall the multi-robot navigation and cooperative box pushing tasks I presented earlier in this chapter many schemas have been programmed for a robot to navigate from a current position to a goal position, and for a robot to push a box towards the goal in coordination with another robot or by itself. In this site clearing task, the low level requirements of the schemas are similar to the above two tasks; thus, the robots can reuse the schemas that have been previously programmed although the high level connections of schemas are different. In addition, each robot is equipped with the distributed ASyMTRe-D reasoning capability, and thus can negotiate with each other to generate the desired connections of schemas to

[†]Here, instantaneous assignment means that only one task is considered by the team at a time. My future work includes applying combinatorial auction that handles multiple tasks at a time.

Table 7.11: Schemas reused in the site clearing task.

Application Name	**Schemas**
Multi-robot navigation	PS_1, MS_1
Cooperative box pushing	PS_1, PS_2, PS_3, PS_4, PS_5 MS_1, MS_2, MS_3, MS_4

Table 7.12: Robot capabilities in the site clearing task.

Robot	**Available Sensor(s)**
R_1	`sonar, laser, camera, comm`
R_2	`laser, camera, comm`
R_3	`sonar, camera, comm`
R_4	`sonar, laser, camera, comm`

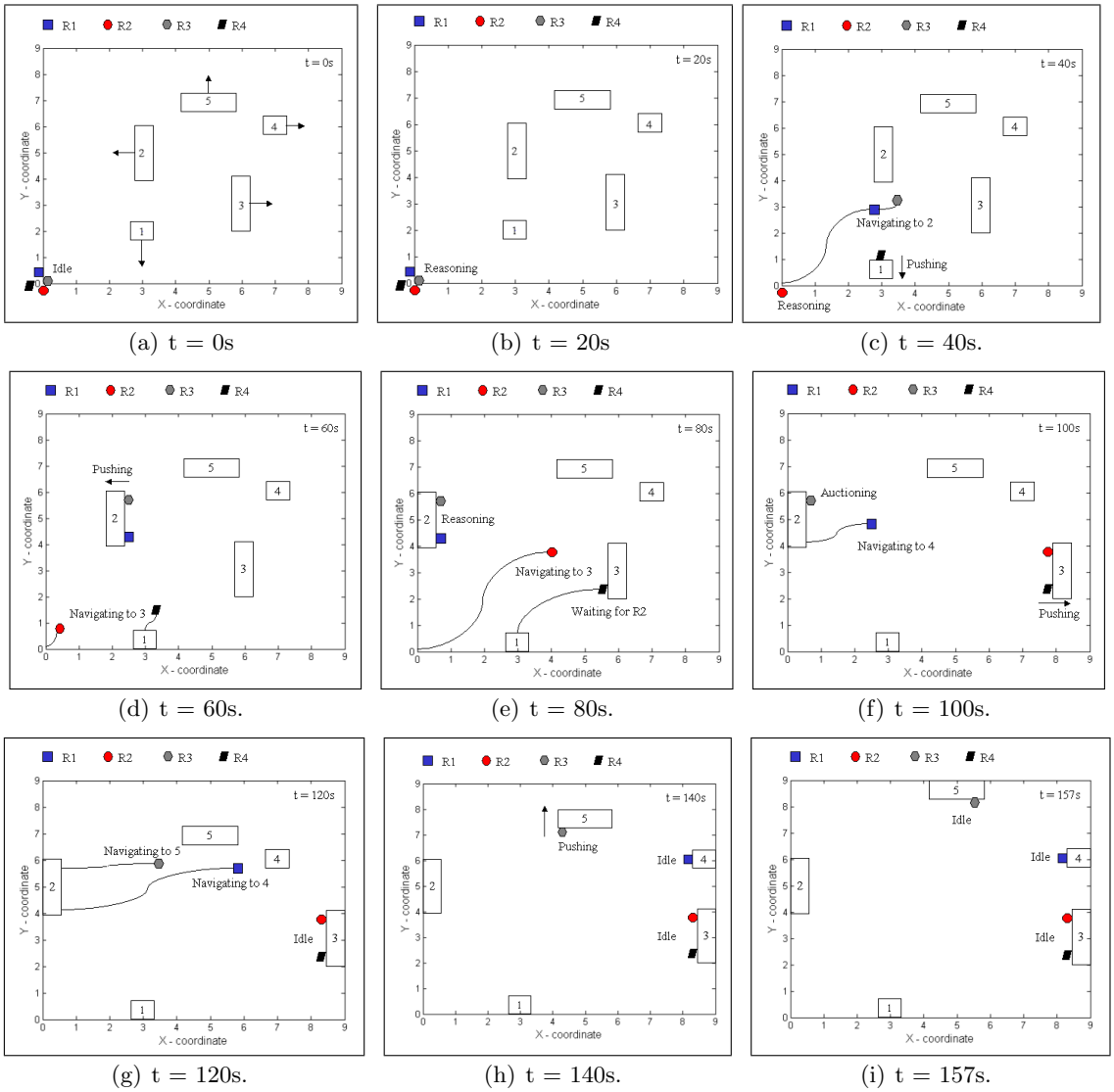
accomplish each task. Table 7.11 shows the previously developed schemas that are reused in the site clearing task.

7.6.3 Simulation setup

The site clearing task has been tested in simulation. In the simulation setup, four heterogeneous robots (as shown in Table 7.12) are required to clean a $10 \times 10m^2$ area with 5 boxes scattered inside, as shown in Figure 7.12 (*a*). The arrow on the box represents the desired pushing direction of the robot(s) in order to remove the box. The number on the box represents the order in which it is removed. Three of the boxes (2, 3, and 5) need to be pushed at both ends, and the other two boxes (1 and 4) only need to be pushed at one end. Initially, four robots start at (0, 0).

As presented in Chapter 5, the time that robots spend reasoning solutions depends on the various timeout settings illustrated in Table 5.2. In the following experiments, timeout 1 is 9s, timeout 2 is 3s, and timeout 3 is 2s. Additionally, Step 3 in Algorithm 2 is skipped to save time. Similar to the distributed negotiation protocol, there are also many timeout settings in the high level task allocator, as follows: (1) the time waiting for incoming bids for a specific task announcement (10s), (2) the time for assigning the task to the winning robot coalition (6s), and (3) the time waiting for confirmations from all coalition members before giving up and assigning the task to the next available coalition (4s). These values can all be fine-tuned to improve the auctioning time.

I ran over 10 logged trails of the site clearing experiment with the above environmental setup, with tasks introduced randomly by the human operator. The robot team is able to accomplish the site clearing task in an average of 151.2s with the standard deviation of 9.1s. The results presented here illustrate the ability of the robot team to form coalitions to solve multiple multi-robot tasks by layering ASyMTRe with auction-based task allocation. During the task execution, I kept a record for the major events of the auctioneer and each robot. Each event record consists of the time and a description of the event. I recorded the following events when:



(a) t = 0s (b) t = 20s (c) t = 40s.

(d) t = 60s. (e) t = 80s. (f) t = 100s.

(g) t = 120s. (h) t = 140s. (i) t = 157s.

Figure 7.12: A series of states of the system that illustrated the task execution process. This is one instance of many runs of the site clearing task. Robot speed is 0.2 *m/s*.

- A task is added by the human operator for auction;
- A task is announced to the robots;
- A task is accepted by a coalition;
- A task is completed by a coalition;
- A robot receives a task announcement;
- A robot forms coalition(s) to solve its current task by negotiating/reasoning with other robots;
- A robot starts navigating and pushing; and
- A coalition completes its current task.

According to the events, robots are always in one of the following states: reasoning, auctioning, navigating, pushing, and idle. A robot starts reasoning with ASyMTRe when it receives a task announcement. A robot is in the auctioning state when it is communicating with the auctioneer to bid for a task, or to accept the task award. A robot is idle when it is waiting for incoming tasks. Figure 7.12 shows one instance of a series of states of the robots and the site at 20 second time intervals. Using the same instance, Figure 7.13 shows the major events from the auctioneer's point of view. Figure 7.14 shows each robot's current state at every second. At time 2, all robots receive the task announcement of removing box 1 and start reasoning to form coalitions. At time 10, coalitions are formed and robots start to bid for the task and wait for the award. At time 13, the task is assigned to R_4, and the rest of the team starts reasoning on the next available task. Notice that at times 24 and 35, R_2 continues to reason on task T_3, but fails to generate any solution because of its limited sensing or computational capability. At time 53, R_2 finally forms a coalition with R_4 to accomplish T_3. In the above experiments, the reasoning and auctioning times are decided by the various timeouts in the distributed ASyMTRe-D negotiation protocol and in the auction algorithm. Fine tuning of the timeout parameters may also result in a shorter task completion time. However, the purpose of this experiment is to show that the ASyMTRe approach can be easily layered with another existing task allocation approach to handle multiple tasks. My future work includes improving the current auction algorithm to minimize the completion time.

In addition to the above illustrative examples, I have also tested the system with more complex setups. I have varied the number of robots from 3 to 6 and the number of boxes from 6 to 10. Results from 20 runs (reported in Table 7.13) have shown that the robots could successfully form a coalition to accomplish each task.

The above experiments illustrate the success of layering ASyMTRe for low-level coalition formation (for solving a single multi-robot task), with a higher level, auction-based task allocation approach (for solving a set of tasks). The resulting approach provides flexible mechanisms for a broad range of realistic multi-robot applications, with the ability of the robot team to generate both strongly cooperative and weakly cooperative solution strategies without predefined solutions, plans, or roles.

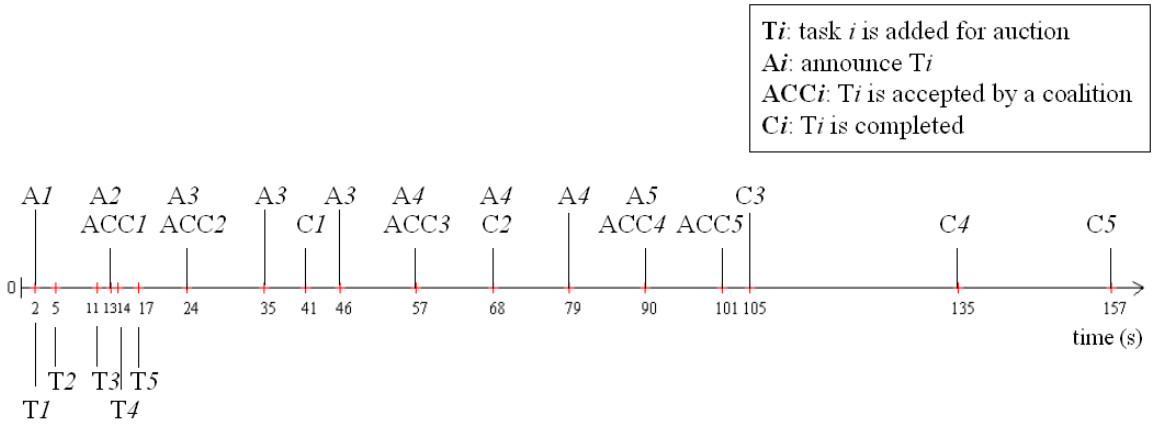


Figure 7.13: Using the same instance as in Figure 7.12, this figure shows the major events from the auctioneer's point of view.

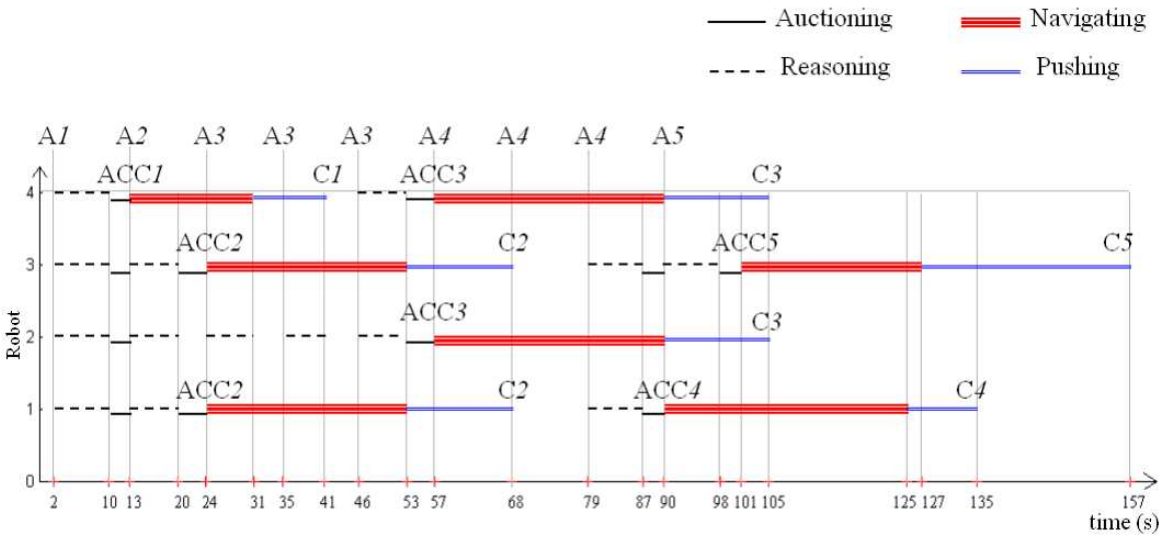


Figure 7.14: Using the same instance as in Figure 7.12, this figure shows the state of each robot during the task execution.

Table 7.13: Average completion time in the box pushing task

Team size	**Number of obstacles**	**Avg. Time**	**Std. Dev.**
3	6	244.2s	16.6s
3	10	341.2s	37.9s
6	6	143.2s	17.1s
6	10	202s	20.6s

Figure 7.15: The site setup for the site clearing task.

7.6.4 Physical robot experiments setup and results

The site clearing task has also been implemented on a physical robot team. The robot team includes two Pioneer robots, each with a laser and a camera. The site is a $4 \times 5m^2$ area with 3 boxes scattered inside. Figure 7.15 shows the environmental setup of the site clearing task. Three boxes are laid out in the hallway. Two of the boxes (1 and 2) are small boxes and the other box is a long box that needs to be pushed at both ends. The objective of the task is to move the boxes to several predefined collection points, which are represented by a red flag. There are three collection points in the environment: two of them are hung on the wall corresponding with the two small boxes, and the third one is two meters away from the long box. To successfully clean this site, I predefine the order in which the boxes are removed (as shown by the number on the box). Figure 7.16 illustrates the major events in the auctioning process. Figure 7.17 shows a series of snapshots taken during one run of the site clearing task. R_1 starts out at a closer point to the boxes than R_2. When T_1 is announced, both robots configure their solutions to remove box 1. Since R_1 takes a shorter time to accomplish this task, it wins the task at the end of the auction. R_2 then configures its solution to remove box 2 when T_2 is announced and wins the task at the end. After both tasks are accomplished, T_3 is introduced to the team and the team generates a solution to cooperatively remove box 3. Figure 7.18 shows that the three boxes have been moved to their nearest collection points at the end of the task.

7.7 Summary

This chapter has presented the experiments that are used to validate the ASyMTRe approach. As shown in Table 7.14, the design of every experiment fulfills certain purposes. The experimental results have demonstrated the contributions of the ASyMTRe work, which include (1) enabling the robots to form coalitions to solve multi-robot tasks or tasks that

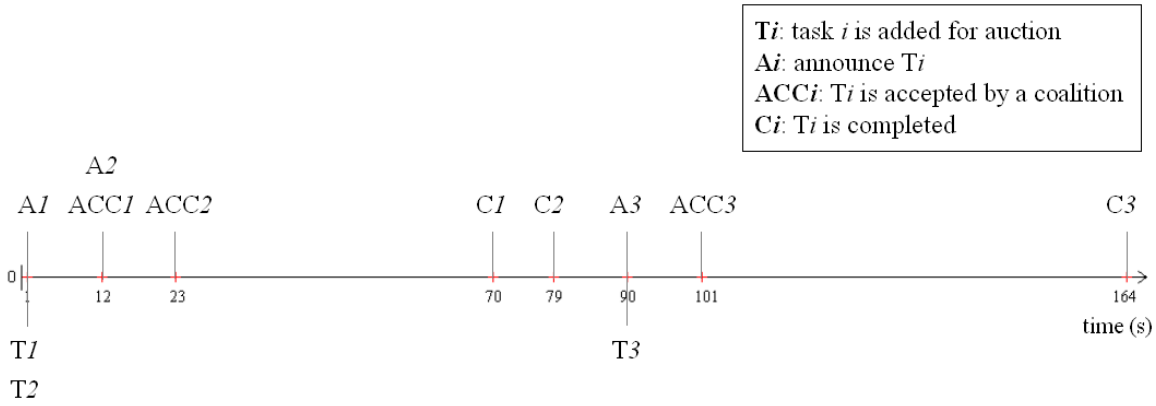


Figure 7.16: Major events from the auctioneer's point of view in the physical experiment.

Figure 7.17: A series of snapshots taken during one run of the site clearing task.

Figure 7.18: Results of the site clearing task.

Table 7.14: Summary of experiments.

Experiment	**Coalition formation**	**Robustness**	**Code Reusability**	**Performance**	**Task allocation**
Multi-robot navigation	X	X			
Cooperative box pushing	X	X	X		
Site clearing	X		X		X
Comparison experiments				X	

are difficult for certain resource-bounded robots to accomplish; (2) increasing the robustness of the application through online solution reconfiguration; (3) allowing robot code reuse for different applications; (4) good performance when using centralized ASyMTRe for smaller teams and distributed ASyMTRe-D for larger teams; and (5) the potential of layering ASyMTRe with high level task allocation approaches to accomplish multiple tasks or more complex tasks.

Chapter 8

Possible Future Extension: Peer-to-Peer Human-Robot Teaming

This chapter outlines a possible extension of the ASyMTRe approach to coalesce human-robot teams to accomplish tasks depending upon each team member's capabilities. According to my prior work on ASyMTRe for multi-robot teams, two different control architectures are implemented: centralized ASyMTRe and distributed ASyMTRe-D. In comparison of their performance, distributed ASyMTRe-D exhibits several advantages over centralized ASyMTRe, since it is robust, flexible and realistic to maintain for heterogeneous teams. Thus, the proposed approach could be built on the distributed ASyMTRe-D approach. This chapter describes the high level ideas of the approach, but its implementation remains as a possible future work. This possible future extension was reported and presented at the AAAI Spring Symposium entitled: "To Boldly Go Where No Human-Robot Team Has Gone Before" [Tang and Parker, 2006a].

The rest of the chapter is organized as follows. Sections 8.1 and 8.2 introduce the problem with a motivating example. Section 8.3 presents the challenging issues for human-robot teaming and reviews the existing approaches that attack these challenges. Section 8.4 describes my proposed approach to human-robot teaming.

8.1 Introduction

The traditional ways of human-robot interaction through teleoperation, or supervisor-subordinate interaction have been extensively explored in the supervisory control literature. However, much work remains to be done to explicitly address the issue of human-robot teaming when humans and robots interact as true team members instead of humans treating robots as tools only. The motivation of peer-to-peer interaction rather than supervisory interaction has been presented in [Marble et al., 2004]. On the one hand, teleoperation has the benefit of reduced danger to humans, better quality of data, etc. Such systems still have limitations such as lapses in communication, situation awareness and ability to handle

failures. On the other hand, purely autonomous robot systems face problems that are extremely difficult to handle without humans' assistance. Thus, there is a great potential for humans and robots to work together as a team and for each team member to contribute to the task objective based on their individual capabilities.

The research on human-robot interaction issues touches many different areas, such as the various interaction modalities, cognitive models, and evaluation methods. In [Fong et al., 2005], the authors present a "Peer-to-Peer Human-Robot Interaction" (P2P-HRI) project, where humans and robots work as partners across a range of configurations to accomplish various space related tasks. To achieve the goal, three major components are introduced: HRI/OS, which is based on a collaborative control model [Fong, 2001] that uses a dialogue system to facilitate the interaction; computational cognitive architectures that model humans to enhance the understanding between humans and robots; and a series of evaluation methods. This project is currently the most comprehensive work that deals with most of the issues in human-robot teaming. The objective of my proposed approach is to explore an alternative way of interaction based on the information that team members need to exchange to accomplish the task collaboratively.

With the premise that humans and robots interact as peers, my proposed work is focused on finding the optimal organization of robots to help humans in accomplishing a task that may require collective work of multiple team members. Since robots vary in their capabilities, some team members may not have the required capabilities to accomplish a task independently, it is important for the robots to work together in a tightly-coupled coalition and share information with each other as needed to accomplish a task. Humans and robots also have different capabilities, thus they can also form a coalition and share information with each other in order to accomplish a task. The ultimate goal is to develop mechanisms for building robot coalitions or human-robot coalitions autonomously as needed for the team to accomplish the task as a whole. Although my proposed approach for human-robot teaming focuses on peer-to-peer interaction, I notice that not every cooperation will be peer-to-peer, thus teleoperation will also be included when necessary.

In previous chapters, I have presented the ASyMTRe approach that enables multi-robot teams to accomplish multi-robot tasks through automated coalition formation. The ASyMTRe approach goes beyond the mapping of capabilities to tasks by abstracting the problem at the *schema* level, rather than the task/sensor level, and by allowing the team to autonomously configure solution strategies with schemas and sensors distributed across team members to accomplish a task. Although these robot systems work well in many situations, pure autonomous robotic systems still have many limitations. For example, performance degrades with unexpected robot failure, or unforeseen environmental changes. In these situations, human intelligence can be incorporated with such a system to increase the team's overall reliability. The ASyMTRe approach provides a solid foundation for developing a system that is suitable for human-robot interaction. Similar to ASyMTRe, the extended system could inherit the characteristics of ASyMTRe and enable the human-robot team to: form robot or human-robot coalitions taking into account both human and robot capabilities and the task requirement; enable information sharing among coalition members to complement each other's capabilities; and configure task solution dynamically instead of using predefined solutions. Since the ASyMTRe approach has been successfully implemented and demonstrated on many multi-robot cooperation tasks, it is interesting to

see how the cooperation mechanism on multi-robot teams can be applied to human-robot teams.

8.2 The Problem

The problem addressed here is: given a task that requires multiple team members to work on, and given a human-robot team, how can the robots cooperate with each other to help the humans in accomplishing the task. Since team members may have different capabilities, it is essential to determine how each team member could contribute to the achievement of the overall task objective.

Usually, humans would design a special plan for a particular team to accomplish a certain task; however, the task solutions are highly dependent on the available team composition. Let us first look at this motivating example on human-robot teams. Suppose a human wants some robots to help him/her move furniture from location A to location B. The types of solutions to accomplish this task are highly dependent upon the team composition. If the robots can localize in the environment and perceive the relative position of the furniture, a straightforward approach would be to select the number of robots as required to move the furniture. However, if some robots do not have the required capabilities, an alternative approach would be for the more capable team members (robots or humans) to guide the less capable robots to fulfill the goal by providing them with the necessary information, for example, the relative position of the furniture, or perhaps, direct commands, such as "turn left", "push harder", etc. In other team compositions, alternative approaches could be imagined. The important point here is that the resulting team behaviors for accomplishing the task could be dramatically different depending upon the combination of capabilities of humans and robots. Thus, instead of redesigning the plan manually, I propose an approach that automatically synthesizes task solutions based on the current team composition. With the success of applying ASyMTRe to multi-robot systems, one possible way to solve the problem is to extend the ASyMTRe approach to ASyMTRe-HRI (ASyMTRe for Human-Robot Interaction) for human-robot applications and address the following challenging issues.

8.3 Challenging Issues in Human-Robot Teaming

In this section, I present the challenges when applying ASyMTRe to human-robot teams and review the related work on how each challenge can be attacked.

8.3.1 How to represent human/robot capabilities?

Since different team capabilities may result in different solution strategies, we need to define team member capabilities for both humans and robots. In ASyMTRe, a robot's capabilities are defined by its sensor and effector capabilities and the corresponding computational capabilities (schemas). However, as has been pointed out by many researchers [Fong et al., 2005, Sofge et al., 2004], I also believe that for humans and robots to interact or collaborate naturally as peers, it is necessary to use a cognitive model onboard a robot to augment this robot's reasoning capabilities, and thus enhance the awareness and facilitate the interaction

between both parties. There are two main reasons for using a cognitive model [Sofge et al., 2004]: (1) the more a robot behaves like a human being, the easier a human can predict and understand its behavior; (2) the same representation shared by both humans and robots facilitates the communication between them.

There are several cognitive architectures that are widely used to model human behaviors and cognitive processes, such as ACT-R [Anderson and Lebiere, 1998] and the cognitive architecture construct used in [Howard, 2005]. ACT-R is one of the most famous cognitive architectures that provides a theory about how human cognition works. It has two types of modules: perceptual-motor modules, which interact with the real world; and memory modules that contains facts and rules of how humans do things. Although ACT-R has been adequately validated by many empirical and psychological studies, it still presents difficulties in large scale applications. The cognitive architecture construct in [Howard, 2005] breaks human information processing into three macro-level mechanisms: perception, cognition, and motor activities. In addition to the general cognitive models, some researchers have also studied reasoning models for collaborative tasks in space, such as Polyscheme for perspective taking [Sofge et al., 2004, Trafton et al., 2005]. Perspective taking is a mechanism for better understanding and reasoning from another agent's perspective. For example, when a human says "the box is to my left", the robot should be able to reason from the human's perspective.

After exploring the applicability of the different cognitive architectures to the ASyMTRe approach, a possible could be to model both human and robot's capabilities with three branches: cognition, perception, and motor capabilities. Note that the last two branches of the proposed model are consistent with the perceptual and motor schemas that are used to represent robot capabilities in the ASyMTRe approach. Adding an extra cognition representation to the robots enables them to better understand and reason about human behaviors. The similar representation of both human and robot capabilities also facilitates the interaction between both parties.

8.3.2 How do humans and robots work as peers?

Humans and robots have different specialties, for example, humans are good at high level control and planning, while robots are precise at sensing and calculation. Thus, human intelligence could be combined with robot intelligence to effectively accomplish a task. Humans and robots interact at different levels. In the taxonomy created by Yanco and Drury [Yanco and Drury, 2004], a human may interact with a robot in the following different roles: supervisor, operator, teammate, mechanic/programmer, and bystander, among which The first three roles are particularly interesting in our case. A supervisor provides the high level mission description to a robot, but does not directly control the behavior of a robot. An operator has more interaction with a robot by teleoperating or intervening the execution of a task. A teammate works collaboratively with a robot to accomplish a task, which is also called peer-to-peer interaction. In ASyMTRe-HRI, the focus is for humans and robots to work naturally as peers; however, teleoperation is also included when necessary.

In ASyMTRe, robots form coalitions to accomplish multi-robot tasks based on the required flow of information of tasks. Similarly, in ASyMTRe-HRI, humans can assign tasks to robots, and robots can negotiate with each other on how to form coalitions to accomplish the task based on their individual capabilities. If no coalitions can be formed to

accomplish the task, humans can assist them by joining in the coalitions with the robots. In the same way, robots can also help humans by providing them the necessary information. For example, a robot capable of global positioning can assist the human operator to arrive at a designated location, and then let the human to do the rest of the work. Humans and robots are working as peers in the sense that humans do not always command robots to perform tasks, giving specific instructions on how to accomplish the task. Instead, both humans and robots may vary their interaction roles or levels of interaction based on the current situation. The benefits of incorporating different levels of interaction are that: (1) robots or humans can quickly respond to each other's requests and provide help; (2) humans have reduced workloads since they do not need to monitor task execution all the time; and (3) different capabilities of team members are efficiently utilized to accomplish a task.

One major challenge when incorporating different levels of interaction is determining how these interaction levels can be activated and mixed in different situations. Research in this direction is sometimes called sliding autonomy [Desai and Yanco, 2005], or mixed initiative teaming [Marble et al., 2004]. In some cases, the system is composed of discrete autonomy levels, varying from pure teleoperation to full autonomy of robots. In other cases, the system is composed of sliding autonomy levels which can be decided by varying a set of parameters. Much work still remains to be done to determine the right level of autonomy for the system based on the team capabilities and the task objective.

8.3.3 What and how do humans and robots communicate?

Humans and robots exchange data depending on different levels of interaction between them. Data communication is bi-directional. At a high level, humans assign tasks to robots with task specification such as defining the goal position for the robots to navigate to. Robots inform humans of their current task-execution status, for example, informing humans that they have accomplished the current task and are waiting for new assignments. At a low level, humans may teleoperate robots through direct commands, or exchange sensing or computational information with robots as needed. Additionally, robots inform humans of their possible failures so that humans can assist them in accessing or recovering from the failures. Note that in ASyMTRe, coalitions are formed according to the flow of information required to accomplish a task. The design ASyMTRe-HRI could also based on the flow of information, such that humans and robots can efficiently communicate information with each other and form coalitions when necessary. For example, a human can help a robot find its way home by teleoperating it with direct commands or giving the relative goal position to the robot.

Humans and robots may have different ways of representing and interpreting the information. For example, humans tend to give abstract information, such as "the goal is to your right" instead of "the goal is at (x, y)", which is more difficult for robots to understand. As described in [Sofge et al., 2004], the use of the spatial language provides a rich and helpful interface for intuitive communication between humans and robots. Precise information (such as coordinates of a goal position) can be replaced by a sequential combination of directives, for example, "go straight for 5 seconds", "turn left for 60 degrees", and "search for a red sign". In addition to the difference in representing and interpreting information, there is also the problem of the frame of reference. For example, when an operator asks the

robot to "locate the object on the left", the robot should be able to determine whether the object is to the left from the operator's reference point or to the robot's reference point. Some experiments [Moratz et al., 2001] have proved that it is more effective for a human operator to use a robot's viewing perspective to decrease the cognitive load of the robot.

In addition to the different interaction levels, the information exchanged between humans and robots also depends on the various interfaces with which they communicate. Current robotics technologies allow the interaction of humans and robots to be implemented in many ways: dialogues, auditory perception, visual perception, behaviors, etc. Dialogue is a promising and realistic approach in many HRI applications. It is a process of sharing information and exchanging ideas between two parties. Examples in this area include [Spiliotopoulos, 2001] and [Fong, 2001]. Sound can also be used to improve HRI in many ways. For example, people tracking could use a combination of auditory and visual experience to enhance the effect [Nakadai and Hidai, 2001]. Humans convey intent through their gaze directions, postures, gestures, and facial displays [Breazeal and Scassellati, 1999]. Interaction through visual perception is a principle way of communication. People also express their inner emotions through behaviors, such as body languages, gestures, and facial expressions. Kismet [Breazeal, 1998] is one of the famous applications that uses facial expressions. For the purpose of this future work, the aim is to demonstrate the ability of ASyMTRe-HRI to coalesce human-robot teams to work collaboratively on a task. Particular model or multimodal communication interface should be employed according to different team capabilities, the particular application, and many other environmental factors. For the cooperative box pushing application, an interface that includes speech recognition/synthesis and maybe gesture recognition is preferred.

8.3.4 How to evaluate team performance?

The goal for human-robot teaming is to create teams of humans and robots that efficiently and effectively utilize the available capabilities of each team member to achieve a task. The purposes of performance evaluation are: (1) to measure the fitness of a particular team composition performing a certain task; and (2) to improve the metrics on which the performance evaluation depends.

Before evaluation, we should determine what characteristics of the system need to be evaluated – the metrics. There are many different types and ways of defining and measuring metrics. In [Fong et al., 2005], they define three categories of top level metrics: *effectiveness*, which measures the task success; *teamwork efficiency*; and *workload*. Additionally, many lower level metrics are defined under each category. Various methods are used to collect and analyze the data collected based on the above metrics: video/audio data, log files, questionnaire, etc. In [Howard, 2005], performance metrics include workload values of human operators, and performance scores for both humans and robots. Instead of using humans subjects (questionnaires) to evaluate the workload, they divide tasks into primitive subtasks and assign workload values by pairwise comparing the effort required to implement each subtask. With multiple measurements available, how these various aspects can be combined to generate an overall evaluation score remains a difficult problem. Many researchers [Rodriguez and Weisbin, 2003,Howard, 2005] have suggested the decomposition

of a task into independent subtasks and the overall score is the summation of the scores for each subtask.

In ASyMTRe, sensing costs and success rates are assigned to schemas and a composite utility is calculated based on the concept of an optimization function for task allocation [Gerkey and Mataric, 2004]. In ASyMTRe-HRI, more complex metrics and measurements need to be considered in the evaluation function.

8.3.5 How to adapt to changes?

Humans and robots work in a dynamic environment, thus they should have strategies for dealing with dynamics. Team members may play different roles throughout a task to accommodate to the changes. For example, a robot can track a supervisor and follow him/her to a goal position, but during the execution, if its sensor for tracking malfunctions, then the supervisor needs to switch his/her role to operator and guide this robot to the goal. This robustness can be achieved through reconfiguring schemas on robots and informing humans of the robots' new need. Learning strategies could be incorporated with the ASyMTRe-HRI approach for human-robot teaming.

8.4 The Approach

In previous chapters, I have presented the ASyMTRe approach that enables heterogeneous robots to form coalitions as needed to accomplish multi-robot tasks. ASyMTRe can be used to autonomously configure solutions for multi-robot teams by connecting schemas within or across robots based on the flow of information required by the task. Coalitions are formed on the fly based on the current team capabilities and the task requirements. Members within the same coalition can share sensory or computational information with each other for the achievement of the task. Additionally, I have also developed a task allocator on top of ASyMTRe to allocate multiple tasks to the robots, with ASyMTRe serving as the lower level solution generator for a single multi-robot task. The same result would be expected for human-robot teams, where each team member contributes to the task based on their individual capabilities.

My proposed approach to human-robot teaming (ASyMTRe-HRI) is built upon prior work on ASyMTRe and task allocation, with the addition of human-robot interaction. As shown in Figure 8.1, the human-robot teaming approach includes the following major steps:

- Humans introduce tasks to the robot team. These tasks are allocated to the team through the task allocator.
- Each team member calls ASyMTRe-HRI to reason about potential solutions by forming coalitions to accomplish a task when necessary.
- Solutions are submitted to the task allocator, the best solution is selected, and the task is assigned to the relevant coalition members. Ideally, to minimize the workload of humans, pure robot coalitions are considered first. When there are no such coalitions available, human-robot coalitions are then considered.

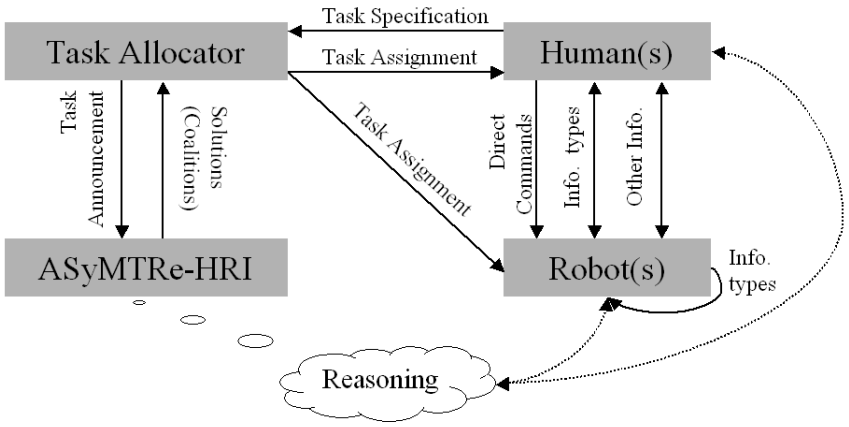


Figure 8.1: The human-robot teaming approach.

- During task execution, team members within the same coalition share information with each other as needed to accomplish the task. Sometimes, humans can provide direct commands to robots for teleoperating.
- Humans and robots interact through dialogue or gesture recognition. The interaction process is an independent process that can be activated at any time.

8.4.1 Agent capabilities

In ASyMTRe-HRI, humans and robots model their capabilities in similar representations to facilitate the interaction and situation awareness among agents. As discussed in Section 8.3, agent capabilities could be modeled as: cognition, perception, and motor capabilities.

First of all, recall in ASyMTRe, a robot's capabilities are represented by its environmental sensors and corresponding schemas, including perceptual schemas, motor schemas, and communication schemas, which will remain the same in ASyMTRe-HRI. Similarly, human capabilities could be represented by primitive behaviors $B = \{B_1, B_2, ...\}$. These behaviors are furthermore categorized as perception or motor-related behaviors. For example, "locate target" is a perceptual behavior, and "traverse" is a motor behavior. For the purpose of calculating solution quality, each human behavior is associated with the human workload and success probability. Workload is used to characterize human performance in terms of the total demand on a person implementing a task [Howard, 2005]. Success probability is the success rate for the person to perform a certain behavior. There are two main methods to assess the workload: pairwise comparison of primitive behaviors and human questionnaires. Pairwise comparison seems to be a good approach for the basic assessment when no human subject are present. Later on, questionnaires could be used to improve the basic assessment. With this addition, a composite score can be calculated for human-robot coalitions in a similar way as the robot coalitions (see Chapter 3 for details).

Second of all, a cognition model could be added to each agent to strengthen the capabilities of the humans and robots to understand and reason from another agent's perspective. Similar to ACT-R, the cognition model may contain two kinds of data: existing facts (such

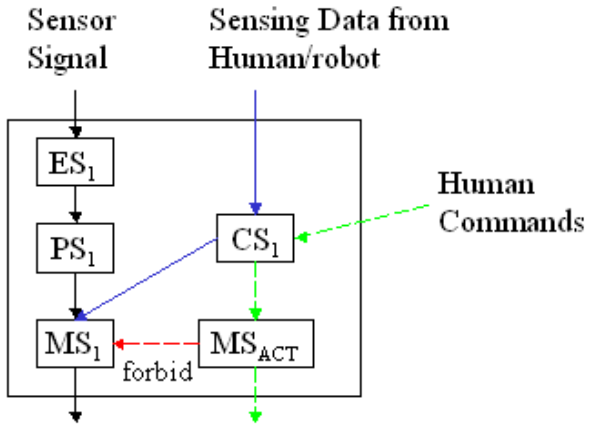


Figure 8.2: An example of how the human commands and sensing data are processed in the behavior control architecture. In this case, $T = \{MS_1, MS_{ACT}\}$. The receiving of a human command will trigger MS_{ACT} and forbid MS_1. The green dashed-line arrows represent the processing of human commands. The blue arrows represent an alternative way of accomplish the task through another team member's help. The black arrows represent the original way of accomplishing the task if the robot has ES_1.

as $1 + 2 = 3$), and rules about how to accomplish a certain thing (such as how to perform addition). The cognition model also takes into account spatial references, in our case, the speaker will make sure that the information uses the listener's frame of reference before sharing it with that listener.

8.4.2 Information sharing

From the information point of view, the types of information that humans could provide include direct commands (e.g., turn left, move straight) and possible sensory information (e.g., the position of an obstacle). Similar to robot schemas, humans behaviors are also associated with information types. I use I^{B_i} and O^{B_i} to represent the sets of input and output information types for behavior B_i. With the association of information types to behaviors, all agents can use the existing distributed ASyMTRe-D negotiation protocol to form coalitions based on the flow of information required to accomplish a task.

For a robot to deal with the information provided by humans, if the information is related to sensing data, it is then processed the same way as the sensing data from actual sensors or perceptual schemas. If the information is related to human commands, the commands will be directly passed to a special motor schema called MS_{ACT}, which takes actions according to the commands without any processing. A MS_{ACT} schema is equivalent to the motor schema that acts based upon sensing data, but it has a higher priority than the other motor schemas because we want to give humans the highest priority in executing tasks. Thus, as long as there are human commands arriving, the MS_{ACT} schema will be activated and it will forbid the other motor schemas. Figure 8.2 shows one example of how robots process the human commands.

8.4.3 Human-robot interaction

In ASyMTRe-HRI, humans will have three different roles: supervisor, operator, and teammate. As a supervisor, a human assigns tasks to robots and monitors their execution. As an operator, a human provides direct commands to teleoperate the robots to accomplish a task. As a teammate, a human works together with the robots, helping each other out through information sharing, etc. A human may switch between different roles according to his/her capabilities and the current situation. A robot's current activity can be interrupted by a human at any time. For example, when a higher-priority task comes, a human operator may want the robot to stop its current task and work on the more urgent task. A robot may initiate a conversation with a human at any time to request help, for example, for task/command clarification. The human who is currently not busy will then review the request and take the corresponding role according to his/her capabilities. For example, a robot, which loses its localization capability because of sensor failures, may request help from a supervisor. The supervisor may then switch to the operator role to teleoperate this robot to the goal position.

Dialogue and gesture recognition could be used as the communication methods between humans and robots. Standard off-the-shelf speech recognition software such as Sphinx and speech synthesis software such as Festival from Carnegie Mellon University can be used. Gesture recognition has also been widely used for human-robot interaction. A minimal gesture recognition system could be developed to complement the dialogue system. In addition, a graphical user interface could be built to facilitate the teleoperation and the information sharing between humans and robots.

8.4.4 Metrics and evaluation

To evaluate the ASyMTRe-HRI system, we could use the top level metrics used in [Fong et al., 2005]: task effectiveness, teamwork efficiency, and workload. According to these top level metrics, many low level metrics are defined and data will be collected for evaluation:

- *Task effectiveness*: The study of how each task is accomplished, including the measurement of task success, time measurement, etc. Time measurement may include the time that the task is waiting to be assigned and the task completion time.

- *Teamwork efficiency*: The study of how coalitions are efficiently formed for task completion and the efficiency in human-robot interaction. To measure the efficiency, data needs to be collected on: (1) solution quality (the composite value of agent cost/workload and task success probability), and compare it with the other possible solutions; (2) the number of problems (any failure, or unexpected situations) occurred during task execution (3) the time taken and the activities of an agent when problems occur; (4) the different roles of humans during task execution; (5) the number of problems resolved or the new problems occurred during interaction; (6) the length of interaction that resolves a problem; and (7) the human's opinion of teamwork efficiency through questionnaire.

- *Workload*: The study of the demand on humans for performing a certain task. The basic workload values are assigned based on pairwise comparing the effort taken to

accomplish each primitive behaviors. More complex behaviors can be a hierarchy of the primitive behaviors, and their workload is the composite value of each primitive behavior. To improve the workload values, human subjects will be used in case studies and they are provided a questionnaire to assess their workload.

As a possible future extension to ASyMTRe, the ASyMTRe-HRI approach could be implemented and tested on many interesting tasks, for example, the site clearing task as mentioned in Chapter 6. In this task, a human would work with two Pioneer robots to clear a specified site with boxes as obstacles. This task scenario provides rich opportunities for human-robot teaming. Different types of interaction modes could be activated. For example, the human and robots may need to work side by side to push a long box to the nearest collection point.

8.5 Summary

This chapter describes the ASyMTRe-HRI approach that is used to enable robots to help humans in accomplishing tasks, with the additional capabilities for the team to form human-robot coalitions to accomplish tasks when required. The ASyMTRe-HRI approach is an extension of the original work on ASyMTRe and its implementation remains a possible future work.

Chapter 9

Summary and Conclusions

9.1 Summary of Contributions

This dissertation makes several contributions to heterogeneous multi-robot cooperation. The most important is the design and development of ASyMTRe – a novel approach that autonomously configures solutions for heterogeneous robots to accomplish tasks. It is particularly suited for generating solutions for multi-robot tasks that require strong cooperation of multiple robots. This approach has been shown to have the following characteristics:

- Generates flexible task solutions based on sensor sharing across robot coalition members and task requirements.
- Forms robot coalitions by connecting schemas within or across robots.
- Provides both fully centralized and fully distributed implementation of the reasoning process.
- Enables software code reuse from one application to another.
- Provides solution reconfigurability for the team to recover from failures.
- Allows multiple tasks to be allocated to suitable robot coalitions.
- Applies to heterogeneous multi-robot teams and a variety of multi-robot tasks.
- Enables resource-bounded robots to accomplish tasks with information sharing from coalition members.
- Provides a promising foundation for human-robot teaming.

The ASyMTRe approach has been implemented on both simulated and physical robot teams performing a variety of tasks. Particularly, there are two distinct implementation versions: centralized ASyMTRe and distributed ASyMTRe. The centralized ASyMTRe configuration algorithm generates a good solution within a short amount of time, but can increase the solution quality when more time is given. However, like many other centralized systems, centralized ASyMTRe faces the problem of single point failure. To increase the

robustness of the reasoning process, the distributed ASyMTRe negotiation protocol was implemented, but it trades off the solution quality to achieve robustness. Additionally, I have also implemented a standard auction-based task allocator that allocates multiple tasks to coalitions with instantaneous assignment. Thus, the ASyMTRe approach can be used as the low-level solution generator that generates potential coalitions to accomplish a certain task, with the high-level task allocator to decide which task been assigned to which coalition.

A variety of simulated and physical robot applications have validated my approach and demonstrated its unique characteristics. The types of tasks that the ASyMTRe approach has been applied to include:

- Multi-robot navigation
- Cooperative multi-robot box pushing
- Site clearing

The ASyMTRe approach is the first approach that enables robots to form coalitions as needed to accomplish multi-robot tasks without predefined plans or solutions. I am not aware of any other robot cooperative control architectures that view robot capabilities in terms of sensors and schemas, and configure solutions at the lower level of schemas, according to the flow of information required by a task – the basic foundation of ASyMTRe.

9.2 Future Work

Several promising research problems follow directly from my current research on ASyMTRe:

- *Solution quality.* In centralized ASyMTRe, I used an anytime algorithm that generates the best solution found so far. However, at the beginning, I use two special orderings to guarantee that a good solution can be generated within a short amount of time. Currently, there are only some experimental results showing that the initial solution is a good starting point, but there is no direct analytical evidence to prove the quality of the first solution. Due to the similarity between my configuration algorithm and the coalition formation algorithm presented in [Shehory, 1998], I plan to analyze the bounds on the solution quality in future work. It has been proved in [Shehory, 1998] that similar algorithms are of low logarithmic ratio bounds to the optimal solution.

- *Adaptive utility function.* Solution quality in the ASyMTRe approach is explicitly measured by linearly combining sensing cost and success probability of a selected solution. Additionally, the cost and probability data are based on the designer's experience on sensor/schema performance or historical data. This is only the first step towards a systematic way of assessing performance of multi-robot systems. In a dynamic environment, these values may change during task execution, thus it is important for the utility function to reflect the changes. In additional to cost and success probability, there are other factors that need to be considered, such as computational complexity, physical complexity, accuracy of solution, frequency of information, interference between robots, etc. How to evaluate the quality based on this variety

of factors remains a difficult problem. My future work includes defining more accurate utility functions for evaluating the multi-robot system, modeling the dynamics in the system, and deriving adaptive utility measurements for systems by incorporating machine-learning techniques.

- *Dynamics of the multi-robot systems.* There are many dynamics in multi-robot applications during the solution generation process or task execution process. In this dissertation, I have presented that the system is able to handle sensor or robot failures, the random insertion of tasks, etc. There are also other situations, for example, robots joining the team dynamically, communication errors, communication delays, etc. These dynamics may either cause the system quality to degrade or to be enhanced. In an ideal situation, we would like the system to handle these dynamics effectively. For example, when a robot joins the team while other robots are performing a task, the team could continue working on the task while evaluating the new team capabilities and generating new solution strategies. If the new solution has a better quality, the team could replace it with their current solutions.

- *Peer-to-peer human-robot teaming.* Researchers have identified the advantages of peer-to-peer human-robot interactions over supervisory interactions or pure autonomous systems. The automated coalition formation ability of the ASyMTRe approach provides a promising foundation for human-robot cooperation, especially peer-to-peer interaction. As shown in Chapter 8, I have proposed to extend the current ASyMTRe approach and design the ASyMTRe-HRI approach for human-robot teaming. Future work includes the implementation of the ASyMTRe-HRI approach on human-robot teams to accomplish tasks that may involve a rich set of human-robot interaction.

- *Large-scale multi-robot applications.* The original ASyMTRe approach is suitable for solving a single task that requires strong cooperation between team members. To enhance the ability of the system to handle more complex tasks (which can be represented as a hierarchy of subtasks, or task tree) or multiple tasks, I also developed an auction-based task allocator on top of ASyMTRe, with the ASyMTRe approach serving as the low level solution generator, and the task allocator allocating the current task to the best fitted coalition. Currently, the task allocator applies instantaneous assignment of the tasks. Future work includes enabling task allocation with time-extended assignment, thus multiple independent tasks (subtasks) can be considered by the robots at once to generate a better plan. I also believe that the ASyMTRe approach for coalition formation can be merged with other, non-auction-based approaches to task allocation, such as the motivation-based approach of ALLIANCE [Parker, 1998b]. It would be interesting to investigate the combination of ASyMTRe and ALLIANCE, as an alternative approach for achieving the merging of coalitions for multi-robot tasks with traditional task allocation techniques.

Bibliography

Bibliography

[Anderson and Lebiere, 1998] Anderson, J. and Lebiere, C. (1998). Atomic components of thought. Erlbaum, Mahawah, NJ.

[Arai et al., 2002] Arai, T., Pagello, E., and Parker, L. E. (2002). Editorial: Advances in multi-robot systems. *IEEE Transactions on Robotics and Automation*, 18(5):655–661.

[Arkin, 1987] Arkin, R. C. (1987). Motor schema based navigation for a mobile robot: an approach to programming by behavior. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 264–271.

[Arkin, 1998] Arkin, R. C. (1998). *Behavior-Based Robotics*. MIT Press.

[Beavers and Hexmoor, 2001] Beavers, G. and Hexmoor, H. (2001). Teams of agents. In *Proceedings of the IEEE System, Man, and Cybernetics Conference*, pages 574–582.

[Beni and Wang, 1989] Beni, G. and Wang, J. (1989). Swarm intelligence in cellular robotics systems. In *Proceedings of NATO Advanced Workshop on Robots and Biological System*.

[Bertsekas, 1988] Bertsekas, D. P. (1988). The auction algorithm: a distributed relaxation method for the assignment problem. *Annals of Operations Research*, 14:105–123.

[Botelho and Alami, 1999] Botelho, S. and Alami, R. (1999). M+: A schema for multi-robot cooperation through negotiated task allocation and achievement. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1234–1239.

[Breazeal, 1998] Breazeal, C. (1998). A motivational system for regulating human-robot interaction. In *Proceedings of the National Conference on Artificial Intelligence*, pages 54–61.

[Breazeal and Scassellati, 1999] Breazeal, C. and Scassellati, B. (1999). How to build robots that make friends and influence people. In *Proceedings of the International Conference on Intelligent Robots and Systems*.

[Brown and Jennings, 1995] Brown, R. G. and Jennings, J. S. (1995). A pusher/steerer model for strongly cooperative mobile robot manipulation. In *Proceedings of 1995 IEEE International Conference on Intelligent Robots and Systems (IROS '95)*, volume 3, pages 562–568.

[Cao et al., 1997] Cao, Y. U., Fukunaga, A. S., and Kahng, A. B. (1997). Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4:1–23.

[Chandra, 2004] Chandra, M. (2004). Software reconfigurability for heterogeneous robot cooperation. Master's thesis, University of Tennessee, Department of Computer Science.

[Cohen et al., 1991] Cohen, P. R., Levesque, H. J., and Nous, H. (1991). Teamwork. *Special Issue on Cognitive Science and Artificial Intelligence*, 25(4):487–512.

[Desai and Yanco, 2005] Desai, M. and Yanco, H. A. (2005). Blending human and robot inputs for sliding scale autonomy. In *Proceedings of the IEEE International Workshop on Robot and Human Interactive Communication*.

[Dias, 2004] Dias, M. B. (2004). *TraderBots: A New Paradigm for Robust and Efficient Multirobot Coordination in Dynamic Environments*. PhD thesis, Robotics Institute, Carnegie Mellon University.

[Dias and Stentz, 2000] Dias, M. B. and Stentz, A. (2000). A free market architecture for distributed control of multirobot system. In *Proceedings of the International Conference on Intelligent Autonomous System*.

[Donald, 1995] Donald, B. R. (1995). Information invariants in robotics. *Artificial Intelligence*, 72(1-2):217–304.

[Donald et al., 1993] Donald, B. R., Jennings, J., and Rus, D. (1993). Towards a theory of information invariants for cooperating autonomous mobile robots. In *Proceedings of the International Symposium of Robotics Research*, pages 293–298, Hidden Valley, PA.

[Donald et al., 1994] Donald, B. R., Jennings, J., and Rus, D. (1994). Analyzing teams of cooperating mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1896–1903.

[Donald et al., 1997] Donald, B. R., Jennings, J., and Rus, D. (1997). Information invariants for distributed manipulation. *International Journal of Robotics Research*, 16(5):673–702.

[Fikes and Nilsson, 1971] Fikes, R. E. and Nilsson, N. J. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208.

[Fong, 2001] Fong, T. (2001). *Collaborative control: a robot-centric model for vehicle teleoperation*. PhD thesis, Carnegie Mellon University.

[Fong et al., 2005] Fong, T., Nourbakhsh, I., Kunz, C., Fluckiger, L., and Schreiner, J. (2005). The peer-to-peer human-robot interaction project. In *Proceedings of AIAA Space*.

[Gerkey and Mataric, 2002] Gerkey, B. and Mataric, M. J. (2002). Sold!: Auction methods for multi-robot coordination. *IEEE Transactions on Robotics and Automation*, 16(5):758–768.

[Gerkey and Mataric, 2004] Gerkey, B. and Mataric, M. J. (2004). A formal analysis and taxonomy of task allocation in multi-robot systems. *International Journal of Robotics Research*, 23(9):939–954.

[Gerkey et al., 2001] Gerkey, B., Vaughan, R., Stoey, K., Howard, A., Sukhatme, G., and Mataric, M. (2001). Most valuable player: A robot device server for distributed control. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1226 – 1231.

[Grosz, 1996] Grosz, B. (1996). Collaborating systems. *AI magazine*, 17(2).

[Hexmoor and Beavers, 2001] Hexmoor, H. and Beavers, G. (2001). Towards teams of agents. In *Proceedings of the International Conference in Artificial Intelligence*.

[Horling and Lesser, 2004] Horling, B. and Lesser, V. (2004). A survey of multi-agent organizational paradigms. Technical Report 04-45, University of Massachusetts.

[Howard et al., 2002] Howard, A., Mataric, M. J., and Sukhatme, G. S. (2002). Mobile sensor network deployment using potential fields: A distributed scalable solution to the area coverage problem. In *Distributed Autonomous Robotic Systems 5: Proceedings of the Sixth International Symposium on Distributed Autonomous Robotic Systems*, pages 299–308. Springer-Verlag.

[Howard, 2005] Howard, A. M. (2005). A methodology to assess performance of human-robotic systems in achievement of collective tasks. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 377–382.

[Jennings and Kirkwood-Watts, 1998] Jennings, J. and Kirkwood-Watts, C. (1998). Distributed mobile robotics by the method of dynamic teams. In *Proceedings of the Fourth International Symposium on Distributed Autonomous Robotic Systems*.

[Jennings, 1995] Jennings, N. (1995). Controlling cooperative problem solving in industrial multi-agent systems. *Artificial Intelligence*, 75(2):195–240.

[Jones et al., 2006] Jones, E., Browning, B., Dias, M. B., Argall, B., Veloso, M., and Stentz, A. (2006). Dynamically formed heterogeneous robot teams performing tightly-coordinated tasks. In *Proceedings of IEEE International Conference on Robotics and Automation*.

[Kalra et al., 2005] Kalra, N., Ferguson, D., and Stentz, A. (2005). Hoplites: a market-based framework for complex tight coordination in multi-robot teams. In *Proceedings of IEEE International Conference on Robotics and Automation*.

[Klusch and Gerber, 2002] Klusch, M. and Gerber, A. (2002). Dynamic coalition formation among rational agents. *IEEE Intelligent Systems*, 17(3):42–47.

[Kube and Zhang, 1993] Kube, C. R. and Zhang, H. (1993). Collective robotic intelligence. *Adaptive Behavior*, 2(2):189–219.

[Kube and Zhang, 1996] Kube, C. R. and Zhang, H. (1996). The use of perceptual cues in multi-robot box-pushing. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2085–2090.

[Lerman and Shehory, 2000] Lerman, K. and Shehory, O. (2000). Coalition formation for large scale electronic markets. In *Proceedings of the International Conference on Multi-Agent Systems*.

[Levesque et al., 1990] Levesque, H. J., Cohen, P. R., and Nunes, J. H. T. (1990). On acting together. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 94–99.

[Lin and Zheng, 2005] Lin, L. and Zheng, Z. (2005). Combinatorial bids based multi-robot task allocation method. In *Proceedings of IEEE International Conference on Robotics and Automation*.

[Low et al., 2004] Low, K. H., Leow, W. K., and Jr., M. H. A. (2004). Task allocation via self-organization swarm coalitions in distributed mobile sensor network. In *Proceedings of the 19th National Conference on Artificial Intelligence*, pages 28–33.

[Luce and Raiffa, 1957] Luce, R. D. and Raiffa, H. (1957). *Games and Decisions*. John Wiley and Songs, Inc.

[Lyons and Arbib, 1989] Lyons, D. M. and Arbib, M. A. (1989). A formal model of computation for sensory-based robotics. *IEEE Transactions on Robotics and Automation*, 5(3):280–293.

[Marble et al., 2004] Marble, J., Bruemmer, D., Few, D., and Dudenhoeffer, D. (2004). Evaluation of supervisory vs. peer-peer interaction with human-robot teams. In *Proceedings of the Hawaii International Conference on System Sciences*.

[Mataric, 1992] Mataric, M. J. (1992). Designing emergent behaviors: From local interactions to collective intelligence. In Meyer, J., Roitblat, H., and Wilson, S., editors, *Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, pages 432–441, Honolulu, Hawaii. MIT Press.

[Mataric et al., 1995] Mataric, M. J., Nilsson, M., and Simsarian, K. T. (1995). Cooperative multi-robot box-pushing. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 556–561.

[Moratz et al., 2001] Moratz, R., Fischer, K., and Tenbrink, T. (2001). Cognitive modeling of spatial reference for human-robot interaction. *International Journal on Artificial Intelligence Tools*, 10(4):589–611.

[Nakadai and Hidai, 2001] Nakadai, K. and Hidai, K. (2001). Real-time auditory and visual multiple-object tracking for robots. In *Proceeding of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI)*.

[Parker, 1993] Parker, L. E. (1993). Designing control laws for cooperative agent teams. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 582–587.

[Parker, 1994a] Parker, L. E. (1994a). ALLIANCE: An architecture for fault tolerant, cooperative control of heterogeneous mobile robots. In *Proc. of the 1994 IEEE/RSJ/GI Int'l Conf. on Intelligent Robots and Systems (IROS '94)*, pages 776–783, Munich, Germany.

[Parker, 1994b] Parker, L. E. (1994b). *Heterogeneous Multi-Robot Cooperation*. PhD thesis, MIT.

[Parker, 1998a] Parker, L. E. (1998a). ALLIANCE: An architecture for fault-tolerant multi-robot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2):220–240.

[Parker, 1998b] Parker, L. E. (1998b). Toward the automated synthesis of cooperative mobile robot teams. In *Proceedings of SPIE Sensor Fusion and Decentralized Control in Robotic Systems II*, volume 3525, pages 82–93.

[Parker, 2003] Parker, L. E. (2003). The effect of heterogeneity in teams of 100+ mobile robots. In Schultz, A., Parker, L. E., and Schneider, F., editors, *Multi-Robot Systems Volume II: From Swarms to Intelligent Automata*. Kluwer.

[Parker et al., 2000] Parker, L. E., Jung, D., Huntsberger, T., and Pirjanian, P. (2000). Opportunistic adaptation in space-based robot colonies: Application to site preparation. In *Proceedings of World Automation Congress*.

[Parker et al., 2004] Parker, L. E., Kannan, B., Tang, F., and Bailey, M. (2004). Tightly-coupled navigation assistance in heterogeneous multi-robot teams. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*.

[Parker and Tang, 2006] Parker, L. E. and Tang, F. (2006). Building multi-robot coalitions through automated task solution synthesis. *Proceedings of IEEE*. Special Issue on Multi-Robot Systems.

[Rodriguez and Weisbin, 2003] Rodriguez, G. and Weisbin, C. (2003). A new method to evaluate human-robot system performance. *Autonomous Robots*, 14(2).

[Sandholm et al., 1999] Sandholm, T., Larson, K., Andersson, M., Shehory, O., and Tohme, F. (1999). Coalition structure generation with worst case guarantees. *Artificial Intelligence*, 111(1-2):209–238.

[Sandholm and Lesser, 1997] Sandholm, T. and Lesser, V. (1997). Coalitions among computationally bounded agents. *Artificial Intelligence*, 94(1):99–137. Special Issue on Economic Principles of Multi-Agent Systems.

[Sandholm et al., 2005] Sandholm, T., Suri, S., Gilpin, A., and Levine, D. (2005). CABOB: a fast optimal algorithm for winner determnation in combinatorial auctions. *Management Science*, 51(3):374–390. Special issue on Electronic Markets.

[Sen et al., 1994] Sen, S., Sekaran, M., and Hale, J. (1994). Learning to coordinate without sharing information. In *Proceedings of AAAI*, pages 426–431.

[Shehory, 1998] Shehory, O. (1998). Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1-2):165–200.

[Shehory and Kraus, 1995] Shehory, O. and Kraus, S. (1995). Coalition formation among autonomous agents: strategies and complexity. In Castelfranchi, C. and Muller, J. P., editors, *Lecture Nots in Artificial Intelligence no. 957*. From Reaction to Cognition.

[Simmons and Apfelbaum, 1998] Simmons, R. and Apfelbaum, D. (1998). A task description language for robot control. In *Proceedings of Conference on Intelligent Robotics and Systems*.

[Simmons et al., 2000] Simmons, R., Singh, S., Hershberger, D., Ramos, J., and Simith, T. (2000). First results in the coordination of heterogeneous robots for large-scale assembly. In *Proceedings of the ISER Seventh International Symposium on Experimental Robotics*.

[Sims et al., 2003] Sims, M., Goldman, C., and Lesser, V. (2003). Self-organization through bottom-up coalition formation. In *Proceedings of the Second International Joint Conference on Autonomous Agents and MultiAgent Systems*, pages 867–874.

[Smith, 1980] Smith, R. G. (1980). The Contract Net Protocol: high-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12).

[Sofge et al., 2004] Sofge, D., Perzanowski, D., and et al. (2004). Cognitive tools for humanoid robots in space. In *Proceedings of IFAC Symposium on Automatic Control in Aerospace*.

[Spiliotopoulos, 2001] Spiliotopoulos, D. (2001). Human-robot interaction based on spoken natural language dialogue. In *Proceedings of the European Workshop on Service and Humanoid Robots*.

[Sycara et al., 1997] Sycara, K., Decker, K., and Williamson, M. (1997). Middle-agents for the internet. In *Proceedings of the International Journal of Computational Intelligence and Applications*.

[Tambe, 1997] Tambe, M. (1997). Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 110(2):215–239.

[Tang and Parker, 2005a] Tang, F. and Parker, L. E. (2005a). ASyMTRe: Automated synthesis of multi-robot task solutions through software reconfiguration. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

[Tang and Parker, 2005b] Tang, F. and Parker, L. E. (2005b). Coalescent multi-robot teaming through ASyMTRe: a formal analysis. In *Proceedings of the IEEE International Conference on Advanced Robotics*.

[Tang and Parker, 2005c] Tang, F. and Parker, L. E. (2005c). Distributed multi-robot coalitions through ASyMTRe-D. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*.

[Tang and Parker, 2006a] Tang, F. and Parker, L. E. (2006a). Automated human-robot teaming throught reconfigurable schemas. In *Proceedings of the AAAI Spring Symposium: To Boldly Go Where No Human-Robot Team Has Gone Before*.

[Tang and Parker, 2006b] Tang, F. and Parker, L. E. (2006b). Layering coalition formation with task allocation. In *Proceedings of the AAAI Workshop: Auction Mechanisms for Robot Coordination*.

[Tidhar et al., 1998] Tidhar, G., Heinze, C., and Selvestrel, M. (1998). Flying together: Modelling air mission teams. *Journal of Applied Intelligence*, 8(3).

[Trafton et al., 2005] Trafton, J., Cassimatis, N., and et al. (2005). Enabling effective human-robot interaction using perspective-taking in robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 49(4).

[Vig and Adams, 2005] Vig, L. and Adams, J. A. (2005). Issues in multi-robot coalition formation. In Parker, L. E., Schultz, A., and Schneider, F., editors, *Multi-Robot Systems Volume III: From Swarms to Intelligent Automata*. Kluwer.

[Werger and Mataric, 2000] Werger, B. B. and Mataric, M. J. (2000). Broadcast of local eligibility for multi-target observation. *Distributed Autonomous Robotic Systems 4*, pages 347–356.

[Yanco and Drury, 2004] Yanco, H. A. and Drury, J. (2004). Classifying human-robot interaction: An updated taxonomy. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pages 2841–2846.

[Zilberstein, 1996] Zilberstein, S. (1996). Using anytime algorithms in intelligent systems. *AI Magazine*, 17(3):73–83.

[Zlot and Stentz, 2006] Zlot, R. and Stentz, A. (2006). Market-based multirobot coordination for complex tasks. *International Journal of Robotics Research*, 25(1). Special Issue on the 4th International Conference on Field and Service Robotics.

Vita

Fang Tang joined the Department of Computer Science at the University of Tennessee-Knoxville (UTK) as Graduate Student in August 2001. She received the M.S. degree in computer science from UTK in 2003, and the B.S. degree in computer science from Sichuan University, Chengdu, China, in 2000.

She has been conducting research in the Distributed Intelligent Laboratory at UTK since 2003, and worked on the Software for Distributed Robotics project funded by DARPA. Her current research focuses on developing mechanisms to enable multiple robots to cooperate by autonomously forming coalitions. Her research interests also include multi-agent systems, human-robot cooperation, machine learning, and educational robotics. She is a student member of IEEE, ACM, and AAAI. After graduation, she is looking forward to joining the Computer Science Department at California Polytechnic State University at Pomona, as an assistant professor.