TraderBots: A New Paradigm for Robust and Efficient Multirobot Coordination in Dynamic Environments

M. Bernardine Dias

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

January 2004

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Thesis Committee:

Anthony Stentz, Chair
Stephen Smith
Jeff Schneider
Tuomas Sandholm

Table of Contents

Table of Figures

Table of Tables

Dedication

FOR my amazing family who selflessly sacrificed so much to give me opportunities beyond my wildest dreams. May this first Ph.D. dissertation in our family be a lasting testimony to God's love and the love of a wonderful family that enabled me to reach the pinnacle of my academic career. I hope this dissertation inspires those who follow to dream the seemingly impossible dreams and to persevere in realizing those dreams despite the many obstacles that make the path to success seem impossible at first.

Acknowledgements

THIS dissertation could not have been completed without the guidance, inspiration, and support of many people. First and foremost, I would like to thank my thesis advisor, Anthony Stentz, for giving me the latitude to explore a variety of research topics over the years, for supporting my numerous endeavors, and for giving me the guidance and inspiration necessary to take this thesis to its conclusion. I would also like to thank the rest of my thesis committee, Jeff Schneider, Stephen Smith and Tuomas Sandholm for their invaluable guidance.

I also wish to thank the members of the Cognitive Colonies project group (Anthony Stentz, Scott Thayer, Bruce Digney, Martial Hebert, Robert Zlot, Aaron Morris, Charles Smart, and Bart Nabbe), and the members of the FIRE project group (Reid Simmons, Anthony Stentz, Stephen Smith, Jeff Schneider, Trey Smith, Dani Goldberg, Drew Bagnell, Vincent Cicirello, David Apfelbaum, and Stuart Anderson), for their invaluable support, advice, collaborative efforts, and guidance. My gratitude also extends to the CTA multirobot project group (Anthony Stentz, Robert Zlot, Marc Zinck, Juan P. Gonzalez, Nidhi Kalra, Dave Ferguson, Andres S. Perez-Bergquist, and Bart Nabbe) for their valuable input and help in implementing and maintaining the robot team that enabled most of the robot results for this dissertation. I especially wish to thank Robert Zlot and Marc Zinck whose collaborations strengthened the robot results in this dissertation.

I am extremely fortunate to have benefited from the tireless efforts of many administrative assistants and staff at Carnegie Mellon University. I would especially like to thank Dottie Marsh, Suzanne Lyons, Heather Farah, Monica Hopes, and Michelle Gittleman who assisted me in numerous ways on countless occasions.

Many friends have helped my inspiration and perseverance – to all of them I am deeply grateful. I am especially indebted to Sarjoun Skaff, Sarah Moody, Ariadna Font Llitjos, Vandana Verma, Charindra Godakanda, Paul Tompkins, Chris and Jen Urmson, Dottie Marsh, Ashley Stroupe, Joelle Pineau, and Devin Balkcom for their countless contributions, encouragements, and sacrifices that enabled me to complete this dissertation successfully.

Lastly and most importantly, I am forever indebted to my family who has been my strength and my guiding light throughout my life. I owe everything I have, everything I know, and everything I am to their love, their dedication, their encouragement, and their unwavering support.

This research was principally sponsored by NASA, under contract "Heterogeneous Multi-Rover Coordination for Planetary Exploration" (contract number NCC2-1243). This work was also sponsored in part by the U.S. Army Research Laboratory, under contract "Robotics Collaborative Technology Alliance" (contract number DAAD19-01-2-0012) and in part by DARPA, under contract “Cognitive Colonies” (contract number N66001-99-1-8921, monitored by SPAWAR). The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies or endorsements, either expressed or implied, of the U.S. Government.

Abstract

THE problem of efficient multirobot coordination has risen to the forefront of robotics research in recent years. The wide range of application domains demanding multirobot solutions motivates interest in this problem. In general, multirobot coordination strategies assume either a centralized approach, where a single robot/agent plans for the group, or a distributed approach, where each robot is responsible for its own planning. Inherent to many centralized approaches are difficulties such as intractable solutions for large groups, sluggish response to changes in the local environment, heavy communication requirements, and brittle systems with single points of failure. The key advantage of centralized approaches is that they can produce globally optimal plans. While most distributed approaches can overcome the obstacles inherent to centralized approaches, they can only produce suboptimal plans because they cannot take full advantage of information available to all team members.

This work develops TraderBots, a market-based coordination approach that is inherently distributed, but also opportunistically forms centralized sub-groups to improve efficiency. Robots are self-interested agents with the primary goal of maximizing individual profits. The revenue/cost models and rules of engagement are designed so that maximizing individual profit has the benevolent effect of, on average, moving the team toward the globally optimal solution. This approach inherits the flexibility of markets in allowing cooperation and competition to emerge opportunistically. The outlined approach addresses the multirobot coordination problem for autonomous robotic teams executing tasks in dynamic environments where it is highly desirable to produce efficient solutions.

This dissertation details the first in-depth study of the applicability of market-based techniques to the multirobot coordination and provides a detailed study of the requirements for robust and efficient multirobot coordination in dynamic environments.

Contributions of this dissertation are the first extensive investigation of the application of market-based techniques to multirobot coordination, the most versatile coordination-approach for dynamic multirobot application domains, the first distributed multirobot coordination-approach that allows opportunistic optimization by “leaders”, the first in-depth investigation of the requirements for robust multirobot coordination in dynamic environments, the most extensively implemented market-based multirobot coordination approach, and the first systematic comparative analysis of multirobot coordination approaches implemented on a robot team.

CHAPTER 1

Introduction

COORDINATING multiple robots to cooperatively complete a task is a difficult problem that has attracted much attention from the robotics research community in recent years. This dissertation advances the state of the art in this research area, detailing a novel approach, "TraderBots", specifically geared towards coordinating multiple robots to succeed in reliably completing a cooperative task in a dynamic environment. The TraderBots approach capitalizes on the strengths of market economies that enable many agents to collectively execute complex tasks with access to limited/incomplete information under highly dynamic conditions. The added capability of reliable and efficient coordination under dynamic conditions enabled by the TraderBots approach allows wider application of multirobot systems.

1.1 Motivation

In this digital age, the demand for technological solutions to increasingly complex problems is climbing rapidly. With this increase in demand, the tasks which robots are required to execute also rapidly grow in variety and difficulty. A single robot is no longer the best solution for many of these new application domains; instead, teams of robots are required to coordinate intelligently for successful task execution. For example, a single robot is not an efficient solution to automated construction, urban search and rescue, assembly-line automation, mapping/investigation of unknown/hazardous environments, and many other similar tasks. Multirobot solutions are paramount for several reasons:

1. A single robot cannot perform some tasks alone, a team is required for successful execution. While in many cases it may be possible to design a single robot capable of executing all tasks, many problems are better suited to team-execution. For example, a single robot can accomplish moving heavy objects if the robot is designed appropriately. However, in many cases, it is simpler to design a team of robots that cooperate to move the heavy objects efficiently. Other application domains such as robotic soccer require a team of robots and cannot be executed with a single robot.
2. A robot team can accomplish a given task more quickly/efficiently than a single robot can by dividing the task into sub-tasks and executing them concurrently in application domains where the tasks can be decomposed. Application domains such as mapping of unknown areas and searching for landmines require careful coverage of a large area. Problems such as these can be easily decomposed into components such that a team of robots can divide the workload and execute sub-portions of the task concurrently, thus completing the overall task more efficiently.
3. A team can make effective use of specialists designed for a single purpose (for example, scouting an area, picking up objects, or hauling payload), rather than

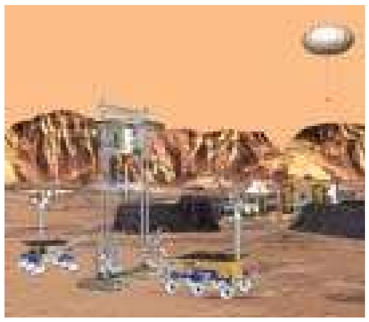
requiring that a single robot with versatile capabilities be a generalist, capable of performing all tasks. This allows more flexibility in designing the robots since a robot that needs to haul heavy payloads can be built with a heavy base for stability and strength, while a robot providing visual feedback can be designed to be more agile and move around with greater speed.

4. A team of robots can localize themselves more efficiently if they exchange localization and map information whenever they sense each other. This allows more robust localization capabilities. In an environment where a single robot would have to rely on landmarks of some sort for localization, a team could have the added advantage of being able to benefit from the localization information of their teammates.
5. A team of robots generally provides a more robust solution by introducing redundancy, and by eliminating any single point of failure as long as there is overlap between the robots' capabilities. For example, a team of robots equipped with cameras, will be a more reliable system for constructing vision-based maps of a dynamic environment because the failure of a single one of these robots will not jeopardize the entire mission.
6. A team of robots can produce a wider variety of solutions than a single robot can, and hence a team can opportunistically respond to dynamic conditions in more creative and efficient ways. Even if a team of robots does not overlap entirely in terms of specialization, the collective resources of the group can be used in creative ways to solve problems. For example, if a diagnostic robot loses its camera during operation, another robot with a camera could aid the diagnostic robot to complete its tasks by providing visual feedback. Similarly, if a rover gets stuck in the mud, one or more of its teammates can assist the stuck robot by pushing it out of the mud.

Thus, for many applications, a team of robots can be used more effectively. Briefly described below are some of the more prominent application domains that would benefit from efficient coordination of multirobot systems:

§ **Autonomous robot teams for operations in remote locations (Remote Operations):**

Many applications in the future will require a team of robots to autonomously execute complex tasks, while humans intervene remotely from time to time to alter the procedure of operations, remedy a situation beyond the capabilities of the robots, or coordinate with the robots to accomplish additional goals. Examples of such application domains are extra-planetary exploration and construction, scientific exploration of hazardous environments, and crop cultivation in underwater environments.



§ **Robotic aids for urban reconnaissance (Urban Reconnaissance):**

Military operations in urban terrain pose fierce constraints such as limited visibility, complex and expansive fortifications, limited intelligence, and the presence of native populations and other non-combatants that prohibit deployment of large forces. Moreover, the use of certain threats, e.g. biological and chemical agents, against both land forces and indigenous population in urban settings is an increasing likelihood. These conditions place both land forces and non-combatants in a highly non-deterministic, dangerous, confrontational, and volatile environment. The development of robotics technology will enable minimally invasive and precise operations that reduce risk to both ground forces and non-combatants by removing humans from dangerous and sometimes confrontational tasks. Potential tasks for robotic systems include mapping/scouting, reconnaissance, security/monitoring, and communications infrastructure.

§ **Robotic aids for urban search and rescue (Urban Search And Rescue):**

Urban Search And Rescue (USAR) workers have forty-eight hours to find trapped survivors in a collapsed structure; otherwise the likelihood of finding victims still alive is nearly zero. Earthquake and other disaster mitigation require rapid and efficient search and rescue of survivors. As recently seen in the USA, Turkey and Taiwan, the magnitude of the devastation of urban environments exceeds the available resources (USAR specialists, USAR dogs, and sensors) needed to rescue victims within the critical first 48 hours. Moreover, the mechanics of how large structures collapse often prevent rescue workers from searching buildings due to the unacceptable personal risk and the added risk to survivors from further collapse of the building. Furthermore, both people and dogs are frequently too big to enter voids, limiting the search to no more than a few feet from the perimeter. Robots can make a significant impact in this domain if made capable of aiding humans in USAR efforts.

§ **Automated warehouse management (Warehouse Management):**

Warehouse operators face the competing challenges of reducing costs while improving customer responsiveness. Order picking represents one of the costliest processes within distribution centers because of high labor content and equipment investment. Operators rely on humans to pick orders and either transport the material with manually driven industrial hand trucks (also known as pallet trucks or pallet jacks) or conveyor systems. An automated approach to case order picking has the potential to capture the benefits of manually driven pallet trucks and conveyor systems. Automated/robotics pallet jacks can roam the distribution center under the warehouse management system's global supervision and move to human pickers stationed to serve one or two aisles. Pickers spend less time traveling and more time picking; the pallets are built as cases are picked, and no major infrastructure changes are required.

§ **Intelligent Environments:**

Intelligent Environments are spaces in which computation is seamlessly used to enhance ordinary activity. They enable tasks historically outside the normal range of human-computer interaction by connecting computers to normal, everyday phenomena that have traditionally been outside the purview of contemporary user-interfaces. Their applications are intelligent rooms and personal assistants. Many familiar environments such as office buildings, supermarkets, schoolrooms, and restaurants are highly likely to incrementally evolve into intelligent environments within the next couple of decades. In these environments, agents can represent different resources and oversee efficient utilization of the resources. These agents can also resolve any conflicts about resource utilization. Moreover, the agents can keep track of maintenance requirements for all resources in the environment. Finally, each human entering the environment can also have a personal representative agent whose goal is to optimize conditions in the environment for the user. In the longer-term robotic assistants can become part of the intelligent environments. The robots, resource-management agents, and humans must integrate and cooperate seamlessly in these intelligent environments.

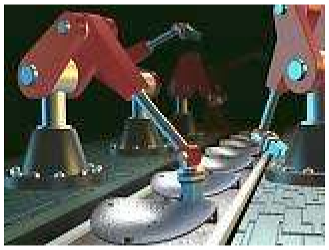
§ **Automated Construction:**

The application domain of automated construction involves the assembly of large-scale structures, such as terrestrial buildings, planetary habitats, or in-space facilities. Such domains need heavy lifting capabilities, as well as precise, dexterous manipulation to connect parts together. A motivating scenario is that of assembling the steel structure of a large building. In such cases, a large crane is used to lift beams and move them near their destinations; a worker near the destination uses hand signals to guide the crane operator; when the beam is close enough, the worker grabs the end and moves it into place. Terrestrial construction tasks, especially in more remote and hazardous areas, can benefit from the help of robotic construction teams. The domain of automated construction, however, is not limited to terrestrial work. Future space facilities, characterized by their immense size and the difficulties of human construction in space will be assembled in part by groups of robots.

§ **Robotic educational and entertainment systems (Education and Entertainment):**

Robotic toys, educational tools, and entertainment systems are rapidly gaining popularity and will continue to do so in the future. Many of these systems will require coordinated efforts by multiple robots. An example in this domain is robotic soccer. “RoboCup” is an international research and education initiative that attempts to foster AI and intelligent robotics research by providing a standard problem, the soccer game, where a wide range of technologies can be integrated and examined and also used for integrated project-oriented education. In order for a robot team to play soccer various technologies, including design principles of autonomous agents, multi-agent collaboration, strategy acquisition, real-time reasoning, robotics, and sensor-fusion, must be seamlessly integrated. RoboCup is a cooperative task for a team of multiple fast-moving robots under a dynamic environment.

§ **Automated production plants and assembly lines (Production):**



A growing trend in production plants is automation. In order to increase production, decrease labor costs, improve efficiency, increase safety, and improve quality in general, more and more industries are seeking to automate their production facilities. This trend demands efficient and robust coordination of heterogeneous multirobot systems. Advantages in the industrial automation domain are a highly controllable environment, well-defined and well-specified tasks, and specifically designed robots.

§ **Robotic exploration of hazardous environments (Hazardous Exploration):**

Exploration of hazardous environments has long been a problem demanding robotic solutions. Some examples of robotic hazardous environment exploration are exploration of extra-planetary regions, exploration of volcanic regions, exploration of disaster zones, exploration and mapping of mines, and exploration of minefields. Many of these tasks require coverage of large spaces under dangerous conditions. Such tasks are best suited for execution by a team of robots.

§ **Robotic cleanup of hazardous sites (Hazardous Clean-up):**

Robots continue to play an important role in cleanup of hazardous sites. Some examples in this domain are robotic minesweeping, robotic cleanup of nuclear waste, and robotic cleanup of disaster zones. Due to the high level of danger involved with these tasks they are highly unsuitable for human execution, and hence desirable for robotic execution. Furthermore, the dangerous circumstances of these applications make the domain more suited for a multirobot system with built-in redundancy.

§ **Agriculture:**

Many groups involved with agricultural work are now seeking automated solutions to their labor problems. Due to the long hours, hard physical work in rough conditions, and tedious and repetitive nature of some of the tasks in this domain, a growing decline in the available pool of labor has become evident. Spraying fields, harvesting, moving containers, and sorting plants are some examples of tasks that can be automated in this domain. For many of these tasks, coordinated teams of robotic agricultural machines promise to provide efficient solutions.

These and other application domains continue to demand more complex and higher standards of performance from automation technology/robotics. These demands will continue to rise in the future. Hence, multirobot coordination research must evolve to meet these demands. While individual applications will make specific demands on a multirobot coordination approach, designing a new approach for each application is not cost effective or efficient. A better methodology is to design one or more general approaches to multirobot coordination which are applicable across many domains, and which can be fine-tuned for specific applications. In order to design such an approach, it is instructive to first understand the requirements of the domain.

This dissertation focuses on multirobot coordination in dynamic environments for application domains such as long-duration operations in remote locations, urban search and rescue, reconnaissance, and exploration of hazardous environments. The requirements of multirobot application domains in dynamic environments are explored in detail next.

1.2 Understanding Multirobot Coordination in Dynamic Environments

Many application domains demand high quality performance from multirobot systems as discussed above. While applications of multirobot systems come in a wide variety, the focus of this dissertation is multirobot coordination in dynamic environments. Hence, it is instructive to examine what requirements must be fulfilled by a general multirobot coordination approach for such application domains. The following characteristics are thought to be an exhaustive list of these requirements:

§ **Robustness**

Robust to robot failure; no single point of failure for the system: This is an important characteristic since most applications in dynamic environments rely on continued progress even if some components of the system fail. . For example, an urban search and rescue operation expects that several robots will malfunction or be destroyed during task-execution, and still require the overall mission to be completed in the best way possible given the remaining resources. Graceful degradation of system performance with component failure is a highly desirable requirement for many applications in general.

§ **Speed**
Quick response to dynamic conditions: Often in dynamic environments, a key to successful task execution is the ability to respond quickly to the dynamic conditions. If information always needs to be channeled to another location for plan modification, conditions can change too rapidly for the planning to keep up. In dynamic application domains that deal with hazardous environments the need for quick response times to dynamic conditions is much higher – slow response to dangerous changes in environmental conditions can have devastating results.

§ **Efficiency**
Opportunistically optimized response to dynamic conditions: This characteristic is desirable in general, and required in some domains. The prevalent dynamic conditions in the targeted application domains require the ability to opportunistically optimize the system response to these conditions for efficiency and success. Often a strategy for executing a task or a strategy for distributing tasks among several robots will cease to be an efficient strategy due to a change in the prevailing conditions. The multirobot system can quickly become highly inefficient if it cannot respond to these changes in conditions and re-strategize accordingly in an opportunistic manner.

§ **Information**
Ability to execute a task without complete/perfect information about the task or the environment: A significant challenge in many dynamic multirobot application domains is the lack of complete and reliable information. Coordination strategies need to be a lot more flexible if all information is not known a-priori. In domains such as urban search and rescue and exploration, often much is unknown about the prevailing conditions of the environment. Hence, robots need to rely on their sensors to discover these conditions. Thus, the information will only be good as the sensing capability and thus uncertainty is introduced. The dynamic nature of the environment further exacerbates the challenge since discovered information cannot be relied on as perfect or sustained. A successful coordination mechanism needs to take all of this into account and deal with the challenges of imperfect information in an efficient manner.

§ **Communication**
Ability to deal with limited and imperfect communication: In general, many application domains cannot realistically guarantee perfect communication among all robots at all times. This characteristic is exacerbated in dynamic domains. Hence, any suitable coordination approach should be robust to communication failures and limits in bandwidth and range of communication.

§ **Resources**
Ability to reason about limited resources: The ability to reason about the limited resources available in a robotic system is very important for optimization purposes. For example, in a scientific exploration task, it is undesirable to use the only robot with some very costly science sensor to perform a simple but risky scouting task.

Thus, the collective resources of the system and their relative worth must be taken into account during the task allocation process. Also, when a robot is assigned a task, its planner must understand the resource requirements for that task in order to enable efficient scheduling. The robot must also take into account events scheduled to occur in the future and the resources that will be required for those events before committing to any new tasks.

§ **Allocation**
Efficient allocation of tasks: A key challenge in coordinating multiple robots is deciding who does what when. Thus, the task allocation mechanism is an important factor in the design of the coordination approach. Factors such as robot capabilities and resources, current commitments of the robots, priority and risk involved with different tasks, task constraints, environmental conditions and predictions for future conditions, and tradeoffs between planning time and execution time need to be considered in order to maximize the efficiency of the task allocation.

§ **Roles**
Efficient adoption of roles: Similar to human teams, robot teams require each robot in the team to play a given role or a set of roles at any given time. For example, a robot may act as a coordinator for a group of robots, by planning strategies for the group's task allocation and execution, while also acting as a map-builder, by incorporating its sensor readings into a map of the environment, and also as a communication router, by passing messages between robots that are out of range from each other but within range of this robot. In many coordination mechanisms robots are restricted to being able to play only a single role in the team at any given time, even if they possess the resources to be able to play multiple roles simultaneously. Efficient role adoption will enable robots to play as many roles as required at any given time based on resource availability, and also allow robots to change in and out of different roles as conditions change.

§ **New Input**
Ability to dynamically handle new instructions from a human operator: In many dynamic application domains, the demands on the robotic system can change during operation. Hence, it may become necessary to assign new tasks, cancel previously assigned tasks, or alter existing tasks. For example, in an urban search and rescue operation, new regions to be explored could be assigned to the robot team based on new information received, previously assigned exploration regions could be cancelled due to newly discovered dangers, and the priorities of different assigned regions for exploration could be altered based on newly discovered information of survivors. A successful coordination mechanism will be able to accommodate all of these requirements.

§ **Fluidity**
Easily able to accommodate the addition/subtraction of robots during operation: Several applications require the ability to introduce new robots into the system during operation. Conversely, robots can exit or malfunction during task execution, especially in hazardous and dynamic environments. A successful coordination mechanism will be able to support these events gracefully.

§ **Heterogeneity**
Ability to accommodate heterogeneous teams of robots: Many multirobot coordination approaches assume homogeneity of the robot team for ease of planning. The coordination problem is more difficult if the robots are heterogeneous. Dynamic environments often require heterogeneous teams in order to be able to deal with the different conditions encountered. Thus, a successful coordination approach will be able to accommodate any team regardless of its homogeneity or heterogeneity.

§ **Extensibility**
Easily extendable to accommodate new functionality: A key characteristic to building a generalized system that can evolve with the needs of the different applications is the ability to easily add and remove functionality as needed. This is identified as extensibility. A common approach is to build the system in a modular fashion such that different modules can be altered or replaced relatively easily according to the requirements of the specific application.

§ **Flexibility**
Easily adaptable for different applications: Since different applications will have different requirements, a widely applicable coordination approach will need to be easily configurable for the different problems it proposes to solve – this is known as flexibility. Instructions and advice on how to reconfigure the mechanism for different applications will also be useful. Identifying important parameters that need to be changed based on the application requirements, instructions on how to change them, identifying components of the mechanism that need to be added/changed based on application requirements, and instructions on how to make these alterations are all important elements of a successful coordination mechanism. A further bonus will be well-designed user interfaces and tools that allow plug and play alterations to the coordination mechanism and automated methods for parameter tuning.

§ **Tight-Coordination**
Ability to coordinate robots in maneuvers that require tight-coordination: Some applications require robots to execute tasks in tight-coordination. For example, if a group of robots have to cooperate to move a heavy object, it requires tight-coordination. While this characteristic is not always required, a generally applicable coordination mechanism will be able to support tasks that require tight-coordination.

§ **Scalability**
Ability to coordinate large numbers of robots: While most application domains don't require large numbers of robots, and limits in resources often prevent deployment of large robot teams, scalability is an attractive quality for a coordination mechanism that aims to be widely applicable.

§ **Learning**
On-line adaptation for specific applications: While a generalized system is often more useful, its application to specific domains usually requires some parameter tuning. The ability to tune relevant parameters automatically in an on-line fashion

is thus a very attractive feature that can save a lot of effort. Thus, the integration of learning techniques to allow the robots to adapt to changing environments and different task domains can be a very powerful feature.

§ **Implementation**

Implemented and proven on robotic system: As with any claim, a proven implementation is far more convincing. Moreover, successful implementation of a coordination mechanism on a robotic system requires discovering and solving many details that are not always apparent in simulation and software systems.

The characteristics described above represent requirements over a wide cross section of application domains. The two tables below illustrates the subset of these requirement characteristics specific to the multirobot applications domains described in section 1.1:

Application	**Robustness**	**Speed**	**Efficiency**	**Information**	**Communication**	**Resources**	**Allocation**	**Roles**	**New Input**
Remote Operations	Y	Y	Y	Y	Y	Y	Y	Y	Y
Urban Reconnaissance	Y	Y	Y	Y	Y	Y	Y	Y	Y
Urban Search And Rescue	Y	Y	Y	Y	Y	Y	Y	Y	Y
Warehouse Management	Y	Y	Y				Y		Y
Intelligent Environments	Y	Y	Y			Y			Y
Automated Construction	Y	Y	Y			Y	Y	Y	Y
Education and Entertainment	Y	Y	Y	Y	Y	Y	Y	Y	Y
Production		Y							
Hazardous Exploration	Y	Y	Y	Y	Y	Y	Y	Y	Y
Hazardous Clean-up	Y	Y	Y	Y	Y	Y	Y	Y	Y
Agriculture	Y	Y							

Table 1: Array of required characteristics for multirobot applications (Part I)

Application	Fluidity	Heterogeneity	Extensibility	Flexibility	Tight Coordination	Scalability	Learning	Implementation
Remote Operations	Y	Y	Y	Y	Y	Y	Y	Y
Urban Reconnaissance	Y	Y	Y	Y		Y		Y
Urban Search And Rescue	Y	Y	Y	Y	Y	Y		Y
Warehouse Management	Y		Y	Y		Y	Y	Y
Intelligent Environments	Y	Y	Y	Y		Y	Y	Y
Automated Construction		Y	Y	Y	Y			Y
Education and Entertainment	Y	Y	Y	Y			Y	Y
Production Plants		Y	Y	Y	Y	Y	Y	Y
Hazardous Exploration	Y	Y	Y	Y		Y		Y
Hazardous Clean-up	Y	Y	Y	Y	Y	Y		Y
Agriculture	Y		Y	Y		Y	Y	Y

Table 2: Array of required characteristics for multirobot applications (Part II)

A brief explanation of why each of the listed application domains requires the characteristics illustrated in Table 1 and Table 2 is presented next:

§ **Remote Operations**

Robot teams carrying out operations in remote locations will require all of the listed characteristics. The team will need to be robust to failures and malfunctions of some team members, able to accept new team members, be efficient and quick in their response to dynamic conditions in order to maximize accomplishments and minimize dangers, and able to deal with incomplete information because many of these operations will take place in regions about which we do not have complete information. The robots will also have limited communication infrastructure and resources, and will need to take these limitations into account when deciding on task allocations and roles in the team. The robot team will likely be heterogeneous to accommodate the different tasks and terrains, need to be capable of executing tightly coordinated tasks, and will need to be able to change their operations based on instructions from human operators and newly discovered information. A successful coordination mechanism for this domain will be easily extended to include new types of tasks, flexible enough to be deployed in different locations, scalable to accommodate different team sizes, amenable to learning techniques for parameter tuning, and proven via implementation on a robotic system.

§ **Urban Reconnaissance**

Robot teams assisting with urban reconnaissance missions will require most of the listed characteristics. The team will need to be robust to failures and malfunctions of some team members, able to accept new team members, be efficient and quick in their response to dynamic conditions in order to maximize accomplishments and minimize malfunctions, and able to deal with incomplete information because many of these operations will take place in areas for which complete information is not known a-priori. The robots will also have limited communication infrastructure and resources, and will need to take these limitations into account when deciding on task allocations and roles in the team. The robot team will likely be heterogeneous to accommodate the different tasks and terrains and will need to be able to change their operations based on instructions from human operators and changing conditions. A successful coordination mechanism for this domain will be easily extended to include new types of tasks, flexible enough to be deployed in different locations, scalable to accommodate different team sizes, and proven via implementation on a robotic system. It is unlikely that robots will need to, or have the time to, execute tightly coordinated tasks or be able to learn any parameters in this task domain due to its highly adversarial and dynamic nature.

§ **Urban Search And Rescue**

Robot teams assisting with urban search and rescue missions will also require most of the listed characteristics. The team will need to be robust to failures and malfunctions of some team members, able to accept new team members, be efficient and quick in their response to dynamic conditions in order to maximize accomplishments and minimize failures, and able to deal with incomplete information because many of these operations will take place in situations where complete information of the disaster zone is not known a-priori. The robots will also have limited communication infrastructure and resources, and will need to take these limitations into account when deciding on task allocations and roles in the team. The robot team will likely be heterogeneous to accommodate the different tasks and terrains, need to be capable of executing tightly coordinated tasks such as moving large obstacles and jointly maneuvering over difficult terrain, and will need to be able to change their operations based on instructions from human operators and changing conditions. A successful coordination mechanism for this domain will be easily extended to include new types of tasks, flexible enough to be deployed in different locations, scalable to accommodate different team sizes, and proven via implementation on a robotic system. It is unlikely that robots will have the time to learn any parameters in this task domain due to its highly dynamic nature and limited repeatability of environmental factors.

§ **Warehouse Management**

Since warehouses are relatively highly controlled environments, automated teams of palette jacks assisting with order picking will not require most of the listed characteristics. The team will still need to be robust to failures and malfunctions of some team members, able to accept new team members, and be efficient and quick in their response to dynamic conditions such as blocked intersections and toppled crates in order to prevent collisions and other damage. Since the environment is controlled and the tasks well specified, information, communication infrastructure and resources will be well known a-priori, and not be limited as much as the previous application domains. Prevailing conditions will still need to be taken into account when deciding on task allocations. The robot team will likely be homogenous and only play a single role. The team will also not need to execute tightly coordinated tasks and it is unlikely that tasks will change during execution.

New tasks will however emerge as new orders arrive at the warehouse. A successful coordination mechanism for this domain will be easily extended to include new types of tasks, flexible enough to be deployed in different warehouses, scalable to accommodate different team sizes, amenable to learning techniques for parameter tuning, and proven via implementation on a robotic system.

§ **Intelligent Environments**

Intelligent environments are also relatively highly controllable environments and hence will not require many of the listed characteristics. The system will still need to be robust to failures and malfunctions of some team members, able to smoothly accept new team members and exit current members, and be efficient and quick in their response to dynamic conditions such as changes in schedules, cancelled meetings, power failures, etc. Since the environment is controlled and the tasks well specified, information, communication infrastructure, task allocations, and roles will be well known a-priori, and not be limited as much as some previous application domains. The robot team will likely be heterogeneous and each team member will probably be designed to play a specific role. The team will not need to execute tightly coordinated tasks but it is highly likely that tasks will change often. New tasks will also emerge as new events get scheduled and resources get used. A successful coordination mechanism for this domain will be easily extended to include new tasks, flexible enough to be deployed in different environments, scalable to accommodate different team sizes, amenable to learning techniques for parameter tuning, and proven via implementation on a robotic system.

§ **Automated Construction**

Automated construction will usually happen within a relatively contained and controlled environment, and hence will not require some of the listed characteristics. The system will still need to be robust to some level of malfunctions of some team members, but it is unlikely that the composition of the team will change. The team needs to be efficient and quick in their response to dynamic conditions such as misalignment of beams and changes in weather conditions. However, since the environment is somewhat controlled and the tasks well specified, environmental information and communication infrastructure will be well known a-priori, and not be limited as much as some previous application domains. The robots will also have limited communication infrastructure and resources, and will need to take these limitations into account when deciding on task allocations and roles in the team. The robot team will likely be heterogeneous to accommodate the different tasks and terrains and will need to be able to change their operations based on changing conditions and possibly instructions from human operators. The team will mostly need to execute tightly coordinated tasks due to the nature of construction but it is highly unlikely that tasks will change often. A successful coordination mechanism for this domain will be easily extended to include new tasks, flexible enough to be deployed in different environments, and proven via implementation on a robotic system. It is unlikely that team sizes will change much and that learning techniques for parameter tuning will help much in this domain.

§ **Education and Entertainment**

Education and entertainment robotics will require most of the listed characteristics. These teams, for example in the robot soccer domain, will need to be robust to failures and malfunctions of some team members, able to accept new team members, be efficient and quick in their response to dynamic conditions in order to maximize accomplishments and minimize damages, and able to deal with incomplete information. The robots will also have limited communication infrastructure and resources, and will need to take these limitations into account when deciding on task allocations and roles in the team. The robot team will likely be heterogeneous to accommodate the different tasks and will need to be able to change their operations based on instructions from human operators. It is highly unlikely that the team will need to execute tightly coordinated tasks or that the team size will be very large. A successful coordination mechanism for this domain will be easily extended to include new types of tasks, flexible enough to be deployed in different locations, amenable to learning techniques for parameter tuning, and proven via implementation on a robotic system.

§ **Production**

Production plants and assembly lines are highly controlled environments and hence will not require most of the listed characteristics. The team will not need to be robust to failures and malfunctions or accept new team members or be efficient in dealing with dynamic conditions since the environment will be highly controlled and humans will most likely deal with any malfunctions. The robots however will need to be quick in their detection and response to dynamic conditions such as malfunctioning parts and obstacles in their path in order to prevent damage. Since the environment is controlled and the tasks well specified, information, communication infrastructure, task allocations, and roles will be well known a-priori, and not be limited as much as some previous application domains. The robot team will likely be heterogeneous and each team member will probably be designed to play a specific role. The team will mostly need to execute tightly coordinated tasks due to the nature of production but it is highly unlikely that tasks will change often. A successful coordination mechanism for this domain will be easily extended to include new types of tasks, flexible enough to be deployed in different facilities, scalable to accommodate large team sizes, amenable to learning techniques for parameter tuning, and proven via implementation on a robotic system.

§ **Hazardous Exploration**

Robot teams assisting with hazardous exploration missions will require most of the listed characteristics. The team will need to be robust to failures and malfunctions of some team members, able to accept new team members, be efficient and quick in their response to dynamic conditions in order to maximize exploration and minimize malfunctions, and be able to deal with incomplete information because many of exploration by nature deals with situation where complete information is not known a-priori. The robots will also have limited communication infrastructure and resources, and will need to take these limitations into account when deciding on task allocations and roles in the team. The robot team will likely be heterogeneous to accommodate the different tasks and terrains and will need to be able to change their operations based on instructions from human operators and changing conditions. A successful coordination mechanism for this domain will be easily extended to include new types of tasks, flexible enough to be deployed in different locations, scalable to accommodate different team sizes, and proven via implementation on a robotic system. It is unlikely that robots will need to execute tightly coordinated tasks or be able to learn any parameters in this task domain due to its highly dynamic and low repetitious nature.

§ **Hazardous Clean-up**

Robot teams assisting with hazardous clean-up missions will also require most of the listed characteristics. The team will need to be robust to failures and malfunctions of some team members, able to accept new team members, be efficient and quick in their response to dynamic conditions in order to maximize accomplishments and minimize failures, and able to deal with incomplete information because many of these operations will take place in situations where complete information of the disaster zone is not known a-priori. The robots will also have limited communication infrastructure and resources, and will need to take these limitations into account when deciding on task allocations and roles in the team. The robot team will likely be heterogeneous to accommodate the different tasks and terrains, need to be capable of executing tightly coordinated tasks such as moving large obstacles and jointly maneuvering over difficult terrain, and will need to be able to change their operations based on instructions from human operators and changing conditions. A successful coordination mechanism for this domain will be easily extended to include new types of tasks, flexible enough to be deployed in different locations, scalable to accommodate different team sizes, and proven via implementation on a robotic system. It is unlikely that robots will have the time to learn any parameters in this task domain due to its highly dynamic nature and limited repeatability of environmental factors.

§ **Agriculture**

Agriculture is a relatively controlled environment and hence will not require some of the listed characteristics. A team of agricultural robots will need to be robust to failures and malfunctions of some team members but will not need to be able to accept new team members or be efficient in dealing with dynamic conditions since the environment will be highly controlled and humans will most likely deal with any unusual conditions. The robots however will need to be quick in their detection and response to dynamic conditions such as malfunctioning parts and obstacles in their path in order to prevent damage. Since the environment is controlled and the tasks well specified, information, communication infrastructure, task allocations, and roles will be well known a-priori, and not be limited as much as some previous application domains. The robot team will likely be homogenous and only play a single role. The team will also not need to execute tightly coordinated tasks and it is unlikely that tasks will change during execution. A successful coordination mechanism for this domain will be easily extended to include new types of tasks, flexible enough to be deployed in different locations, scalable to accommodate different team sizes, amenable to learning techniques for parameter tuning, and proven via implementation on a robotic system.

The descriptions above indicate that the more dynamic application domains require more of the listed characteristics, while the application domains with more controlled environments have less requirements. Thus, a multirobot coordination approach designed to address the requirements of dynamic environments needs to incorporate all of the listed characteristics in order to be successful in a wide range of application domains. Therefore, designing a robust coordination mechanism for multirobot systems in dynamic environments is a highly challenging task.

1.3 Thesis Statement

This thesis asserts that the use of market techniques in the TraderBots multirobot coordination mechanism enables efficient and robust multirobot coordination in dynamic environments.

1.4 Thesis Roadmap

This dissertation presents a novel market-based approach to multirobot coordination, TraderBots, designed to satisfy all of the characteristics required for efficient and robust multirobot coordination in dynamic environments. Chapter 2 explores the problem description and related work in more detail, followed by a detailed description of the TraderBots approach in Chapter 3.
The principal benefits of the TraderBots approach are examined in detail next; Chapter 4 examines the different means by which robustness is ensured. Chapter 5 examines proposed solutions to encouraging efficient solutions, focusing on the benefits and challenges of introducing coalitions and leaders.

A comparative study of the TraderBots approach and two other fundamental approaches that bracket the solution space for the multirobot coordination problem is explored in Chapter 6, and a presentation and analysis of overall experimental results can be found in Chapter 7. This dissertation concludes in Chapter 8 with a summary of this thesis, a list of contributions made by this dissertation towards advancing the state of the art in robotics literature, the impact of this dissertation on other research efforts, and a description of future work. Appendix 1 presents details of the different implementations to date of the TraderBots approach.

CHAPTER 2

Problem Description

THE past decade has witnessed a growing emphasis in research topics highlighting coordination of multirobot systems. This emphasis is generated by the increasing demand for automation in application domains where a single robot is no longer capable of performing the necessary tasks, and/or multiple robots can accomplish the same tasks more efficiently. Coordinating a multirobot system is more complicated than controlling a single robot, not only because of the increased number of robots to be coordinated, but also due to the added complication of requiring the robots to work together in an intelligent manner to achieve assigned goals efficiently and robustly. Dynamic environments, malfunctioning robots, communication disruptions, and multiple user requirements add to the complexity of the multirobot coordination problem. Cao et al. [24], Dias and Stentz [40] and Matari [74] explore some of these issues, and present summaries of some of the principal efforts in this field of research.

2.1 Problem Statement

Prior to the work detailed in this dissertation, the multirobot coordination problem had not been solved for dynamic domains where highly sub-optimal solutions can be very costly and a single point of failure is unacceptable. Mainly lacking in the prior work is a multirobot coordination approach that reasons in an efficient fashion about resource utilization and task allocation while maintaining the ability to respond quickly and robustly to dynamic conditions. This dissertation details the design and implementation of a novel market-based approach, TraderBots, which solves the multirobot coordination problem in dynamic environments in a robust and opportunistically efficient manner, and enables a detailed analysis of the effectiveness of applying economic strategies to solve the multirobot coordination problem. The TraderBots approach is designed to satisfy all of the characteristics in the previous chapter identified as being necessary for addressing the problem of efficient and robust coordination of multiple robots in dynamic environments.

A principal advantage of the TraderBots approach is its ability to opportunistically optimize resource utilization. A simple example is illustrative. Consider a system where we want two items to be retrieved. Two robots are available to do the task.

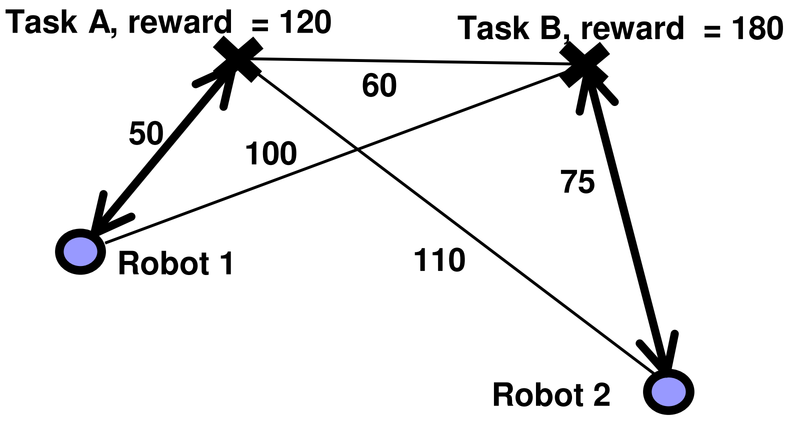


Figure 1: An illustration of simple reasoning

The robots incur costs in driving, as indicated by the numerical labels in Figure 1. The agent who owns the two tasks holds an auction to determine the allocation of these tasks. Lets assume that the robots are only capable of reasoning about single-task bids. Hence, the robots bid on each of the tasks separately. Robot 1 bids the cost, plus a profit of 20 for task A (120). Robot 2 cannot compete for this task since its costs alone are 220. If the auctioneer agent allocates the tasks in a greedy fashion, Robot 1 wins task A and, for similar reasons, Robot 2 wins Task B. Both robots are happy, since each makes a profit. The auctioneer is happy, since the tasks are allocated quickly and with reasonable efficiency in terms of the cost.

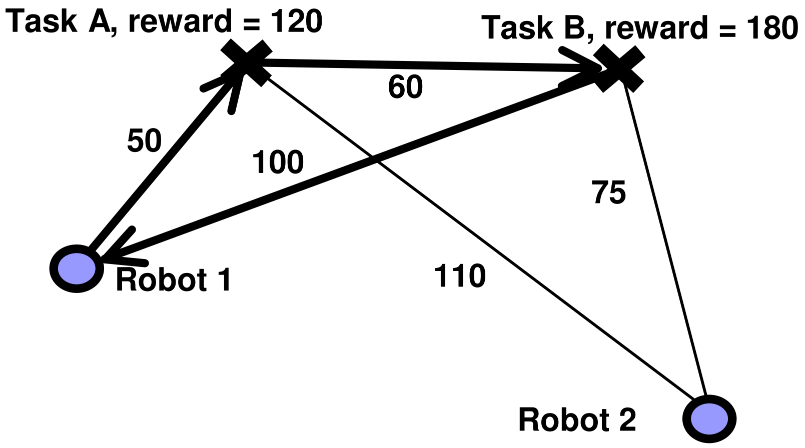


Figure 2: An illustration of more complex reasoning

Now the robots will continue to auction their incomplete tasks among themselves, always seeking a more profitable solution. Thus, consider the case when Robot 2 puts up Task B for auction. Robot 1 must now reason about its complete tour. Thus, if Robot 1 adds Task B immediately after Task A before returning to its base as illustrated in Figure 2, an additional cost of 110 is incurred. If Robot 2 can subcontract Task B, it will save 150 in cost. Therefore, it will do so if it pays out less than 150. Thus, Robot 2 can place a reservation price of 150 on Task B notifying the bidders that it will not pay more than 150 for subcontracting Task B. Robot 1 determines that the new route will add 110 in cost to its expenses and sees that this additional cost is lower than the reservation price. Therefore, it will take the subcontract if it will receive more than 110. A reasonable strategy is for Robot 1 to bid 130 and win Task B from Robot 2. Both robots are happy, since they increased their profits, and the system produces a more efficient solution because the global task was accomplished at lower cost.

Note that the same solution could be achieved if Robot 2 was playing acting as a leader and ascertained through observation or via communication that a better plan was to offload Task B to Robot 1. Robot 2 wouldn't lead by coercion; instead, it can use the additional profit to "buy off" the participant, Robot 1, and thus convince Robot 1 to complete both tasks. Good plans generate more profits that can be used to gain participants. Note further that this plan could also have been proposed to both robots by a third robot playing a leader role.

The TraderBots approach is not, however, the optimal solution for all multirobot application domains. One concern is that there may be difficulties designing appropriate cost and revenue functions to represent some problems accurately. Furthermore, centralized approaches will work best for multirobot applications in static environments where there are only a few robots required to execute a task – in such scenarios a centralized approach can often produce optimal solutions to the multirobot coordination problem. Similarly, reactive methods will provide less complicated coordination-approaches for multirobot applications where efficiency is not a paramount concern. Thus, the TraderBots approach is best suited for multirobot tasks in dynamic environments where efficient solutions are highly preferred. The principal assumptions made in this dissertation, and the resulting limitations, are addressed in the following section. Different approaches to solving the multi-agent coordination problem are reviewed in sections 2.3, 2.4, and 2.5.

2.2 Assumptions

In order to limit the scope of this dissertation, several assumptions are made. These assumptions and the resulting limitations are discussed next. Other work that addresses these limitations is also noted for the interested reader.

§ **Cooperative Tasks**
This dissertation only addresses application domains where robots are assigned cooperative tasks; the overall mission does not require robots to interact in an adversarial manner. The TraderBots approach itself, however, is not limited to being suitable for cooperative tasks. Bererton et al. ([10], [11]) investigate the application of market-based techniques for adversarial multirobot coordination domains (laser-tag).

§ **Truthful Agents**
The complications that arise from deceitful agents are beyond the scope of this dissertation. This assumption is made predominantly due to the fact that the designer of the robot team can ensure that the robots are truthful in the examined

application domains. While the topic of deceitful agents has not been directly addressed for market-based approaches in multirobot domains to date, many relevant lessons can be learned from related research in the domain of software agents and on-line auctions. A relevant example is work by Brainov and Sandholm ([17], [18]).

§ **Discernible Cost and Revenue Structures**
Discerning the structure of costs and revenues for some application domains can be complicated. This concern is not directly addressed in this dissertation. Application domains chosen for testing the TraderBots approach in this dissertation are limited to easily identifiable cost and revenue structures. Recent work by Bererton et al. [10] (based on work by Guestrin and Gordon [60]) address this concern by introducing techniques to discern suitable reward functions for multirobot coordination using market-based techniques.

§ **Ability to Align Local and Global Costs and Revenues**
The principal limitation of this dissertation stems from the assumption that local and global costs and revenues can be aligned. That is, the work done to date assumes that individual robots maximizing individual profits can be utilized to maximize team profits and thereby provide efficient solutions. This assumption trivially holds true for application domains where the team cost and team revenue can be discerned by a summation of the individual costs and individual revenues respectively. Application domains involving Traveling Salesman Problem (TSP)-like task structures are amenable to this assumption. Since many multirobot coordination application domains addressed in the literature to date fall within this category, this assumption does not excessively limit the applicability of the work presented in this dissertation. Moreover, research by Tumer and Wolpert [121] in the domain of Collectives present techniques for applying reinforcement learning methods to derive functions that align local and global revenue functions.

§ **Simple Tasks**
A final assumption made in this dissertation concerns the structure of tasks assigned to the robots. This dissertation does not address issues of task decomposition or complex tasks that require multiple robots to work in tight coordination in order to accomplish the task. All tasks considered to date in the application of the TraderBots approach assume that tasks are primitive tasks independently executable by a single robot. However, the TraderBots approach is not limited in capability to handling simple tasks. Research by Zlot and Stentz ([128], [129], [130]) demonstrates the ability to handle task decomposition and loosely coordinated tasks using market-based techniques, and Kalra and Stentz [67] explore techniques for accomplishing tight-coordination within the framework of market-based multirobot coordination.

2.3 Centralized Approaches

The benefits of using multiple robots to accomplish certain tasks are many, as described in Chapter 1. However, simply increasing the number of robots assigned to a task does not necessarily solve a problem more efficiently; multiple robots must cooperate to achieve efficiency. The difficulty arises in coordinating many robots to perform a single, global task. One approach is to consider the robot team to be a single robot "system" with many degrees of freedom. That is, a single robot or central computer is designated as the "leader" and is responsible for planning the actions of the entire group. This leader coordinates the group to perform the specified task. The members of the group convey relevant state information to the leader, and execute the plans generated by the leader, as illustrated in Figure 3. Some examples of such centralized approaches can be found in work done by Caloud et al. [23], Chaimowicz et al. [26], Jensen and Veloso [65], Brummit and Stentz [21], Simmons et al. [104], Švestka and Overmars [115], and Burgard et al. [22].

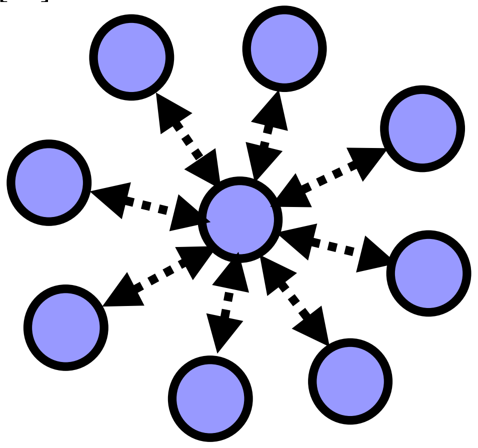


Figure 3: Illustration of centralized coordination of a team

The principal advantage of such centralized approaches is that optimal plans can be produced if the team size is sufficiently small and the environment is sufficiently static. Under these conditions, the leader can take into account all the relevant information conveyed by the members of the team and generate an optimal plan for the team. However, centralized approaches suffer from several disadvantages.

In a centrally coordinated system, the system response to changes in the environment is sluggish since all relevant information must be conveyed to the leader before any action can be taken. Another weakness with this approach is that it produces a highly vulnerable system. That is, if the leader (the central planning unit) malfunctions, a new leader must be available or the entire team is disabled. Due to this reason, design of the architecture is made more complex because the designer must decide how many agents should be capable of being leaders. If all agents are able to be leaders, the potential for wasted resources is high since only a single leader is usually required at any given time. However, if only one agent is able to be a leader, the system is made highly vulnerable. Finally, the approach often has stringent and heavy communication requirements because information is required from all agents in order to compute efficient plans. If the system is required to produce optimal solutions, the difficulty is increased.

Optimal coordination is computationally difficult—the best-known algorithms are exponential in complexity. Thus, an approach striving to compute globally optimal solutions becomes intractable for teams larger than a few. Additionally, the approach assumes that all relevant information about the robots and their environment can be transmitted to a single location for processing and that this information does not change during the time that an optimal plan is constructed. These assumptions are unrealistic for problems in which the environment is unknown and/or changing, communication is limited, and robots behave in unpredictable ways.

2.4 Distributed Approaches

Local and distributed approaches address the problems that arise with centralized, globally coordinated methods by distributing the planning responsibilities among all members of the team. Each robot operates largely independently, acting on information that is locally available through its sensors. Thus, each robot plans its course of action based on its local observations.

This allows fast response to dynamic conditions and decreases the communication requirements. A robot may coordinate with other robots in its vicinity, perhaps to divide a problem into multiple sub-problems or to work together on a sub-task that cannot be accomplished by a single robot. Typically, little computation is required, since each robot only plans and executes its own activities. Also, less stringent constraints on communication are required, since the robots only communicate with others in their vicinity, as illustrated in Figure 4. The robots are better able to respond to unknown or changing environments, since they sense and respond to the environment locally. Moreover, the system is more robust since the entire team's performance no longer depends on the guidance of a single leader. In general, no single point of failure exists for distributed systems and the approach scales easily to accommodate large numbers of robots. The approach works best for problems that can be decomposed into largely unrelated sub-problems, or problems for which a desired group behavior results from the aggregate of individual behaviors and interactions.

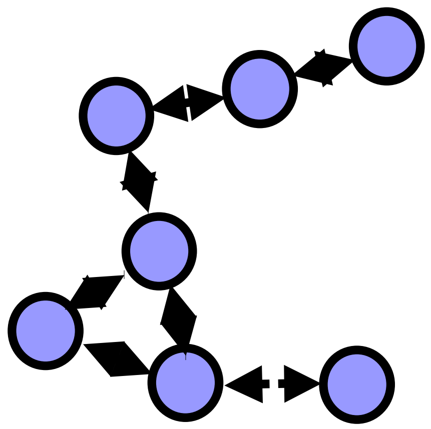


Figure 4: Illustration of distributed coordination of a team

Many research efforts have modeled distributed systems inspired by biology ([5], [20], [29], [30], [75], [101]). Others have designed systems based on fluidics and similar physics-based concepts ([12], [28], [36], [124], [126]). Some have chosen to pursue rule-based, heuristic-based and model-based approaches ([34], [56], [117]). Economy-based models have inspired still others ([57], [92], [107], [111], [126]). In general, cooperation in distributed approaches can be divided into two categories: fortuitous cooperation and planned cooperation. Work done by Arkin and Balch [5], Matari, [75], Goldberg and Matari, [54], Brooks [20], Salido et al. [91], Schneider-Fontán and Matari, ([101], [102]), Arkin ([3], [4]), Parker ([83], [84]), Maio and Rizzi [73], Goldman and Rosenschein [56], Desai et al. [36], Böhringer et al. [12], Alur et al. [1], Osawa [79], and Pagello et al. [80] are examples of distributed systems in the fortuitous cooperation category. Some examples in the planned cooperation category are work done by Kaminka and Tambe [68], Decker and Lesser [34], Tambe et al. [119], Tambe [117], Weiß [125], Jennings and Kirkwood-Watts [63], Veloso et al. [123], Noreils [78], Bonasso et al. [13], Stentz and Dias [111], Golfarelli et al. [57], Sandholm [92], Smith [107], Gibney et al. [53], Collins et al. [31], and Jennings and Arvidsson [62]. All of these approaches aim to solve the multirobot/multiagent coordination problem in a distributed manner so that the difficulties inherent in a centralized system are circumvented.

However, the principal drawback of many distributed approaches is that they often result in highly sub-optimal solutions because all plans are often based solely on local information. Stentz and Dias [111] propose a market-based approach which aims to opportunistically introduce pockets of centralized optimal planning into a distributed system, thereby exploiting the desirable properties of both distributed and centralized approaches.

2.5 Economy-based Approaches

Smith [107] first introduced the concept of using an economic model to control multiagent systems as the Contract Net protocol. Many groups have since adopted similar strategies for controlling multiagent systems. Work done by Krovi et al. [70], Faratin et al. [46], Jung et al. [66], Brandt et al. [19], Wellman and Wurman [126], Smith [107], Gibney et al. [53], Collins et al. [31], Jennings and Arvidsson [62], Sandholm [92], and Sycara and Zeng [116] are examples of economy-based sofware-agent systems. In contrast, work done by Laengle et al. [71], Simmons et al. [105], Botelho and Alalmi [15], Dias and Stentz [39], Gerkey and Matari, [49], and Golfarelli et al. [57] are examples of economy-based control-architectures applied to multirobot systems.

Economic approaches are not without their disadvantages. Negotiation protocols, mapping of task domains to appropriate cost functions, and introducing relevant de-commitment penalty schemes can quickly complicate the design of a control-architecture. Furthermore, some negotiation schemes can drastically increase communication requirements.

Many research efforts have focused on developing optimization techniques applicable to distributed multiagent systems. Sandholm and Lesser [93], Sandholm et al. [99], and Excelente-toledo et al. [45] have proposed methods of allowing breaching of contracts as an optimization mechanism in economy-based approaches. Sandholm and Suri [96] examine the use of combinatorial auction schemes for optimizing negotiation.

2.6 Multi-agent versus Multirobot Coordination

Many characteristics differentiate software-agent domains from situated-agent (robotic) domains [100]. Some principal differences between robotic systems and software systems are highlighted next.

Tasks assigned to robotic agents can vary significantly from tasks in software domains. Also, robotic agents often deal with more restricted resources and robotic systems often have to deal with more restricted communication. Failures occur with higher frequency, and in a wider variety in robotic systems. Partial observability due to limits in sensing capability, and more prominent manifestations of latencies due to interactions with hardware systems are other distinguishing features of robots from software agents. Furthermore, robotic systems have to be able to accommodate larger error bounds in performance since they often deal with faulty sensors and interact with real-world environments. Finally, robotic systems often require very different solutions to recover from faults (for example, one robot pushing another robot that is stuck, two robots cooperating to lift a heavy obstacle, etc.). Thus, controlling multirobot systems can be a significantly different problem compared to controlling multiple software agents.

2.7 Matrix of Technical Problems Solved in Related Work

The following matrix illustrates the accomplished characteristics of the best-known multirobot coordination architectures developed to date (Note that only publications pertaining to the design of a complete multirobot coordination architecture were evaluated in this matrix):

§ **Robustness:**	**Robust to robot failure; no single point of failure for the system**
§ **Speed:**	**Quick response to dynamic conditions**
§ **Efficiency:**	**Opportunistically optimized response to dynamic conditions**
§ **Information:**	**Ability to execute a task without complete/perfect information about the task or the environment**
§ **Communication:**	**Ability to deal with limited and imperfect communication**
§ **Resources:**	**Ability to reason about limited resources**
§ **Allocation:**	**Efficient allocation of tasks**
§ **Roles:**	**Efficient adoption of roles**
§ **New Input:**	**Ability to dynamically handle new instructions from a human operator**
§ **Fluidity:**	**Easily able to accommodate the addition/subtraction of robots during operation**
§ **Heterogeneity:**	**Ability to accommodate heterogeneous teams of robots**
§ **Extensibility:**	**Easily extendable to accommodate new functionality**
§ **Flexibility:**	**Easily adaptable for different applications**
§ **Tight-Coordination**	**Ability to coordinate robots in maneuvers that require tight-coordination**
§ **Scalability**	**Ability to coordinate large numbers of robots**
§ **Learning:**	**On-line adaptation for specific applications**
§ **Implementation:**	**Implemented and proven on robotic system (N- no implementation, S - simulation or software agents, P – partially implemented on a robotic system, Y - fully implemented on a robotic system)**

N = Not accomplished, **P** = Partially accomplished, **Y** = Fully accomplished, **F** = Five robots or less, **T** = Tens of robots, **H** = Hundreds of robots

Reference #	**Robustness**	**Speed**	**Efficient**	**Information**	**Communication**	**Resources**	**Allocation**	**Roles**	**New Input**	**Fluidity**	**Heterogeneity**	**Extensibility**	**Flexibility**	**Tight-Coordination**	**Scalability**	**Learning**	**Implementation**
[1]	Y	Y	N	Y	Y	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
[5]	Y	Y	N	Y	Y	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
[12]	N	N	N	N	Y	Y	Y	Y	N	N	P	P	N	Y	Y	N	Y
[15]	Y	Y	Y	P	P	N	Y	N	N	N	Y	Y	P	N	Y	N	S
[21]	N	N	Y	Y	N	N	Y	N	Y	P	N	P	N	N	N	N	S
[22]	N	N	Y	Y	N	N	Y	N	N	P	P	P	N	N	N	N	Y
[23]	P	P	P	Y	Y	P	P	N	P	Y	Y	Y	Y	N	Y	N	P
[26]	P	Y	P	Y	P	N	P	P	N	P	Y	P	P	Y	Y	N	Y
[28]	P	P	P	P	Y	N	P	N	N	N	N	N	N	Y	N	N	N
[36]	N	Y	P	P	Y	N	N	N	N	N	N	N	N	Y	Y	N	S
[45]	P	P	P	P	Y	N	Y	Y	Y	P	Y	P	Y	P	Y	N	S
[49]	Y	Y	P	Y	P	Y	P	P	P	Y	Y	Y	Y	Y	Y	N	Y
[56]	P	N	P	Y	Y	N	N	N	N	P	Y	P	P	P	Y	N	S
[57]	P	P	P	P	Y	N	N	N	P	N	Y	Y	P	P	Y	N	S
[61]	P	N	N	Y	Y	N	N	N	N	P	P	N	P	P	Y	N	S
[63]	Y	Y	P	Y	Y	N	N	N	P	Y	Y	Y	P	Y	Y	N	Y
[65]	N	P	Y	Y	N	N	P	N	N	P	Y	Y	Y	N	N	N	S
[66]	P	P	P	Y	P	P	P	N	N	P	Y	Y	Y	Y	Y	N	S
[71]	P	Y	Y	Y	Y	P	Y	P	P	Y	Y	Y	P	Y	P	N	Y
[73]	N	Y	N	Y	Y	N	N	N	N	P	N	N	N	N	Y	N	S
[78]	P	Y	P	N	Y	P	P	N	P	P	Y	Y	Y	Y	N	N	Y
[79]	N	P	N	Y	Y	N	P	Y	N	N	P	P	P	Y	Y	N	S
[80]	P	Y	N	Y	Y	N	N	N	N	N	P	P	P	P	Y	N	S
[83]	Y	P	P	Y	Y	N	P	P	P	Y	Y	Y	Y	Y	Y	Y	Y
[102]	Y	P	P	Y	Y	N	P	N	P	Y	P	P	N	P	Y	N	Y
[104]	N	N	N	N	N	N	Y	N	P	N	Y	P	P	Y	N	N	Y
[105]	Y	Y	P	Y	Y	P	Y	N	N	Y	P	Y	P	Y	N	N	P
[115]	N	P	P	N	Y	N	Y	N	N	N	N	N	N	N	N	N	S
[117]	Y	Y	Y	Y	P	N	Y	P	P	Y	Y	Y	Y	Y	P	P	P
[119]	Y	Y	Y	Y	P	N	Y	P	P	Y	Y	Y	Y	Y	Y	Y	P
[123]	Y	Y	P	P	Y	N	P	N	N	N	P	N	N	P	Y	N	Y

Table 3: The coverage of characteristics required by multirobot applications by multirobot coordination architectures designed to date

2.8 Multirobot Coordination

A closer examination of the architectures that satisfied many of the desired characteristics is useful. Note that many others (for example [44]) have published coordination schemes geared towards specific domains such as the soccer domain – these are not explored in detail here.

Arkin [4] provides a biologically inspired architecture where individual agents make reactive decisions based on motor-schema. Introducing cooperation through recruitment behaviors enhances the resulting fortuitous group-behavior. This eliminates the necessity for direct communication between agents. However, this system is incapable of optimizing its response to dynamic conditions. The agents also do not explicitly reason about optimal use of their limited resources, optimal adoption of roles, and optimal task distribution among the group. Furthermore, the group of agents cannot easily accommodate new tasks, resources, and roles assigned to the group. Finally, this work does not include implementation results from a robotic system – results are all in simulation.

Caloud et al. [23] introduce another successful architecture, "GOPHER". The architecture is broken down into four layers: task decomposition, task allocation, motion planning, and execution. The task allocation process is partially centralized by involving a central task processing system (CTPS) which announces tasks to all available robots within communication range. There is no apparent plan however to recover from failure or malfunction in the CTPS. Moreover, although robots make bids for different tasks offered by the CTPS, it is not clear how these bids are formed, or on what basis the costs are computed. Furthermore, any robot in the midst of executing a task will not participate in any other transactions until it has completed its current tasks. These characteristics detract from optimal resource utilization and optimized reaction to dynamic conditions. Also, no explicit reasoning about optimal utilization of limited resources and corresponding adoption of roles is evident. Finally, the implementation of the architecture is at preliminary stages, and no method of on-line adaptation for specific applications is discussed.

Chaimowicz et al. [26] present an architecture for tightly coupled multirobot coordination. The architecture is based on the concept of at least one leader being identified at any give time within a group, and the others being followers. Some of the followers can be leaders for other agents in the group – so a hierarchy could exist. (In the implementation only a single leader was present at any time). Leadership can be changed by request and by resignation. Conflicts are resolved using a priority-based approach. If any deadlock is detected, command is relinquished to a human operator. No discussion is presented about what would happen if a leader is disabled before it can resign. Thus, the system is not fully robust. Also, since a robot could be assigned as a leader, even though it isn't optimally placed to become a leader, due to the current leader's resignation, the system will not always produce an optimal response to dynamic conditions. No consideration is provided about optimal management of resources. The roles are limited to leaders and followers. On-line optimization/adaptation schemes are not discussed, and extensibility, flexibility and fluidity of the system are limited.

Gerkey and Matari, ([49], [52]) present “MURDOCH”; a completely distributed, resource-centric, publish/subscribe communication model. A key feature in this approach is that all communication is resource-centric, and never name-based. Thus, the claim is that all messages are addressed in terms of the resources required to do the task. All tasks are allocated based on a single-round auction scheme. The auctioneer determines a winner and notifies the bidders. The winner is awarded a time-limited contract to complete the task. The auctioneer is responsible for monitoring the progress of the task execution. To do this, the auctioneer periodically sends contract renewal messages to the winner, which sends back corresponding acknowledgements. These messages will have to be addressed by name and will increase the communication requirements of the system since the auctioneer and winner will have to remain within communication range (or periodically return to positions within communication range) to renew the contract. Furthermore, since the auctioneer assumes a fault if a renewal message is not acknowledged and reassigns the task, several robots could attempt to complete the same task if acknowledgements are not received on time or some acknowledgement messages are lost. The instantaneous greedy task scheduling in MURDOCH does not allow for opportunistic optimization. Also, no discussion of on-line adaptation is presented.

Laengle et al. [71] introduce KAMARA, a distributed control-architecture for controlling the different components of a complex robot. This architecture could arguably be applied to the multirobot coordination problem. The architecture is based on the concept of generating an agent that represents each component of the robot. Each agent is composed of a communicator that handles communication, a head that is responsible for the planning and action selection, and a body which handles execution. Each body consists of one or more executive components which can be agents themselves. The agents can form teams to execute cooperative tasks when tight coordination is not necessary. If tight coordination is required, a special agent is generated which oversees the coordination of the agents involved by seizing control of these agent bodies from their heads. Task allocation occurs via negotiation where mediation is carried out by a selected agent in the candidate group for executing the task. Although disabled agents are compensated for because they cannot participate in the negotiation and hence cannot be assigned tasks, no method of ensuring completion of tasks assigned to the disabled agent is presented. Moreover, each agent is only able to execute a single task at a time. This can be highly sub-optimal in multirobot coordination because either resource utilization will be non-optimal, or an agent will be required to represent each resource on each robot which will drastically increase the required negotiation between agents. Also, the presented architecture has no method for on-line optimization.

Parker [83] designed ALLIANCE to be a fault-tolerant and adaptive multirobot coordination architecture. Essentially a behavior-based system, ALLIANCE has the added benefit of motivational behaviors such as impatience and acquiescence which prompt the robots to complete tasks when other robots fail to execute them, and to give up on tasks they cannot complete. While these motivational behaviors allow robot teams to be fault tolerant and adaptive, they do not enable the robots to respond to dynamic conditions in a quick and opportunistically optimal manner. Furthermore, the robots do not reason about the limited resources available to them and attempt to optimize utilization of these resources. No allocation of tasks is performed in this architecture. Instead, the high-level tasks are programmed into the behavior sets of the robots. This scheme does not promote optimized task allocation, and doesn't allow new types of tasks to be dynamically assigned. Parameterized learning methods are explored for on-line optimization. In Parker's discussion of other multirobot coordination schemes, her main argument against negotiation schemes is that they have not been proven on robotic systems. The proposed work for this Ph.D. thesis aims to overcome Parker's challenge.

"Teamcore", developed by Tambe et al. [119] is a most successful architecture included in Table 3 in terms of satisfying the largest number of the identified criteria. The general idea of this approach is to provide the architecture with general-purpose teamwork coordination capabilities and to adapt these capabilities for specific applications via machine learning. The architecture is formed by generating Teamcore proxies to represent each participant in the group (the participants can be robots, humans, or software agents). Each proxy is equipped with a general teamwork model, STEAM [117], which automates its coordination with other proxies on its team. STEAM is based on a hierarchical reactive plan architecture, with teamwork capabilities inserted as reactive team plans which instantiate joint intentions [64] for the team. The proxies can then adapt at four different levels: autonomy, execution, monitoring, and information requirements. While this system is highly adaptable and successful in satisfying many of the criteria in Table 3, it lacks the ability to reason about and optimize the use of limited resources available to the robots. Further, it has not been implemented and proven on a robotic system.

In summary, although many multirobot coordination approaches have been implemented on robotic systems with varying levels of success, no approach that satisfies all of the criteria for efficient multirobot coordination in dynamic domains has been designed and implemented to date. Mainly lacking are approaches that are capable of efficient and robust coordination of teams of robots in dynamic environments. Furthermore, no in-depth analysis has been carried out to investigate the applicability of economic methods to the multirobot coordination problem.

Stentz and Dias [111] first envisioned the TraderBots approach in 1999. This was the first detailed investigation of the concept of using a market approach to coordinate multiple robots to cooperatively complete a task. This work builds on the contract net protocol by Smith [107], its extension by Sandholm and Lesser [98], and the general concepts of market-aware agents developed by Wellman and Wurman [126]. This work investigated the methodology of applying market mechanisms to intra-team robot coordination (i.e. in typically non-competitive environments) as opposed to competitive multirobot domains and competitive inter-agent interactions in domains such as E-commerce. Simulation results using this approach were produced by Dias and Stentz [39] and by Thayer et al. [120], and robot results were presented by Zlot et al. [131]. Dias and Stentz [40] further developed the TraderBots methodology and started a detailed investigation of the multirobot coordination problem in dynamic environments, which is completed in this dissertation. Numerous further enhancements of the TraderBots approach have been investigated and implemented in simulation and on robots ([37], [41], [43], [55], [106]) culminating in the most sophisticated implementation to date of the approach on robots described in a recent publication by Dias et al. [43]. Further extensions of the TraderBots approach are being investigated by Zlot and Stentz ([128],

[129], [130]) and Kalra and Stentz [67]. Details of the TraderBots approach are presented in the next chapter.

CHAPTER 3

TraderBots Approach

IT is now instructive to examine in detail the TraderBots approach presented in this dissertation. TraderBots is a coordination mechanism designed to inherit the efficacy and flexibility of a market economy, and to exploit these benefits to enable robust and efficient multirobot coordination in dynamic environments. Consider a team of robots assembled to perform a set of tasks in a dynamic environment. Consider further, that the team of robots is modeled as an economy, and each robot in the team is modeled as a self-interested trader in this economy. Thus, the goal of the team is to complete the tasks successfully while maximizing overall profits (i.e. the difference between revenue and cost), while each robot will aim to maximize its individual profit. Thus, robots will hold auctions and submit bids to determine task allocations within the team, and the different tasks and information will be the commodities traded in the economy. A system such as this will inherit many desirable characteristics from the market methodology. The competitive element of the robots bidding for different tasks enables the system to decipher the competing local information of each robot (note that there is no central agent evaluating information and planning for the entire system), while the common currency provides grounding for the competing local costs in relation to the global priorities/value of the tasks being performed. Presented next is a more detailed examination of these characteristics.

An economy is essentially a population of agents (i.e., citizens) producing a global output. The agents coordinate with each other to produce an aggregate set of goods. Centralized economies suffer from an inability to gather all salient information, uncertainty in how to optimize with gathered information, and sluggish responsiveness to changing conditions. Additionally, if economic output is divided equally amongst the entire population, individuals have little incentive to work harder or more efficiently than what is required to minimally comply with the economic plan. Individual input is de-coupled from individual output. The net effect is a sluggish, brittle, and inefficient economy.

Market economies are generally unencumbered by centralized planning; instead, individuals are free to exchange goods and services and enter into contracts as they see fit. Despite the fact that individuals in the economy act primarily to advance their own self-interests, the aggregate effect is a highly productive society. Individuals are often in the best position to understand their needs and the means to satisfy them. Thus, individuals reap the direct benefits of their own good decisions and suffer the direct consequences of their bad ones. At times they cooperate with other members of the society to achieve an outcome greater than that possible by each member alone. At times they compete with other members to provide goods or services at the lowest possible cost, thus eliminating waste and inefficiency. But at every turn, the individual members act to reap the greatest profit for themselves. This dissertation presents a method for applying these powerful techniques to the task of coordinating a team of robots in dynamic environments. An important aspect of the TraderBots approach is that the robots are self-interested. Note however that the robots are only self-interested within the domain of capabilities allowed to them. Since we design the system, we are able to influence the actions and capabilities of the robots/traders to a large extent. For example, robotic dishonesty is not allowed in order to simplify architectural component design and overall system complexity.

While other approaches have adopted cooperative models for robot coordination, it is not clear that cooperative motivations in general have any significant advantage over the presented approach. Designing generic incentives to motivate cooperation in multirobot applications is not trivial, although such schemes could be better suited for some applications. Furthermore, it is not clear that cooperative incentives would not lead to interference between robots for some applications. While designing appropriate cost and revenue models can also be non-trivial for some applications, in many multirobot application domains the TraderBots approach enables efficient and robust coordination by providing a mechanism that inherits the flexibility and many benefits of market mechanisms as described next.

3.1 Cost, Revenue, and Profit

Costs and revenues dictate to a large extent the performance of a market-based approach. A function, **trev**, is needed to map possible task outcomes onto revenue values. Another function, **tcost**, is needed that maps possible schemes for performing the task onto cost values. As a team, the goal is to execute some plan **P** such that profit, **trev(P) – tcost(P)**, is maximized. Note that **trev** and **tcost** are not always simple functions – they can potentially be complex functions involving statistical distributions and vectors of different components (for example, the cost function can be a combination of time spent, fuel expended, and CPU cycles utilized, and the revenue function can be a combination of the priority placed on the task by the operator, the risk involved with executing the task, and the extent to which specialized resources are required for task execution). The cost and revenue functions are designed to reflect the nature of the application domain. Thus, these functions need to reflect characteristics important to the domain such as priorities for task completion, hard deadlines for relevant tasks, and acceptable margins of error for different tasks.
But it is not sufficient to define just the revenue and cost functions for the team. These functions must provide a means for distributing team revenue and assessing team costs to individual robots. Preferably, these individual revenues and costs are assigned based on factors over which the individuals have direct control. For example, if the task is to find and retrieve a set of objects, the team's revenue, **trev**, could be the number of objects retrieved (converted to a "cash" value), and the team's cost, **tcost**, could be the amount of energy consumed by the entire team to find the objects. The individual robot revenues and costs, **rrev** and **rcost**, could be the cash value of the number of objects turned in, and the energy expended, respectively, by *that* individual robot. Therefore, in this case, the sum of the individual revenues and costs equals the team's revenues and costs. However, the distribution is not even: individuals are compensated in accordance with their contribution to the overall task, based on factors that are largely within the control of the individual. An important point to note is that each robot does not estimate costs based solely on resource consumption (for example fuel spent or time taken) but can also

include estimates of opportunity cost in this computation. The opportunity cost can take into account factors such as what other opportunities are available to the robot, what competition it faces for winning a given task, and what special capabilities the robot possesses (for example the team may only have one robot with a gripper which makes that robot's time more valuable and that robot should give preference to gripper-related tasks unless there is a pressing need for it to execute a task that can be executed by another robot). An individual that maximizes its own personal production and minimizes its own personal cost receives a larger share of the overall profit. Note that the revenue/cost models and rules of engagement are designed so that maximizing individual profit has the benevolent effect of moving the team toward the globally optimal solution. This is a crucial element to prevent unfavorable interactions between individual and team profit maximization. Therefore, by acting strictly in their own self-interests, individual robots maximize not only their own profit but also the overall profit of the team.

It is instructive to note that since costs and revenues drive its design, the TraderBots approach can be easily applied to many different applications. Another important feature of this approach is its ability to accommodate several revenue providers. That is, a number of operators could connect to a single group of robots and offer the robots revenue in return for different services. This also prevents the revenue provider from becoming a single point of failure for the system. If for some reason the connection from the operator to the team of robots is disabled (a likely scenario in application domains such as urban reconnaissance), a different connection can be established if necessary, but any tasks (whose completion don't require the connection) conveyed to the robot team before the communication failure can be completed even if the connection cannot be re-established.

3.2 Bidding, Price, and Negotiation

Robots receive revenue and incur costs for accomplishing a specific team task, but the team's revenue function is not the only source of income. A robot can also receive revenue from another robot in exchange for goods or services. For example, a robot may not be equipped to find objects for which the team function provides revenue, but it can transport the objects to the goal once they have been found. Therefore, this haulage robot provides a service to the robots that find the objects, and it receives payment for performing such a service.

In general, two robots have incentive to deal with each other cooperatively if they can produce more aggregate profit together than apart—such outcomes are win-win rather than zero-sum. The *price* dictates the payment amount for the good or service. Determining this price is an important aspect of the TraderBots approach. Assume that robot **A** would like to purchase a service from robot **B**. Robot **B** incurs a cost **Y** for performing the service. Robot **A** can make additional revenue of **X** if **B** performs the service for it. Therefore, if **X > Y**, both parties have an incentive to execute the deal. But how should the composite profit, **X - Y**, be divided amongst the two parties? It may sound fair to split the winnings **(X - Y) / 2** by setting the price at **(X + Y) / 2**. This is certainly a possibility. But robots **A** and **B** may have other opportunities—they may be considering other deals that contend for the same money and resources. Since these factors may be hidden or complex, a common approach is to *bid* for a good or service until a mutually acceptable price is found. For example, robot **A** could start by bidding a price of **Y** (i.e., robot **A** receives the entire profit). Robot **B** could decline and counter with a bid of **X** (i.e., robot **B** receives the entire profit). The idea is to start by bidding a price that is personally most favorable, and then successively retreat from this position until a price is mutually agreed upon. Note that a given robot can negotiate several potential deals at the same time. It begins by bidding the most favorable price for itself for all of the deals, successively retreats from this position with counter bids, and closes the first deal that is mutually acceptable. The deal will be mutually acceptable only if it results in increased profits for both robots. Thus, the deal will move the global solution towards its optimal.

Note also that a deal can be multi-party, requiring that all parties agree before any part of the deal is binding. Since multiple rounds of negotiations could be time-consuming, time restrictions will have to be imposed on all bidding. These time restrictions can vary from bid to bid, and between application domains. Based on basic economic intuition, the negotiated price will tend toward the intersection of the supply and demand curves for a given service. However, several rounds of bidding can still be too time-consuming for some applications – especially in dynamic environments. Hence, another possibility is to hold single-round auctions where each robot bids a price that is higher than its estimated cost for completing the task (this could include opportunity cost calculations) and low enough to be competitive by its estimates.

If a service is in high demand or short supply, we can expect that the negotiated price will be high. This information will prompt other suppliers to enter the fray, driving the price down. Likewise, if demand is low or supply high, the low price will drive suppliers into another line of business. Thus, price serves to match supply to demand. Note that in an efficient market with perfect information, the price will optimize the matching of supply and demand. Finally, it is important to note that price and bidding are low bandwidth mechanisms for communicating aggregate information about costs. When consumers decide between purchasing apple juice or orange juice for breakfast, they do not analyze land acreage dedicated to the two crops, the costs of producing each, the demand for each, and the impact of weather and pest infestations. Instead, they merely look at the price of each and weigh them against their own personal preferences. The price, however, *encodes* all of these factors in a concise fashion that enables them to make a locally optimal decision based on low-bandwidth information available at the point of sale.

3.3 Cooperation and Competition

As described in the previous section, robots interact with each other to exchange goods and services. Two robots are *cooperative* if they have complementary roles; that is, if both robots can make more profit by working together than by working individually. Generally, robot teams foster cooperation between members of different types (heterogeneous). For instance, a robot able to grasp and lift objects and a robot able to transport objects could team together to provide a pick-and-place service that neither one could offer independently.

Conversely, two robots are *competitive* if they have the same role; that is, if the amount of profit that one can make is negatively affected by the presence of the other robot. Generally, robot teams foster competition amongst members of the same type (homogeneous). For instance, two robots that are able to transport objects compete for the services of a given grasping robot, thus driving the price down. Either one could charge more money if the other were not present.

These delineations are not strict however. Subgroups of heterogeneous robots could form that provide a given service. These subgroups would compete with each other, thus providing an example where robots of different types compete rather than cooperate with each other. Heterogeneous robots could also compete if the same task can be accomplished in different ways. Conversely, two robots of the same type may cooperate by agreeing to segment the market. Homogeneous robots can also cooperate if accomplishing a specific task requires more than one robot. For example, several robots with grasping capability may need to cooperate in order to move a heavy object. The flexibility of the market-model allows the robots to cooperate and compete as necessary to accomplish different tasks, regardless of the homogeneity or heterogeneity of the team.

3.4 Self Organization

Conspicuously absent from the market is a rigid, top-down hierarchy. Instead, the robots organize themselves in a way that is mutually beneficial. Since the aggregate profit amassed by the individuals is directly tied to the success of the task, this self-organization yields the best results.

Consider a group of ten robots. An eleventh robot, **A**, offers its services as their leader. It does not become their leader by coercion or decree, but by convincing the group that they will make more money by following its plans than by acting individually or in subgroups. **A** does this by investigating plans for utilizing all ten robots. If **A** comes up with a truly good plan, it will maximize profit across the whole group. The prospective leader can use this large profit to bid for the services of the group members, and of course, retain a portion of the profit for itself. Note that all relevant robots will have to commit to the plan before it can be sold. The leader may be bidding not only against the individuals' plans, but also against group plans produced by other prospective leaders. Note that the leader acts both as a benevolent agent and a self-interested agent; it receives personal compensation for efforts benefiting the entire group.

But there is a limit to this organization. As the group becomes larger, the combinatorics become intractable and gathering all of the relevant information to produce a good plan becomes increasingly difficult. A leader will realize this when it can no longer convince its subjects (via bidding for their services) to follow its plans. The process of building coalitions/sub groups and enabling leaders to do an efficient job can be a complex endeavor. These topics are explored in more detail in Chapter 5. Fully centralized and fully distributed reactive approaches are two extremes along a continuum of solutions to the multirobot coordination problem. The introduction of leaders and coalitions allows the TraderBots approach to slide along this continuum in the direction of improved efficiency in an opportunistic manner.

3.5 Learning and Adaptation

The robot economy can learn new behaviors and strategies as it executes its tasks. This learning applies to both individual behaviors and negotiations as well as to those of the entire team. Individual robots may learn that certain strategies are not profitable, or that certain robots are apt to break a contract by failing to deliver the goods or proper payment. Individuals may also learn successful bidding strategies or which deals to offer when. The robot team may learn that certain types of robots are in over-supply, indicated by widespread bankruptcy or an inability to make much money. Conversely, the robot team may learn that certain types of robots are in under-supply, evidenced by excessive profits captured by members of the type. Thus, the population can learn to exit members of one type and enter members of another. Moreover, in this approach, successful agents are able to accumulate wealth and perpetuate their winning strategies because of their ability to offer higher payments to other agents. The different ways in which learning can benefit the TraderBots approach are explored in greater detail in Chapter 6.

One of the greatest strengths of the TraderBots approach is its ability to deal successfully with dynamic conditions. Since a market economy does not rely on a hierarchical structure for coordination and task assignment, the approach is highly robust to changes in the environment, including malfunctioning robots. Disabling any single robot does not jeopardize the system's performance. By adding escape clauses for "broken deals", any tasks undertaken by a robot that malfunctions can be re-bid to other robots, and the entire task can be accomplished. Also, escape clauses will provide the robots with further flexibility to adapt to dynamic conditions and accept more attractive offers that are generated, although some caution is necessary here to maintain stability and prevent cycles of broken deals. Thus, the TraderBots model allows the robots to deal robustly with dynamic environments in an opportunistic and adaptive manner. A more detailed examination of how the TraderBots approach is designed to ensure robustness is presented in Chapter 4.

Due to its distributed nature, the TraderBots approach can respond quickly (within the time constraints placed on bidding) to dynamic conditions. An added benefit with this approach is that it can also respond efficiently in terms of specified cost-criteria to dynamic conditions. The challenges encountered and solutions proposed for encouraging higher levels of efficiency in the design and implementation of TraderBots are explored in greater detail in Chapter 5. Finally, a key factor for efficiency and flexibility in the TraderBots approach is the agents' ability to efficiently manage their resource utilization and role adoption. This approach allows for efficient resource and role management, because each agent is driven to make individual-rational decisions at any given time and because resource and role constraints are taken into account during cost estimation before making bids. Furthermore, no artificial limits are placed on the robots in terms of how many roles they can play or how many resources they can use at any given time.

The TraderBots approach can be further illustrated via a couple of examples of its application in multirobot task domains. Two such examples are presented next.

3.6 Illustrative Example I

The first chosen example is that of a warehouse with automated pallet jacks that work in conjunctions with humans to pick and package orders; a less dynamic application domain where some control over the environment is possible. Controlling multiple automated pallet jacks in a warehouse is a challenging task. The underlying challenge is to provide efficient scheduling and coordination of the pallet jacks to maximize throughput and minimize labor costs. This involves maximizing the time spent by human workers picking orders, the operation most difficult to automate, and minimizing their unproductive travel time. Thus, the workers are dedicated to specific storage areas in the warehouse and the pallet jacks drive themselves from worker to worker in the necessary sequence to fill orders.

If the TraderBots approach is applied to the automated warehouse domain, agents representing the pallet jacks (PJ-traders) will trade to win order filling, delivery, and other tasks the pallet jacks are capable of performing. The PJ-traders will negotiate with agents representing restrictive areas on their routes (toll-traders) so that they are able to make cost-effective bids and reach their goals with minimal delays. The toll-traders will be responsible for controlling traffic in these areas and thereby preventing collisions, bottlenecks, and delays. Agents representing human workers (HW-traders) will negotiate on their behalf for loading, sorting, and other such tasks that will maximize the efficiency and utility of the laborers. Finally, agents representing the Warehouse Management System (WMS) will be responsible for announcing orders issued by the WMS and orchestrating efficient execution of other instructions from the WMS.

Each agent can choose to play one or more complementary roles at any point in time if the agent is managing the set of resources required by those roles. Maximizing individual profit will be each agent's motivation for choosing whether or not to adopt a particular role. The following roles will correspond to the duties of the different agents specified above:

§ **Loader – loading pallets (needs to be done by a human)**
§ **Transporter – transporting pallets (can be done by a human or a pallet jack)**
§ **Order Filler – efficiently filling an order (can be done by any agent with sufficient computing capability)**
§ **Leader – computing efficient plans for a group of agents (can be done by any agent with sufficient computing capability)**
§ **Computer – utilizing computing resources for evaluation of a problem (can be done by any agent with sufficient computing capability)**
§ **Traffic Cop – monitoring a given area to prevent collisions and traffic congestion (can be done by any agent with sufficient computing capability)**

Let's assume that only pallet jack agents and WMS agents manage sufficient computing resources to adopt multiple roles. The agents representing laborers and restrictive areas don't have sufficient computing resources to manage more than a single role. In general, an agent will be able to assume more roles if it has access to more resources. Agents orchestrate efficient planning and scheduling by switching between these roles as necessary. For example, an agent assuming the Order Filler role can plan a sequence of routes through the warehouse assuming that there is no contention for resources (i.e., human loaders, intersections, one-way routes etc.). If it can't "buy" everything it needs, it will need to modify its plans. On the other hand, if it can get what it needs but the prices are high, then that is an opportunity to assume the role of a Coordinator. As a Coordinator it can coordinate Order Filler plans with Loader and Transporter plans to produce a globally more efficient solution and sell the results to the participants. The more computing time/resources available, the more complicated that reasoning could be, involving many agents and resulting in even greater global efficiency. The system is thus an "anytime" system, producing good results that get better if time permits. Examined next is a more specific scenario in the automated warehouse domain.

Consider the case of two automated pallet jacks (J1 and J2) working in a warehouse. The warehouse also employs two human laborers (H1 and H2) for picking orders, and has a Warehouse Management System (W) that oversees order picking. H1 is assigned to pick orders from a storage location (S1) that contains tomatoes and cabbages, while H2 is assigned to a storage area (S2) containing tomatoes, eggs, and potatoes. Two intersections (I1 and I2) and one passageway (P) in the warehouse only allow for a single pallet jack to pass though at a time. In this scenario the above warehouse components would be represented by the agents j1, j2, h1, h2, w, i1, i2, and p respectively. Assume all costs are time-based. For example, the cost for traveling from a start position to a goal is directly proportional to the time it takes to move from start to goal. For simplicity, let as assume the WMS designates sufficiently high, equal rewards for all orders. Note that in reality, the reward offered for each assignment can vary depending on factors such as the urgency of the order, the size and complexity of the order, and the relationship with the customer. Also assume that the agents determine the bid price as a percentage of the difference between the offered reward and the cost. Let the cost of loading one unit of any of the produce also be fixed. Figure 5 below shows the layout of a simplistic warehouse as described above, the costs of all possible routes, the locations of the key components in the warehouse, and the different items stored in the storage areas. In this scenario the agents j1 and j2 reside on the corresponding pallet jacks, the agents h1 and h2 reside in the storage areas S1 and S2 respectively, the agents i1, i2, and p reside on computers with access to surveillance capabilities in I1, I2, and P, and the agent w resides on the central computing system which supports W. Let us further assume that communication is possible at all times between all agents, either point-to-point or routed through a central computer.

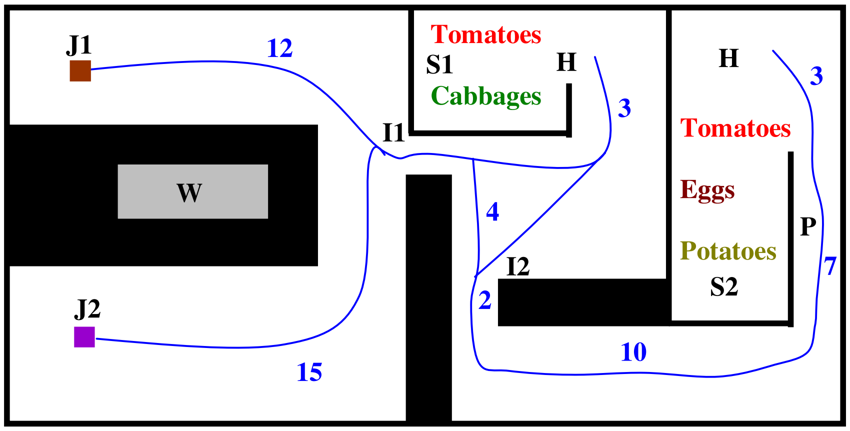


Figure 5: Illustration of warehouse setting

Now consider the case where **W** receives an order for cabbages and tomatoes. Agent **w** will play the role of order filler and auction the order transportation to **j1** and **j2**. Agents **j1** and **j2** will estimate their costs to carry out the order and choose the route which allows them to fill the order with the lowest cost. Thus, **j1** and **j2** will both choose the route to **S1** and back which will cost each of them **42** and **48** respectively. Based on this estimate, plus the estimate payment to get the pallets loaded, **j1** is able to make a lower bid and hence will win the assignment from **w** to play the role of transporter. Once the assignment is won, **w**, based on the route chosen by **j1**, will negotiate with **i1** for the right of passage through **I1** during the estimated required time intervals, and also with **h1** for the services of **H1** to pick and load the appropriate units of produce into the pallet. Assuming no conflicts, **i1** will charge the standard cost, **2**, for each pass through **I1**, and **h1** will charge the standard cost per unit of produce loaded into the pallet. Then **i1** and **h1** will reserve the relevant time slots in their schedules while playing the roles of traffic cop and loader respectively. Thus, an efficient solution will be produced.

However, this is a simplistic scenario. Consider the slightly more complex scenario where the order is for eggs and tomatoes. Once again, **j1** will win the assignment, but this time will choose to deal entirely with **S2**. But what happens if **H2** is currently occupied and hence causes a delay? This will translate to a greater cost for **J1** to complete the order. If the increase in cost is sufficiently high (the delay is sufficiently long), **j1** will choose the alternate route of first heading to **S1** for the tomatoes and then to **S2** for the eggs. Thus, the added cost of traveling to **S1** is offset by the lower delay at **S2** and hence the overall cost of completing the order is minimized; that is, efficiency is achieved. Other possible scenarios include j2 being able to fulfill part of the order due to its current schedule overlapping with that of j1's schedule – for example, if j1 was filling an order at S1 and had to travel to S2 for its next order, it could carry the tomato portion of j1's order to S2 with it which would then only need to be transferred to J1 at S2. This would of course be done for a price paid by j1 to j2. Note that in a more sophisticated system, an idle agent assuming the role of leader could realize through monitoring that a large number of orders requiring stops at **S2** are queuing up and produce a plan where **H1** moves to **S2** temporarily. If this plan results in greater profits for all involved parties, the leader agent will be able to easily sell the plan to the participating agents thus improving efficiency in the warehouse operations. Many other complex scenarios can be explored in this domain; the more dynamic the scenario, the more creative the agents will need to be in their trading and planning in order to generate efficient solutions for filling the arriving orders. However, many aspects of the automated warehouse domain are controllable, and it is likely that the daily stream of orders will be somewhat predictable and hence prior planning and availability of the necessary resources will help facilitate efficient solutions for most of the order filling without the need for many optimization techniques. The next example explores the application of TraderBots in the remote operations domain, which is more dynamic and less controllable.

3.7 Illustrative Example II

In this example we explore a scenario in the remote operations domain. Domains in which complex tasks must be performed in environments that are largely inaccessible to humans motivate this scenario. The applicability of the TraderBots approach to this scenario is explored in detail by Dias et al. [41] in a previous publication. Robots, largely autonomously, performing the required tasks becomes necessary because the environment is largely inaccessible to humans. From within this domain, we choose the scenario of a team of robots maintaining and operating a Martian outpost. For this scenario (which can include tasks such as exploration, habitat construction, space facility construction, and maintenance operations), the tasks can be performed safer, better, faster, and cheaper using teams of heterogeneous robots similar to the manner in which our society utilizes workforces consisting of specialists in diverse fields to perform similar tasks.

For the foreseeable future, mobile robots will serve as the remote sensors and data collectors for scientists. To create an outpost for such long-term exploration, the robots need to assemble solar power generation stations, map sites and collect science data, and communicate with Earth on a regular basis. In this scenario on the order of ten robots are sent many with different capabilities. Some of the robots specialize in heavy moving and lifting, some in science data collection, some in drilling and coring, and some in communication. The rovers have different, but overlapping, capabilities – different sensors, different resolutions and fields of view, even different mobility, such as wheeled and aerial vehicles. Figure 6 is a NASA artist's depiction of such a scenario.

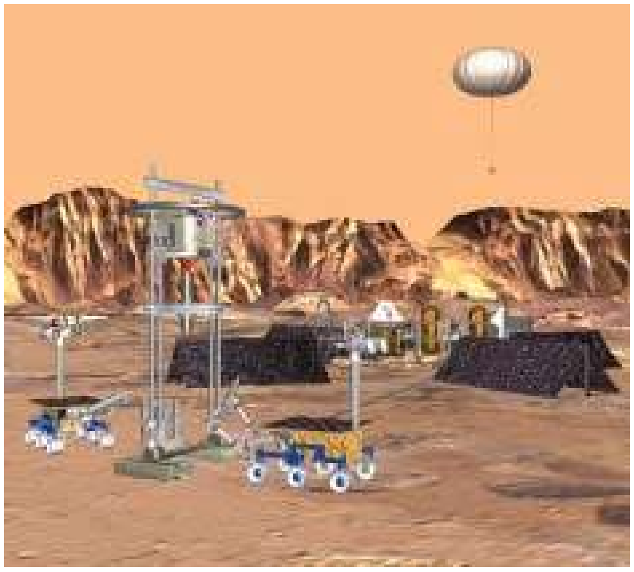


Figure 6: Conceptual Illustration of a Multirobot Martian Outpost (Illustration produced by courtesy of Jet Propulsion Laboratory)

The rovers cooperatively search for a location suitable in size and terrain for a base station. Once such a location is found, rovers with appropriate capabilities form several teams to construct the base station capable of housing supplies and generating energy. Two rovers carry parts, such as solar panels, that are too large for a single rover. Complementary capabilities are exploited – for example, to align and fasten trusses rovers with manipulators receive assistance from camera-bearing rovers that position themselves for advantageous viewing angles.

Meanwhile, other rovers begin general exploration of the region. To start, several scouting robots (perhaps joined by aerial vehicles) quickly survey the region. Scientists on Earth (and perhaps the rovers themselves) identify sites within the region that have high likelihood to contain interesting science data. Rovers with specialized sensing instruments are sent to investigate. If a particular subtask requires more intensive scrutiny, additional rovers with appropriate capabilities are brought in. For instance, to perform a seismographic test, one rover could transmit pulses into the ground while other rovers further away receive the reflected signals. Meanwhile, several rovers move to high ground to provide an inter-robot communications network. The exploration teams dynamically reconfigure depending on what is found and which rovers are available to help.

Rover failures are addressed by dispatching a rover with diagnostic capabilities. The diagnostic rover can use its cameras to view the failed robot to see if it can be aided in the field (e.g., if it has a stuck wheel or is high-centered), or it may drag the rover back to the base station to be repaired by replacement of failed modules. In the meantime, another robot with the same (or similar) capabilities can be substituted, so as to complete the original task with minimal interruptions.

At any given time, different teams of rovers may be involved in exploration, base-station construction/maintenance, and rover diagnosis/repair. Many tasks will be time critical, requiring execution within hard deadlines (e.g., repair of a failed power generation station) or synchronization with external events (communication satellite visibility, periods of sunlight). The teams form dynamically, depending on the task, environment, and capabilities and availability of the various robots to best meet mission requirements over time. The rovers negotiate their individual roles, ensure safety of the group and themselves, and coordinate their actions, attempting as a group to avoid unnecessary travel time, to minimize reconfiguration and wait time, and to prefer more reliable alternatives in cases of overlapping capabilities. The challenge is to keep all the robots healthy and busy in appropriate tasks in order to maximize the scientific data collected.

Similar scenarios exist for domains such as habitat construction and in-space facility construction and maintenance. For instance, consider an inspection robot that has identified a failed component on the Space Station. It tries to assemble a team of robots to replace the failed component. After negotiation, a courier robot (capable of retrieving the necessary replacement part) and a repair robot (capable of swapping-out the failed device) take responsibility for the repair task, leaving the inspection robot free to continue inspection. While the courier collects the replacement part, the repair robot evaluates the problem and plans its course of action, possibly seeking additional aid if it encounters unexpected difficulties it is unable to resolve. Upon arrival with the replacement part, the courier and repair robot coordinate to complete the task.

While these descriptions demonstrate the highly dynamic nature of the remote operations domain, it is more instructive to explore a more simplistic example in this domain. Thus, we explore the scenario of applying the TraderBots approach to a heterogeneous group of robots stationed at an extra-planetary outpost for scientific exploration of the region. The initial task assigned to the team is as follows:

Map a designated region (large-scale) and characterize any interesting sites as follows:

> *If at any site "X1", "X2", … or "Xn" is observed, perform the corresponding science experiment and if the experiment results in "Y1", "Y2", … or "Yn", analyze a sample and record the analysis.*

The scenario unfolds in four stages:

1. The robots able to climb slopes well will disperse to the highest points within sight and scout out the area at high level

2. Based on the findings of the first stage, small teams of scouts will deploy to explore designated areas in greater detail
3. Based on the findings of the second stage, designated sites will be further characterized by the performance of science experiments
4. Based on the findings of the third stage, sample returns to the processing facility at the base station will be recommended

Furthermore, communication with the earth will only be available within a specific window of time each day. During this window, the robot with the most powerful antenna available will be required to locate to a high point in order to be able to successfully communicate with operators on earth. The team of robots will only have a fixed data storage capacity, and hence have high incentive to offload the gathered data to earth each day. All robots will be able to communicate only within a limited range. Included in the team are the following robots:

Robots:	Resources:
R1, R2, R3	Radio, 2 cameras, 1 laser
R4, R5, R6	Radio, 1 camera, 1 manipulator, science instrument, fast computer
R7, R8, R9	Radio, tool-kit, 2 manipulators, 1 camera

These robots will be able to adopt the following roles if they have the necessary resources:

Roles:	Required Resources:
Communicator	Radio
Mapper	2 cameras + laser
Leader	Fast computer
Diagnoser/Fixer	1 camera, 2 manipulators, tool kit
Sampler/Scientist	1 camera, science instrument, 1 manipulator, fast computer

The execution of the scenario will include the following activities:

§ Exploration tasks are distributed among robots, and robots deploy, keeping in mind their maximum range for communication.
§ Optimization by leaders can happen with respect to positioning communicators and distribution of exploration tasks to robots.
§ New robots can enter the scene at any point during operations.
§ Whenever an "X" is detected, a capable robot has to perform the required science experiment and collect a sample if the result is a "Y".
§ Some robots can be disabled and its assignments will have to be re-distributed.
§ The disabled robots will also have to enlist the aid of a diagnoser/fixer robot.
§ Some disabilities will not be fixable, and creative solutions will emerge to complete the task. For example, one of the scientists could lose a camera and have to cooperate with a robot with a camera to do its experiments

The costs for this scenario will be based on fuel consumption and time. The commodity of value will be information (or scientific data). Each task is assigned a reward based on the value if the task being accomplished. Each robot houses a robot trader (RT) that participates in the TraderBots economy on behalf of the robot. The base station is built on a high location appropriate for communicating with earth and the main computer in the base station houses an operator trader (OT) that participates in the TraderBots economy on behalf of the earth-bound team of scientists.

During the first stage of the scenario, the OT auctions out scouting positions to be visited by robots capable of playing the mapper role to the group of RTs. Each scouting task is assigned a reward proportional to the estimated information to be gained from that particular position. RT1, RT2 and RT3 will bid for these tasks based on the estimated cost of executing the scouting tasks and thus the scouting tasks are distributed among the robots with the necessary resources for executing the task. Based on the findings of the scouts, new scouting tasks could be generated. For example, an area assigned for a scouting task can prove to be inaccessible and hence need to be removed from the list. Similarly, an area assigned as good for scouting can prove to be occluded by a previously unknown obstacle and hence become less useful. Conversely, the view from an assigned scouting location can reveal another location that can provide valuable information, and thus generate a new scouting task.

The operations in the second, third, and fourth stages depend to a large extent on the limits in computation and the range of communication. If communication range limits are sufficiently large to accommodate robots being in communication with each other and with the base station at all times (or most of the time), stages two, three, and four can occur simultaneously. That is, the exploration regions identified by OT in stage 1, based on the requirements of the scientists and the findings of R1, R2, and R3, are auctioned off to R1, R2, and R3 who play the mapper role and examine the specified targets searching for observations X1 through Xn. If any of these observations are made, a new task is generated to perform the necessary experiment. These experimentation tasks are auctioned out to R4, R5, and R6 who can play the sampler/scientist role and perform the necessary experiments, and if the experiment results in any of the specified Y1 through Yn results, they can return a sample to the base station. If communication range is limited but computation is not very limited, robots can be offered tasks that require them to play the role of communicator such that a link of communication is maintained between the OT and the robots at all times (or most of the time) even if each robot is not within the required communication range. If both communication range and computation is very limited, the latter three stages will more likely unfold as three separate stages rather than in parallel.

Note that many things can go wrong in the scenario described above. Robots could have malfunctions or get stuck and require the aid of R7, R8, or R9 acting as a diagnoser/fixer. This is accomplished by a diagnose/fix task being generated by the malfunctioning robot and auctioned off to R7, R8, or R9. Note that the robots with diagnose/fix capabilities can perform other tasks such as acting as a communicator and assisting with sample returns. However, these robots charge a higher price (taking into account their opportunity cost) for doing tasks other than diagnosis and fixing for which they are best suited since it is crucial that these robots remain available for dealing with malfunctions. If a robot is completely disabled a diagnoser will discover this and announce its death to the other robots and OT, who in turn will seek to redistribute tasks they had awarded to the dead robot. All gathered information is stored at the base station and the OT decides what information should be discarded if storage capacity is exceeded. The OT can also decide to cancel awarded tasks if storage capacity is close to being exceeded, and generate communication-to-earth tasks when the window of opportunity is present. The priority of communicating information to earth, and hence the reward offered for the task, will depend on the current state of events and requirements to communicate with operators on earth. Finally, any of the RTs or the OT can choose to play the role of a leader at any given time if they are idle or if they believe they can earn more profit by providing a new strategy for a group of robots executing tasks.

This example thus illustrates how the TraderBots approach can accommodate many of the requirements identified for successful multirobot coordination in dynamic environments. The means by which TraderBots satisfies each of the identified requirements is explored in more detail next.

3.8 Proof Of Concept

An initial version of the market-based architecture was developed and tested as a proof of concept for a distributed sensing task in a simulated interior environment[1]. A group of robots, located at different starting positions in a known simulated world, are assigned the task of visiting a set of pre-selected observation points for sensing tasks.

This problem is equivalent to a variation of the distributed traveling salesman problem (TSP), where the observation points are the cities to visit. The robot colony is structured as illustrated below (Figure 7 shows a snapshot of the robot colony at a given time and the double-headed arrows indicate communication channels).

Each robot is equipped with a map of the world, which enables it to calculate the cost associated with visiting each of these cities. The costs are the lengths of the shortest paths between cities in an eight-connected grid, interpreted as money. Let $\mathbf{c_{ij}}$ be the cost for the $\mathbf{j}^{th}$ robot to visit the $\mathbf{i}^{th}$ city from the ($\mathbf{i\text{-}1}^{th}$) city in its tour (where the $\mathbf{0}^{th}$ city is the starting location).
The robot cost function for the $\mathbf{j}^{th}$ robot is computed as follows:

$$\mathbf{rcost(j)} = \sum_{\mathbf{i=1}}^{\mathbf{n_j}} \mathbf{c_{ij}}$$

where $\mathbf{n_j}$ is the number of cities in the tour for robot $\mathbf{j}$. The team cost function is:

$$\mathbf{tcost} = \sum_{\mathbf{j=1}}^{\mathbf{m}} \mathbf{rcost(j)}$$

where **m** is the number of robots. The team revenue and robot revenue functions, **trev** and **rrev**, are determined by the negotiated prices. The maximum available team revenue is chosen to exceed team costs for reasonable solutions to the problem.

[1] This work was published in IAS-6 (Dias and Stentz [39]).

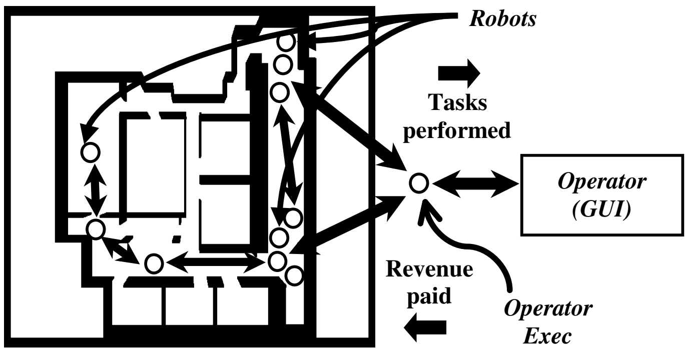


Figure 7: Organizational structure for robots engaged in distributed mapping

All robots (bidders) adopt the same simplistic strategy of bidding a fixed percentage of the maximum profit they can obtain. According to this strategy, if a task is on offer for a maximum price of **r**, and the cost to carry out the task is **c**, a robot computes its bid **b** as follows:

$$\mathbf{b = 0.9*(r-c) + c}$$

Thus, the robots bid for each city based on their estimated costs to visit that city.

The interface between the human operator and the team of robots is a software agent, the *operator trader (OpTrader).* The *OpTrader* conveys the operator's commands to the members of the team, manages the team revenue, monitors the team cost, and carries out the initial city assignments. Being a self-interested agent, the *OpTrader* aims to assign cities quickly while minimizing revenue flow to the team. In our initial implementation (Implementation 1 in Chapter 7), the *OpTrader* adopts the following greedy algorithm for assigning tasks:

- Announce all cities to all robots and wait for all incoming bids
- Insert each incoming bid in a priority queue with the lowest bid claiming the highest priority
- Assign m cities (one to each robot) starting with the highest priority bid. (Note, once a city is assigned from the priority queue, all other bids for that city and all other bids submitted by that robot are removed from the queue before making the next assignment)
- Delete all bids, and call for re-bids for all remaining cities
- Repeat procedure until all n cities are assigned

Once the *OpTrader* has completed the initial city assignments, the robots negotiate amongst themselves to subcontract city assignments. Unlike a bartering system where robots can only make "city-for-city" deals, the TraderBots approach allows robots to make "city-for-revenue" deals, thereby enabling transactions between robots that have a task distribution where only a one-way task transfer reduces cost, and by enabling transactions that reduce team costs but increase a robot's individual costs. Each of the robots, in turn (the initial implementation is fully synchronous), offer all the cities on its tour (individually) to all the other robots for a maximum price equal to the offerer's cost reduction by removing that city from its tour. Each bidder then submits a bid for that city greater than the cost for adding the city to its tour. Note that a bidder only submits a bid for a city if it can make at least a fixed minimum profit by inserting that city into its tour. In order to estimate the additional cost of inserting a city into its tour, the bidder evaluates the cost of inserting that city at each point of the tour and picks the point of insertion that results in the lowest cost increase. In this initial implementation, only single-city deals are considered, and the robots continued to negotiate amongst themselves until no new, mutually profitable deals are possible. Thus, negotiations cease once the system settles into a local minimum of the global cost.

A few experiments were carried out to evaluate the performance of the TraderBots approach implemented in simulation:

Experiment 3.1: A comparison of the global solution with and without inter-robot trading

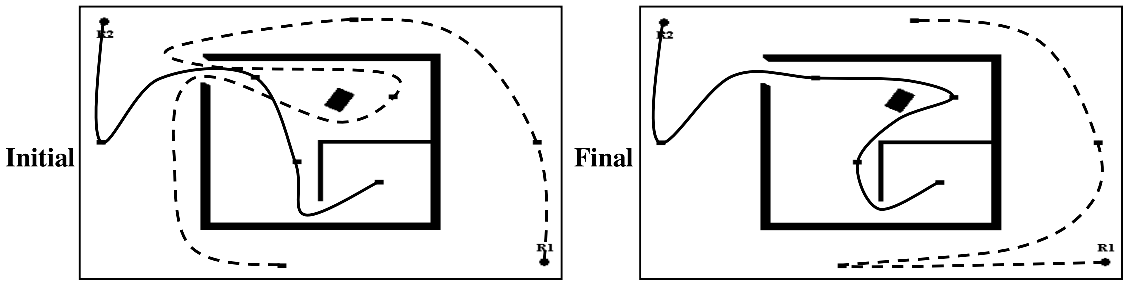


Figure 8: Initial assignments and final tours for 2 robots and 8 cities (14.7% decrease in team cost)

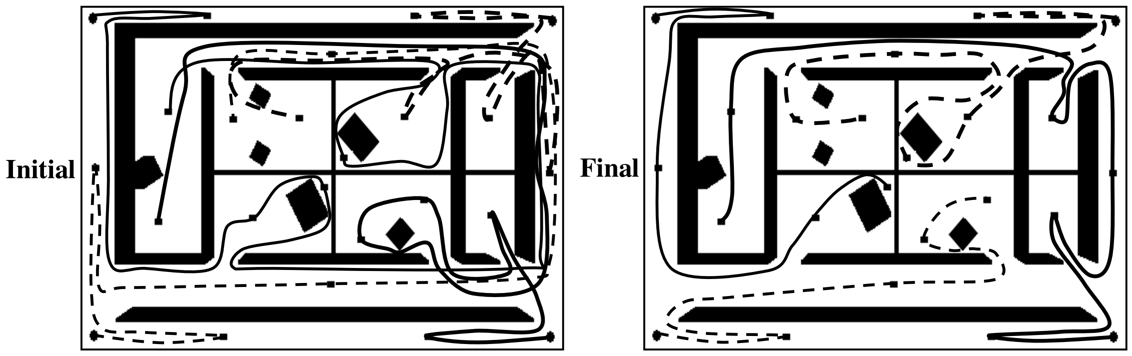


Figure 9: Initial assignments and final tours for 4 robots and 20 cities

Experiment 3.2: Plot of global cost versus number of trades

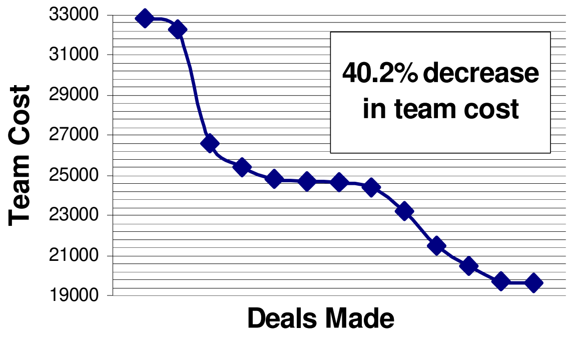


Figure 10: Team cost reduction during inter-robot negotiation for the example in Figure 9

Note that the reported decrease in team costs was calculated on the operator executive's initial greedy assignment and not on a random assignment that would have resulted in a significantly higher initial team cost on average. Further comparisons to globally optimal solutions and other greedy solutions will be carried out in the near future. Although many features of the market-based architecture were not implemented in this preliminary version, initial experiments clearly show effective global plans with low team costs.

Experiment 3.3: Preliminary testing was also carried out to evaluate the system response to dynamic conditions in a scenario similar to exploration of a partially known world[2]. The operator designated a set of observation points to be visited in a simulated world. Robots began their exploration based on the tours they won after negotiations had ceased. Whenever a robot approached a doorway, a new observation point was dynamically triggered inside the "newly observed" room. When a new goal was triggered, the robots ceased their current tours and began negotiations to determine new tours that were most profitable in light of the new information. Thus, they were able to adapt to the dynamically added input.

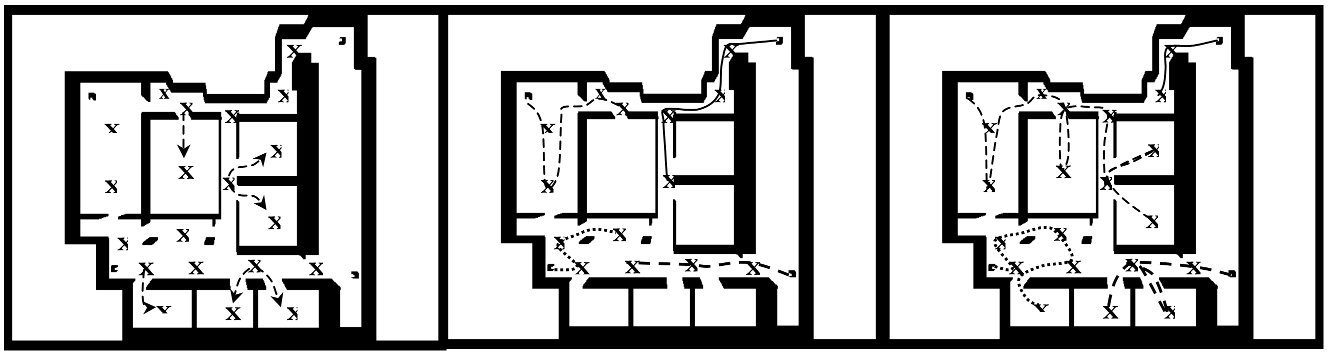


Figure 11: Results from Dynamic re-negotiations

[2] This work was published in SPIE 2000 (Thayer et al. [120])

The illustration above shows the evolution of four robots' tours in a 20-city exploration scenario.[3] Initially the operator designated 14 cities. Whenever a robot reached a doorway, a city was triggered in the "newly discovered" room. Thus, six additional cities were triggered and the robots adapted their tours to allow visiting all the cities.

An initial version of the approach was also implemented on the Cognitive Colonies robotic system. Many valuable lessons for architectural design, especially regarding communication and real-time system issues, were learned through the process of this implementation. The aim of the Colonies project was to build a group of robotic aids for urban reconnaissance. Ten PioneerII-DX robots, built by Activmedia Incorporated, (described in greater detail as Implementation 2.1 in Appendix 1) are used in the project.

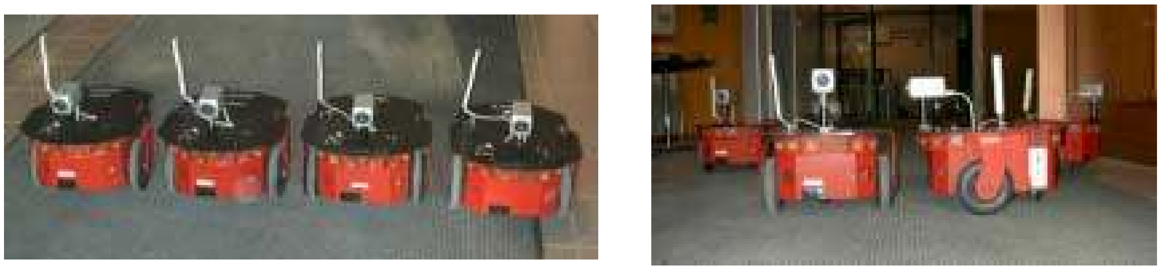


Figure 12: Pioneer Robot Team

The robots are each equipped with onboard computing as well as 16 sonar sensors for obstacle detection and a forward pointing camera for map construction. The distributed TSP scenario is tested using 4 of these robots in a cluttered environment. The robots only negotiate with the *OpTrader* in this implementation. Inter-robot negotiation was not implemented due to some communication problems that have since been sorted out. The operator is able to specify goals to be visited via a GUI. These goals are then bid out to the robots by the *OpTrader* and assigned according to the greedy algorithm described in the simulation implementation above.

The robots start at different positions in the Field Robotics Center (FRC) high-bay at Carnegie Mellon University and are able to complete the assigned tours. Robots are also able to dynamically handle new tasks assigned during execution as shown in Experiment 3.3.[4]

This initial implementation of TraderBots on the Pioneer robots was further enhanced by Zlot et al. [131] (described in further detail as Implementation 2.2 in Chapter 7) and tested on a distributed exploration and mapping task. In this implementation, the TraderBots approach seeks to maximize benefit (information gained) while minimizing costs (in terms of the collective travel distance), thus aiming to maximize utility. The system is robust in that exploration is completely distributed and can still be carried out if some of the team members lose communications or fail completely. All communications are asynchronous.

[3] A movie of this evolution is available at http://www.frc.ri.cmu.edu/projects/colony/frcworld.shtml
[4] Some videos of robots performing a distributed sensing task are available at http://www.frc.ri.cmu.edu/projects/colony

The effectiveness of this approach was demonstrated through successful mapping results obtained with the team of robots. Zlot et al. [131] found that by allowing the robots to negotiate using the market architecture, exploration efficiency was improved by a factor of 3.4 for a four-robot team.

Thus, initial implementations proved the TraderBots approach to be a highly promising solution for efficient and robust multirobot coordination. These key characteristics of robustness and efficiency, as applied to the TraderBots approach, are explored in greater detail in the following chapters.

CHAPTER 4

Ensuring Robustness

ROBUSTNESS is crucial for any robot team, especially when operating in dynamic environments. The physicality of robotic systems and their interactions with the environment make them highly prone to malfunctions of many kinds. Three principal categories in the possible space of robot malfunctions are communication failures, the loss of the use of some robot resources (or partial robot malfunction), and robot death. This chapter addresses these three categories and explores means by which the TraderBots approach ensures robustness and promotes graceful degradation in team performance when faced with malfunctions.

4.1 Introduction

Many multirobot applications demand some level of robustness to malfunctions. The requirement for robustness becomes increasingly crucial when the application domain requires the robots to interact within a highly dynamic environment and where prior information about the environment is sparse. Applications such as urban reconnaissance, urban search and rescue, planetary exploration, and hazardous cleanup inherently include hazardous conditions that will cause robotic malfunctions with high probability. Key to the success of these applications is the team's ability to gracefully degrade their performance and maximize the efficiency with which the available resources are used to complete the task. Many multirobot coordination approaches deal with malfunctions in different ways. The three main categories of malfunctions, identified above, are explored next.

4.1.1 Communication Failures

Communication failures are abundant in many application domains. These failures can vary from occasional loss of messages to complete loss of communication. Different approaches handle losses in communication using a variety of strategies. Balch and Arkin [7] study the importance of communication in team coordination using reactive approaches. They study the effectiveness of three types of communication, no communication, state communication, and goal communication, for three kinds of tasks, foraging, consuming, and grazing. Note that grazing is a task with implicit/environmental communication since the completion of the task alters the environment and thus implicitly communicates that the task has been done and prevents repetition. Balch and Arkin draw three conclusions from their study: communication significantly improves performance in tasks with little or no environmental communication, explicit communication is not essential for tasks that result in implicit communication, and more complex communication strategies offer little improvement over simple communication.

As described by Balch and Arkin, some approaches forego communication altogether and robots make action decisions entirely independent of decisions made by teammates ([84], [90]). Other approaches forego explicit communication, but instead, coordinate team actions by basing action selection on observed environmental clues [7], by choosing actions based on anticipated actions of teammates [122], or by socially attentive monitoring of teammates to gauge their progress in task execution, and intervening if necessary [68]. Yet another approach without explicit communication is to select actions based on a set of pre-defined rules, triggered by environmental cues or observation of specific team formations or actions. An example of such an approach is the locker-room-agreement used by Stone and Veloso [112] in the robotic soccer domain. None of these coordination approaches are affected by failures in communication. However, they are also unable to effectively use information that can improve team performance if shared with teammates. Roth et al. [89] and Vail and Veloso [122] show that teams can perform more effectively if teammates coordinate and share information.

If the robots explicitly communicate with each other, there are still several methods to ensure graceful degradation in performance with communication failures and limitations. Stone and Veloso ([112], [113]) present a set of techniques for dealing with communication-based coordination of robot teams in adversarial environments with unreliable, high-cost communication. They address five challenges of communication: identifying messages targeted to a specific robot, dealing with active interference from hostile agents due to sharing a single communication channel, effectively sharing a low-bandwidth communication channel, being robust to lost messages, and maximizing the likelihood that the team is operating based on a cohesive strategy despite unreliable communication and autonomous team-members. Each message is equipped with a field identifying its target to meet the first challenge. Hostile agent interference is avoided by encrypting the time-stamp on the message according to a pre-determined locker-room-agreement. The team shares the low-bandwidth channel by responding to messages after a pre-defined delay specific to each member's ID, and is robust to message loss because each member is always active regardless of what messages it receives or not, and because communication is used to improve task execution rather than enable it. Finally, a cohesive team-strategy is maintained by following a pre-defined strategy based on environmental cues, enhanced by time-stamped messages indicating switches in strategy when possible.

Not all domains are adversarial. Parker's ALLIANCE architecture [83] does not reason about hostile agents, but encourages fault tolerance in several ways. The use of broadcast communication (as opposed to point-to-point communication) and the design of behaviors such that robots announce their current intentions but never require responses, eliminate the dependence of coordination on multi-step communication, thus improving robustness. Furthermore, the team uses time-slice communication so that each agent gets exclusive use of the single communication channel periodically.

Today, communication for robot teams is not always limited to low-bandwidth, single-channel communication. A method to ensure robustness to message loss in less stringent application domains is the use of acknowledgements, as in the original Contract Net Protocol by Smith [107]. In this approach, the receiver acknowledges the receipt of each message. While this approach adds a level of robustness to message loss, some limitations are evident. The acknowledgement can be lost as easily as the message, the acknowledgements add to the communication load, and the approach does not explicitly deal with the scenario of complete loss of communication.

4.1.2 Partial Robot Malfunction

Relatively little work has been done to investigate efficient use of partially malfunctioning robots. When a robot malfunctions partially, it loses the ability to effectively use some of its resources but retains the ability to use others. Inherent in this definition of partial malfunction is the robot's ability to plan for itself of the ability to communicate with a planning agent; if the robot loses this capability, it is considered dead. Many reactive and behavioral approaches (for example Arkin [4], Balch and Arkin [7], Parker [83], and Rus et al. [90]) are resilient to partial robot malfunctions because robots execute tasks independent of what other team members do, and hence all tasks with no specific time deadlines are accomplished as long as at least one capable robot remains active. Gerkey [49] demonstrates a fault-tolerant auction scheme that decomposes a cooperative box-pushing task into short-duration pusher and watcher/coordinator tasks. Since the tasks span only a short duration, the team re-evaluates the progress of the task frequently and thus recovers from faults by reassigning short duration tasks designed to adapt to the most current state. However, these approaches do not reason about efficient utilization of remaining active resources on the partially malfunctioning robots.

One of the difficulties in dealing with partial robot malfunctions is detecting the malfunction. A host of literature on fault detection and identification demonstrate different techniques that enable robots to detect and identify their own faults. However, relatively little work has been done to address handling detected faults in a team. Techniques such as socially attentive monitoring (Kaminka and Tambe [68]) and regular monitoring of the task/environment state and adapting to it (Gerkey [49] and Rus et al. [90]) allow teammates to discover faults that the robot cannot detect itself.

Once a fault is detected, fewer techniques have been proposed to deal with them. Bererton and Khosla ([9], [11]) analyze the merits and challenges of repairing robots when failures are detected. Some investigated strategies are towing faulty robots to a base station, cooperative repair, and self-repair. Another common scenario that can be viewed as a partial robot malfunction is the loss of robot energy. Since low battery power interferes with task execution, and since robots can plan to recover from this condition by recharging, it fits the category of partial robot malfunctions. Michaud [77] and Gerkey [49] both investigate recharging techniques for robot teams.

4.1.3 Robot Death

Robot death is similar to the case of partial robot malfunction, except that the affected robot cannot aid in the recovery from the malfunction in any way. Most of the research in fault tolerance (Parker [83], Gerkey [49], and Rus et al. [90] for example) deals with robot death. As with partial robot malfunctions, many reactive and behavioral approaches are resilient to robot death because robots execute tasks independent of what other team members do, and hence all tasks with no specific time deadlines are accomplished as long as at least one capable robot remains alive.

The detection problem is more difficult for robot death since the dead robot cannot detect its own death and reallocate its tasks. A common method of detecting robot death is to monitor a heartbeat (a periodic signal) from robots and assume the robot is dead if the heartbeat is not detected. Other methods of monitoring such as Kaminka's and Tambe's socially attentive monitoring [68] can also be used to detect the death of teammates.

Once a dead robot is discovered, any tasks assigned to that robot must be reassigned or the dead robot must be repaired. Bererton's and Khosla's work on robot repair ([8], [11]) can be applicable to some cases of robot death. Note that in the cases where malfunctioning robots or dead robots can be repaired and return to the team, the coordination approach needs to be fluid in order to accommodate both the exit of the dead robots and the entrance of the repaired robots.

The TraderBots approach is capable of handling all three categories of robot malfunctions. The different strategies used in the TraderBots approach to gracefully handle these malfunctions are examined in detail next.

4.2 Handling Communication Failures

The TraderBots approach does not depend on communication to complete tasks. Communication mainly plays the role of enabling improved efficiency in the generated solutions. If the robots in the team are presented with a task they need to execute, they are able to execute the task without communication if the tasks do not explicitly require communication. However, the availability of communication can dramatically improve the efficiency of task allocation if the robots are allowed to trade. Zlot et al. [131] investigate the performance degradation of the team, in the TraderBots approach, given the absence of communication. The reported results show that inter-robot trading improves efficiency of the solution by a multiple of 3.4 for a team of 4 robots if communication is flawless, in comparison to the case where no communication is allowed. However, even in the absence of communication the task is complete.

Newer implementations of the TraderBots approach are made more robust to communication failures. Message loss is expected and often witnessed resulting in only minor degradations in solution efficiency. Strategies used to improve robustness are: frequent auctioning and bidding which help reallocate tasks among robots, the absence of assumptions that all robots will participate in any auction, monitoring of communication connectivity to robots that have subcontracted tasks, and constant scheduling of assigned tasks for execution.

However, in a case where only the OpTrader is aware of all tasks, and the tasks are divided among the robots, a scenario such as a combination of communication failure between the OpTrader and all robots, plus the death of a robot with assigned tasks can result in the task assigned to the dead robot to remain incomplete. Thus, domains where completion of the global task (i.e. all tasks assigned to the team) must be guaranteed (if resources are available) require a somewhat different strategy. A possible strategy for these domains is to disseminate knowledge of all tasks to all robots, as would be the case in many reactive approaches. If each robot maintains a list of all tasks assigned to the team, and if each task completion results in an announcement of all completed tasks to all robots within communication range, robots can choose to execute tasks not awarded to them after they have completed the tasks awarded to them and negotiate payment for the tasks via the OpTrader. Whenever a robot chooses to execute a task that wasn't assigned to it, an announcement of this intention can improve efficiency since other robots would no attempt to execute the same task. This strategy guarantees that all tasks are completed with the possible inefficiency of some tasks being repeated depending on communication fidelity. Note that this strategy is only required if specific tasks are assigned to the team. In the case where robots dynamically generate tasks (i.e. where the same tasks can be generated by other robots given the necessary time), as in work published by Zlot et al. [131], such strategies are unnecessary.

4.3 Handling Partial Robot Malfunctions

Detecting partial robot malfunctions in the TraderBots approach is achieved by monitoring the resources available to the robots. While specific algorithms for fault detection and identification are beyond the scope of this dissertation, in general, the resource manager's loss of access to a particular resource, the trader's loss of access to its resource manager, the discovery of an unforeseen depletion of a resource, or the discovery that the accrued cost in attempting to complete a task surpasses the estimated cost for that task, can indicate a partial robot malfunction.

Once a trader discovers a partial robot malfunction, it attempts to sell all tasks it cannot complete to other robots even if it has to take a loss for some of the trades. If however trading becomes impossible due to a coupling with loss of communication, then the relevant strategy described in the previous section needs to be used for the case where a static set of tasks, all of which must be completed, is assigned to the team. Thus, graceful degradation with malfunctions of team performance is achieved. If the malfunction occurs with the trader, then it falls into the category of robot death.

4.4 Handling Robot Death

Once a robot is incapable of trading, it is considered dead. In this case, the robot cannot aid in the detection or recovery process. Several methods can be employed to allow teammates to discover robot death as discussed above in section 4.1.3. The TraderBots approach can deal with detected robot deaths by attempting to discover all trades that affected the dead robot. This can be done in several ways. Some possibilities are explored next. Each trader can keep track of awards it makes and receives. Thus, if a

robot death is detected, each trader checks to see if it has awarded any tasks to the dead robot, or if it has won any tasks from the dead robot.
If a robot *A* has awarded a task to the dead robot, *D*, it makes an announcement to the remaining robots to find out if the robot *D* has traded that task to another robot. If such a trade is discovered, the robot that is currently committed to executing the task, robot *B*, compares costs with robot *A*. The robot that can execute the task more profitably gets awarded the task. However, robot *B* may not have won the task directly from robot *D*, but instead won it from robot *C*, who in turn won it from robot *D*. One possibility for dealing with this scenario is for robot *B* to pay a breach penalty to robot *C* and generate a new contract directly with robot *A*. Another possibility is for robot *C* to cut its losses and for robot *A* and robot *B* to make a deal profitable to both of them. Yet another simpler strategy is for robot *A* to simply be satisfied that the task will be executed by robot *B* and hence simply update its information that the task will be executed by robot *B*. Many other similar solutions are possible. If on the other hand robot *A* cannot discover any robot that is currently committed to executing that task, the task is added back to robot *A*'s commitment list. Note that this can result in the task being repeated since robot *A* may simply not have been able to discover the robot executing the task due to communication limitations, or since robot *D* may have completed executing the task before it died. The premise of such an implementation would be that it is better to repeat the execution of a task rather than leave any task incomplete if possible.

If robot *C* realizes it has won a task from the dead robot *D*, a simple strategy is for robot *C* to complete the task at a loss even though it will not be compensated by robot *D*. A slightly more complex strategy is for robot *C* to attempt discovering robot *A* that awarded the task to robot *D* and attempt to get paid by robot *A* for completing the task. If robot *C* cannot discover robot *A*, it can attempt to get paid directly by the OpTrader for completing the task. Numerous such strategies can be applied to guarantee task completion if sufficient robots remain active. The best strategy can be picked depending on the demands of the particular application domain. Finally, note that the TraderBots approach easily accommodates fluidity by allowing repaired robots or new robots to enter the team since any available robot can participate in the frequently conducted auctions.

4.5 Implementation

An implementation of the TraderBots approach on a team of Pioneer II DX robots enables investigation of how successfully TraderBots deals with robot failures in each of the three categories described above. The details of the robotic system used in this implementation, and the details of the TraderBots implementation itself are found in Appendix 1. Details specific to the study of robustness are presented in this section. In the robotic implementation, each robot navigates using a basic set of behaviors as described in Appendix 1. When the robots are not executing tasks, they remain stationary at their current locations. Implementation details for each of the three categories are detailed next.

4.5.1 Handling Communication Failures

The TraderBots approach does not depend on communication to complete tasks. Communication mainly plays the role of enabling improved efficiency in the generated solutions. Zlot et al. [131] investigate the performance degradation of the team, in the TraderBots approach, given the absence of communication. Newer implementations of the TraderBots approach are made more robust to communication failures. Message loss is expected and often witnessed resulting in only minor degradations in solution efficiency. Strategies used to improve robustness are: frequent auctioning and bidding which help reallocate tasks among robots more efficiently, the absence of assumptions that all robots will participate in any auction, monitoring of communication connectivity to robots that have subcontracted tasks, and continuous scheduling of assigned tasks for execution as tasks are completed.

However, in a case where only the OpTrader (an interface agent responsible for trading on behalf of the operator) is aware of all tasks, and the tasks are divided among the robots, a scenario such as a combination of communication failure between the OpTrader and all robots, plus the death of a robot with assigned tasks can result in the task assigned to the dead robot remaining incomplete. Thus, domains where completion of the global task (i.e. all tasks assigned to the team) must be guaranteed (if resources are available) require a different strategy. A possible strategy for these domains is to disseminate knowledge of all tasks to all robots, as would be the case in many reactive approaches. Note that this strategy is only required if specific tasks are assigned to the team. In the case where robots dynamically generate tasks (i.e. where the same tasks can be generated by other robots given sufficient time), as in work published by Zlot et al. [131], this strategy is unnecessary.

In the current implementation, it is possible for more than one robot to believe it is responsible for executing the same task if communications are imperfect. For example, when robot A awards a task to another (robot B), an acknowledgment is sent from B to A. If the acknowledgment is lost, then robot A does not know if B has accepted the task. In that case both A and B will maintain responsibility for completing the task. It is also possible that this duplication of tasks can be repaired: one of the robots may subsequently try to auction the task, in which case the other will be likely to win it as its marginal cost for the task is 0.

4.5.2 Handling Partial Robot Malfunctions

Detecting partial robot malfunctions in the TraderBots approach is achieved by monitoring the resources available to the robots. While specific algorithms for fault detection and identification are beyond the scope of this paper, in general, the TaskExec's (the module responsible for task execution) loss of access to a particular resource, the Trader's loss of access to its TaskExec, the discovery of an unforeseen depletion of a resource, or the discovery that the accrued cost in attempting to complete a task surpasses the estimated cost for that task, can indicate a partial robot malfunction. Once a Trader discovers a partial robot malfunction, it attempts to sell all tasks it cannot complete to other robots even if it has to take a loss for some of the trades. (The trader still attempts to maximize profit, so any losses will be minimized). If however trading becomes impossible due to a coupling with loss of communication, then the relevant strategy described in the previous section needs to be used for the case where a static set of tasks, all of which must be completed, is assigned to the team. Thus, graceful degradation with malfunctions of team performance is achieved. If the malfunction occurs with the Trader, then it falls into the category of robot death. In this implementation, robots were able to detect malfunctions caused by disconnection of the on-board SICK laser used for obstacle detection and mapping, and gyro errors caused by sudden drastic rotations of the robot due to a wheel getting stuck. Ongoing implementation efforts also include detection and appropriate handling of low battery conditions that require the robot to head back to a re-charging station.

4.5.3 Handling Robot Death

Once a robot is incapable of trading, it is considered dead. In this case, the robot cannot aid in the detection or recovery process. Several methods can be employed to allow teammates to discover robot death as discussed above in section 4.1.3. The TraderBots approach can deal with detected robot deaths by attempting to discover all trades that affected the dead robot. Each trader can keep track of awards it makes and receives. Thus, if a robot death is detected, each trader checks to see if it has awarded any tasks to the dead robot, or if it has won any tasks from the dead robot.

If a robot has awarded a task to the dead robot, it makes an announcement to the remaining robots to find out if they subcontracted that task from the dead robot. If such a trade is discovered, the two robots re-negotiate their deal with respect to that task. If a robot cannot discover any robot that is currently committed to executing that task, the task is added back to its commitment list. Note that this can result in the task being repeated due to communication limitations. The premise of such an implementation would be that it is better to repeat the execution of a task rather than leave any task incomplete, if available resources permit. Ongoing implementation efforts include enabling the TraderBots approach to gracefully and robustly accommodate robot death. Results in detecting and handling robot death will be added in final submission of this paper if it is accepted for publication. Finally, note that the TraderBots approach easily accommodates fluidity by allowing repaired robots or new robots to enter the team since any available robot can participate in the frequently conducted auctions. Initial experiments demonstrating this capability are reported in results accepted for publication in the proceedings of the 2004 conference on Intelligent Autonomous Systems [43]. A limitation in the current implementation is that detection of a robot death is indistinguishable from a severe communications failure since the only way robots detect one another is via communication. It would be possible to improve this if the robots additionally had some other mode of detecting/monitoring each other (for example, by using a camera or some other sensor).

4.6 Experiments

An implementation of the TraderBots approach on a team of 3 Pioneer robots enables the reported results. The robot team consists of a homogenous set of off-the-shelf mobile robot platforms outfitted with additional sensing and computing. The chosen application is a distributed sensing problem where 3 robots are tasked with gathering sensory information from 23 designated locations of interest in a large dynamic environment. This translates into a version of the traveling salesman problem (TSP) with the robots being represented by multiple salesmen following paths instead of tours (i.e. without the requirement that robots need to return to their starting locations) and where all the robots can start from different base locations – this is known as the multi-depot traveling salesman path problem (MD-TSPP). The tasks can be considered as cities to be visited where the costs are computed as the time taken to traverse between cities. A task is completed when a robot arrives at a city. The global task is complete when all cities are visited by at least one robot. The global cost is computed as the summation of the individual robot cost, and the goal is to complete the global task while minimizing the number of robot-hours consumed. When the robots are not executing tasks, they remain stationary at their current locations.

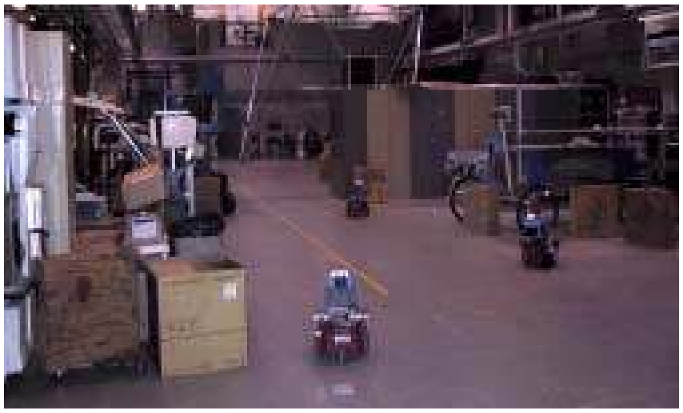


Figure 13: Photograph of test-site

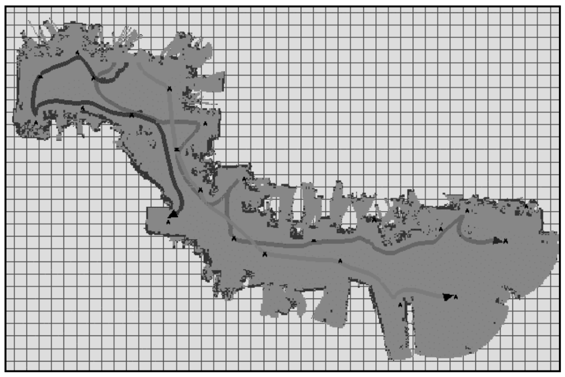


Figure 14: Map of environment showing assigned goals and robot paths (grid squares = 1m×1m)

Each robot is responsible for optimizing its own local schedule (i.e. given a set of tasks, the robots attempt to find the optimal TSPP solution to their local problem instance). In general, the TSPP is NP-hard, so approximation algorithms are often used when the problem instances encountered are large. In the implemented TSPP scenario, all valuations are derived from inter-point distance costs. These costs are estimated using a D* path planner [108] with the robot's local map as input. The experiments performed, using this implementation, are described next.

§ Communication Failures

Experiment 4.1:Communication statistics over several runs with fixed set of tasks

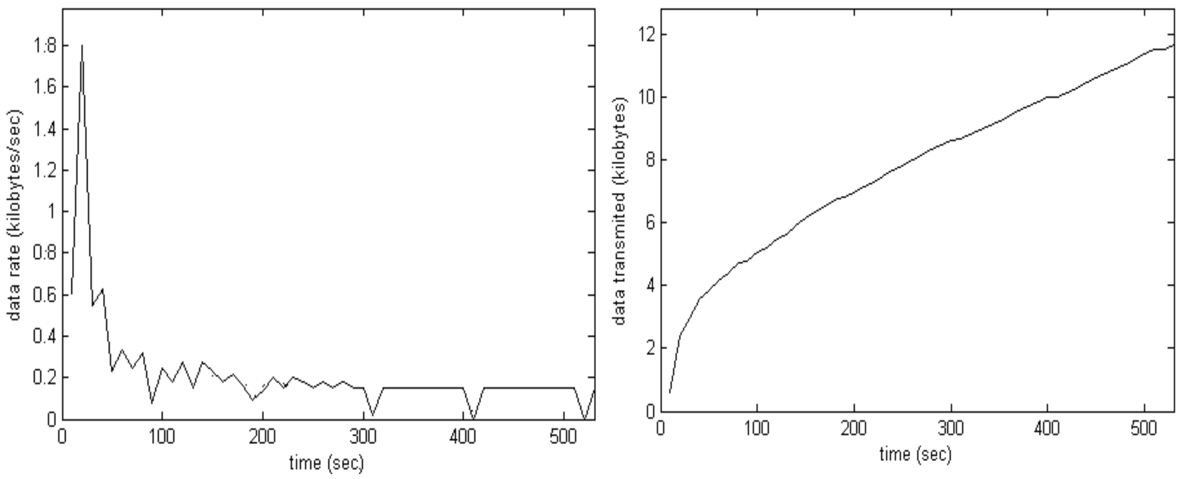


Figure 15: Trading-Related Communication Traffic

Figure 15 shows an analysis of communication traffic sent by one trader using the *TraderBots* approach. Figure 6a illustrates the evolution of the data rate, in kilobytes/second, over time, in seconds, and Figure 6b shows the cumulative data transmitted, in kilobytes, over the same time period. Message types are auctions, bids, hunts, awards, and acknowledgments. When the robot is first deployed, communication peaks as a result of the initial auction. As tasks are executed and knowledge is gathered, tasks continue being traded at a lower rate. The steady state communication rate is due to the continuing trading mechanism (auctions, bids, awards) and messages to maintain knowledge of trader state (hunts, acknowledgments).

Experiment 4.2: A set of exploration runs with staged communication failures ranging from 100% of messages being delivered to 0% of fidelity, with a fixed set of tasks. Communications between robots are blocked at different percentage levels (20%, 40%, 50%, 60%, 80%, and 100%) and the corresponding performance is reported.

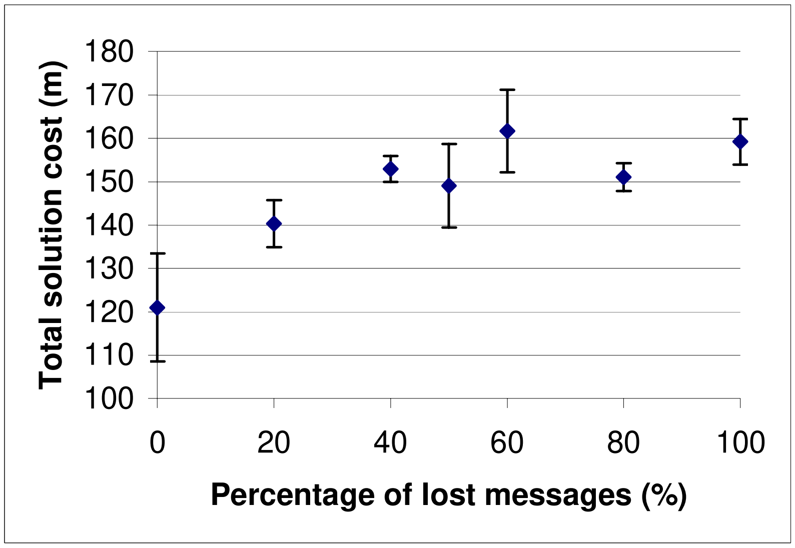


Figure 16: System Performance with Communication Failures

The first set of experiments investigates the effect of communication failures on the team performance. Inter-robot communication is blocked at different percentages and the resulting solution recorded. Figure 16 shows the variation of solution cost with the percentage of message loss. While the solution cost increases with loss of communication until approximately 60% loss, further communication loss has little added effect on performance. The reason for this is that when the loss rate is significant but not too large, it is often the case that tasks are subcontracted, but their award acceptance acknowledgement message does not reach the seller. When this happens, the task ends up being duplicated, thereby increasing the global cost. When the loss rate is high, the trades do not progress to this stage as often and this effect is not seen as frequently (e.g. the bids or the award are already lost, so the task is not awarded). Since our initial solution based on the initial OpTrader auctions is reasonably good, the result is that we sometimes can do better with 100% loss rate than with 60% loss. We hypothesize that if we start off with a worse solution (for example, an initial random allocation), then we would expect that this function would be more monotonic. Also, if we enable the robot death handling, then there would be more duplications of tasks at the high loss rates when a death was detected and robots try to make up for the tasks of the "dead" robots.

§ **Partial Robot Failures**

Experiment 4.3: A set of exploration runs, with staged randomly induced partial malfunctions, with a fixed set of tasks. The laser is turned off or a gyro error is introduced at a specific point during execution and the resulting performance is reported.

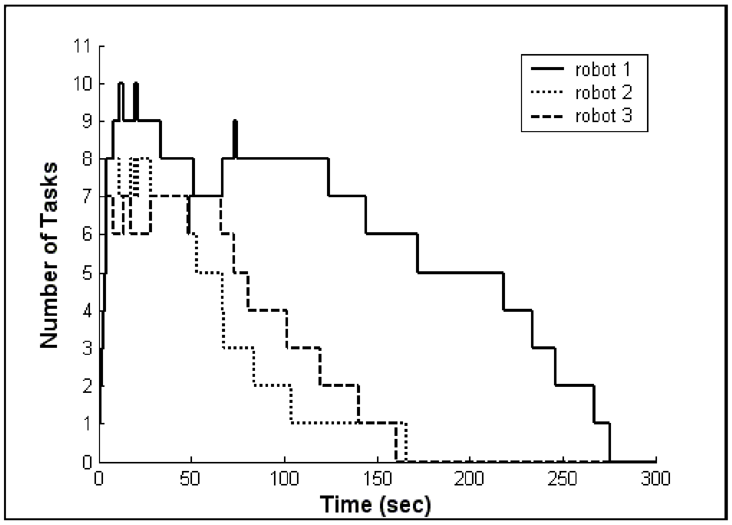


Figure 17: Nominal Performance

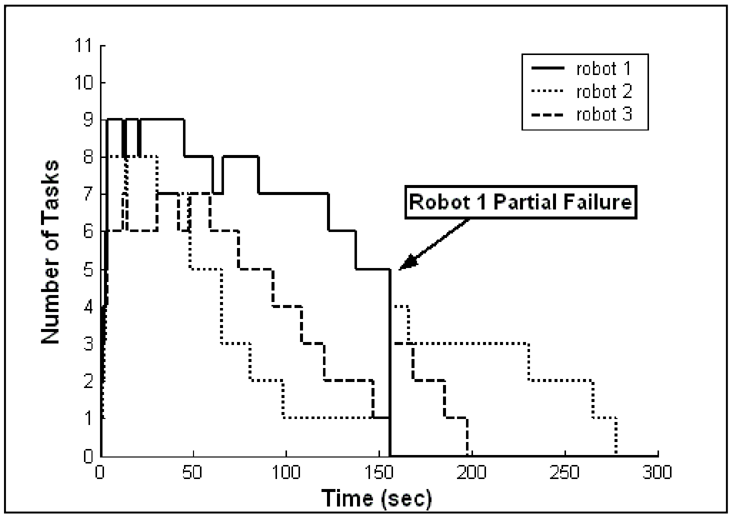


Figure 18: Partial Robot Malfunction

Figure 17 and Figure 18 show the variation in the number of tasks assigned to each robot over time for a nominal run and a partial robot failure run respectively. While the number of tasks gradually decreases with time as tasks are executed in a nominal run, when a partial failure occurs, that robot immediately trades away all of its tasks attempting to minimize its losses, and hence the malfunctioning robot has a sudden loss in the number of assigned tasks. The other two robots have a sudden gain since they are assigned the unfinished tasks of the malfunctioning robot.

The results from all experiments are reported in Table 4 below.

	Cost (m)		**Tasks (#)**	
Description	**Average**	**+/-**	**Success**	**Failure**
Nominal	121	12	21.0	2.0
Partial Failure	140	5	22.0	1.0
20% msg. loss	140	5	24.0	0.3
40% msg. loss	153	3	24.7	2.0
50% msg. loss	149	10	24.0	0.7
60% msg. loss	162	9	25.3	0.7
80% msg. loss	151	3	22.3	0.7
100% msg. loss	159	5	21.0	2.0

Table 4: Performance Statistics

For each experiment, the mean cost, the standard deviation in the cost over the three runs of the experiment, the number of tasks that succeeded, and the number of tasks that failed are shown. Note that tasks can fail for several reasons due to the dynamic environment and conditions. Note further that tasks are sometimes duplicated and hence the addition of succeeded and failed tasks sometimes exceeds the number of assigned tasks (23). Future implementations will be able to better deal with duplicate tasks as follows. If a trader is selling task x and another robot already has committed to task x then that robot will bid very low for the task and win it (its marginal cost is 0). When the robot is awarded the task, it should check if it is a duplicate, and if so it should be discarded.

4.7 Limitations and Future Work

This chapter presents a comprehensive study of how the TraderBots approach is robust to failures. Three categories of failure are identified and explored in this study: communication failures, partial robot failures, and robot death. All three categories of failure are studied for a team of 3 Pioneer robots assigned a distributed sensing task. Ongoing work introduces random combination of failures at random times during the experiment, to gauge the effect on the overall performance. Introduced failures include communication failures, partial robot malfunctions, and robot deaths. Some robots are also re-introduced into the team following a simulated revival from death.

Disallowing robots to recover from failures and not investigating cooperative means of robot repair thus far limits this study. Adversarial domains are not addressed either. Future work includes developing techniques for more efficient use of partially malfunctioning robots, examining strategies for cooperative recovery from failures, and more rigorous testing of the robustness of TraderBots in different scenarios.

CHAPTER 5

Encouraging Efficiency

THE work presented in this chapter explores different methods of encouraging efficient solutions when applying the TraderBots approach. Considered strategies include frequent trading, some effects of opportunistic optimization with leaders in market-based multirobot coordination, and negotiation strategies for improving solution efficiency, namely the capability to perform different combinations of multi-party and multi-task trades.

5.1 Introduction

Efficiency is another key requirement of many multirobot coordination domains. While a few applications do not require efficient solutions, many applications require efficiency in automated solutions, and most applications benefit greatly from increased efficiency. However, often a tradeoff exists between efficiency, robustness, resources, and latency. Producing the optimal solution can be a long and computationally expensive process and often requires a centralized solution that can degrade robustness. Dynamic environments further exacerbate the requirements for producing optimal solutions. Hence, a more reasonable strategy, applicable to more application domains, is to introduce strategies for opportunistically encouraging efficiency in the produced solutions while maintaining robustness in the system. This chapter explores some strategies for opportunistic methods of encouraging solution efficiency in the TraderBots approach.
Many groups have pursued different optimization methods. Coalition formation techniques to optimize multi-agent coordination have been proposed by Sandholm and Lesser [94], Sandholm et al. [97], Zlotkin and Rosenschein [132], and Shehory and Kraus [103]. Tambe [118] proposes tracking behavior of other agents as an optimization technique. Rosin and Belew [88], and Matos and Sierra [76] adopt evolutionary methods for enhancing performance. Other optimization techniques have been proposed by Lux and Marchesi [72], Huber and Durfee [61], Castelfranchi and Conte [25], Panzarasa and Jennings [81], Sandholm and Lesser [98], Khuller et al. [69], Cheng et al. [27], and Andersson and Sandholm [2].

An important contribution of this work is the development of a "leader" role that allows a robot with the necessary resources to assess the current plans of a group of robots and provide more optimal plans for the group. The leader can gain knowledge of the group's current state via communication or some form of observation. A prospective leader can use the profits generated by an optimized plan to bid for the services of the group members, and retain a portion of the profit for itself. The leader may bid not only against the individuals' plans, but also against group plans produced by other prospective leaders. In this work we implement a preliminary version of the leader capability by means of a combinatorial exchange, as proposed in [40]. Centralized and distributed approaches are two extremes along a continuum. The introduction of leaders allows the market-based approach to slide along this continuum in the direction of improved profitability in an opportunistic manner. Furthermore, this work addresses one of the key limitations of our early implementation of this approach: the restriction of negotiations to single-party, single-task deals. In many cases, this restriction limits the global cost reduction, since the robots do not have the negotiation tools to reason their way out of shallow, local minima. The work presented here extends these tools to permit multi-party and multi-task deals with better global cost reduction potential.

5.2 Frequent Inter-Robot Trading

The TraderBots approach allows robots to auction tasks in their commitment list to other robots if a profitable trade can be made. The frequency at which robots opt to auction their tasks can affect the efficiency of the solution. In our most current implementation, robots offer for auction all tasks that are not scheduled for execution every 10-15 seconds. The implementation also encourages staggered auctions so that all robots are not holding auctions in synchrony. This allows robots to participate in a variety of auctions without complicating bid-estimation by requiring that bidders estimate their chance of winning bids submitted to different auctions. Robots are allowed to participate in multiple auctions without waiting for the outcome of any single auction. Due to the frequency of re-auctioning, and the dynamic nature of the environment that affects the cost estimation for bids, the solution efficiency does not suffer greatly because the robots do not take into account bids that have been submitted but not yet awarded.
Experiments to determine the effects of different levels of participation in auctions are shown next. In these experiments, run in simulation by the FIRE project group (see Appendix 1 for further details) with special thanks to Dani Goldberg, execution is delayed until market quiescence. In other words, when every RoboTrader that wants to (or can) participate in auctions has done so for every auction available, and no tasks have been bought or sold, then the market is deemed to have quiesced. This means that a stable allocation (or local minimum) has been found. Only after quiescence do the RoboTraders execute their tasks. Delayed execution is useful in that it allows us to study the performance of the market under different parameters without the confounding effect of having tasks executing, and thus disappearing, from the market.

The experimental parameters we examined form a progression in terms of participation in RoboTrader auctions. The parameters are:

§ **Maximum Awards Per Auction (MAPA):**
The maximum number of tasks awarded by the OpTrader in a single auction, influencing the quality of the initial allocation.

§ **No RoboTrader Auctions (RTno):**
RoboTraders never auction their tasks, so the OpTrader allocation is never altered. Participation in RoboTrader auctions is nonexistent.

§ **OpTrader Auctions First (OTfirst):**
The OpTrader auctions all of its tasks first, and only when it is finished do the RoboTraders conduct their own auctions. Participation in RoboTrader auctions is limited to occur only after OpTrader auctions.

§ **Trader Auctions May Conflict (Tconf):**
The OpTrader and RoboTraders conduct simultaneous auctions. In order for the RoboTraders to be able to assess their costs accurately and clear their auctions, the following constraint is imposed: a RoboTrader may participate as a bidder in at most one external auction and that auction must terminate before its own auction is cleared. Participation in RoboTrader auctions may occur during the entire trial, but it is limited by conflicts.

§ **Round Robin Auctions (Trobin):**
The OpTrader and RoboTraders participate in a round robin series of auctions, where each trader has the exclusive right to hold an auction during its turn. While not necessarily efficient or realistic, the round robin auction-synchronization mechanism enables full participation in RoboTrader auctions while maintaining accurate costing.

§ **Relaxed Participation (RTrelax):**
Each RoboTrader is allowed to participate as a bidder in all external auctions, regardless of how many there are and how they overlap with its own auction. Similar to Trobin, this allows full participation in RoboTrader auctions, but with the possible drawback of inaccurate costing.

For each of the MAPA values of 1, 3, and 6, we performed experiments with each of the five other parameters, giving 15 experimental combinations. We ran 5 trials of each combination, with the completion criterion that all 50 rocks be characterized. The experimental data collected for each trial included: the final total cost of the solution, the degree of participation in RoboTrader auctions, and the number of RoboTrader trades made (Table 1). In all experiments, task execution was delayed until auction quiescence. As a baseline for comparison with our cost values, we ran experiments to find an optimal solution to our problem using a genetic algorithms (GA) approach. The best total cost achieved was 777.

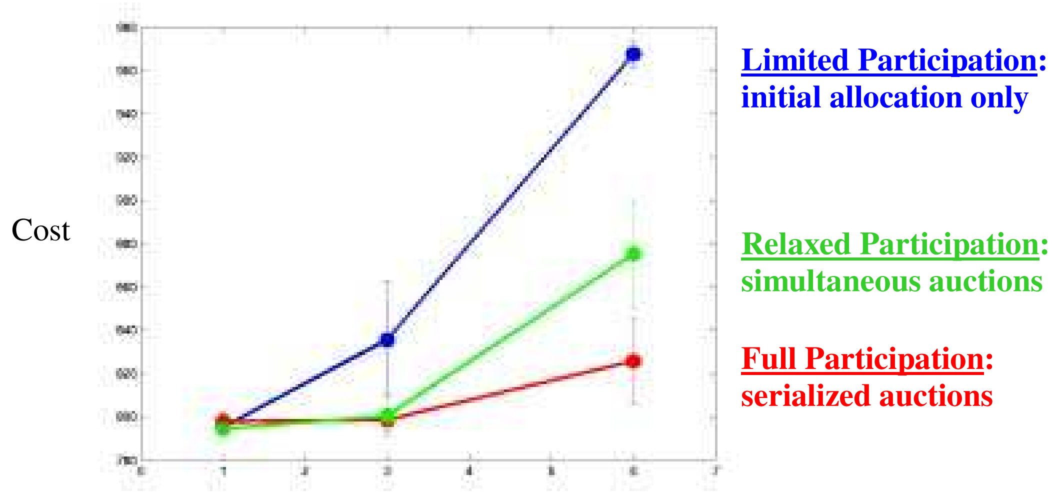


Figure 19: Team Cost for different auction participation schemes

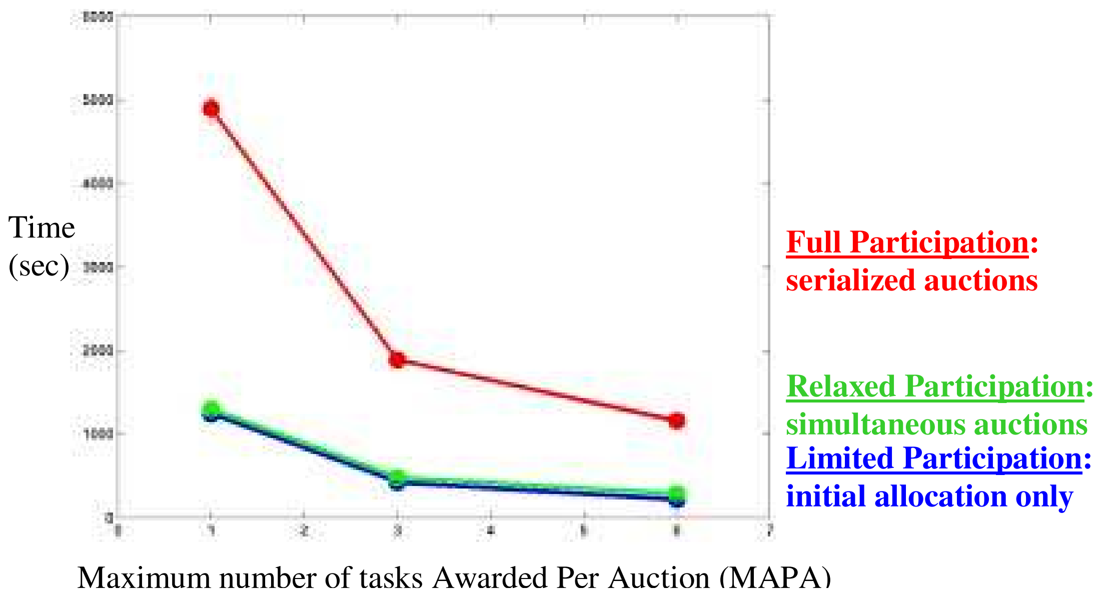


Figure 20: Time to reach solution for different auction participation schemes

There are a number of general trends in the data that are notable in the cost data of Table 5, Table 6, and Table 7. One trend is that the quality of the solution degrades (i.e., total cost increases) as the MAPA value increases. The reasons for this are fairly straightforward. In each auction in our system, the OpTrader never awards more than a single task to each RoboTrader bidder, possibly producing inefficiencies. For example, if the OpTrader awards three tasks in a particular auction (MAPA ≥ 3), they are to three different RoboTraders, when a better allocation might have had two of the tasks going to the same bidder. Thus, smaller MAPA values tend to lead to better initial allocations, though more OpTrader auctions must be held.

A good solution, given the current limitations of our system, relies significantly on the quality of the initial allocation made by the OpTrader (i.e., with a low MAPA value). When MAPA=1, the allocations provided by all of the variations are essentially equally good. When the initial OpTrader allocation degrades (at MAPA values of 3 and 6), having RoboTraders auctions tends to improve the solution, as is evident from comparing RTno to OTfirst, Tconf, Trobin, and RTrelax.

In addition to conducting RoboTrader auctions, Table 5, Table 6, and Table 7 show that increased participation in those auctions tends to improve the solution. One item of note is that even though Trobin and RTrelax show full participation, for MAPA=6 the cost of RTrelax is significantly greater (at a p-value of 0.01). This may be attributed to the inaccurate costing that can arise in the RTrelax case.

	Mean Total Cost		
MAPA	**6**	**3**	**1**
Rtno	967 *(6.1)*	836 *(26.6)*	795 *(1.7)*
Otfirst	920 *(15.7)*	809 *(7.2)*	795 *(1.5)*
Tconf	921 *(17.3)*	837 *(40.3)*	795 *(4.4)*
Trobin	826 *(19.9)*	798 *(5.3)*	798 *(4.4)*
RTrelax	875 *(24.8)*	800 *(9.9)*	794 *(2.0)*

Table 5: Mean total cost of the solution. Standard deviations are shown in parentheses.

	Mean Number of Participants per Auction		
MAPA	**6**	**3**	**1**
Rtno	0 *(0)*	0 *(0)*	0 *(0)*
Otfirst	1.3 *(1.5)*	1.4 *(1.7)*	1.6 *(1.8)*
Tconf	1.1 *(1.5)*	1.1 *(1.4)*	1.7 *(1.9)*
Trobin	5.0 *(0)*	5.0 *(0)*	5.0 *(0)*
RTrelax	5.0 *(0)*	5.0 *(0)*	5.0 *(0)*

Table 6: Mean number of participants in each RoboTrader auction. Standard deviations are shown in parentheses.

	Number of Tasks Sold in RoboTrader Auctions		
MAPA	**6**	**3**	**1**
Rtno	0 *(0)*	0 *(0)*	0 *(0)*
Otfirst	3.2 *(1.5)*	1.6 *(0.9)*	0 *(0)*
Tconf	4.6 *(2.9)*	2.0 *(1.0)*	1.4 *(1.1)*
Trobin	14.6 *(1.1)*	4.0 *(0.7)*	0 *(0)*
RTrelax	23.2 *(1.8)*	7.0 *(4.6)*	0.4 *(0.5)*

Table 7: Mean number of tasks sold between RoboTraders. Standard deviations are shown in parentheses.

Table 7 shows at least part of the reason why greater participation in RoboTrader auctions helps to improve the solutions. Improved participation tends to facilitate tasks being sold between RoboTraders, as is clear by comparing the values for Trobin and RTrelax to the other cases. This increase in trades is particularly significant and important with high MAPA values because of the potential for inefficiencies in the OpTrader allocation. Thus, with high participation, the RoboTraders have the ability to improve the solution before it irrevocably settles into an inefficient local minimum.

5.3 Clustering Tasks

The capability to negotiate multi-task deals greatly enhances the market approach because it allows a robot to escape some local minima in task allocation solutions.[5] However, if the robots bid on every possible combination of tasks, the number of bids submitted will grow exponentially with the number of tasks. Consequently, processing these bids will be impossible for more than a few tasks. Hence, some form of clustering algorithm is necessary to determine the clusters of tasks to bid on. The possibilities for such clustering algorithms are numerous [82]. The work shown in this chapter was published in 2002 by Dias and Stentz [37].

The clustering algorithm used in this work is chosen to ensure a span in size (from single-task clusters to a wholly inclusive cluster) and task membership (i.e. ensure that every task is included in at least one cluster). These properties are important because a robot cannot necessarily predict the interaction of the clusters it offers with the tasks of other bidders, and hence, needs to give the allocator ample flexibility in offloading tasks. The chosen clustering algorithm operates as follows:

1. Create a list of edges spanning all tasks on offer (N), where each edge joins two tasks and the cost of the edge represents the distance in cost space between the two tasks. A low edge value implies, but does not guarantee, that two tasks can be performed more cost-effectively together than apart.
2. Sort the edge list from lowest to highest cost.
3. Form the first group of clusters by creating a single-task cluster for each task on offer.
4. For cluster sizes ranging from 2 to N, recursively form new clusters by adding the next best available edge (an edge is unavailable if it is either already included in a previous cluster or if the edge connects two tasks which are not included in any of the previous clusters) to a cluster in the previous cluster-list. (Note, when new clusters are formed, all previous clusters are preserved). Thus, recursively form a forest of minimum spanning trees (MSTs) [32] ranging in size from 1 to N.

This algorithm can be applied in general to determine which tasks are best dealt with in clusters, without computing every possible cluster. Suitable variations of this algorithm (or others) can be chosen to enable multi-task negotiations in different task domains. The presented work is verified on a multi-depot distributed traveling salesman problem (TSP), and hence, the MSTs are decomposed into tours as follows. If a newly added edge breaks the continuity of the tour, the MST is adjusted by removing one of the edges connecting to the newly added edge and adding the necessary edge to preserve the

[5] Demange et al. [35] present a more detailed examination of multi-item auctions.

continuity of the tour with the least addition to the cost of the tour. Note that this change still preserves the bounds of the MST, which guarantees that the cost of the tour does not exceed twice the optimal cost. This holds true for metric cost spaces where the triangle inequality is preserved.

Allowing robots to include the offloading of an owned cluster when bidding to accept a new cluster of tasks further enhances the bidding capability of the robots.

5.4 Using Leaders

In this work we implement a preliminary version of the leader capability by means of a combinatorial exchange, as proposed in [40].

5.4.1 Combinatorial Exchange for Multi-Party Optimizations

A combinatorial exchange (a market where bidders can jointly buy and sell a combination of goods and services within a single bid) is chosen to enable multi-party optimizations for a team. A combinatorial exchange enables a leader to locally optimize the task assignments of a subgroup of robots and to potentially achieve a greater global cost reduction. Many researchers including Sandholm and Suri [95] have presented valuable insight on how to efficiently implement and clear combinatorial exchanges for E-commerce applications. However, many of these tools are relatively complex and are not used in this work for simplicity. Instead, the basic recommendation of searching a binary bid tree is applied. The chosen implementation for clearing the combinatorial exchange in this work is a depth first search on a binary tree where each node of the tree represents a bid and the binary aspect of the tree represents accepting or rejecting that bid. The tree is pruned to disallow accepting multiple bids from any single bidder, and to disallow exchanging of any single task more than once. Note that the pruning does not affect the solution except by improving the runtime.

The preliminary version of the leader role in the market approach is implemented as follows. A leader queries surrounding robots to discover what tasks they have to offer and their current states, and re-allocates tasks within the group using the combinatorial exchange mechanism. Note that this is just one way in which the leader can reduce the cost within the group (and thereby the global cost). Other schemes could involve the leader using different mechanisms to re-distribute tasks and even generating new tasks to coordinate the group more efficiently. Moreover, some tasks (for example, cooperative automated construction and cooperative maneuvering of large objects) may require tight coordination where a leader has to closely monitor the progress of individual team members and accordingly direct the efforts of other members of the team.

5.4.2 Competing Local Groups

When leaders are allowed to opportunistically optimize sub-groups, occasions could arise where two leaders are in competition for the services of the robots that overlap between the two groups. If a robot bids on tasks from both leaders, it could win both bids and be unable to perform them or find it unprofitable to do so. There are several ways to address this “synchronization” issue. For example, broken deals with a penalty can be allowed, or bids can be stamped with an expiration time during which they are valid and offers can be dealt with on a first-come-first-serve or last-come-first-serve basis. In the work presented here, the groups are allowed to negotiate in round robin fashion, thus forcing serial synchronization.

5.5 Implementation

The proposed multi-task and multi-party enhancements are developed and tested in a simulated distributed sensing task. A group of robots, located at different starting positions in a known simulated world, are assigned the task of visiting a set of pre-selected observation points. This problem is a variation of the multi-depot distributed traveling salesman problem, where the observation points are the cities to visit. Note that many multirobot application domains require an effective solution to the distributed traveling salesman problem. The costs are the lengths of the straight-line paths between locations, interpreted as money. Let $\mathbf{c_{ij}}$ be the cost for the $\mathbf{j}^{th}$ robot to visit the $\mathbf{i}^{th}$ city from the ($\mathbf{i\text{-}1}^{th}$) city in its tour (where the $\mathbf{0}^{th}$ city is the starting location).
The robot cost function for the $\mathbf{j}^{th}$ robot is computed as follows:

$$\mathbf{rcost(j)} = \sum_{\mathbf{i=1}}^{\mathbf{n_j}} \mathbf{c_{ij}}$$

where $\mathbf{n_j}$ is the number of cities in the tour for robot $\mathbf{j}$.

The team cost function is:

$$\mathbf{tcost} = \sum_{\mathbf{j=1}}^{\mathbf{m}} \mathbf{rcost(j)}$$

where **m** is the number of robots.

The team revenue and robot revenue functions are determined by the negotiated prices. All robots (bidders) adopt the same simplistic strategy of bidding a fixed 10% markup above the cost of completing the task. According to this strategy, if an announced task costs **c** to execute, a robot computes its bid as **1.1 c**. Thus, the robots bid for each city based on their estimated costs to visit that city. Similarly, if a robot offers up a task that will cost it **c** to execute, in an attempt to buy the services of another robot to complete that task, the maximum price it offers for this service is set as **0.9 c**.

Tasks and robot positions are randomly generated within a 100x100 world, and initial task allocations are made by randomly distributing the tasks among the robots. Heterogeneous robot capabilities are considered by restricting some robots' capabilities such that they can only process single-task (ST) deals, while other robots can process multi-task (MT) deals. Robots capable of playing leader roles are allowed the additional capability of performing multi-party (MP) optimizations via either a single-goods exchange or a combinatorial exchange, depending on their capability. Sections 5.5.1 through 5.5.4 describe in detail the scenarios of robots negotiating in the absence of a leader (TPST and TPMT) and the optimization scenarios with leaders (MPST and MPMT). Section 5.5.5 describes the scenario where robots have limited communication range and hence can only trade within subgroups.

5.5.1 Two-Party, Single-Task (TPST) Negotiations

In this case, once the initial random task assignments are made, each of the robots, *in turn*, offers all its assigned tasks to all the other robots, *in turn*. Thus, interactions are limited to two parties at any given time as illustrated in Figure 21.

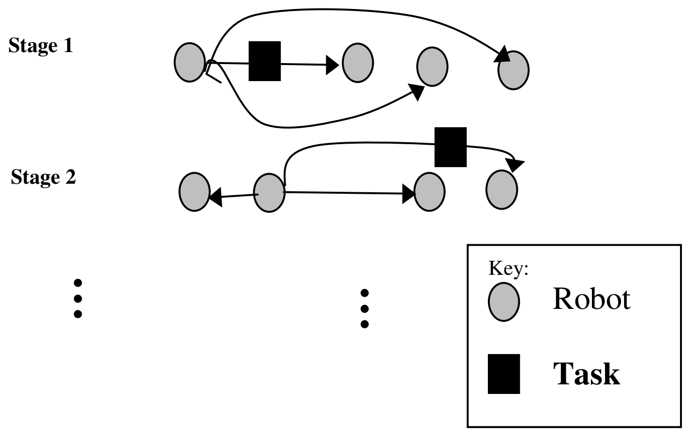


Figure 21: TPST Illustration

Each bidder then submits a bid for each task. In order to estimate the additional cost of inserting a task into its queue, the bidder uses the cluster generation algorithm described above to generate an MST with its current queue of tasks plus the offered task, and computes the cost difference between the resulting and original queues. The offerer accepts the most profitable bid it receives. The cost of the offerer's resulting queue is computed by removing from its queue the task that was transferred through the winning bid, clustering the remaining tasks using the clustering algorithm, and computing the cost of the resulting queue. Hence, in the TPST scenario, only single-task (ST) deals are considered, and pairs of robots continue to negotiate amongst themselves in round-robin fashion until no new, mutually profitable deals are possible. Therefore, negotiations cease once the system settles into a local minimum for the global cost function.

5.5.2 Two-Party, Multi-Task (TPMT) Negotiations

In this case, the previous case is repeated with clusters of tasks being the atomic unit of the negotiations as shown in Figure 22.

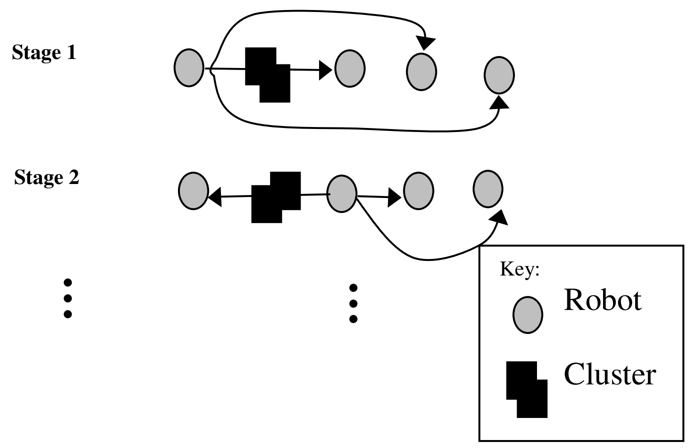


Figure 22: TPMT Illustration

That is, the initial assignments are followed by each of the robots, *in turn*, offering all of its assigned tasks to all the other robots, *in turn*. The robots then bid for clusters of these tasks. Once again, costs are computed by using the clustering algorithm to cluster all tasks under consideration and compute the cost of the resulting queues, and negotiations are always between two robots.

5.5.3 Leader Performing Multi-Party Single-Task (MPST) Optimizations

A leader, whose capability is restricted to dealing in single-task deals, is introduced in this case. The leader queries all the robots, and gathers all the tasks of all the robots along with each robot's state information. The leader then sets up an exchange by formulating single-task bids for the robots in the sub-group based on the gathered information. The exchange used in the MPST scenario is a single-task exchange (i.e. a single bid can contain buying of a single task and selling of another single task). The exchange is then cleared to maximize the leader's profit. These interactions are illustrated in Figure 23.

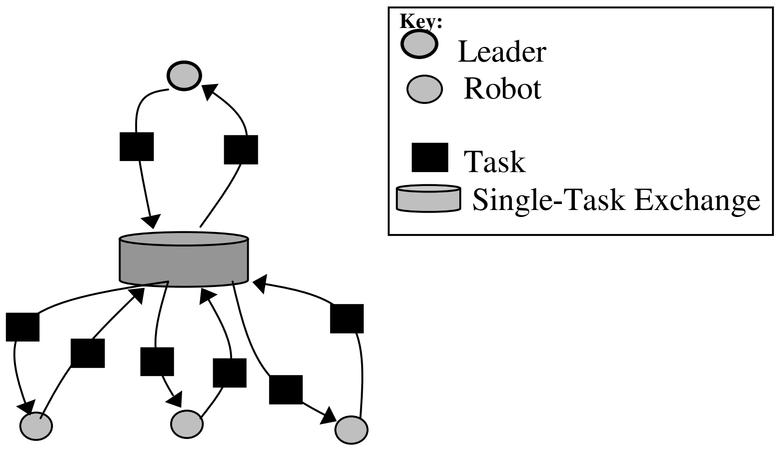


Figure 23: MPST Illustration

This process is repeated until the exchange cannot produce any further profit, and the corresponding task re-allocation is proposed to the sub-group of robots. If the leader's plan reduces the global cost, the resulting excess profit can be distributed among the entire subgroup (including the leader) such that the robots in the subgroup accept the leader's task re-allocation.

5.5.4 Leader Performing Multi-Party, Multi-Task (MPMT) Optimizations

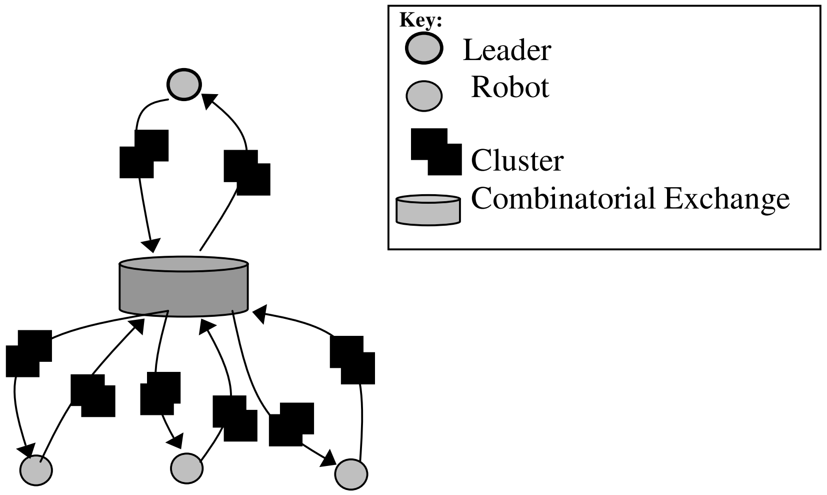


Figure 24: MPMT Illustration

Here, the previous case was repeated with the added capability of the leader to process MT bids as shown in Figure 24. That is, the leader sets up and clears a combinatorial exchange to determine the re-allocation of tasks. In a combinatorial exchange, clusters of tasks can be bought and sold within a single bid.

5.5.5 Multiple Competing Local Groups

This set of experiments involved 8 robots divided into 3 groups of 4 robots each (with the middle group overlapping the other two groups) and 10 tasks. Trading and optimization with leaders are restricted to within the subgroups. The robots are evenly spread throughout a 2000x2000 world and the cities (tasks) are randomly generated. Scenarios with and without leaders, and with ST-capable and MT-capable robots are considered.

5.6 Experiments

Experiment 5.1: Different negotiation strategies described above.
Figure 25 and **Figure 26** show the final tours of each robot for a 2-robot, 10-city TSP and a 4-robot, 10-city TSP respectively. In both figures, the robots are shown as circles and the cities are shown as squares.

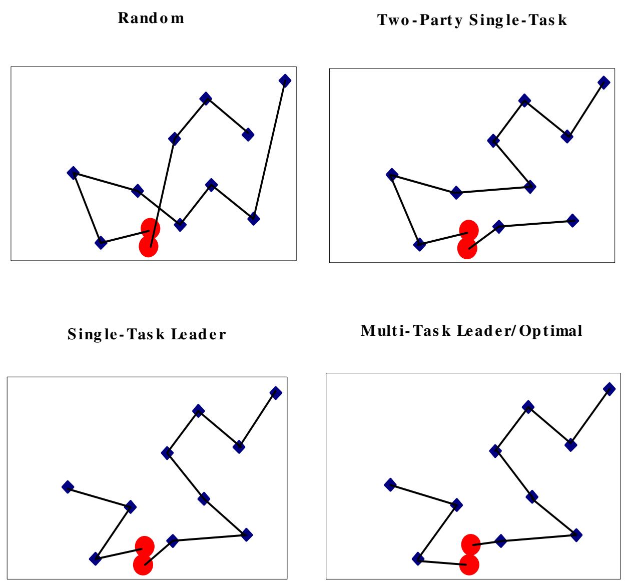


Figure 25: Solutions to a 2-robot, 10-task TSP with and without leader-optimization

	Cost	Itns	Improved	Opt. Error
Random	351	-	0.0 %	65.6 %
No Leader				
2 ST	256	2	25.9 %	21.4 %
2 MT	231	1	33.0 %	9.0 %
ST Leader	245	2	29.0 %	16.2 %
MT Leader	227	1	34.4 %	7.0 %
Optimal	212	-	38.6 %	0.0 %

Table 8: Performance averaged over 100 randomly generated 2-robot, 10-task TSPs

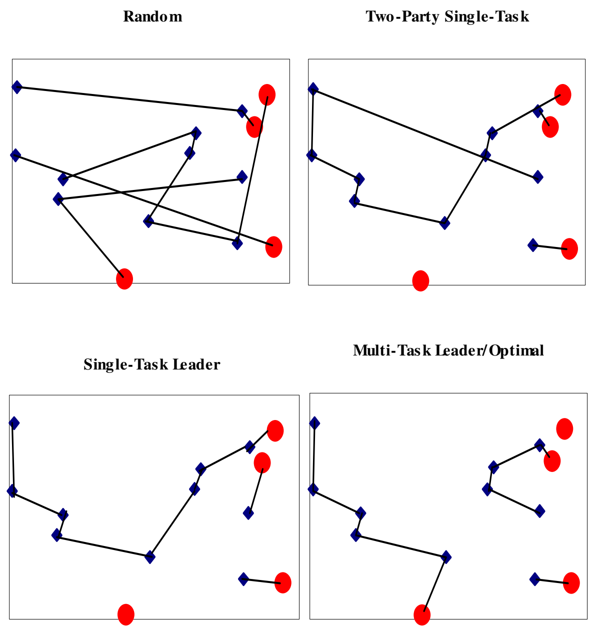


Figure 26: Solutions to a 4-robot, 10-task TSP with and without leader-optimization

The first illustration in each figure shows the tours after the initial random allocation of tasks. The second illustration shows the resulting tours after the robots have completed TPST deals and reached a local minimum in global cost. The third and fourth illustrations show the results of the MPST and MPMT scenarios. In the illustrated cases, the optimal allocation is reached in the MPMT scenario.

	Cost	Itns	Improved	Opt. Error
Random	411	-	0.0 %	124.6 %
No Leader				
4 ST	230	5	42.7 %	27.7 %
2ST+2MT	222	5	44.6 %	23.3 %
1ST+3MT	209	4	47.8 %	16.2 %
4MT	197	4	50.9 %	9.7 %
ST Leader	218	3	45.8 %	21.1 %
MT Leader	193	2	51.8 %	7.5 %
Optimal	183	-	-	0.0 %

Table 9: Results averaged over 100 randomly generated 4-robot (heterogeneous), 10-task TSPs

Table 8, Table 9, and Table 10 report the performance averaged over 100 randomly generated task distributions for the 2-robot-10-task case, the 4-robot-10-task case, and the 4-robot-20-task case respectively. As evident from these results, on average, an MT-capable leader can improve the profit of the group significantly. An ST-capable leader can only improve the profit of the group on average for groups of robots where there are at most 50% MT-capable robots.

	Cost	Iterations	Improved
Random	725	-	0.0%
No Leader			
4 ST	400	10	44.1%
2ST+2MT	388	9	45.7%
1ST+3MT	359	7	49.8%
4MT	336	5	53.0%
ST Leader	373	6	47.7%
MT Leader	322	3	54.9%

Table 10: Performance averaged over 100 randomly generated 4-robot (heterogeneous), 20-task TSPs

Experiment 5.2: Overlapping groups
Figure 27 and Table 11 illustrate preliminary results for the competing subgroup scenario. The subgroups of robots are circled in Figure 27, which depicts the results of a single run. Table 11 reports the performance averaged over 100 randomly generated task distributions. Again, the results show that on average the local optimization with leaders improves the global profit.

	Cost	Iterations	Improved
Random	9091	-	0.0%
No Leader			
4 ST	4598	8	48.9%
2ST+2MT	4379	9	51.2%
ST Leader	4312	6	52.1%
MT Leader	3687	6	58.9%

Table 11: Performance averaged over 100 randomly generated 8-robot (heterogeneous), 10-task TSPs with 3 overlapping groups of 4 robots each

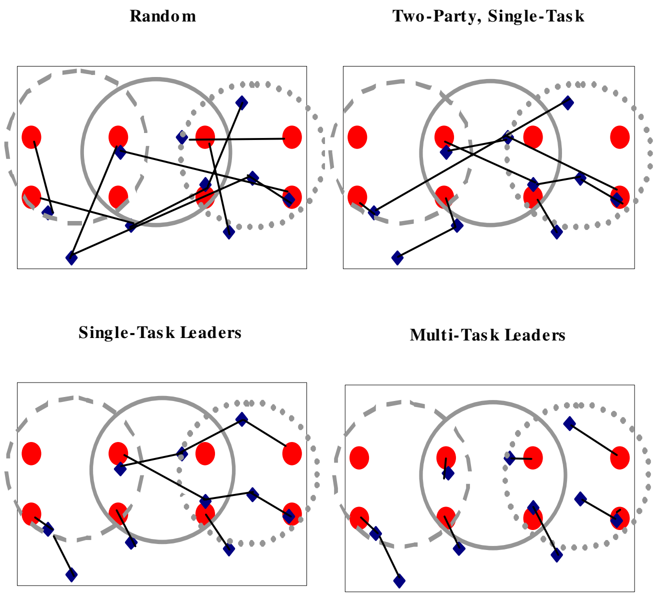


Figure 27: Solution for TSP with 3 overlapping subgroups of 4 robots each and 10 tasks

5.7 Limitations and Future Work

Presented results show that leaders can considerably reduce global costs in market-based multirobot coordination. Initial experiments for optimizing within robot sub-groups with leaders also proved promising. Future work includes implementing these capabilities on a robot team and further extensions of the market approach. Proposed enhancements include more detailed analysis of optimizing with leaders, dealing with time constraints, and experimentation with different task domains. The goal of this work is to produce an efficient and robust market-based multirobot coordination architecture.
The presented work only addresses scenarios where leaders run exchanges to optimize task allocation within a group of robots. Some leaders are also capable of clustering tasks and hence can conduct combinatorial exchanges. It is also possible to have combinatorial exchanges and leaders as distinct entities within the economy. For example, there could be a leader that simply clusters tasks and sells these *cluster plans* to a combinatorial exchange. Note that the leader is not selling the *actual* cluster of tasks—just a plan for which tasks to cluster. The exchange could then buy all of the component tasks, sell off the resultant cluster, and pay a fee to the leader. The presented results indicate that the benefit from the ability to cluster tasks and participate in multi-task negotiations exceeds the benefit from the ability to perform multi-party negotiations. Leaders could also use other approaches to generate plans for a subgroup of robots. Finally, a leader could simply act as a means of enabling trade between subgroups of robots who are otherwise unable to communicate, thus enriching the possible trades.

CHAPTER 6

Comparison of Three Coordination Mechanisms

THIS chapter presents first steps towards a comparative study of three multirobot coordination schemes that span the spectrum of coordination approaches. On one end of the spectrum is a fully centralized approach, where the leader runs on a more powerful computer and can produce optimal solutions. On the other end of the spectrum of approaches is a fully distributed behavioral approach with minimal planned interaction between robots. Finally, in the middle of the spectrum is the TraderBots market approach. The dimensions for comparison are chosen based on the characteristics identified in Chapter 1 of this dissertation as being important for multirobot coordination. Evaluations are based on the performance of each approach in accomplishing a distributed sensing task. This is a new direction of research and hence only preliminary results are presented in this chapter. Thus three selected approaches are implemented in simulation and compared across only two of the identified dimensions to date.

6.1 Motivation

The growing demand for robotic solutions to increasingly complex and varied problems has dictated that a single robot is no longer the best solution for many application domains; instead, teams of robots must coordinate intelligently for successful task execution. Driven by these demands, many research efforts have focused on the challenge of multirobot coordination. Chapter 1 in this dissertation presents a detailed description of multirobot application domains and their demands, and show that robot teams are more effective than a single robot in many application domains. Simply increasing the number of robots assigned to a task does not necessarily solve a problem more efficiently; multiple robots must cooperate to achieve high efficiency. The difficulty arises in coordinating many robots to perform a complex, global task. Dynamic environments, malfunctioning robots, and multiple user requirements add to the complexity of the multirobot coordination problem as detailed in Chapter 1.

Multirobot coordination mechanisms span a spectrum of approaches ranging from fully centralized approaches to fully distributed approaches. At one end of the spectrum, centralized approaches design the team so that a single robot or central computer acts as a "leader" and is responsible for planning the actions of the entire group. This methodology usually requires that the group members report most recent state information to the leader who uses this information to coordinate the group. The principal advantage of such centralized approaches is that they allow optimal planning since the decision-making agent has access to all relevant information when planning for the group. However, it is commonly considered that they suffer from several disadvantages including sluggish response to dynamic conditions, intractable solutions for large teams, communication difficulties, and the leader becoming a central point of failure. If the environment is highly dynamic, the leader may not be able to generate plans at a sufficiently fast rate to keep up with changes in state information from the group members. Furthermore, the robots, dependent on the leader for action-decisions, may not be able to avoid dangerous situations in time. Also, the ability to generate optimal solutions for a complex group-coordination problem diminishes drastically as the size of the group grows. Another important consideration in multirobot systems is communication. A fully centralized approach demands that all group members remain in communication with the leader at all times so that the leader can re-plan whenever a new situation is encountered. Fully centralized approaches also require high-bandwidth communication because the robots have to communicate all their state information to the leader on a regular basis so that the leader can generate informed plans. Finally, in a fully centralized system, the success of the group is tightly coupled to the performance of the leader. Hence, if the leader malfunctions or is disabled, the entire group becomes ineffective. For all of these reasons, fully centralized approaches seem best suited for small groups of robots operating in controlled, static environments with global communication.

On the other end of the spectrum, fully distributed, reactive approaches address many of the problems inherent to a centralized approach by distributing the planning responsibilities amongst all members of the team. Each robot operates independently, relying on its local sensory information and planning its actions accordingly. Any cooperation between team members is often fortuitous and each robot tends to act as though it is the only member of the team. Many research efforts have modeled such distributed systems inspired by biology. The principal drawback of these reactive approaches is that they often result in highly sub-optimal solutions because all plans are based solely on local information and hence, efficient execution of the global goal, by coordinating the resources of the team, isn't prioritized in the local planning. However, to their advantage, reactive approaches are often very simple to implement and overcome many of the disadvantages of fully centralized approaches.

Recently, economy/market-based multirobot coordination has gained popularity. The general concept of these market-based approaches is that the robots act largely independently in terms of planning for themselves, but are able to take into account team resources by trading tasks with team members. Communication is limited to offers and awards of tasks, and bids for tasks, and hence often consists of low-bandwidth communication. Economic approaches can also maintain the benefits of distributed approaches and opportunistically incorporate pockets of centralized optimization [37]. However, economic approaches are not without their disadvantages. Negotiation protocols, mapping of task domains to appropriate cost functions, and introducing relevant de-commitment penalty schemes can quickly complicate the design of a coordination approach. Furthermore, some negotiation schemes can drastically increase communication requirements. Thus, all of these factors must be considered when designing a market-based approach.

The goal of this chapter is to report on a new direction is our research that aims to conduct a meaningful comparison between three approaches that span the spectrum of multirobot coordination mechanisms discussed above. The three chosen approaches are a fully centralized approach, a fully distributed behavioral approach and our TraderBots market-based approach. Rabideau et al. [85] conduct a similar comparative study between a centralized planner that does not guarantee the optimal solution, a distributed planner, and a single-round-auction Contract Net approach [107]. They evaluate these coordination schemes for a simulated 3-rover, 20-rock, geological science scenario where rovers sample spectra of rocks on Mars. They conclude that the Contract Net approach performs best but takes up the most CPU cycles. The work presented in this paper differs in that the market approach used is not limited to a single round of bidding, the chosen approaches span the spectrum of those currently used, and the chosen approaches are evaluated along many different dimensions. Also, the comparison is carried out not only in simulation, but also on a robotic system.

An early paper by Rus et al. [90] evaluates four approaches for a furniture-moving task: a global planning approach, a local planning approach, a behavioral approach with communication, and a behavioral approach without communication. The goal of the paper however is not to compare methods of coordination, but instead to investigate what manipulative tasks can be done by robots without global control, communication, planning, or synchronization.

Another recently published multirobot comparative study is that of Gerkey and Matari, ([50], [51]). This work aims to compare different multirobot coordination schemes in terms of complexity and optimality. However, the proposed framework for comparison makes the limiting assumption that only one task can be assigned to any robot. Thus, the framework limits the performance and hence the analysis of some of the approaches under consideration, where the approaches do not limit task assignment to a single task per robot.

Bererton [11] proposes a comparison between the Guestrin Gordon MDP solution to multirobot coordination and a market approach inspired by the TraderBots approach. The proposed comparison is in the adversarial laser-tag game. However, to date, none of the proposed comparative work is published. In previous work, Bererton and Khosla [9] present comparison results in simulation between a repairable team of robots and a non-repairable team, and show that repairable robots have a higher average mean time to failure for larger team sizes.

Finally, Parker's seminal work on fault tolerant multirobot coordination using the ALLIANCE architecture [83] devotes a section to a comparison with negotiation-based coordination approaches. While no comparative results are presented, Parker argues that the principal disadvantage of negotiation-based approaches is that they have not been proven on robotic systems. This dissertation addresses Parker's criticism by presenting a detailed study and implementation of a negotiation-based multirobot coordination approach. Furthermore, this chapter extends Parker's comparative analysis, and presents an implemented comparison between the TraderBots negotiation-based approach, a centralized approach, and a behavioral approach.

The comparison work in this chapter spans many more dimensions than published in any previous work and presents the first comparative analysis implemented on robots. Comparisons such as these are extremely useful for determining the strengths and weaknesses of different approaches when choosing a suitable approach for a given application.

6.2 Dimensions of Comparison

The scope of this comparison is limited to evaluating the performance of three approaches that span the spectrum of multirobot coordination mechanisms as described in the previous section. The selected approaches will be evaluated along the dimensions identified in Chapter 1 as being important to multirobot application domains. These dimensions are listed below, along with the corresponding questions to be answered by the comparison. (Note that this is a new direction in our research and hence the presented results are limited to comparisons along only two of the identified dimensions).

§ **Robustness:**
Can the approach deal with robot death and partial robot malfunction?

§ **Speed:**
How quickly can the approach respond to dynamic conditions?

§ **Efficiency:**
How does the efficiency of the solution vary for this approach?

§ **Information:**
Can the approach handle dealing with an unknown and changing environment, by relying on sensor feedback from robots to gather information about the world?

§ **Communication:**
Can the approach deal with limited-range and limited-bandwidth communication between robots?

§ **Resources:**
Does the approach reason about limited resources?

§ **Allocation:**
How efficiently does the approach allocate tasks?

§ **Roles:**
Does the approach allow for efficient adoption of roles?

§ **New Input:**
Can the approach handle new assignments and changes to current assignments from the operator during execution?

§ **Fluidity:**
Is the approach able to accommodate the addition/subtraction of robots during operation?

§ **Heterogeneity:**
Can the approach handle heterogeneous teams of robots?

§ **Extensibility:**
Is the approach easily extendable to accommodate new functionality?

§ **Flexibility:**
Is the approach easily adaptable for different applications?

§ **Tight-Coordination:**
Can the approach handle tightly coordinated tasks?

- **Scalability:**
  How well does the approach scale with the number of robots?
- **Learning:**
  Can the approach be integrated with learning techniques for on-line adaptation to specific application environments?
- **Implementation:**
  Can this approach be validated via implementation on a robot team?

This chapter presents a comparative analysis of three selected approaches that span the spectrum of solutions to the multirobot coordination problem across the 17 dimensions listed above. While most of the dimensions are explored by an implementation in simulation or an implementation on a robotic system, some of the dimensions are examined by examples of other implementations, or by argument due to the nature of the characteristic, or the complexity of the required implementation to investigate that characteristic. Note that the 17th dimension is necessarily covered due to all three approaches being implemented both in simulation, and on a robotic system.

6.3 Scenario And Implementation Details

The chosen application is that of a distributed sensing problem where robots are tasked with gathering sensory information from various designated locations of interest. This translates into a version of the multirobot traveling salesman problem (MTSP) with paths instead of tours (i.e. without the requirement that robots need to return to their starting locations) and where all the robots can start from different base locations – this is known as the multi-depot vehicle routing problem. Hence, the tasks can be considered as cities to be visited where the costs are computed as the time taken to traverse between cities. A task is completed when a robot arrives at a city. The global task is complete when all cities are visited by at least one robot. The global cost is computed as the summation of the individual robot costs. Thus, the goal is to complete the global task while minimizing the number of robot-hours consumed. The implementation of the three approaches in simulation and on a team of robots is described next.

Due to the preliminary nature of this work, the comparisons are analyzed via an implementation of the three chosen approaches in simple simulation. In the simulation implementation, each robot travels by taking a fixed step (based on its speed) each control-loop-cycle along the straight line connecting its current position to the next city on its task queue. When the robots are not executing tasks, they remain stationary at their current locations. Implementation details for each of the three approaches are presented below. The key challenge in the implementation is to allow each approach to maximize its benefits and minimize its disadvantages while maintaining the fundamental philosophy of the approach and while keeping the playing field leveled in order to make a useful comparison. There are numerous ways to implement such a comparison, and no matter which implementation is chosen, there will be limitations. Hence, this work attempts to choose the least contentious methodology for the comparison. Thus, the team of robots is kept the same across approaches. That is, the comparison is based on the common scenario where a team of robots is available for executing a task and the comparisons are based on coordinating the team in different ways to successfully execute the task. In order to keep the playing field level, the algorithms used for the centralized approach are also used for individual tour queue optimization in the market and behavioral approaches. This is explained in more detail in the following sections.

- **Centralized Approach**
  The centralized approach is implemented with one robot in the team acting as the leader who does all the planning for the group. The centralized planner is implemented using an exhaustive depth-first search (DFS) with some pruning, which produces an optimal solution for the task allocation problem. The robots transmit state information to the leader, the leader generates a plan based on this information, and the team executes the instructions of the leader to the best of its ability. In the centralized approach all robots do not start execution until the leader completes planning for the group. Thus, it is assumed that the leader alone does all the planning-related computation.

- **Behavioral Approach**
  The behavioral approach is implemented with each robot planning and executing all its actions independently. Each robot uses the same DFS optimization algorithm used in the centralized approach to plan the optimal scheduling of its own tasks. The robots all plan to execute all tasks since no team coordination is planned. However, to level the playing field somewhat, as each robot completes a task, it announces the completion of that task to all the other robots. When a robot learns that a task has been completed, it removes that task from its queue and re-schedules its task-queue accordingly. In the behavioral approach, each robot stops execution until its own planning/re-planning is complete and then resumes execution of any remaining tasks.

- **Market (TraderBots) Approach**
  The market approach is implemented with an OpTrader agent (the OpTrader is assumed to reside on one of the robots) that is responsible for translating and trading on behalf of the operator. The robots can all plan for themselves using the same algorithm as in the behavioral approach. The robots can also trade their tasks with other robots during execution. In this implementation, only single-task trades are allowed and hence the market solution can get stuck in local minima. In the market approach, execution is not started until initial negotiations with, and allocations by, the OpTrader are completed. Thereafter the robots keep trading tasks during execution as opportunities for profitable trades arise. Each robot has a specified frequency at which it trades. In this implementation, all robots trade with the same frequency (once every 5 steps), but at staggered intervals. The OpTrader uses a greedy algorithm for auction clearing, where the *N* best allocations are made each round, one award per bidder, where *N* is the number of robots in the team. Any tasks not auctioned off in a round are re-announced and cleared during subsequent rounds of auctioning. During the inter-robot trading, each robot announces all of its current tasks for auction, and awards the task corresponding to the single most profitable bid it receives.

All interactions among the robots occur in a synchronous manner in this simulation. Each robot is allowed a time-slice within a large time-cycle to carry out its computation and execution. The experiments performed, using this implementation, are described below in section 6.4.

6.4 Comparisons

This section describes the relevant comparisons of the three chosen approaches along the dimensions identified in section 6.2. For all the experiments, a fixed set of randomly generated observation tasks are assigned to the robots, and the global task is complete as soon as all the observation points are visited by at least a single robot. If for some reason the global task fails, a percentage of completion is determined using the percentage of observation points visited. If a generated observation point is inaccessible due to obstruction, the inability to complete that task is not considered a failure. Robots will give up trying to reach a goal if the accrued cost to reach the goal exceeds the estimated cost by a significant percentage.

All experiments used a set of randomly generated tasks in a 2000m by 2000m obstacle-free world. The robots start off in random positions within this world. The operator announces all tasks to the group at the beginning. In all the approaches, the robots stop execution during their planning stage.

6.4.1 Heterogeneity:

Experiment 8.1: In keeping with the chosen scenario, heterogeneity is introduced into the system by allowing different robots to move at different speeds in simulation.

The goal in this comparison is to study the performance of the three approaches when the group of robots is heterogeneous. The chosen aspect of heterogeneity is robot speed. Since the optimized quantity is the number of robot-hours, the speed of each robot will affect the quality of the solution.

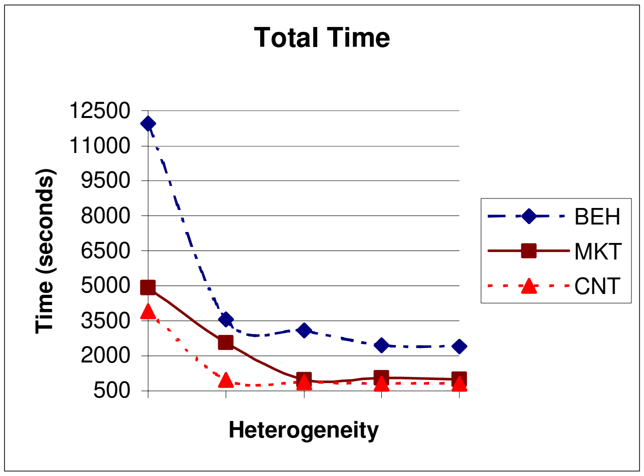


Figure 28: Heterogeneous Teams

Figure 28 shows the variation in total time for solutions produced by the three approaches for teams of 4 robots with varying degrees of heterogeneity where robots can have speeds of either 1m/s or 5m/s. On the far left all 4 robots have the minimum speed of 1m/s and on the far right all 4 robots have the maximum speed of 5 m/s. The 3 points in the middle represent the cases of mixed speeds where the points from left to right indicate scenarios where 1, 2, and 3 robots have a speed of 5m/s respectively, while the others have speeds of 1m/s.

	5 Runs	100 Runs
BEH	3451	3049
MKT	1834	1545
CNT	1206	1129

Table 12: Results For Heterogeneous Teams With Speeds Randomly Allocated Between The Minimum Speed Of 1m/S And The Maximum Speed Of 5m/S

Table 12 shows a summary of results where each robot is assigned a random speed between the max and min. All three approaches are able to improve cost by taking into account heterogeneity. The performance ranking of the three approaches remains the same as in the previous section.

6.4.2 Scalability:

Experiment 8.2

The goal in this comparison is to study the performance of the three approaches for different team sizes. It is assumed that one of the robots acts as the leader in the centralized approach and as the OpTrader in the market approach. All robots are otherwise assumed to be homogeneous.

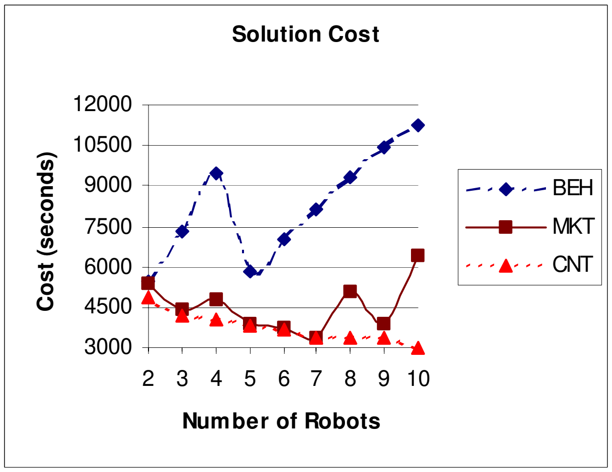


Figure 29: Cost Comparison For A Single Run

Figure 29 shows a plot of the global cost versus the team size for a single run each for different team sizes ranging from 2 to 10. The plot shows data gathered for the same randomly generated set of 10 tasks. As expected the optimal solution cost is achieved by the centralized approach. The behavioral method produces fairly sub-optimal solutions as predicted, and the market method sits in between.

(Note that this comparison was only run for one set of randomly generated task distributions because of the high time consumption of the centralized method for larger team sizes.)

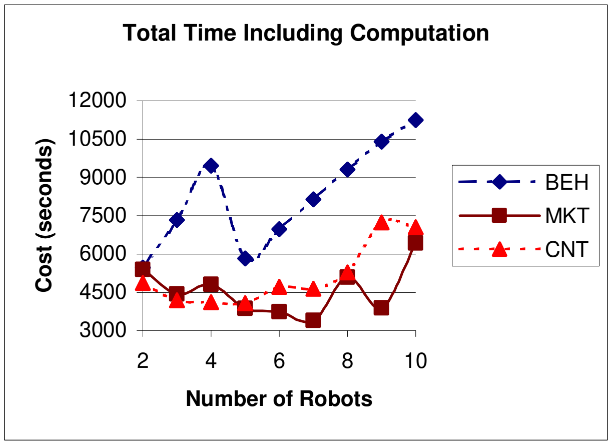


Figure 30: Total Time Comparison For Single Run

However, if we observe the total time taken, including computing time, then the centralized approach exceeds the cost of the solution generated by the market approach for team sizes exceeding 5, as seen Figure 30. The behavioral method still exceeds the cost of the other two approaches.

The same experiments were then repeated averaged over 100 runs. Here too we see a similar trend when looking at the solution cost (see Figure 31) and the total time taken including the computation time (see Figure 32).

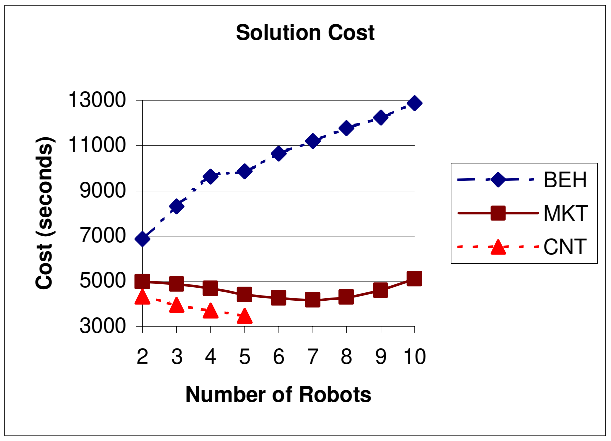


Figure 31: Solution Costs Averaged Over 100 Runs

Note once again that we were only able to generate results for up to 5-robot teams for the centralized approach due to time constraints. Hence we were unable to observe how close the total time taken by the centralized and market methods compare as the number of robots grows. However, based on time performance for a smaller number of runs, it is highly probable that the market approach matches if not exceeds the performance of the centralized approach for larger numbers of robots.

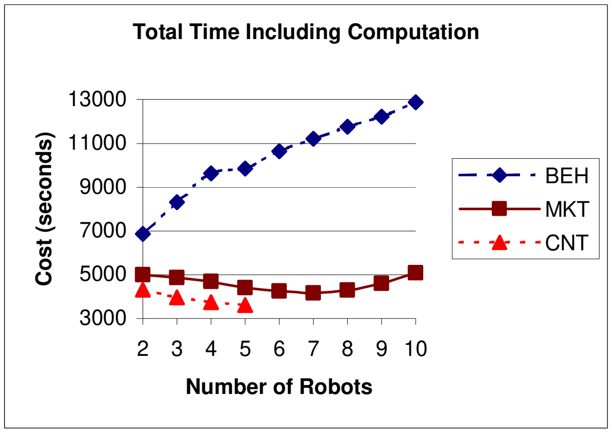


Figure 32: Total Time Averaged Over 100 Runs

Another interesting aspect to notice is that the global cost increases with the number of robots for the behavioral approach. This is because there are more robots moving towards the same goals until the robot that gets there first eliminates them from the task queues. Interestingly though, if the average execution time per robot is monitored (see Figure 33) it is clear that this reduces with the number of robots for all three approaches. Thus, it is clear that all three approaches are tending towards reducing the number of robot-hours spent. Thus far, it appears that the centralized and market approaches perform best. However, one benefit of the behavioral approach is that it requires very low computation time. Therefore, it is informative to observe the average computation time per robot for the presented results. Figure 34 shows the variation of the computation time for the different approaches plotted against the number of robots. This figure clearly shows the centralized method at a disadvantage compared to the other two methods. Although the difference is small in comparison to that with the centralized approach, the behavioral approach on average takes slightly less time than the market approach to generate solutions. This makes sense because in addition to computing the optimal arrangement of its tasks, the robots in the market approach spend some computation time on announcing, clearing, and submitting bids to task auctions.

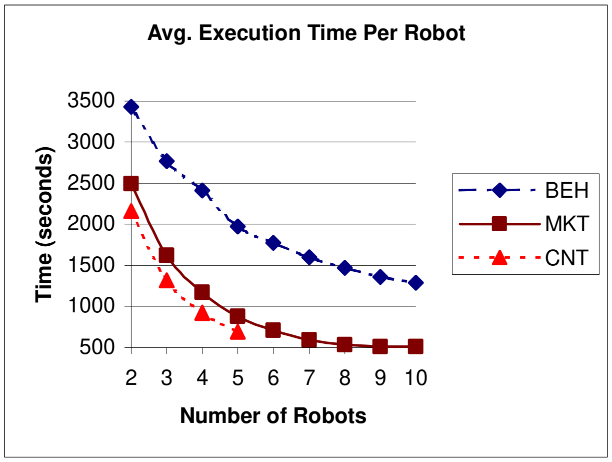


Figure 33: Average Execution Time Per Robot

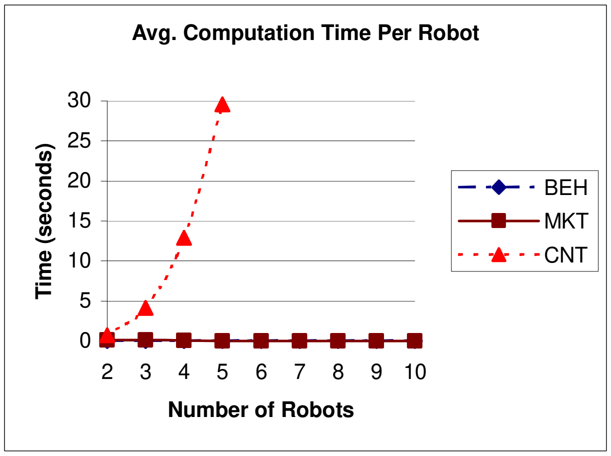


Figure 34: Average Computation Time Per Robot

Thus, overall, the market method can provide solutions with significantly better global costs than the behavioral approach using computation time significantly less than the centralized approach. Note however that for applications where the optimal solution must be generated, the centralized approach will suit best. Also, for applications where optimality of the solution is of no consequence, the behavioral method is fast and simple and hence becomes the approach of choice. Note that the market approach can generate better solutions if allowed to trade clusters of tasks [37]. However this improvement will increase computation time somewhat.

6.5 Limitations and Future Work

This chapter presents a comparative study between three multirobot coordination schemes that span the spectrum of coordination approaches; a fully centralized approach that can produce optimal solutions, a fully distributed behavioral approach with minimal planned interaction between robots, and a market approach which sits in the middle of the spectrum. Several dimensions for comparison are proposed based on characteristics important to multirobot application domains, and experiments to compare the three approaches along each of the identified dimensions are presented. The study is aimed at providing useful information for determining the strengths and weaknesses of these approaches when choosing a suitable approach for a given multirobot application. The comparative study thus far is limited to comparisons along only two of the identified dimensions, by implementation only in simulation, by simple implementations of the three approaches, and by a single fixed scenario. Future work will address these limitations and will also extend the methodology to perform comparisons that were beyond the scope of this dissertation, including comparisons with more sophisticated implementations of the three coordination approaches and a wider variety of scenarios.

CHAPTER 7

Overall Performance of TraderBots

THIS chapter reports the overall performance of the TraderBots approach. Each of the 17 characteristics identified in Chapter 1 are revisited here and examined with respect to the TraderBots approach. This chapter serves to summarize the overall capability of the TraderBots approach and to illustrate its ability to satisfy all identified requirements for successful multirobot coordination in dynamic environments. Some experiments showcasing the versatility of the approach are also presented here.

7.1 Satisfying the Required Characteristics

Chapter 1 identifies several characteristics required for successful multirobot coordination in dynamic environments. This section examines how the TraderBots approach addresses these identified requirements.

§ **Robustness**
Since the TraderBots approach has no single leader it has no single point of failure. Note that although a single agent (OT) could represent the operator, if one OT fails for some reason, a new one can be initiated and it can gather information from the robot traders (RTs) to assess their current operational status. Thus, although some information may be lost with the failure of agents or robots, the system performance will degrade gracefully and the robot team will always aim to complete the assigned tasks with the functioning resources if possible. Different means of ensuring robustness in the TraderBots approach are explored in Chapter 4 of this dissertation.

§ **Speed**
Since each robot in the TraderBots approach will make decisions for itself, the system as whole can respond more quickly to dynamic conditions than if new information had to be conveyed to a central leader agent and the robot had to wait for a new plan from its leader before acting.

§ **Efficiency**
The TraderBots approach can respond to dynamic conditions efficiently since the prevailing conditions are taken into account during negotiations and since the traders continue to seek profitable new trades and re-trades frequently. For example, if new conditions arise which make an assigned task no longer profitable for a robot it will try to sell that task to another robot who may find it profitable due to/in spite of the new conditions. Different means of encouraging efficiency in the solutions produced by the TraderBots approach are examined in Chapter 5 of this dissertation.

§ **Information**
The TraderBots approach does not require a-priori complete information. Since trading occurs frequently and repeatedly, new information can be taken into account during negotiations as it is received. At any point in time the robots will use the best information they have to plan and execute assigned tasks in an opportunistically efficient manner. Most experiments performed to date with the TraderBots approach, and presented throughout this dissertation, do not assume the availability of a-priori information about the environment.

§ **Communication**
TraderBots rely on communication to trade. However, the robots will execute opportunistically efficient plans based on the available communication. Hence, the tasks will still be completed, albeit more inefficiently, even if the robots cannot trade – i.e. the system degrades gracefully with degradations in communication. At no point is the system reliant on perfect communication due to the requirements of the TraderBots approach. The ability of the TraderBots approach to deal with imperfect communication is examined in greater detail in Chapter 4 of this dissertation.

§ **Resources**
The available resources on the robots are always taken into account when estimating costs and savings for tasks being traded in this approach. Furthermore, the TraderBots approach allows for an executive that can monitor the addition and depletion of resources and roles on the robots and notify the trader of changes to the robot capability so that the current capabilities of the robot are taken into account during negotiations. Traders can also take into account their opportunity costs when estimating bids. Thus, a robot with a more specialized capability will bid higher on a risky mundane task even if it can do it and hence will be less likely to be assigned a task not suitable for its expertise.

§ **Allocation**
In the TradeBots approach, factors such as robot capabilities, risks, and task constraints can be considered when estimating costs and savings for tasks during trading. Note that different cost and revenue functions can result in different allocations of tasks. Hence, designing appropriate cost and revenue functions can be very important in this approach. However, imperfect cost and revenue functions could be altered to some extent via learning. Task allocation in the TraderBots approach has been evaluated in comparison to numerous other approaches with highly favorable results. A comparison to the optimal solution for simpler allocation problems is shown in Chapter 5 of this dissertation. Comparisons to allocations made by a centralized approach and allocations made by a reactive approach are examined in Chapter 6 of this dissertation. Finally, comparisons to greedy and random allocations are shown in the results section of this chapter.

§ **Roles**
In this approach, robots are not restricted to being able to play a limited number of roles at any given time. Robot capabilities are derived at any time from available resources at that time and commitments in the future. If a robot needs to switch out of a current role, it can decide to do so by selling the tasks that require it to play that

role or deciding to default on its commitment and paying a penalty. Implementation of role management is beyond the scope of this dissertation. However, an action selection method such as Stroupe's MVERT [114] can be used to efficiently manage multiple roles in conjunction with the Trader Bots approach.

§ **New Input**
In many dynamic application domains, the demands on the robotic system can change during operation. Hence, it may become necessary to assign new tasks, cancel previously assigned tasks, or alter existing tasks. In the TraderBots approach, new task assignments can be handled if they can be communicated to a capable robot, and changes to existing tasks and/or cancellations can be handled as long as it is communicated to the relevant robot in time for the change to be made before execution. Some changes can be accommodated even if the robot is notified during execution of the altered task, depending on how far the execution has progressed. The ability of the TraderBots approach to successfully handle new input is demonstrated in experiments reported at the end of Chapter 3 and in the results section of this chapter.

§ **Fluidity**
If a new robot enters the team, it can join the team activities by participating in any on-going trading. If the failure of a robot can be detected (by means of a heart-beat or occasional pinging of the robot) all tasks assigned to that robot could be re-auctioned to other robots thereby assuring the completion of tasks as long as the necessary resources are available. If a robot's malfunction is only partial, it can subcontract portions of a task it can no longer complete to other capable robots, or transfer the whole task to another robot via an auction. Some experiments demonstrating the TraderBots ability to accommodate changes in team size during operation are shown in Chapter 4, Chapter 6, and the results section of this chapter.

§ **Heterogeneity**
The TraderBots approach makes no assumptions about the heterogeneity or homogeneity of the team. Each trader will make decisions about trading based on the resources of the robot. Hence the approach works well on heterogeneous teams as well as on homogeneous teams. Preliminary experiments demonstrating the TraderBots approach applied to a heterogeneous team are reported in Chapter 6 of this dissertation.

§ **Extensibility**
The TraderBots approach allows for the addition and subtraction of different levels of functionality in a modular fashion since all of its components are implemented in a modular fashion. The tasks, resources, roles, cost functions, reward functions, and trading capabilities can be tailored to the specific needs of each application.

§ **Flexibility**
The TraderBots approach is not specifically geared to a single application domain. While the modularity of the TraderBots approach allows the approach to be applied to different task domains, an on-line learning capability can enhance the flexibility of the approach by autonomously tuning the market parameters to adapt to the application domain. Many publications, including this dissertation, provide

instructions and guidance for implementing the TraderBots approach for different applications. One current implementation of TraderBots is being improved to include a user interface and tools to allow plug-and-play system that can be used easily in many application domains.

§ **Tight-Coordination**
This dissertation does not specifically address tight coordination between multiple robots. However, the TraderBots approach can accommodate tasks that require tight coordination in different ways including those applied by Kalra and Stentz [67] and Gerkey and Matari, [49]. The basic idea is that the overall tightly coordinated task can be decoupled into subtasks and auctioned out to different participants. The task description can include information about the task decomposition by representing tasks as decomposable trees as shown by Zlot and Stentz ([128], [129], [130]). One of the subtasks of the tightly coordinated task could be a monitoring task that allows a robot to watch the progress of the task and stop execution if one or more participants fail or get out of synchrony. This increases the robustness of the task execution.

§ **Scalability**
While most application domains don't require large numbers of robots, and limits in resources often prevent deployment of large robot teams, scalability is an attractive quality and needs to be understood for any approach before it is adopted. TraderBots does not intrinsically place limits on the number of robots it can accommodate due to its distributed nature. However, some auction clearing algorithms can take a lot longer based on how many robots participate in the auction, and hence, some optimization techniques become less efficient as the number of robots on the team grows. However, even with a large team of robots, the TraderBots approach is applicable and can produce robust and reasonably efficient solutions if sufficient resources are available. A preliminary examination of the scalability of the TraderBots approach is reported in Chapter 6 of this dissertation.

§ **Learning**
The TraderBots approach is designed as a generalized approach to multirobot coordination and hence its application to specific domains usually requires some parameter tuning. Some of these parameters can be tuned automatically via an on-line learning module. A learning module can enable the robots to autonomously tune certain important parameters according to the prevailing conditions, and thereby improve efficiency in performance. Learning techniques can also help with recognizing partial malfunctions by building correlations between failures and specific types of assignments that require a small set of common resources.

§ **Implementation**
The TraderBots approach has been successfully implemented, tested, and proven in simulation and on a robot team. Some initial experiments that serve as a proof of concept are examined next. More detailed experiments and implementation details are reported in Appendix 1 of this dissertation.

Thus, the TraderBots approach is designed to satisfy all of the characteristics necessary to successfully coordinate multiple robots in dynamic environments.

7.2 Experiments, Results, and Discussion

Many application domains demand high quality performance from multirobot systems. Chapter 1 identifies several requirements for a successful multirobot coordination approach. Some of these requirements are used to evaluate the versatility of the current implementation of the *TraderBots* approach. The following characteristics are examined: the robustness to malfunctions, the ability to execute a task with incomplete information about the environment, the ability to deal with imperfect communication, the ability to dynamically handle new instructions from the operator during execution, the flexibility to execute different types of tasks, and the ability to accommodate the addition of a robot to the team during operation. The chosen application is a distributed sensing problem where robots are tasked with gathering sensory information from various designated locations of interest. Figure 4a shows a graph of the 25mx45m area in which 30 cities (tasks), illustrated as triangles, are assigned to a team of robots to visit. Figure 4b shows a photograph of the cluttered dynamic environment, graphed in Figure 4a, where the reported experiments were carried out. This translates into a version of the traveling salesman problem (TSP) with the robots being represented by multiple salesmen following paths instead of tours (i.e. without the requirement that robots need to return to their starting locations) and where all the robots can start from different base locations – this is known as the multi-depot traveling salesman path problem (MD-TSPP). The tasks can be considered as cities to be visited where the costs are computed as the time taken to traverse between cities. A task is completed when a robot arrives at a city. The global task is complete when all cities are visited by at least one robot. The global cost is computed as the summation of the individual robot cost, and the goal is to complete the global task while minimizing the number of robot-hours consumed. When the robots are not executing tasks, they remain stationary at their current locations.

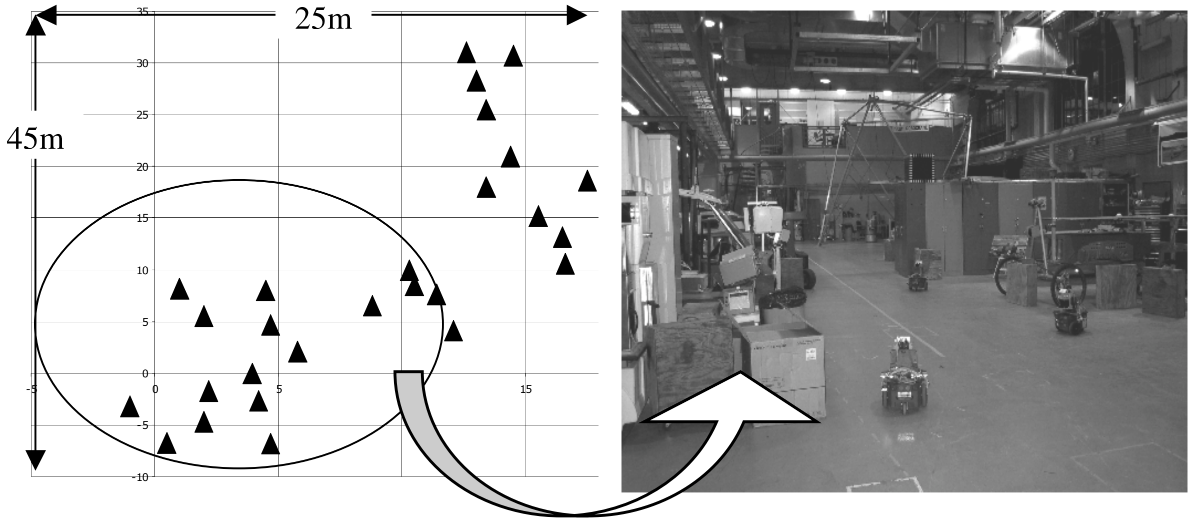


Figure 35: Distributed Sensing Tasks and Experimental Environment

Each robot is responsible for optimizing its own local schedule (i.e. given a set of tasks, the robots attempt to find the optimal TSPP solution to their local problem instance). In general, the TSPP is NP-hard, so approximation algorithms are often used when large problem instances are encountered. Additionally, the problem encountered is an online variant of the TSPP – cities are arriving whenever a robot is awarded a task in an auction and are being removed whenever a task is traded to another robot. When adding a task to the tour, it is inserted into the tour at the location that results in the smallest increase in marginal cost. Insertion heuristics have been shown to have constant factor approximation guarantees for some point orderings, but in general they have a performance guarantee of ($\lceil \log n \rceil + 1$) for n-city tours [6]. Tours are also optimized as a whole whenever tasks are added or removed. If the number of tasks is at most 12, the optimal solution is computed using a depth first search-based algorithm. If the number of tasks exceeds 12, computing the optimal solution is too time-intensive, and hence a minimum spanning tree-based 2-approximation algorithm is used [33] if the resulting tour has a lower cost than the current tour. Tour optimization is also performed whenever a task is completed or failed as costs between cities may have changed due to new map information from recent sensor readings. In the implemented TSPP scenario, all valuations are derived from inter-point distance costs. These costs are estimated using a D* path planner [108] with the robot's local map as input.

Experiment 1: This experiment measures the performance, averaged over a set of 3 runs, of the nominal case for 4 robots engaged in a distributed sensing task.
Experiment 2: This experiment investigates how a partial robot malfunction (simulated by killing the *TaskExec* process) at a random time during a run affects the nominal performance. Reported results are averaged over a set of 3 runs for 4 robots engaged in a distributed sensing task.
Experiment 3: This experiment investigates how communication failures (simulated by deleting 10% of messages passed between robots) affect the nominal performance. Reported results are averaged over a set of 3 runs for 4 robots engaged in a distributed sensing task.
Experiment 4: This experiment investigates the effect of new operator input during execution (where 4 random tasks are cancelled at random times during execution). Reported results are averaged over a set of 3 runs for 4 robots engaged in a distributed sensing task.
Experiment 5: This experiment investigates the performance for the 3-robot case using a random allocation for comparison with the *TraderBots* allocation. Reported results are averaged over a set of 3 runs for 3 robots engaged in a distributed sensing task.
Experiment 6: This experiment investigates the performance for the 3-robot case using a greedy allocation for comparison with the *TraderBots* allocation. Reported results are averaged over a set of 3 runs for 3 robots engaged in a distributed sensing task.
Experiment 7: This experiment investigates the nominal performance for the 3-robot case, for comparisons in efficiency with random and greedy allocations reported in experiments 5 and 6. Reported results are averaged over a set of 3 runs for 3 robots engaged in a distributed sensing task.
Experiment 8: This experiment investigates the effect of adding a new robot to the team at a random time during execution. Reported results are averaged over a set of 3 runs for a distributed sensing task.

Experiment 9: This experiment investigates the flexibility of the implemented *TraderBots* approach by applying it to an exploration task where a set of 4 robots dynamically generate locations to be visited in order to cooperatively build a map of a previously unknown world. Reported results are averaged over a set of 3 runs, each of 5-minute duration.

	# Robots	**Tasks Assigned**	**Tasks Handled**	**Team Cost**
Experiment 1	4	30	30	154.4
Experiment 2	4	30	30	150.3
Experiment 3	4	30	30	190.0
Experiment 4	4	30-4	27	140.9
Experiment 5	3	30	30	232.1
Experiment 6	3	30	30	162.0
Experiment 7	3	30	30	139.0
Experiment 8	3+1	30	30	139.1
Experiment 9	4	22	22	154.8

Table 13: Experimental Results

Table 13 reports the results of experiments 1-9. Experiment 1 shows a team of robots able to accomplish the 30 assigned tasks with a cumulative cost of ~150 robot-seconds nominally. Experiment 2 shows that an induced malfunction in one of the robots is handled gracefully without loss to solution efficiency. Experiment 3 shows that a 10% loss in communication raises the solution cost to ~190 robot-seconds, mainly due to repeated tasks because of lost acknowledgements, but does not prevent the handling of any of the tasks. Experiment 4 results in a drop in the solution cost due to the cancellation of 4 tasks during the experiments – note that on average only 3 of the tasks were cancelled before execution since tasks were chosen at random to be cancelled and a task chosen for cancellation was on average completed prior to the time of cancellation.

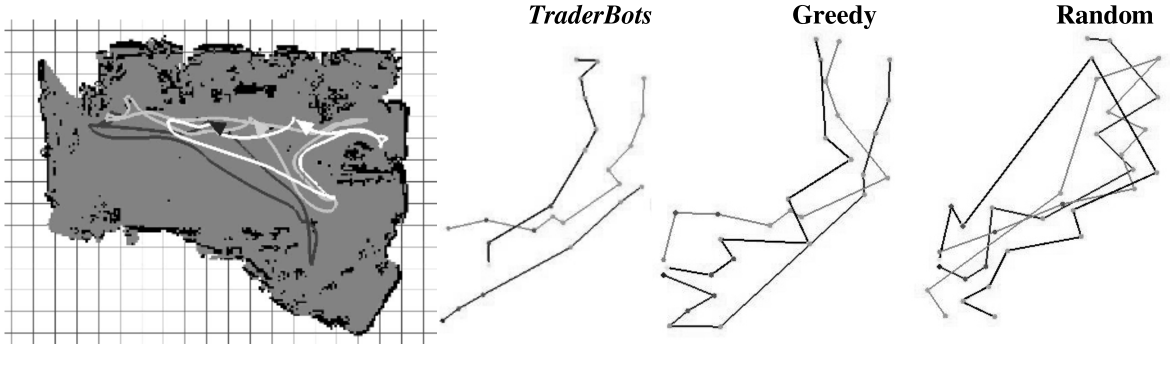


Figure 36: Generated Map From Exploration Task and Comparison of Allocations

Experiment 8 shows that the *TraderBots* approach can accommodate the addition of a robot during operations. However, the new robot was started in a different start position from the previous experiments and hence the resulting costs cannot be compared. Experiment 9 shows the results when the robots are tasked with generating suitable observation points in order to collectively build a map of the environment. Figure 34a shows an example of a map collaboratively generated by the robots. Figures 34b, 34c, and 34d show a graphical comparison of allocations made using random, greedy, and *TraderBots* approaches in experiments 5-7. Note that the robot paths overlap the least in the *TraderBots* allocation and they overlap the most in the random allocation as expected. This result is reflected in the corresponding costs shown in Table 13. In previous publications Dias and Stentz report comparisons, in simulation, of the efficiency of the *TraderBots* allocation to the optimal allocation [37], to a centralized allocation [38], and to a fully distributed allocation [38]. Implementations of the TraderBots approach to date have proved to be highly successful in multirobot coordination, especially in dynamic environments. Movies showing successful mapping of different environments by a team of robots coordinated using the TraderBots approach can be viewed at the CTA web site: https://www.cs.cmu.edu/afs/cs.cmu.edu/project/cta/www/

CHAPTER 8

Conclusion

THIS dissertation describes a market-based architecture for coordinating a group of robots to achieve a given objective. The architecture is inherently distributed, but also opportunistically forms centralized sub-groups to improve efficiency, and thus approach optimality. Robots are self-interested agents, with the primary goal of maximizing individual profits. The revenue/cost models and rules of engagement are designed so that maximizing individual profit has the benevolent effect of moving the team toward the globally optimal solution. This architecture inherits the flexibility of market-based approaches in allowing cooperation and competition to emerge opportunistically. Therefore, this approach is well suited to address the multirobot control problem for autonomous robotic colonies carrying out complex tasks in dynamic environments where it is highly desirable to optimize to whatever extent possible. Future work will develop the core components of a market-based multirobot control-architecture, investigate the use of a negotiation protocol for task distribution, design and implement resource and role management schemes, and apply optimization techniques to improve system performance.

8.1 Summary

This dissertation describes a market-based architecture for coordinating a group of robots to achieve a given objective. The architecture is inherently distributed, but also opportunistically forms centralized sub-groups to improve efficiency, and thus approach optimality. Robots are self-interested agents, with the primary goal of maximizing individual profits. The revenue/cost models and rules of engagement are designed so that maximizing individual profit has the benevolent effect of moving the team toward the globally optimal solution. This architecture inherits the flexibility of market-based approaches in allowing cooperation and competition to emerge opportunistically. Therefore, this approach is well suited to address the multirobot control problem for autonomous robotic colonies carrying out complex tasks in dynamic environments where it is highly desirable to optimize to whatever extent possible. Future work will develop the core components of a market-based multirobot control-architecture, investigate the use of a negotiation protocol for task distribution, design and implement resource and role management schemes, and apply optimization techniques to improve system performance.

8.2 Contributions

The contributions of this dissertation to the robotics literature are as follows:

- **Most comprehensive study to date of the requirements of multirobot coordination in dynamic environments**
  This dissertation reports the most comprehensive study, to date, of the requirements for successful multirobot coordination in dynamic environments. Chapter 1 of this dissertation examines several application domains that benefit from multirobot coordination, and identifies several characteristics required of a successful coordination approach.

- **First extensive investigation of the application of market-based techniques to multirobot coordination**
  This dissertation also accomplishes the first extensive investigation of applying market-based techniques to coordinate multiple robots engaged in cooperative tasks. Results reported in this dissertation clearly show that market techniques, applied in the form of the TraderBots approach, can be highly successful in coordinating multirobot systems in dynamic environments.

- **Most widely applicable coordination-approach for dynamic multirobot application domains that require efficient solutions**
  The TraderBots approach detailed in this dissertation demonstrates the capability to satisfy all of the identified characteristics required for successful multirobot coordination in many application domains. An examination of other multirobot coordination approaches shows that no other approach satisfies all the identified requirements. Thus, the TraderBots approach is the most widely applicable multirobot coordination, currently available.

- **First distributed multirobot coordination-approach that allows opportunistic optimization by "leaders"**
  Opportunistic optimization using leaders, as described in Chapter 5 of this dissertation, is a highly promising strategy for encouraging efficiency that has not been explored in previous work. This strategy will be further investigated in future implementations of the TraderBots approach.

- **Most extensively implemented market-based multirobot coordination approach**
  While other groups have implemented different variations of market-based techniques, this dissertation reports the most extensively implemented market-based multirobot coordination approach, to date. The numerous implementations of the TraderBots approach are detailed in Appendix 1, and the features of all these implementations are highlighted throughout the chapters of this dissertation.

8.3 Impact of Dissertation Work

The methodology introduced by this dissertation has inspired research carried out by many other groups. Highlights of the impact of this dissertation are summarized below.

§ **Task Abstraction in the TraderBots framework by Rob Zlot and Anthony Stentz**
Zlot and Stentz ([128], [129], [130]) investigate task abstraction using tree structures within the TraderBots framework. The participants in the market are permitted to bid on nodes representing varying levels of task abstraction, thereby enabling distributed planning, task location, and optimization among the robot team members. Results in simulation and on a robot team demonstrate that this approach can introduce a significant improvement on the total solution cost for the team.

§ **Tighlty Coordinated Tasks within the TraderBots framework by Nidhi Kalra and Anthony Stentz**
Klara and Stentz [67] investigate methods of applying market-based techniques to domains that require tightly coupled, distributed coordination of multiple robots. The robots achieve tight coordination by repeatedly evaluating fine-grained actions, reacting to their team members' actions in the process.

§ **Other research inspired by the TraderBots approach**
Several other groups are also engaged in research inspired by the TraderBots approach presented in this dissertation. Some examples are a comparison of market-based techniques and Markov Decision Process (MDP) techniques for multirobot cooperative repair by Bererton and group [11], the use of market-based techniques in conjunction with Reib graphs to ensure coverage for de-mining purposes by Ioannis Rekleitis and Ai Peng New in Howie Choset's biorobotics lab at Carnegie Mellon University (http://voronoi.sbp.ri.cmu.edu/), and an examination of specialization in a multirobot team coordinated using a market approach carried out by Gabe Reinstein and Austin Wang at the Massachusetts Institute of Technolosgy (presentation: http://www.ai.mit.edu/courses/6.834J/lectures/advanced%20lecture_11.6.ppt, document:http://www.ai.mit.edu/courses/6.834J/final%20projects/Austin_TreasureHunt%20Paper.doc).

§ **References**
Publications of the TraderBots approach serve as reference material for many academic courses, and are favorably referenced in numerous publications on the topic of multirobot coordination.

8.4 Future work

This dissertation describes in detail a successful implementation of market-based techniques, in the form of the TraderBots approach, to coordinating multiple robots engaged in cooperative tasks in dynamic environments. The work presented in this dissertation can be extended in many ways. The goal of the future work continues to be producing a fully functional market-based coordination approach capable of efficient and robust multirobot coordination in dynamic environments. Important extensions to be considered in future extensions of this dissertation are:

§ Reasoning more completely about resources

§ Implementation of roles

§ Investigation of more sophisticated techniques for optimizing with leaders

§ Extension of the TraderBots approach to accommodate tasks with deadlines and domains that require non-linear cost functions

§ Testing the TraderBots approach on tightly-coordinated tasks, more heterogeneous teams, and more complex tasks

§ Testing the performance of the TraderBots approach over longer-duration, and more complex missions

APPENDIX 1

Implementing TraderBots

APPENDIX 1 details the different considerations that should be taken into account when implementing the TraderBots approach, and also reports details of the evolution to date of implementation of the TraderBots approach in simulation and on robots.

Architectural Framework

The architectural structure for the TraderBots approach can be viewed as shown below in Figure 38. The illustration is tailored to the distributed mapping application and shows the architectural form for each robot in the team. It is organized in layers. In the bottom layer are the resources under the robot's control, such as sensors, computers, and communication devices. These resources are available to the robot to perform its tasks—some unused resources can be leased to other robots in the team if there is a demand for them (although this feature has not yet been implemented). For example, if a robot is not using its entire computing capacity, it can do another robot's data processing for a fee. The next layer consists of the agent's roles for accomplishing tasks. This layer has not been fully implemented to date. Roles are application-specific software modules that implement particular robot capabilities or skills required for members in the team, such as acting as a communication router or generating optimal plans for the team as a leader. The roles utilize resources in the layer below to execute tasks that require a robot to adopt those roles. Roles execute tasks that match their specific capabilities. They receive assignments from the trader and can be monitored by an executive. As they execute their tasks they may generate other tasks or subtasks to be bid out to the other robots. These new tasks are communicated to the trader.

At the top layer in the architecture, the RoboTrader coordinates the activities of the agent and its interactions with other agents. All of the planning is carried out at this top layer. The trader bids on tasks for the robot to perform and offers tasks for sale. It passes on tasks it wins to an executive who matches tasks to roles, schedules the roles to run, and resolves any contention for resources. The trader could be equipped with an on-line learning module (not implemented to date) that enables it to perform better over time by adapting to the specific application and environment. But the architecture for a single robot does not complete the picture.

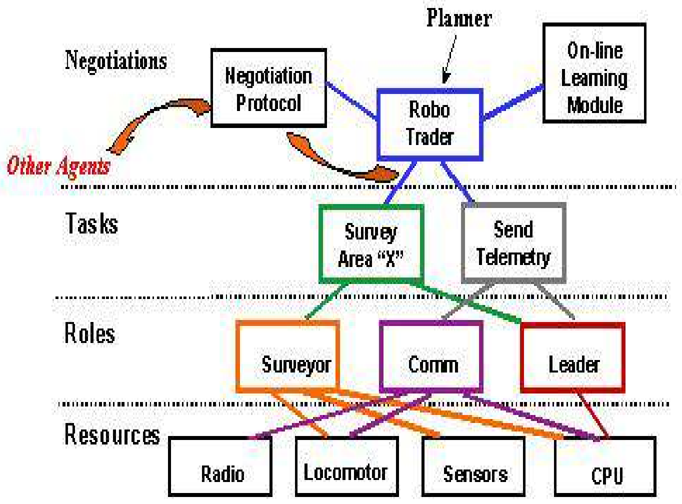


Figure 38: Architectural management of resources, tasks, and roles on a single robot

Figure 39 below illustrates potential high-level interaction between a group of robots and two users:

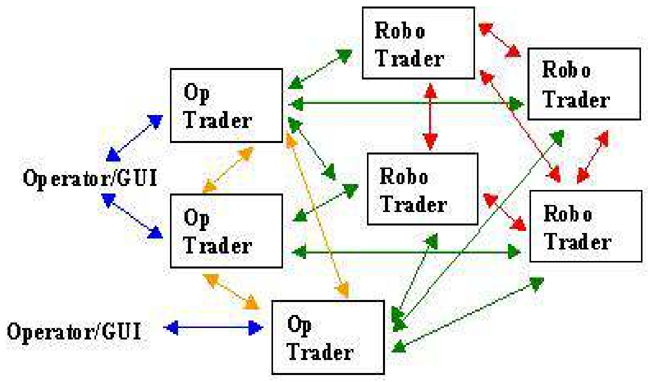


Figure 39: Interaction between robots and operator

As shown above, the operators can communicate high-level tasks to the interface agents known as the "Operator Traders" (or the OpTraders). The OpTraders then interpret the operator's commands and translate them into tasks that the robots can recognize. Next these tasks are bid out to the RoboTraders on the robots within communication range. The OpTraders could also negotiate amongst themselves.

Architectural Components

The architectural components interact with each other as illustrated in Figure 38. A crucial component of implementing the approach will be investigating different designs for cost and revenue models that reflect the requirements of the chosen application domain. They could potentially be complex functions involving statistical distributions and vectors of different components (for example, the cost function could be a combination of time spent, fuel expended, and CPU cycles utilized). These functions will need to reflect aspects such as priorities for task completion, hard deadlines for relevant tasks, and acceptable margins of error for different tasks. Other factors to be considered here are rewards for partially completed tasks, compensation for imperfect or incomplete state information, and errors in task execution. The negotiation protocol will require bids, calls for bids, and contracts to allow negotiation. Also, bidding mechanisms are necessary to compute bids and determine when to place a bid, and when to wait. Finally, penalty mechanisms could become necessary to allow agents to break deals when profitable.

The OpTrader will need to communicate with the operator(s), the robots, and any other OpTrader(s). Through these communications, the OpTrader will receive assignments that will be decomposed into tasks that will be bid out to the robots via the negotiation protocol. An OpTrader may also subcontract assignments it receives to other OpTraders. Thus, each OpTrader will maintain a portfolio of tasks to be assigned to the robots and subcontracts. An auction mechanism and perhaps some form of scheduler will be necessary to determine how best to match robots to assignments. Finally, each OpTrader will keep track of its wealth by updating its personal account after each relevant transaction. The core components of the trader are illustrated below in Figure 40:

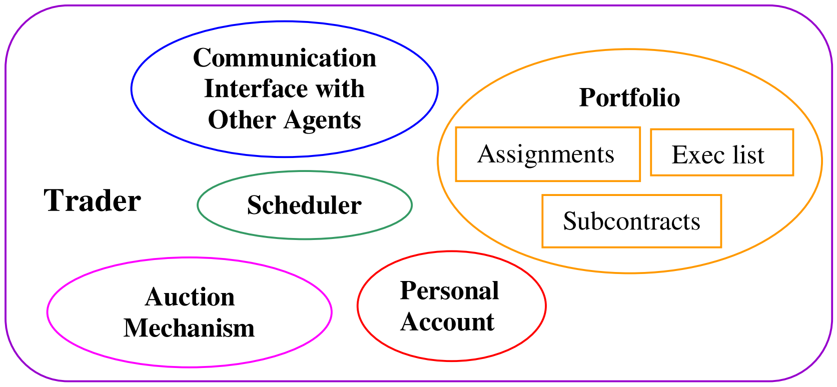


Figure 40: Core components of a trader

The robot traders will need to communicate with the other robots and the OpTrader(s). Through these communications, the robots will receive assignments (that may be further decomposed into sub-tasks) that will either be subcontracted out to the other robots via the negotiation protocol, or executed. Thus, each robot trader will maintain a portfolio of assignments, tasks sent to the executive to be executed, and subcontracts. An auction mechanism and some form of scheduler will be necessary to determine which tasks to send to the executive and which tasks to re-assign to other robots. Finally, each robot trader will keep track of its wealth by updating its personal account after each relevant transaction. Thus, the operator-traders and the robot-traders will be identical except for the operator-traders' ability to translate commands from the operator into tasks that are recognized by the robots, and the robot-traders' ability to schedule tasks for execution. Note that robots or operator-traders could then break down these tasks into sub-tasks.

Implementation 1: Cognitive Colonies Simulation

An initial version of the TraderBots approach was developed by the Cognitive Colonies group (http://www.frc.ri.cmu.edu/projects/colony/), including Tony Stentz, Scott Thayer, and Bruce Digney, and tested on a distributed sensing problem in several simulated interior environments built by Vanessa De Gennaro and Brian Fredrick. A group of robots, located at different starting positions in known simulated worlds, were assigned the task of visiting a set of pre-selected observation points as shown below in Figure 41. This problem is similar to a version of the distributed traveling salesman problem, where the observation points are the cities to visit. Each robot was equipped with a map of the world, which enabled it to calculate the cost associated with visiting each of these cities. The costs were the lengths of the shortest paths between cities in an eight-connected grid, interpreted as money. Thus, the robots bid for each city based on their estimated costs to visit that city.

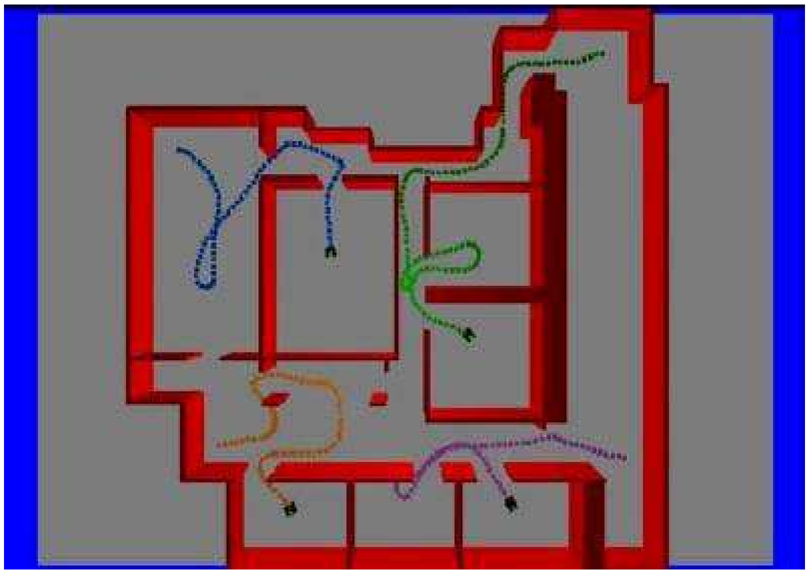


Figure 41: Screen shot from simulator used in Cognitive Colonies project

The interface between the human operator and the team of robots was a software agent, the operator executive (exec). The exec conveyed the operator's commands to the members of the team, managed the team revenue, monitored the team cost, and carried out the initial city assignments. Being a self-interested agent, the exec aimed to assign cities quickly while minimizing revenue flow to the team. In our initial implementation, the exec adopted a greedy algorithm for assigning tasks.

Once the exec had completed the initial city assignments, the robots negotiated amongst themselves to subcontract city assignments. Each of the robots, in turn (the initial implementation was fully synchronous), offered all the cities on its tour (individually) to all the other robots for a maximum price equal to the auctioneer's cost reduction by removing that city from its tour. Each bidder then submitted a bid for that city greater than the cost for adding the city to its tour. In this initial implementation, only single-city deals were considered, and the robots continued to negotiate amongst themselves until no new, mutually profitable deals were possible. Thus, negotiations ceased once the system settled into a local minimum of the global cost.

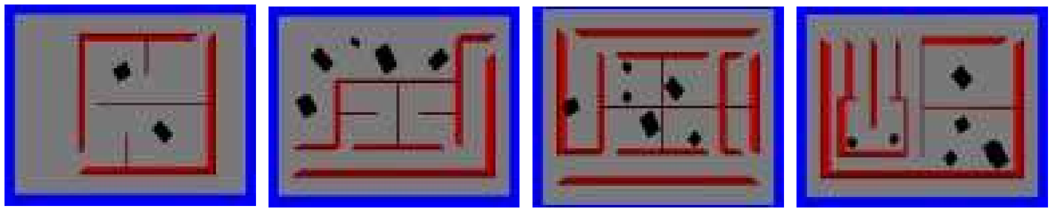


Figure 42: Sampling of simulated worlds used in Cognitive Colonies project

The worlds shown above are a sample of the worlds we used to run test simulations. If you click on the floorplan of Worlds 1 through 4 on the project web page, http://www.frc.ri.cmu.edu/projects/colony/sim_gal.shtml, you can view a simulator movie run in that world and read a description explaining the movie. If you click on the Field Robotic Center (FRC) World floor plan, shown on the same web page, you will be taken to a listing of movies, each illustrating different scenarios. The FRC World simulations are important because they indicated the performance we could expect for our implementation on the robotic platform described next.

Implementation 2.1: Cognitive Colonies Robots

An initial version of the approach was next implemented on the Cognitive Colonies robotic system. The aim of the Colonies project was to build a group of robotic aids for urban reconnaissance. Ten PioneerII-DX robots, built by Activmedia Incorporated, (shown in Figure 43) were used in the project.

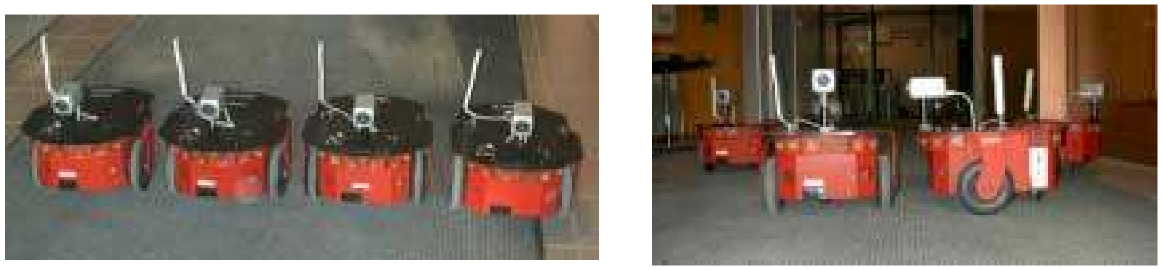


Figure 43: Robot team used in Cognitive Colonies project

The robots were each equipped with onboard computing as well as 16 sonar sensors for obstacle detection and a forward pointing camera for map construction. The distributed TSP scenario was initially tested using 4 of these robots in a cluttered environment. The robots only negotiated with the *OpTrader* in this implementation. Inter-robot negotiation was implemented later. The operator was able to specify goals to be visited via a GUI.

Implementation 2.2: Cognitive Colonies Robots II

This initial implementation of TraderBots on the Pioneer robots was further enhanced by Zlot et al. [131] and tested on a distributed exploration and mapping task.

Figure 44: Team of Pioneer II-DX robots

The experiments were run on the same team of ActivMedia PioneerII-DX robots. Each robot is equipped with a ring of 16 ultrasonic sensors, which are used to construct occupancy grids of the environment as the robot navigates. Each robot is also equipped with a KVH ECORE TM 1000 fiber optic gyroscope used to track heading information. Due to the high accuracy of the gyroscopes (2-4 degrees drift/hr), we use the gyro-corrected odometry at all times rather than employing a localization scheme. Using purely encoder-based dead reckoning the positional error can be as high as 10% to 25% of the distance traveled for path lengths on the order of 50 – 100m, while using gyro-corrected odometry reduces the error to the order of 1% of the distance traveled.

Implementation 3: FIRE simulation

A third implementation focuses on the space application domain, and more specifically, presents simulation results for market-based coordination of a group of heterogeneous robots engaged in information gathering on a Martian outpost.
In this implementation the market-based, multi-robot planning capability, is designed as part of a distributed, layered architecture for multi-robot control and coordination[6]. More specifically, this architecture is an extension to the traditional three-layered robot architecture (illustrated in Figure 45) that enables robots to interact directly at each layer – at the behavioral level, the robots create distributed control loops; at the executive level, they synchronize task execution; at the planning level, they use the TraderBots approach to assign tasks, form teams, and allocate resources.

[6] See Goldberg et al.[55] and Simmons et al. [106] for more details about this layered architecture.

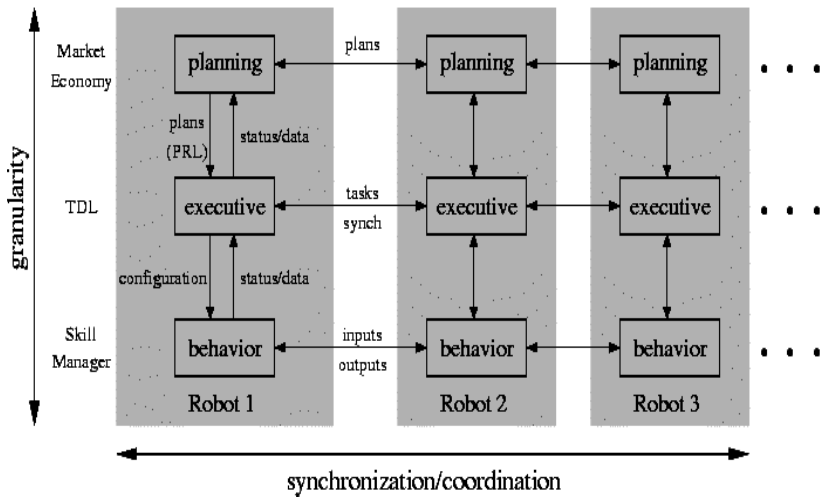


Figure 45: Extended three-layer architecture

This implementation is tested using a 3D graphical simulator developed for the project (see Figure 46).

Figure 46: Screen shot from 3D graphical simulator

The market-based planning layer of each robot has two main components: a "trader" that participates in the market, auctioning and bidding on tasks, and a “scheduler” that determines task feasibility and cost for the trader, and interacts with the executive layer for task execution. The focus of the development and testing of the current system has been on a characterize task that will fit within the broader scenario of the Martian outpost. In this task, a user/scientist specifies a region on the Mars surface, indicating that rocks within that region are to be characterized with an appropriate sensing instrument. The scientist may also specify the locations of rocks, if known.

Implementation 4: CTA Robots

An implementation of the *TraderBots* approach on a team of Pioneer robots enables the reported results. The details of the robotic system used in this implementation are presented next.

Robotic Platform

The robot team (shown in Figure 1) consists of a homogenous set of off-the-shelf mobile robot platforms outfitted with additional sensing and computing. Serving as the mobility platform is an ActivMedia Pioneer II DX indoor robot. A Mobile Pentium 266 with MMX is the main processor. Attached is a 1-gigabyte hard drive for program and data storage and 802.11b wireless card for ad-hoc communication between robots. Encoder data from the drive wheels is collected onboard from which dead reckoning position (x, y, θ) is calculated. Encoders provide a relatively accurate measure of linear travel, but relatively inaccurate angle measurement, such that small errors in angle compound over time resulting in large displacement errors. A solution to these pose errors is the addition of alternate angle measurements using a fiber optic rate gyroscope (KVH E-Core 1000). The gyroscope provides highly stable and accurate angle measurement (four degrees drift per hour). Robots sense their environment using an 180 scanning laser range finder (SICK LMS 200). Horizontal scan-range-data is incorporated with position data to create a 2D map. In addition to providing information to the operator, the map is used for local navigation and cost estimation during trading.

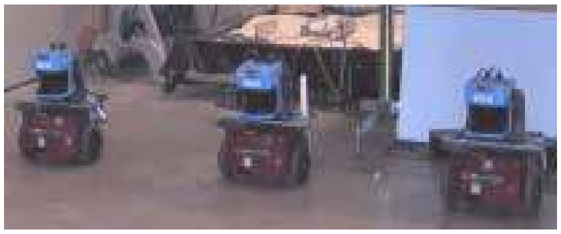


Figure 47: Robot Team

Architecture

Design and implementation of the system supporting the *TraderBots* architecture was focused on extensibility and scalability. The system can be conceptualized as a 4-tier structure (as illustrated in Figure 2): hardware, hardware abstraction, autonomous navigation, and multi-robot (inter-robot) communication. The hardware layer consists of the motors, encoders, laser sensor and gyroscope. A process called *RobotCntl*, which serves as a hardware abstraction to higher-level processes, controls all components of the hardware layer. *RobotCntl* manages the state of the hardware, collects, timestamps, and provides access to data, and interprets and executes hardware control commands from higher-level processes.

Two separate processes, *TaskExec* and *DataServer*, in conjunction with the hardware abstraction layer accomplish autonomous navigation. *TaskExec* executes local navigation with a map provided from *DataServer*. *DataServer* aggregates position and laser range data from the hardware abstraction level and provides maps to other processes that require map information. In addition to receiving map data from *DataServer*, *TaskExec* broadcasts its position to other robots and receives the position of other robots through the *CommRelay*. These positions are placed in *TaskExec*'s navigation map as obstacles to implement a collision avoidance mechanism between robots. At the highest level of control is the *Trader* process. The *Trader* is responsible for coordinating with other robots through *CommRelay* and determining task allocations. Once tasks are allocated, the *Trader* maintains a schedule for its commitments and periodically sends tasks to be executed to the *TaskExec*. The *Trader* also keeps a local map for cost estimation during trading. Note that different versions of D* ([108], [109], [110]) maps are used for cost estimation during trading and for navigation during task execution.

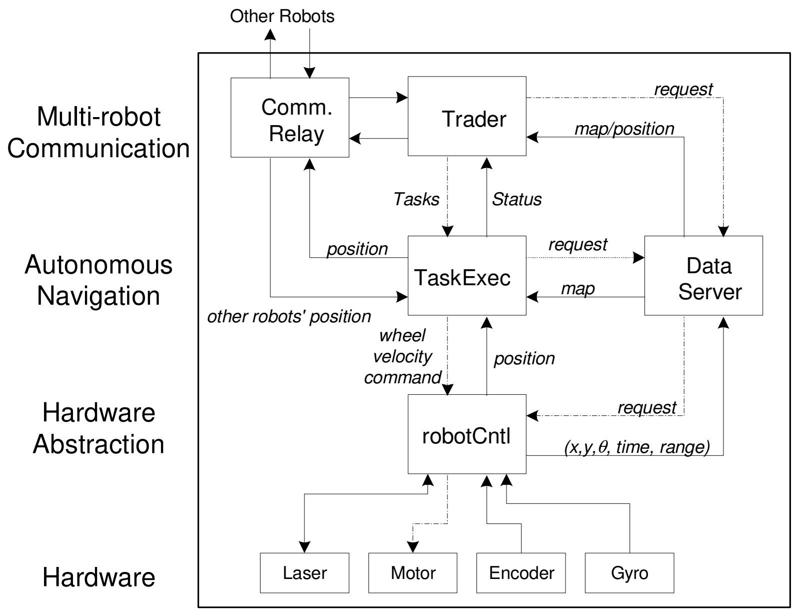


Figure 48: Architectural Layout

Communication

Communication between modules occurs in two ways: intra-robot (between modules on a single robot) and inter-robot (between modules on different robots). These two instances use different techniques reflecting their unique situations. Intra- robot communication happens between processes on one robot such as *TaskExec* and *RobotCntl* or *Trader* and *DataServer*. These links are assumed to be high-speed and reliable since the processes run on the same robot. The basic assumption is that this channel is high bandwidth, low latency and reliable. In this implementation we use a communication package called RTC (Real Time Communication) [86], which provides inter-process-communication between processes on the same machine or machines with reliable links. Inter-robot communication differs from intra-robot communication with respect to bandwidth and reliability. Inter-robot communications use wireless Ethernet that is orders of magnitude less capable in terms of bandwidth in comparison to intra-robot communication, and suffers from reliability problems due to radio interference. In order to avoid re-transmission problems in an unreliable wireless environment, we use UDP (User Datagram Protocol), a connectionless datagram protocol built on IP (Internet Protocol), for transmitting data between robots. All RTC messages destined for another robot are sent to the *CommRelay* and packaged as UDP messages. The UDP messages are then sent via UDP to the destination robot and received by that robot's *CommRelay*. They are then converted back into RTC messages and sent to the appropriate modules using the intra-robot communication protocol.

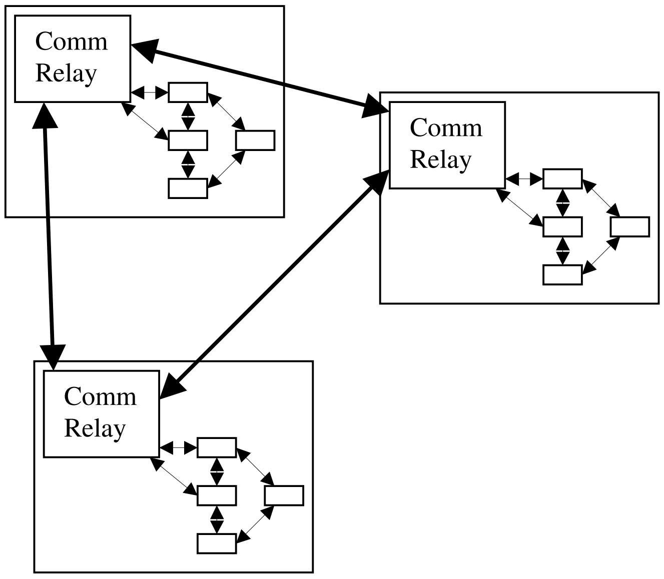


Figure 49: Inter-robot and Intra-robot Communication

As described above, communication between processes on different robots is realized through a point-to-point UDP-based message-passing scheme. Thus, each *RoboTrader* is not instantly able to determine which other *RoboTrader*s it is connected to at any given time. In order to keep track of which other traders are reachable, each *RoboTrader* sends out a periodic hunt signal to all existing robots whether they are known to be alive or not. All traders that receive the hunt signal record the sender as connected, and send an acknowledgement (ACK). The original sender waits a predetermined amount of time (10 seconds) for ACKs. The senders of any ACKs that arrive within the time interval are recorded as being connected, and at the end of the time interval all other traders are marked as disconnected. Additionally, the senders of any other signals (e.g. auction calls, bids) can opportunistically be marked as connected by the recipients of these messages.

It is possible for a connected *RoboTrader F* to become perceived as disconnected at a later time if a trader *T* who had detected that robot previously ceases to communicate with it. This can happen both in the case of a communication problem (out of range or a malfunction) or a robot death. When the disconnection occurs, *T* waits for a specified interval (1 minute) to attempt to reconnect to *F* either through the hunt-acknowledge protocol, or by receiving any other message from *F*. If no such message arrives, then T assumes that there is a problem with *F*. To handle the possible fault, *T* first asks the other connected traders if they can connect to *F*. This may be possible if *F* is out of communications range of *T*, but is within range of some other robot *R* that is also reachable by *T*. If any other traders are connected to *F*, then *T* reverts to believing that *F* is alive and begins the 1-minute disconnection timer once again (in case *F* suffers a fault before reconnecting to *T*). Otherwise, *T* assumes *F* has suffered a robot death and thus is out of commission until it receives a message from *F* again. The handling of robot death and other robustness issues in the *TraderBots* approach is reported in detail in a recent publication submission to the International Conference on Robotics and Automation [42].

Execution

The *TaskExec* module performs the execution level of the architecture. This module is in charge of monitoring and arbitration of tasks, allowing for sequential and/or parallel execution. The *TaskExec* module combines the virtues of SAUSAGES [58] and DAMN [87] to create a task network in which simultaneous tasks can have their outputs combined through an arbiter. The basic building blocks for the task network are tasks. Tasks share a common structure that allows them to be transparently called by the *TaskExec* independent of the specific function that the task performs. Thanks to this common structure, tasks can be dynamically added and removed from the task network.

The most important member functions of a task are:

§ *startTask()*: this function is called once by the *TaskExec* before the task is executed for the first time.
§ *runOnce()*: this function is called once each execution cycle, for as long as the task is active
§ *endTask()*: this function should be called by the task itself, when its termination criterion has been met. The TaskExec will also call it if the task executes beyond its assigned termination time.

The *TaskExec* is the executor of the task network. It starts, executes, monitors and terminates tasks as required. It also allows for dynamic changes in the task network and operates as follows:

§ Process inputs from sensors, and put them in maps and data structures that are accessible to all tasks
§ Check start-conditions of all tasks. If the start conditions of one or more tasks are satisfied, start the tasks by calling the *startTask()* member of the tasks. Tasks

will be considered active from this moment until their termination. There are two kinds of start conditions for a task: (1) at a specified time, and (2) after completion of its predecessor: It is also possible to condition the start on the successful termination of the task, and specify a different task to be executed if the predecessor fails.

§ Call *runOnce()* for all the tasks that are active. Each task processes the changes in the world and generates an output, or a vote. If a task has complete control of a resource, the output can be a direct command to the resource. If the task has shared control of a resource, the output of the task will be a vote on the desired behavior of the controlled resource. If a task has finished, it calls *endTask()*, to indicate its termination.

§ Check termination conditions for all active tasks. If a task remains active beyond its scheduled execution time, the *TaskExec* will terminate the task.

Tasks can have control of two types of resources: exclusive and shared. Exclusive resources (for example science instruments on a space exploration robot) are unique to a task, and can be controlled directly from the task. Shared resources (for example motors and multi-purpose sensors) are common to several tasks, and need to be arbitrated to perform an action. The most common kind of arbitrated resources are steering angle and speed. In the current implementation, all the tasks that participate in the selection of a steering angle and speed share a set of arcs with a different curvature and speed associated to each one of them. When the *runOnce()* function is called, the tasks issue votes on each one of the arcs. After all the tasks have been called for the current execution cycle, the *TaskExec* combines the votes from all the active tasks and executes the arc corresponding to the winning vote. The *RoboTrader* sends sequences of tasks to the *TaskExec* to be executed. In the current implementation the *TaskExec* maintains a single execution queue, and new tasks are added to the end to the current execution queue. If the queue was empty before the arrival of new tasks, the new tasks are executed immediately. The *TaskExec* reports success or failure of an executed task to the *RoboTrader* when a task terminates.

Trading

Trading is a key component of the *TraderBots* approach. A *RoboTrader* assigned to each robot is responsible for opportunistically optimizing the tasks the robot commits to executing. An *OpTrader* serves as an interface agent between the operator and the robot team. Each trader maintains a portfolio in which it keeps track of its commitments, schedule, currently executing tasks, and tasks it trades to others. Two forms of contract types are allowed during trading: subcontracts and transfers. If the contract type is a subcontract, it implies the auctioneer is interested in monitoring the progress of the task and will hence expect a report when the task is completed; payment is made only after the subcontracted task is completed. Note that a subcontracted task can be traded in turn to another robot, but only as another subcontract. Each robot only needs to keep track of the robot it won the subcontract from and the robot it subcontracted the task to. Once the task is executed, the completion of the task is reported along the chain of robots linked by the subcontracts until the initial auctioneer is notified. If on the other hand, the contract type is a transfer, payment is made as soon as the task is traded, and no further communication concerning that task is necessary between the auctioneer and bidder. Each trader has an internal alarm that prompts it to auction all tasks in its schedule periodically. Note that tasks being executed are removed from the schedule and hence cannot be traded. This implementation decision was based on the assumption that a task cannot be transferred once it is started. For application domains where this assumption is not true, this restriction can be removed. In contrast, in application domains where idle time for robots is highly costly, introducing a larger execution window by sending a higher number of tasks to the TaskExec and removing them from future auctions will be more suitable. A trader initiates an auction by sending out a call for bids. Traders within communication range compute and submit bids to this auction. Once the specified deadline expires, the auctioneer resolves the call by making a profit-maximizing allocation based on the bids it received. If a trader receives an award for a bid it submitted, it accepts or rejects that award based on its current state. Note that an award is binding after it has been accepted. Two methods of call resolution are used in the current implementation of *TraderBots*. The *RoboTrader*s assign at most the single most profitable bid submitted to the auction. The *OpTrader,* and *RoboTrader*s who discover they are in a fault state due to a malfunction, use a greedy algorithm for resolving calls so that tasks are allocated more rapidly; this greedy allocation is done because they cannot execute the tasks themselves and in the case of a malfunction, because the robot can expect a robot death with higher probability and hence aims to reassign tasks quickly. The greedy algorithm assigns the most profitable bid submitted by each trader that participates in the auction while assuring that no task gets assigned more than once and no bidder gets assigned more than one task during each auction. All bids are limited to single-task bids in this implementation. Dias and Stentz explore the comparative advantages of different negotiation strategies in a previous publication [37].

In order to participate in an auction, robots need to calculate the costs of tasks. A robot announcing an auction must determine its reservation price, i.e. the highest price it is willing to pay to subcontract or purchase a task. A robot bidding in an auction must calculate the expected cost of the tasks being offered. These valuations are based on marginal costs – the difference in between the cost of the current schedule with those tasks and the cost of the schedule without those tasks. For a single task, an auctioneer's valuation is the savings resulting from removing that task from its schedule and reordering the remaining tasks in the schedule in an efficient manner. A bidder's marginal cost for a single task is the estimated cost of efficiently inserting the task into its schedule. When a trader initiates an auction, the call is sent to all robots marked as connected by that trader. This allows the trader to clear the auction as soon as it receives bids from all connected robots, rather than waiting for the auction deadline.

References

[1] Alur, R., Das, A., Esposito, J., Fierro, R., Grudic, G., Hur, Y., Kumar, V., Lee, I., Ostrowski, J., Pappas, G., Southall, B., Spletzer, J., and Taylor, C. J., "A Framework and Architecture for Multirobot Coordination", Seventh *International Symposium on Experimental Robotics*, 2000.

[2] Andersson, M. R., and Sandholm, T. W., "Sequencing of Contract Types for Anytime Task Reallocation", *Agent-Mediated Electronic Trading, Lecture Notes in Artificial Intelligence*, Volume 1571, pp.54-69, 1999.

[3] Arkin, R. C., "Behavior-Based Robotics", *MIT Press*, Cambridge, Massachusetts, 1998.

[4] Arkin, R. C., "Cooperation without Communication: Multiagent Schema-Based Robot Navigation", *Journal of Robotic Systems*, Vol. 9, No.3, pp. 351-364, 1992.

[5] Arkin, R. C., Balch, T., "AuRA: Principles and Practice in Review", *Journal of Experimental & Theoretical Artificial Intelligence*, Vol. 9, No. 2/3, pp.175-188, 1997.

[6] Bafna, V., Kulyanasundaram, B., and Pruhs, K., "Not All Insertion Methods Yield Constant Approximate Tours in the Euclidean Plane", *Theoretical Computer Science*, 125(2), pp.345-353, 1994.

[7] Balch, T. and Arkin, R.C., "Communication in reactive multiagent robotic systems", *Autonomous Robots*, 1(1): pp.27-52, 1995.

[8] Bennewitz, M., Burgard, W., and Thrun, S., "Optimizing schedules for prioritized path planning of multi-robot systems", *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2001.

[9] Bererton, C., and Khosla, P., "An Analysis of Cooperative Repair Capabilities in a Team of Robots", *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp.476-482, 2002.

[10] Bererton, C., Gordon, G., and Thrun, S., "Auction Mechanism Design for Multi-Robot Coordination", *submitted to NIPS*, 2003.

[11] Bereton, C., "Repairable Robot Teams: Modeling, Planning, and Construction", *Thesis Proposal, Carnegie Mellon University*, http://www-2.cs.cmu.edu/~curt/Thesis_Proposal/Thesis_Proposal.pdf, 2003.

[12] Böhringer, K., Brown, R., Donald, B., Jennings, J., and Rus, D., "Distributed Robotic Manipulation: Experiments in Minimalism*", Proceedings of the International Symposium on Experimental Robotics (ISER)*, 1995.

[13] Bonasso, R. P., Firby, R. J., Gat, E., Kortenkamp, D., Miller, D. P., and "Slack, M. G., "Experiences with an Architecture for Intelligent, Reactive Agents*", Journal of Experimental & Theoretical Artificial Intelligence*, Volume 9(2/3), pp.237-256, 1997.

[14] Botelho, S. C., and Alami, R., "A multi-robot cooperative task achievement system", *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp.2716-2721, 2000.

[15] Botelho, S. C., and Alami, R., "M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement", *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp.1234-1239, 1999.

[16] Botelho, S. C., and Alami, R., "Multi-robot cooperation through the common use of 'mechanisms'", *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp.375-380, 2001.

[17] Brainov, S. and Sandholm, T., "Auctions with Untrustworthy Bidders", *Proceedings of the IEEE Conference on Electronic Commerce*, 2003.

[18] Brainov, S. and Sandholm, T., "Contracting with Uncertain Level of Trust", *Computational Intelligence 18(4):501-514, Special issue on Agent Technology for Electronic Commerce*, 2002.

[19] Brandt, F., Brauer, W., and Weiß, G., "Task Assignment in Multiagent Systems based on Vickrey-type Auctioning and Leveled Commitment Contracting", *Cooperative Information Agents IV*, *Lecture Notes in Artificial Intelligence*, Volume 1860, pp.95-106, 2000.

[20] Brooks, R. A., "Elephants Don't Play Chess", *Robotics and Autonomous Systems*, Vol. 6, pp.3-15, 1990.

[21] Brumitt, B. L., Stentz, A., "Dynamic Mission Planning for Multiple Mobile Robots", *Proceedings of the IEEE International Conference on Robotics and Automation*, No. 3, pp. 2396-2401, 1996.

[22] Burgard, W., Moors, M., Fox, D., Simmons, R., and Thrun, S., "Collaborative Multi-Robot Exploration", *Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco CA, April 2000.

[23] Caloud, P., Choi, W., Latombe, J-C., Le Pape, C., and Yim, M., "Indoor Automation with Many Mobile Robots", *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp.67- 72, 1990.

[24] Cao, Y. U., Fukunga, A. S., and Kahng, A. B., "Cooperative mobile robotics: Antecedents and directions", *Autonomous Robots*, 4:1-23, 1997.

[25] Castelfranchi, C., and Conte, R., "Limits of economic and strategic rationality for agents and MA systems", *Robotics and Autonomous Systems*, Volume 24, pp.127-139, 1998.

[26] Chaimowicz, L., Sugar, T., Kumar, V., and Campos, M. F. M., "An Architecture for Tightly Coupled Multi-Robot Cooperation", *IEEE International Conference on Robotics and Automation*, 2001.

[27] Cheng, Y-Q., Wu, V., Collins, R. T., Hanson, A. R., and Riseman, E. M., “Maximum-weight bipartite matching technique and its applications in image feature matching”, *SPIE Conference on Visual Communication and Image Processing*, 1996.

[28] Chevallier, D., and Payandeh, S., “On Kinematic Geometry of Multi-Agent Manipulating System Based on the Contact Force Information”, *The 6th International Conference on Intelligent Autonomous Systems (IAS-6)*, pp.188-195, 2000.

[29] Cicirello, V., and Smith, S., “Insect Societies and Manufacturing”, *The IJCAI-01 Workshop on Artificial Intelligence and Manufacturing: New AI Paradigms for Manufacturing*, 2001.

[30] Cicirello, V., and Smith, S., “Wasp Nests for Self-Configurable Factories”, *Agents '01, Proceedings of the Fifth International Conference on Autonomous Agents*, 2001.

[31] Collins, J., Jamison, S., Mobasher, B., and Gini, M., “A Market Architecture for Multi-Agent Contracting”, *Technical Report 97-15*, University of Minnesota, 1997.

[32] Cormen, T. H., Leiserson, C. E., and Rivest, R. L. eds. 1990. *Introduction To Algorithms*. Cambridge, MA: The MIT Press/McGraw-Hill Book Company.

[33] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C., “Introduction to Algorithms (Second Edition)”, *MIT-Press*, 2001.

[34] Decker, K. S., and Lesser, V. R., “Designing A Family Of Coordination Algorithms”, *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pp. 73-80, 1995.

[35] Demange, G., Gale, D., and Sotomayor, M., “Multi-Item Auctions”, *The Journal of Political Economy* 94(4), pp.863-872, 1986.

[36] Desai, J. P., Ostrowski, J., and Kumar, V., “Controlling formations of multiple mobile robots”, *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2864-2869, 1998.

[37] Dias, M. B. and Stentz, A., “Opportunistic Optimization for Market-Based Multirobot Control”, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2002.

[38] Dias, M. B., and Stentz, A., “A Comparative Study between Centralized, Market-Based, and Behavioral Multirobot Coordination Approaches”, *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.

[39] Dias, M. B., and Stentz, A., “A Free Market Architecture for Distributed Control of a Multirobot System”, *The 6th International Conference on Intelligent Autonomous Systems (IAS-6)*, pp.115-122, 2000.

[40] Dias, M. B., and Stentz, A., “A Market Approach to Multirobot Coordination”, *Technical Report, CMU-RI -TR-01-26, Robotics Institute, Carnegie Mellon Universit*y, 2001.

[41] Dias, M. B., and Stentz, A., “TraderBots: A Market-Based Approach for Resource, Role, and Task Allocation in Multirobot Coordination”, *Technical Report, CMU-RI -TR-03-19, Robotics Institute, Carnegie Mellon Universit*y, August 2003.

[42] Dias, M. B., Zlot, R., M., Zinck, M., and Stentz, A., “Robust Multirobot Coordination in Dynamic Environments”, *IEEE International Conference on Robotics and Automation (ICRA) 2004*, Submitted 2003.

[43] Dias, M. B., Zlot, R., Zinck, M., Gonzalez, J. P., and Stentz, A., “A Versatile Implementation of the TraderBots Approach for Multirobot Coordination”, *Proceedings of the 8th Conference on Intelligent Autonomous Systems (IAS-8)*, 2004.

[44] Emery, R., Sikorski, K., and Balch, T., “Protocols for Collaboration, Coordination, and Dynamic Role Assignment in a Robot Team”, *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp.3008-3015, 2002.

[45] Excelente-Toledo, C. B., Bourne, R. A., and Jennings, N. R., “Reasoning about Commitments and Penalties for Coordination between Autonomous Agents”*, Proceedings of the 5th International Conference on Autonomous Agents*, 2001.

[46] Faratin, P., Sierra, C., Jennings, N. R., “Negotiation decision functions for autonomous agents”, *International Journal of Robotics and Autonomous Systems*, Volume 24(3-4), pp. 159-182, 1997.

[47] Farinelli, A., Grisetti, G., Iocchi, L., Nardi, D., “Coordination in Dynamic Environments with Constraints on Resources”, *IROS Workshop on Cooperative Robotics*, 2002.

[48] Fierro, R., Das, A., Spletzer, J., Hur, Y., Alur, R., Esposito, J., Grudic, G., Kumar, V., Lee, I., Ostrowski, J. P., Pappas, G., Southall J., and Taylor, C. J., “A framework and architecture for multirobot coordination”*, International Journal of Robotics Research*, 2003, To appear.

[49] Gerkey, B. P. and Matari, , M. J., “Sold! Market methods for multi-robot control”, *IEEE Transactions on Robotics and Automation Special Issue on Multi-Robot Systems*, submitted March 2001.

[50] Gerkey, B. P., “On Multi-Robot Task Allocation”, *Ph. D. Thesis*, *University of Southern California*, 2003.

[51] Gerkey, B. P., and Matari, , M. J., “Multi-Robot Task Allocation: Analyzing the Complexity and Optimality of Key Architectures”, in *Proceedings of 2003 IEEE International Conference on Robotics and Automation*.

[52] Gerkey, B. P., and Mataric´, M. J., "A market-based formulation of sensor-actuator network coordination", *Proceedings of the AAAI Spring Symposium on Intelligent Embedded and Distributed Systems*, pp.21-26, 2002.

[53] Gibney, M. A., Jennings, N. R., Vriend, N. J., and Griffiths, J. M., “Market-Based Call Routing in Telecommunications Networks Using Adaptive Pricing and Real Bidding”, *3rd Inernational Workshop on Multi-Agent Systems and Telecommunications (IATA-99)*, pp.50-65, 1999.

[54] Goldberg, D. and Matari, , M. J., “Robust behavior-based control for distributed multi-robot collection tasks”, *Technical Report IRIS-00-387*, USC Institute for Robotics and Intelligent Systems, 2000.

[55] Goldberg, D., Cicirello, V., Dias, M. B., Simmons, R., Smith, S., Smith, T., and Stentz, A., “A Distributed Layered Architecture for Mobile Robot Coordination: Application to Space Exploration”, Proceedings of the 3rd International NASA Workshop on Planning and Scheduling for Space, 2002.

[56] Goldman, C. V., and Rosenschein, J. S., “Emergent Coordination through the Use of Cooperative State-Changing Rules”, Proceedings of the Twelfth National Conference on Artificial Intelligence, pp. 408-413, 1994.

[57] Golfarelli, M., Maio, D., Rizzi, S., “A Task-Swap Negotiation Protocol Based on the Contract Net Paradigm”, Technical Report CSITE, No. 005-97, 1997.

[58] Gowdy, J., “SAUSAGES: Between Planning and Action”, *Technical Report CMU-RI-TR-94-32, Robotics Institute, Carnegie Mellon University*, 1994.

[59] Grocholsky, B., Makarenko, A., and Durrant-Whyte, H. F., “Information-Theoretic Coordinated Control of Multiple Sensor Platforms”, *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2003.

[60] Guestrin, C., and Gordon, G., “Distributed planning in hierarchical factored MDPs”, *Uncertainty in Artificial Intelligence (UAI)*, Volume 18, 2002.

[61] Huber, M. J., and Durfee, E. H., “Deciding When To Commit To Action During Observation-Based Coordination”, *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pp. 163-170, 1995.

[62] Jennings, B., and Arvidsson, Å., “Co-operating Market/Ant Based Multi-agent Systems for Intelligent Network Load Control”, *Proceedings of the 3rd International Workshop on Intelligent Agents for Telecommunication Applications (IATA)*, *LNAI*, Vol. 1699, pp. 62-75, 1999.

[63] Jennings, J., Kirkwood-Watts, C., “Distributed Mobile Robotics by the Method of Dynamic Teams”, *4th International Symposium on Distributed Autonomous Robotic Systems*, 1998.

[64] Jennings, N., “Controlling cooperative problem solving in industrial multi-agent systems using joint intentions”, *Artificial Intelligence* 75, pp.195-240, 1995.

[65] Jensen, R. M., Veloso, M. M., “OBDD-based Universal Planning: Specifying and Solving Planning Problems for Synchronized Agents in Non-Deterministic Domains”, *Lecture Notes in Computer Science*, No. 1600, pp. 213-248, 1999.

[66] Jung, H., Tambe, M., and Kulkarni, S., “Argumentation as Distributed Constraint Satisfaction: Applications and Results”, *Proceedings of the 5th International Conference on Autonomous Agents*, 2001.

[67] Kalra, N., and Stentz, A., “A Market Approach to Tightly-Coupled Multi-Robot Coordination: First Results,” *Proceedings of the ARL Collaborative Technologies Alliance Symposium*, 2003.

[68] Kaminka, G. A., and Tambe, M., “Robust Agent Teams via Socially-Attentive Monitoring”, *Journal of Artificial Intelligence Research (JAIR)*, Volume 12, pp. 105-147, 2000.

[69] Khuller, S., “On-line algorithms for weighted bipartite matching and stable marriages”, *Theoretical Computer Science*, Volume 127(2), pp.255-267. 1994.

[70] Krovi, R., Graesser, A. C., and Pracht, W. E., “Agent Behaviors in Virtual Negotiation Environments”, *IEEE Transactions on Systems, Man, and Cybernetics*, Volume 29, pp.15-25, 1999.

[71] Laengle, T., Lueth, T. C., Rembold, U., and Woern, H., “A distributed control architecture for autonomous mobile robots – implementation of the Karlsruhe Multi-Agent Robot Architecture (KAMARA)”, *Advanced Robotics*, Volume 12, No. 4, pp. 411-431, 1998.

[72] Lux, T., Marchesi, M., “Scaling and Criticality in a Stochastic Multi-Agent Model of a Financial Market”, *Nature*, Vol. 397, No. 6719, pp. 498-500, 1999.

[73] Maio, D., and Rizzi, S., “Unsupervised Multi-Agent Exploration of Structured Environments”, *Proceedings of First International Conference on Multi-Agent Systems*, pp.269-275, 1995.

[74] Matari, , M. J., “Issues and Approaches in the Design of Collective Autonomous Agents”, *Robotics and Autonomous Systems*, Vol. 16, pp. 321-331, 1995.

[75] Matari, , M., “Coordination and Learning in Multi-Robot Systems”, *IEEE Intelligent Systems*, pp. 6-8, 1998.

[76] Matos, N., and Sierra, C., “Evolutionary Computing and Negotiating Agents”, *Agent-Mediated Electronic Trading, Lecture Notes in Computer Science*, Volume 1571, pp. 126-150, 1999.

[77] Michaud, F., and Robichaud, E., "Sharing charging stations for long-term activity of autonomous robots", *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2002.

[78] Noreils, F. R., “Toward a Robot Architecture Integrating Cooperation between Mobile Robots: Application to Indoor Environment”, *The International Journal of Robotics Research*, Volume 12(1), pp.79-98, 1993.

[79] Osawa, E., “A Metalevel Coordination Strategy For Reactive Cooperative Planning”, *Proceedings of First International Conference on Multi-Agent Systems (ICMAS-95)*, pp. 297-303, 1995.

[80] Pagello, E., D'Angelo, A., Montsello, F., Garelli, F., Ferrari, C., "Cooperative Behaviors in Multi-Robot Systems through Implicit Communication", *Robotics and Autonomous Systems*, Vol. 29, No. 1, pp. 65-77, 1999.

[81] Panzarasa, P., and Jennings, N. R., "Social Influence and The Generation of Joint Mental Attitudes in Multi-Agent Systems", *Proceedings of the Eurosim Workshop on Simulating Organisational Processes*, 2001.

[82] Papadimitriou, C. H., and Steiglitz, K. eds. 1998. Combinatorial Optimization: Algorithms and Complexity. Mineola, NY: Dover Publications, Inc.

[83] Parker, L. E., "ALLIANCE: An Architecture for Fault Tolerant Multi-Robot Cooperation", *IEEE Transactions on Robotics and Automation*, Vol. 14, No.2, pp. 220-240, 1998.

[84] Parker, L. E., "Designing Control Laws for Cooperative Agent Teams*", Proceedings of IEEE International Conference on Robotics and Automation*, pp.582-587, 1994.

[85] Rabideau, G., Estlin, T., Chien, S., and Barrett, A., "A Comparison of Coordinated Planning Methods for Cooperating Rovers", in *Proceedings of AIAA 1999 Space Technology Conference*, Albuquerque, NM.

[86] Real-Time Communications (RTC), http://www.resquared.com/RTC.html.

[87] Rosenblatt, J., "DAMN: A Distributed Architecture for Mobile Navigation", *Doctoral Dissertation, Technical Report CMU-RI-TR-97-01, Robotics Institute, Carnegie Mellon University*, January, 1997.

[88] Rosin, C. D., and Belew, R. K., "Methods for Competitive Co-Evolution: Finding Opponents Worth Beating", *Proceedings of the Sixth International Conference on Genetic Algorithms*, pp. 373-380, 1995.

[89] Roth, M., Vail, D., and Veloso, M., "A world model for multi-robot teams with communication", *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2003.

[90] Rus, D., Donald, B., and Jennings, J., "Moving furniture with teams of autonomous robots", in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 235-242, 1995.

[91] Salido, J., Dolan, J., Hampshire, J., and Khosla, P. K., "A modified reactive control framework for cooperative mobile robots", *Proceedings of the International Conference of Sensor Fusion and Decentralized Control*, *SPIE*, pp 90-100, 1997.

[92] Sandholm, T., "An Implementation of the Contract Net Protocol Based on Marginal Cost Calculations", *Proceedings, Eleventh National Conference on Artificial Intelligence (AAAI-93)*, pp. 256-262, 1993.

[93] Sandholm, T., and Lesser, V., "Advantages Of A Leveled Commitment Contracting Protocol", *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pp. 126-133, 1996.

[94] Sandholm, T., and Lesser, V., "Coalition Formation Among Bounded Rational Agents", *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pp. 662-669, 1995.

[95] Sandholm, T., and Suri, S. 2000. Improved Algorithms for Optimal Winner Determination in Combinatorial Auctions and Generalizations. Proceedings of the National Conference on Artificial Intelligence (AAAI).

[96] Sandholm, T., and Suri, S., "Improved Algorithms for Optimal Winner Determination in Combinatorial Auctions and Generalizations", *National Conference on Artificial Intelligence (AAAI),* pp. 90-97, 2000.

[97] Sandholm, T., Larson, K., Andersson, M., Shehory, O., and Tohmé, F., "Coalition structure generation with worst case guarantees", *Artificial Intelligence*, Volume 111(1-2), pp.209-238, 1999.

[98] Sandholm, T., Lesser, V., "Issues in Automated Negotiation and Electronic Commerce: Extending the Contract Net Framework", *Proceedings, First International Conference on Multiagent Systems (ICMAS-95)*, pp. 328-335, 1995.

[99] Sandholm, T., Sikka, S., and Norden, S., "Algorithms for Optimizing Leveled Commitment Contracts", *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 535-540, 1999.

[100] Scerri, P., Modi, P. J., Shen, W-M., and Tambe, M., "Are Multiagent Algorithms Relevant for Robotics Applications? A Case Study of Distributed Constraint Algorithms", *ACM Symposium on Applied Computing*, 2002.

[101] Schneider-Fontán, M., Matari, , M. J., "A Study of Territoriality: The Role of Critical Mass in Adaptive Task Division", *Proceedings, From Animals to Animats 4, Fourth International Conference on Simulation of Adaptive Behavior (SAB-96)*, MIT Press/Bradford Books, pp. 553-561, 1996.

[102] Schneider-Fontán, M., Matari, , M. J., "Territorial Multi-Robot Task Division", *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 5, 1998.

[103] Shehory, O., and Kraus, S., "Task Allocation Via Coalition Formation Among Autonomous Agents", *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pp. 655-661, 1995.

[104] Simmons, R., Apfelbaum, D., Fox, D., Goldman, R. P., Haigh, K. Z., Musliner, D. J., Pelican, M., and Thrun, S., "Coordinated Deployment of Multiple, Heterogeneous Robots", *Proceedings of the Conference on Intelligent Robots and Systems (IROS)*, Takamatsu Japan, 2000.

[105] Simmons, R., Singh, S., Hershberger, D., Ramos, J., and Smith, T., "First Results in the Coordination of Heterogeneous Robots for Large-Scale Assembly*", Proceedings of the International Symposium on Experimental Robotics (ISER)*, 2000.

[106] Simmons, R., Smith, T., Dias, M. B., Goldberg, D., Hershberger, D., Stentz, A., and Zlot, R. M., "A Layered Architecture for Coordination of Mobile Robots", Multi-Robot Systems: From Swarms to Intelligent Automata, Proceedings from the 2002 NRL Workshop on Multi-Robot Systems, 2002.

[107] Smith, R., "The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver", *IEEE Transactions on Computers*, Vol. C-29, No. 12, 1980.

[108] Stentz, A., "Optimal and Efficient Path Planning for Partially-Known Environments", *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1994.

[109] Stentz, A., "The D* Algorithm for Real-Time Planning of Optimal Traverses", *Technical Report CMU-RI-TR-94-37, Carnegie Mellon University Robotics Institute*, 1994.

[110] Stentz, A., "The Focussed D* Algorithm for Real-Time Replanning", in *Proceedings of the 1995 International Joint Conference on Artificial Intelligence*, pp. 1652-1659, 1995.

[111] Stentz, A., and Dias, M. B., "A Free Market Architecture for Coordinating Multiple Robots", Technical report, CMU-RI-TR-99-42, Robotics Institute, Carnegie Mellon University, 1999.

[112] Stone, P., and Veloso, M., "Communication in Domains with Unreliable, Single-Channel, Low-Bandwidth Communication", *Collective Robotics*, pp. 85–97, Springer Verlag, 1998.

[113] Stone, P., and Veloso, M., "Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork", *Artificial Intelligence*, 110(2), pp.241-273, 1999.

[114] Stroupe, A., "Collaborative Execution of Exploration and Tracking using Move Value Estimation for Robot Teams (MVERT)", *Ph.D. Thesis, Carnegie Mellon University*, 2003.

[115] Švestka, P., Overmars, M. H., "Coordinated Path Planning for Multiple Robots", *Robotics and Autonomous Systems*, Vol. 23, No. 4, pp. 125-133, 1998.

[116] Sycara, K., and Zeng, D., "Coordination of Multiple Intelligent Software Agents*", International journal of Intelligent and Cooperative Information Systems*, Volume 5(2-3), pp.181-211, 1996.

[117] Tambe, M., "Towards Flexible Teamwork", *Journal of Artificial Intelligence Research*, Vol. 7, pp. 83-124, 1997.

[118] Tambe, M., "Tracking Dynamic Team Activity", *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 1996.

[119] Tambe, M., Pynadath, D. V., Chauvat, N., Das, A., and Kaminka, G. A., "Adaptive Agent Integration Architectures for Heterogeneous Team Members", *Proceedings of the International Conference on MultiAgent Systems*, pp.301-308, 2000.

[120] Thayer, S. M., Dias, M. B., Digney, B. L., Stentz, A., Nabbe, B., and Hebert, M., "Distributed robotic mapping of extreme environments" *Proceedings of SPIE* Vol. 4195: Mobile Robots XV and Telemanipulator and Telepresence Technologies VII, 2000.

[121] Tumer, Kagan, Agogino, Adrian K., and Wolpert, David, "Learning sequences of actions in collectives of autonomous agents", First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), pp.378-385, 2002.

[122] Vail, D., and Veloso, M., "Dynamic multi-robot coordination", *Multi-Robot Systems: From Swarms to Intelligent Automata*, Volume II, pp.87-100, 2003.

[123] Veloso, M., Stone, P., and Bowling, M., "Anticipation as a key for collaboration in a team of agents: A case study in robotic soccer", *Proceedings of SPIE Sensor Fusion and Decentralized Control in Robotic Systems II*, volume 3839, 1999.

[124] Wang, ZD., Piñeros, C. V., Takahashi, T., Kimura, Y., and Nakano, E., "Arrangement and Control of a Cooperative Multi-Robot System for Object Transportation", *The 6th International Conference on Intelligent Autonomous Systems (IAS-6)*, pp.196-203, 2000.

[125] Weiß, G., "Achieving Coordination through Combining Joint Planning and Joint Learning", *Technical Report FKI-232-99*, Institut für Informatik, TU München, 1999.

[126] Wellman, M. P., and Wurman, P. R., "Market-Aware Agents for a Multiagent World", *Robotics and Autonomous Systems*, Volume 24, pp.115-125, 1998.

[127] Zhang, J., Ferch, M., and Knoll, A., "Carrying Heavy Objects by Multiple Manipulators with Self-Adapting Force Control", *The 6th International Conference on Intelligent Autonomous Systems (IAS-6)*, pp.204-211, 2000.

[128] Zlot, R., and Stentz, A., "Efficient Market-based Multirobot Coordination for Complex Tasks", *The International Journal or Robotics Research (IJRR)*, Submitted 2003.

[129] Zlot, R., and Stentz, A., "Market-based Multirobot Coordination Using Task Abstraction", *International Conference on Field and Service Robotics (FSR)*, 2003.

[130] Zlot, R., and Stentz, A., "Multirobot Control Using Task Abstraction In a Market Framework", *Collaborative Technology Alliances Conference*, 2003.

[131] Zlot, R., Stentz, A., Dias, M. B., and Thayer, S., "Multi-Robot Exploration Controlled By A Market Economy", *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2002.

[132] Zlotkin, G., and Rosenschein, J. S., "Coalition, Cryptography, And Stability: Mechanisms For Coalition Formation In Task Oriented Domains", *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pp. 432-437, 1994.