CRES Technical Report CRES-03-012
Center for Robotics and Embedded Systems, University of Southern California, Los Angeles

ON MULTI-ROBOT TASK ALLOCATION

by

Brian Paul Gerkey

A Dissertation Presented to the
FACULTY OF THE GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(COMPUTER SCIENCE)

August 2003

Contents

List Of Tables

List Of Figures

Abstract

Despite more than a decade of experimental work in multi-robot systems, important theoretical aspects of multi-robot coordination mechanisms have, to date, been largely ignored. To address part of this negligence, this dissertation focuses on the problem of multi-robot task allocation (MRTA). Most work on MRTA has been *ad hoc* and empirical, with many coordination architectures having been proposed and validated in a proof-of-concept fashion, but infrequently analyzed.

With the goal of bringing some objective grounding to this important area of research, this dissertation presents a formal study of MRTA problems. A domain-independent taxonomy of MRTA problems is given, and it is shown how many such problems can be viewed as instances of other, well-studied, optimization problems. By casting MRTA problems in the well-understood framework of optimization, a half-century of work in operations research, game theory, economics, and network flows can be adapted for use in robotic domains. This dissertation demonstrates how such theory can be used for analysis and greater understanding of existing approaches to task allocation, and suggests how the same theory can be used in the synthesis of new approaches.

Although vital for the advancement of the field, analysis is only one component of a comprehensive research agenda. Sensor-actuator systems such as robots present a rich, complex problem domain that can exhibit significant levels of noise and uncertainty. One must implement and empirically validate proposed MRTA algorithms. Thus this dissertation also presents experimental work with an auction-based task allocation system, implemented on physical robots. Optimization theory is used to explain how and why this approach is successful.

This kind of empirical work is a natural complement to formal analysis, and can serve the crucial role of suggesting modifications to the formal model that is analyzed. Such experiments require substantial supporting software infrastructure. This dissertation describes the underlying software facilities that were developed for experimental use in the study of MRTA, as well as for more general use. Specifically discussed is the Player/Stage project, which produces high-quality Open Source software to support robotics research, with the goal of providing a standard platform for mobile robot experimentation and simulation.

Chapter 1

Introduction

Over the past decade, a significant shift of focus has occurred in the field of mobile robotics as researchers have begun to investigate problems involving multiple, rather than single, robots. From early work on simple loosely-coupled tasks such as foraging (e.g., Matarić (1992), Arkin, Balch & Nitz (1993)) to recent work on sophisticated team coordination for robot soccer (e.g., Stone & Veloso (1999), Asada, Kitano, Noda & Veloso (1999)), the complexity of the multi-robot systems being studied has increased. This complexity has two primary sources: larger team sizes and greater heterogeneity of robots and tasks. As significant achievements have been made along these axes, the bar has been raised; it is no longer a sufficient demonstration of multi-robot coordination to show, for example, only two robots observing targets (Parker 1999*b*), or a large group of robots only flocking (Matarić 1995). Rather today we reasonably expect to see larger and larger robot teams engaged in concurrent and diverse tasks over extended periods of time.

1.1 Multi-Robot Task Allocation (MRTA)

As a result of the focus on multi-robot systems, multi-robot coordination has received significant attention. In particular, multi-robot task allocation (MRTA) has recently risen to prominence. Originally a side-show to other problems, MRTA has now become a key research issue in its own right. As researchers

design, build, and use cooperative multi-robot systems, they invariably encounter the question: "which robot should execute which task?"

One model, especially popular in the field of swarm robotics (e.g., Fukuda, Nakagawa, Kawauchi & Buss (1988), Deneubourg, Theraulaz & Beckers (1991)), is *emergent* cooperation: the robots do not explicitly work together, but group-level cooperative behavior emerges from their interactions with each other and the world. This kind of cooperation is observed in nature, and appears to be the evolutionary favorite for cooperation among non-human animals. An emergent system, when constructed skillfully, can be extremely effective and is likely the simplest and most elegant solution to a problem. However, it is a solution to a *specific* problem, and if robots are to be generally useful, they must be capable of solving a variety of problems. Furthermore, emergent cooperation solutions have traditionally been applied to extremely large, homogeneous robot groups, and have relied heavily on redundant skills across the group to achieve good overall performance. In particular, these techniques are a poor fit for small- to medium-scale systems, as well as systems in which the robots have different skills.

This dissertation is instead concerned with methods for *intentional* cooperation (Parker 1998). In this model, robots cooperate explicitly and with purpose, often through task-related communication. As compared with emergent cooperation, intentional cooperation is better suited to the kinds of real-world tasks that humans might want robots to do. If the robots are deliberately cooperating with each other, then, intuitively, humans can deliberately cooperate with them, which is a long-term goal of multi-robot research. Furthermore, intentional cooperation has the potential to better exploit the capabilities of heterogeneous robots. In this work, the use of intentional cooperation is at the level of *task allocation*, and need not propagate to the level of task execution. Importantly, there is no prescription or proscription of any particular method for implementing the details of a task. For example, if a foraging task is assigned to a team of robots because they are best fit for the job, they can execute the task in any way they wish, from probabilistic swarming to classical planning.

1.2 Toward formal analysis

The question of task allocation must be addressed, even for relatively simple multi-robot systems, and the importance of task allocation grows with the complexity, in size and capability, of the system under study. Yet the empirically validated methods remain primarily *ad hoc* in nature, and relatively little has been said regarding the general properties of cooperative multi-robot systems. After a decade of research, while many such architectures have been proposed, the field lacks even a primitive prescription for how to design a MRTA system. Similarly, there has been little attempt to evaluate or compare, either analytically or empirically, the proposed architectures.

This dearth of formal work on multi-robot coordination is, in part, to be expected. A new field, such as robotics, generally begins as an experimental science. It is not until after sufficient collective experience is accumulated that one can begin to analyze the assembled evidence in order to find general trends and make formal statements describing and predicting system behavior. This dissertation suggests that research in multi-robot coordination has reached this point and that there is now the potential to move the field from a primarily experimental science to a more formal, analytical one. The goal is to bring some formalism to the study of task allocation in multi-robot systems by providing tools for analysis of existing methods and synthesis of new ones.

Of course, there has already been one incarnation of robotics as a formal undertaking in the time of so-called Good Old-Fashioned Artificial Intelligence (Russell & Norvig 1995). Proponents of this paradigm suggested that one could program robots by having them build abstract symbolic models of their environments and solve problems by reasoning from first principles about that model. Then system designers could, for example, prove theorems regarding their robots' behavior. Unfortunately this way of thinking, which brought unparalleled success with tasks like chess and checkers, has been a spectacular failure in *embodied* domains, including robotics, and I am not suggesting a return to it. Rather, I propose to follow in the footsteps of the natural sciences: accrue experimental evidence regarding some system or phenomenon until one can propose a plausible descriptive model. Such a model will of course not be

perfect, as it will not capture every salient detail of system. However, because it is constructed *from the experimental evidence*, such an imperfect model will still be a powerful tool for describing multi-robot systems and provide a common framework in which to study them.

There are many ways in which one could build this kind of model or framework. In this dissertation I present a particular framework for studying MRTA, based on organizational theory from several fields, including operations research, economics, scheduling, network flows, and combinatorial optimization. I show how this framework can be used to classify MRTA problems, and evaluate and compare proposed solutions. I also suggest how the framework can be extended to provide practical advice regarding the design of algorithms for solving MRTA problems. This framework is not meant to be final or exhaustive and indeed it has some limitations. However, I believe that the ideas presented in this dissertation constitute a starting point on a path toward developing a fundamental understanding of problems involving MRTA, as well as other aspects of multi-robot coordination.

1.3 Contributions

This dissertation makes five primary contributions to the study of multi-robot systems:

1. **Common organizational framework**. I show how MRTA problems can be discussed formally within the context of relevant organizational theory (Chapter 3). This novel perspective provides a common framework in which to study many seemingly different MRTA problems. As opposed to studying such problems in isolation and in an *ad hoc* manner, the use of a common theoretical framework provides the opportunity to make general and lasting conclusions about the nature of a wide variety of MRTA problems. Furthermore, since the underlying theory is domain-independent, this framework need not be limited to robot problems; rather it can be applied more generally to the study of autonomous coordination, which would include multi-agent systems, sensor-actuator networks, and intelligent embedded systems.

2. **Validated economic solution**. Within this organizational framework, theory from economics and operations research suggests that synthetic price-based markets can be used to efficiently solve certain task allocation problems. I show by experimentation with teams of robots that this prediction holds true, even for relatively complex MRTA problems (Chapter 4). My work with MURDOCH, an auction-based MRTA architecture, represents the first application of market-oriented techniques to the coordination of physical robots.

3. **Taxonomy of MRTA problems**. I propose a taxonomy of MRTA problems that is based on the common organizational framework (Chapter 5). This taxonomy differentiates MRTA problems by characteristics such as whether or not multi-robot tasks are allowed. I show where many familiar MRTA problems fall within this taxonomy. Such classification in turn allows for an understanding of how various MRTA problems both differ from and relate to each other.

4. **Analysis of existing solutions**. Using results from combinatorial optimization and economics in particular, I analyze several existing MRTA architectures, including MURDOCH (Section 5.2.3). I derive domain-independent characteristics, such as expected solution quality, of the algorithms that underlie these architectures. This analysis provides the opportunity to objectively evaluate and compare different solutions, an important undertaking that has, to date, been missing in the field.

5. **Supporting infrastructure**. Because multi-robot teams are complex embodied systems that can exhibit significant amounts of noise and uncertainty, theoretical analysis alone is insufficient for their study. Proposed solutions *must* be implemented and tested, either in the physical world or in realistic simulation. This testing requires substantial supporting software infrastructure, the importance of which is often underestimated. To this end, I (in collaboration with others) maintain the Player/Stage project, which produces the robot device server Player and the multi-robot simulator Stage (Chapter 6). I lead development of Player and have contributed significantly to Stage. Originally designed specifically to support robot experiments in the USC Robotics Research Lab, Player and Stage have recently come into widespread use and now represent indispensable infrastructure in many research

labs around the world. In large part because of its Open Source status, the Player/Stage project is an important contribution to and resource for the research community.

1.4 Outline

The rest of this dissertation is organized as follows. In the next chapter, I discuss previous work relevant to the topic at hand. Since this dissertation is concerned with the construction and application of a formal framework of MRTA, I focus on similar frameworks from the literature.

Chapter 3 gives background on the organizational theory that I leverage in studying MRTA problems. To ground the discussion, I show how a particular MRTA problem can be understood as an instance of the well-known Optimal Assignment Problem (OAP), and describe how this problem is approached in several disciplines, including operations research, economics, scheduling, network flows, and combinatorial optimization. These disciplines provide different perspectives on the OAP and suggest a variety of techniques for solving it.

As explained in Chapter 3, there is a natural economic interpretation of the some MRTA problems. In Chapter 4, I describe MURDOCH, an auction-based MRTA architecture that that I have developed and extensively validated through experimentation with physical robots, using the infrastructure discussed in Chapter 6. MURDOCH is a variant of the well-known Contract Net Protocol, or CNP (Smith 1980, Davis & Smith 1983). As such, this architecture provides, in a distributed manner, a tradeoff between communication overhead and solution quality, producing greedy solutions at a low cost.

In Chapter 5, I propose a taxonomy of MRTA problems, relying on the theory discussed in Chapter 3. I show where many common MRTA problems fall within this taxonomy, and suggest how relevant theory and techniques from various organizational disciplines can be used to understand and solve them. Several MRTA architectures from the literature (including MURDOCH) are classified and analyzed, with the goal of determining their algorithmic characteristics, such as communication overhead and solution quality.

In order to explore MRTA problems, such as the ones described in Chapter 5, one must construct and experimentally evaluate multi-robot systems. These experiments require substantial supporting infrastructure, both software and hardware. This infrastructure must be *embodied*, either in the physical world or in realistic simulation. In Chapter 6 I motivate and describe the underlying facilities that I developed for experimental use in this dissertation, as well as for more general use.

I conclude in Chapter 7 with a summary of the contributions of this dissertation. I also discuss ongoing and future work toward formalizing problems in multi-robot systems.

Chapter 2

Related work

As explained in the previous chapter, the goal of this dissertation is to establish a formal, yet practical, understanding of problems involving multi-robot task allocation (MRTA). To that end Chapters 3 & 5 develop an analytical MRTA framework that is based on organizational theory from disciplines such as operations research. The current chapter discusses previous work related to this topic, concentrating on similar frameworks developed in the robotics and multi-agent systems communities. Discussion of previous work that is related to other, more specific topics can be found elsewhere in this dissertation, including Sections 4.5 & 6.1.5.

2.1 Robotics

Research in multi-robot systems has been focused primarily on construction and validation of working systems, rather than analysis of problems and solutions. As a result, in the literature, one can find a great many *architectures* for multi-robot coordination, but relatively few formal *models* of multi-robot coordination. I do not attempt in this chapter to cover the various proposed and demonstrated architectures; rather I treat the prominent models. For discussion and analysis of several key architectures, see Section 5.2.3. Some models for *single*-robot control have been proposed and tested, including Subsumption (Brooks 1986) and motor schemas (Arkin 1987). While such models can serve as guides to programming and describing the

behavior of individual robots, they have not been shown to apply meaningfully to systems composed of multiple robots.

Formal models of coordination in multi-robot systems tend to target medium- to large-scale systems composed of simple, homogeneous robots, such as the CEBOTS (Fukuda et al. 1988). Agassounon & Martinoli (2002) explored the tradeoffs between using a coarse, macroscopic model of such systems and using detailed, microscopic models of the individuals. Lerman & Galstyan (2002) presented a physics-inspired macroscopic model of a cooperative multi-robot system and showed that this model accurately describes the behavior of physical robots engaged in stick-pulling and foraging tasks. They also also showed how the system behavior can be explained by Boolean network theory (Galstyan & Lerman 2002). Although this kind of model is *descriptive*, is it not *prescriptive* in that it does not suggest how to design control or coordination mechanisms for multi-robot systems. Klavins (2002*a*) developed a prescriptive model for microscopic self-assembling robots, suggesting how to automatically generate local assembly rules that produce target shapes.

Though simple and elegant, such models are insufficient for domains involving complex tasks or requiring precise control. To study the case of complex tasks, Donald, Jennings & Rus (1997*a*) proposed the formalism of *information invariants*. This formalism models the information requirements of a coordination algorithm and provides a mechanism to perform reductions between algorithms. Information invariants were used to study cooperative multi-robot box-pushing (Donald, Jennings & Rus 1997*b*). Spletzer & Taylor (2001) developed a prescriptive control-theoretic model of multi-robot coordination and showed that it can be used to produce precise multi-robot box-pushing. Lynch & Mason (1995) had earlier applied a similar control-theoretic model to box-pushing with dexterous manipulators.

Relatively little work has been done on formal modeling, analysis, or comparison of multi-robot coordination at the level of task allocation. One exception is the work of Chien, Barrett, Estlin & Rabideau (2000), who developed a baseline geological scenario and used it to compare three different planning approaches to coordinating teams of planetary rovers. Also, Klavins (2002*b*) showed how to apply the theory

of communication complexity (Kushilevitz & Nisan 1997) to the study of multi-robot coordination algorithms. Finally, Jennings & Kirkwood-Watts (1998) described the method of *dynamic teams*, concentrating on programmatic structures that enable the specification of multi-robot tasks.

2.2 Multi-agent systems

As compared with robotics, research in multi-agent systems (MAS), also known as distributed artificial intelligence (DAI), has produced a significant amount of formalism regarding coordination. Perhaps the best-known model is the *belief-desire-intention* (BDI) model (Georgeff & Lansky 1987). The BDI formalism, which models reasoning agents, is based on folk-psychology ideas about human motivations (Bratman, Israel & Pollack 1988). To my knowledge, this formalism has not been applied (or otherwise shown to be applicable) to robot systems, and there is some debate within the multi-agent community as to how well it models agent systems (Georgeff, Pollack, Tambe & Wooldridge 1999).

The BDI formalism has been used in the development of several multi-agent coordination frameworks, including *SharedPlans* (Grosz & Kraus 1993) and *joint intentions* (Cohen & Levesque 1991, Jennings 1995). In particular, the joint intentions model allows for the specification of pre-conditions that must be satisfied for agents to begin a coordinated activity. Furthermore, the model provides methods for agents to determine how they should behave during a coordinated activity and how to assess group performance. The joint intentions framework has been instantiated in a multi-agent coordination architecture, called STEAM (Tambe 1997), which was demonstrated in a simulated helicopter coordination scenario. The architecture was extended to provide some amount of lookahead in order to efficiently handle persistent teams (Tambe & Zhang 2000).

Other popular frameworks for describing multi-agent systems include the Markov Decision Process (MDP) model and the Partially Observable Markov Decision Process (POMDP) model (Sondik 1978, Singh, Jaakkola & Jordan 1994). Even for relatively small systems, such Markovian models are notoriously difficult to use directly for policy-generation. Thus researchers have investigated a variety of

methods for factoring, distributing, and otherwise decomposing these models into smaller pieces that can be tractably solved (Milch & Koller 2000, Guestrin, Koller & Parr 2001, Jung & Tambe 2003, Nair, Tambe & Marsella 2003). Such an approach necessarily incurs a penalty in terms of solution quality, and even with decomposition, these methods are currently computationally limited to small-scale systems. However, the general approach of probabilistically modeling multi-agent domains in a Markovian manner has the potential to accurately represent complex systems, as well as generate high-quality policies.

Multi-agent coordination problems have also been modeled as distributed constraint satisfaction problems (Yokoo, Durfee, Ishida & Kuwabara 1998), including the work of Clearwater, Huberman & Hogg (1991). Jung, Tambe & Kulkarni (2001) explored the use of argumentation as a method for solving such problems, and others (e.g., Modi, Jung, Tambe, Shen & Kulkarni (2001), Modi, Shen, Tambe & Yokoo (2003)) have described methods for solving dynamic distributed constraint satisfaction problems.

Perhaps most relevant to the topic of this dissertation is the work of Pynadath & Tambe (2002), who propose a unified framework for evaluating multi-agent coordination algorithms. Their framework, the *Communicative Multiagent Team Decision Problem* (COM-MTDP), allows for a theoretically sound complexity analysis of various general classes of multi-agent coordination problems. The framework has also been applied to the study of specific multi-agent problems, and can be used to predict the solution quality of particular multi-agent coordination algorithms.

There exist several other formal frameworks for studying multi-agent problems and algorithms. van Alstyne (1997) examined distributed systems from the point of view of *network organizations*, focusing on their organizational structure. d'Inverno, Luck & Wooldridge (1997) proposed a negotiation framework called *cooperation structures*, and Kraus, Wilkenfeld & Zlotkin (1995) have explicitly studied the effect of the passage of time in such negotiations.

2.3 Summary

In this chapter, I have presented work relevant to the primary topic of this dissertation, which is the development of a formal framework for studying MRTA problems. As can be seen, although relatively little such work has been done in the field of robotics, a significant effort to this end has been made in the agent community. It is an open question as to whether theories and techniques developed for multi-agent systems can be applied to robot domains; there has recently been some interest in resolving this question (Scerri, Modi, Tambe & Shen 2003). In the next chapter, I lay the theoretical foundation for a framework of task allocation that is aimed specifically at the study of the multi-robot systems.

Chapter 3

The problem

I view the problem of problem of multi-robot task allocation (MRTA) as fundamentally organizational in nature. That is, given some resources, some tasks, and a performance metric, the goal of the system is to maximize performance on the tasks, subject to the resource constraints. Such organizational problems have been studied scientifically for over a century, with early efforts concentrated in economics (Edgeworth 1881). Interest in these problems has since arisen in several fields, including game theory (von Neumann & Morgenstern 1964), operations research (Gale 1960), scheduling (Brucker 1998), network flows (Ahuja, Magnanti & Orlin 1993), and combinatorial optimization (Korte & Vygen 2000). Each field has a particular view of organizational problems and although the solutions that they produce are often very similar, each viewpoint provides additional insight into the nature of the problems under study.

At first glance, describing MRTA in terms of resources and tasks may seem too abstract to be useful in designing real multi-robot systems. To the contrary, this general problem statement in fact produces methods for both synthesis and analysis of MRTA systems by permitting the introduction of theoretical results from the rich body of work concerned with organizational problems. I discuss and apply these methods in Chapters 4 & 5. In this chapter, I briefly provide background in the relevant theory arising in various disciplines. In order to motivate and ground the discussion of organizational theory, I first explain the relevance to MRTA of the Optimal Assignment Problem, a classic organizational problem that has been independently explored in many fields.

3.1 The Optimal Assignment Problem

The *Optimal Assignment Problem* (OAP) (Gale 1960) is a well-known problem that was originally studied in game theory (von Neumann 1947) and then in operations research, in the context of personnel assignment (Thorndike 1950, Dwyer 1954). A recurring special case of particular interest in several fields of study, this problem can be formulated in many ways. Given the application domain of MRTA, it is fitting to describe the problem in terms of jobs and workers.

Definition 3.1 (Optimal Assignment Problem (OAP)) : Given are m workers, each looking for one job; and n possibly weighted jobs, each requiring one worker. Also given for each worker is a nonnegative skill rating estimating his/her performance for each job (if a worker is incapable of undertaking a job, then the worker is assigned a rating of zero for that job). The goal is to assign workers to jobs so as to maximize overall expected performance, taking into account the priorities of the jobs and the skill ratings of the workers.

The MRTA problem can be posed similarly:

Definition 3.2 (Multi-Robot Task Allocation (MRTA)) : Given are m robots, each capable of executing one task; and n possibly weighted tasks, each requiring one robot. Also given for each robot is a nonnegative efficiency rating estimating its performance for each task (if a robot is incapable of executing a task, then the robot is assigned a rating of zero for that task). The goal is to assign robots to tasks so as to maximize overall expected performance, taking into account the priorities of the tasks and the efficiency ratings of the robots.

This definition of MRTA is clearly restricted, in several respects. For example, most task allocation problems are not static; they are dynamic decision problems that vary in time with phenomena such environmental changes and robot failures. I address such questions in detail in Chapter 5. For the remainder of the current discussion, I use Definitions 3.1 & 3.2 as examples because they are simple and instructive.

In the following sections, I show how the OAP and MRTA are viewed and solved in several fields. First, however, I address the question of how, in multi-robot systems, to calculate performance estimates, which are also called *utility* estimates.

3.1.1 Utility

Utility is a unifying, if sometimes implicit, concept in economics, game theory, and operations research, as well as multi-robot coordination. The idea is that each individual can somehow internally estimate the

value (or the cost) of executing an action. It is variously called fitness, valuation, and cost. Within multi-robot research, the formulation of utility can vary from sophisticated planner-based methods (Botelho & Alami 1999) to simple sensor-based metrics (Gerkey & Matarić 2002*c*). I posit that utility estimation of this kind is carried out somewhere in every autonomous task allocation system (for more examples, see Chapter 5). Since each system uses a different method to calculate utility, I now give an instructive and generic, yet practical, definition of utility for multi-robot systems.

It is assumed that each robot is capable of estimating its fitness for every task of which it is capable. This estimation includes two factors, both task- and robot-dependent:

- expected quality of task execution, given the method and equipment to be used (e.g., the accuracy of the map that will be produced using a laser range-finder)
- expected resource cost, given the spatio-temporal requirements of the task (e.g., the power that will be required to drive the motors and laser range-finder in order to map the building)

Given a robot R and a task T, if R is capable of executing T, then one can define, on some standardized scale, Q_{RT} and C_{RT} as the quality and cost, respectively, expected to result from the execution of T by R. I can now give a combined, nonnegative utility measure[1]:

$$U_{RT} = \begin{cases} Q_{RT} - C_{RT} & \text{if } R \text{ is capable of executing } T \text{ and } Q_{RT} > C_{RT} \\ 0 & \text{otherwise} \end{cases}$$

Thus, given a robot A that can achieve a task T with quality $Q_{AT} = 20$ at cost $C_{AT} = 10$ and a robot B that can achieve the same task with quality $Q_{BT} = 15$ at cost $C_{BT} = 5$, there should be no preference between them when searching for efficient assignments, for:

$$U_{AT} = 20 - 10 = 10 = 15 - 5 = U_{BT}.$$

[1] I thank Michael Wellman for suggesting this formulation.

Regardless of the method used for calculation, the robots' utility estimates will be inexact for a number of reasons, including sensor noise, general uncertainty, and environmental change. These unavoidable characteristics of the multi-robot domain will necessarily limit the efficiency with which coordination can be achieved. I treat this limit as exogenous, on the assumption that lower-level robot control has already been made as reliable, robust, and precise as possible and thus that we are incapable of improving it. When I discuss "optimal" allocation solutions, I mean "optimal" in the sense that, given the union of all information available in the system (with the concomitant noise, uncertainty, and inaccuracy), it is impossible to construct a solution with higher overall utility; this notion of optimality is analogous that used in optimal scheduling (Dertouzos & Mok 1983).

It is important to note that utility is an extremely flexible measure of fitness, into which arbitrary computation can be placed. The only constraint on utility estimators is that they must each produce a single scalar value such that they can be compared for the propose of ordering candidates for tasks. For example, if the metric for a particular task is distance to a location and the robots involved employ a probabilistic localization mechanism, then a reasonable utility estimation might be to calculate the distance to the target using the center of mass of the current probability distribution. Other mechanisms, such as planning and learning, can likewise be incorporated into utility estimation. No matter the domain, it is vital that *all* relevant aspects of the state of the robots and their environment be included in the utility calculation. Signals that are left out of this calculation but are taken into consideration when evaluating overall system performance are what economists refer to as *externalities* (Simon 2001) and their effects can be detrimental, if not catastrophic.

3.1.2 Formalism

The MRTA problem can now be stated as an instance of the OAP. Given:

- the set of m robots, denoted $I_1, \ldots, I_m$
- the set of n prioritized tasks, denoted $J_1, \ldots, J_n$ and their relative weights $w_1, \ldots, w_n$

- U_{ij}, the nonnegative utility of robot I_i for task J_j, $1 \leq i \leq m, 1 \leq j \leq n$

Assume:

- Each robot I_i is capable of executing at most one task at any given time.
- Each task J_j requires exactly one robot to execute it.

These assumptions, though somewhat restrictive, are necessary in order to reduce MRTA to the classical OAP, which is given in terms of single-worker jobs and single-job workers. I relax these assumptions and analyze the consequences in Chapter 5. It is worth noting that in most existing MRTA work, these same assumptions are made, though often implicitly.

The problem is to find an optimal allocation of robots to tasks. An allocation is a set of robot-task pairs:

$$(i_1, j_1) \ldots (i_k, j_k),\ 1 <= k <= \min(m, n)$$

Given the assumptions made above, for an allocation to be *feasible*, the robots $i_1 \ldots i_k$ and the tasks $j_1 \ldots j_k$ must be unique. The benefit (i.e., expected performance) of an allocation is the weighted utility sum:

$$U = \sum_{l=1}^{k} U_{i_l j_l} w_{j_l}. \tag{3.1}$$

The utilities U_{ij} are usually represented in a matrix form, called the *utility matrix* (also called the *rating matrix*). Some algorithms (e.g., the Hungarian method; see Section 3.2.2) are designed to minimize, rather than maximize, the sum (3.1). In this case, we can construct a *cost matrix* K from a utility matrix U by subtracting each element in U from the maximum value in U:

$$\begin{aligned} U_{max} &= \max_{i,j} \{U_{ij}\} \\ K_{ij} &= U_{max} - U_{ij},\ \forall i, j. \end{aligned}$$

Any assignment that minimizes the resulting cost sum in K also maximizes the original utility sum in U. Some algorithms require that the utility (or cost) matrix be symmetric; i.e., $m = n$. Given an asymmetric utility matrix, we can construct a symmetric utility matrix by adding zero rows or zero columns as necessary, and not carry out any resulting assignments that involve those rows or columns. When only n is mentioned in the complexity results given below, it is assumed that the problem is symmetric.

3.2 Operations research and linear programming

It was optimization problems like the OAP that spurred George Dantzig, while working for the United States Air Force shortly after WWII, to develop the theory of *linear programming* (LP) and invent the simplex method (Dantzig 1982); see Section 3.2.2. In so doing, Dantzig, with the help of others, including Tjalling Koopmans and John von Neumann, founded the field that has come to be known as "operations research" or "management science" (Dorfman 1984). To this day, much of operations research is concerned with LP problems, and LP techniques are in widespread use in many areas, especially industrial resource allocation applications.

The canonical maximum linear programming problem is formulated as follows (Gale 1960):

Definition 3.3 (Maximum Problem for a Linear Program) : Given an $m \times n$ matrix A, an n-vector b, and an m-vector c, find a nonnegative m-vector x such that

$$xc \text{ is a maximum}$$

subject to

$$xA \leq b.$$

The canonical minimum problem can be formulated analogously:

Definition 3.4 (Minimum Problem for a Linear Program) : Given an $m \times n$ matrix A, an n-vector b, and an m-vector c, find a nonnegative m-vector x such that

$$xc \text{ is a minimum}$$

subject to

$$xA \geq b.$$

In either case, if there exists a vector x that satisfies the constraints, then the problem is called *feasible* and the vector x is a *feasible solution*. A feasible solution is called an *optimal solution* if there is no other feasible solution that produces a higher (lower) value in the function to be maximized (minimized).

Often the elements of the solution vector x are required to be integers; in this case, the problem is called an *integral linear program* (ILP). The most general case of such problems, commonly called the *0-1 integer programming problem*, is $\mathcal{NP}$-hard (Ahuja et al. 1993). However, many interesting ILPs (including the OAP) exhibit special structure that facilitates their solution.

3.2.1 MRTA as an integral linear program

The MRTA problem can be cast as an ILP in the following way (Gerkey & Matarić 2003*a*):

Find n^2 nonnegative integers α_{ij} that maximize

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_{ij}U_{ij}w_j \tag{3.2}$$

subject to

$$\begin{aligned}\sum_{i=1}^{n}\alpha_{ij} &= 1, \quad 1 \leq j \leq n \\ \sum_{j=1}^{n}\alpha_{ij} &= 1, \quad 1 \leq i \leq n.\end{aligned} \tag{3.3}$$

The sum (3.2) is just the overall system utility, while (3.3) enforces the constraint that we are working with single-robot tasks and single-task robots (note that since α_{ij} are integers they must all be either 0 or 1). Given an optimal solution to this problem (i.e., a set of integers α_{ij} that maximizes (3.2) subject to (3.3)), we construct an optimal task allocation by assigning robot i to task j only when $\alpha_{ij} = 1$.

By creating a *linear* program, we restrict the space of task allocation problems that we can model in one way: the function to be maximized (3.2) must be linear. Importantly, there is no such restriction on the

manner in which the components of that function are derived. That is, individual utilities can be computed in any arbitrary way, but they must be combined linearly.

3.2.2 The *simplex* and Hungarian methods

Early in the study of linear programming, Dantzig (1951) introduced the *simplex method* for solving linear programs. The simplex method has since become one of the most celebrated algorithms in mathematical computing and is still the algorithm of choice for solving linear programs. For an explanation of the simplex method, see for example Gale (1960), Dantzig (1963), Schrijver (1986), or Winston (1991). The basic idea of the method is to traverse the edges of the polygon that underlies the linear program until an optimal vertex is found (Schrijver 1986). Dantzig, Orden & Wolfe (1955) made an important revision to the simplex method in order to prevent the occurrence of cycles during this traverse. Although the worst-case complexity of the simplex method is exponential in the size of the input (Klee & Minty 1972), the average-case complexity for certain classes of problems is polynomial (Borgwardt 1982), and the method is known to work very well in practice[2] (Spielman & Teng 2001).

While it can be solved by a general-purpose linear programming algorithm such as the simplex method, the OAP exhibits a special structure that can be exploited in order to construct more efficient algorithms. The first (and most famous) such specialized algorithm is Kuhn's (1955) *Hungarian method*, so named because it is based on the work of the Hungarian mathematicians D. König and J. Egerváry. Given an $n \times n$ cost matrix, The Hungarian method can be summarized in the following way (Winston 1991):

Algorithm 3.1 (The Hungarian method) :

1. Construct the *reduced cost matrix* by subtracting from each element the minimum element in its row and the minimum element in its column.

2. Find the minimum number of horizontal and vertical lines required to cover all the zeros in the reduced cost matrix. If n lines are required, then an optimal assignment is available in the covered zeros. If fewer than n lines are required, go to Step 3.

[2]Because it is an integral problem, the OAP would require use of the *binary* simplex method, which will not necessarily exhibit such amenable performance.

3. Find the smallest nonzero uncovered element in the reduced cost matrix. Subtract that value from each uncovered element of the reduced cost matrix and add it to each twice-covered element in the reduced cost matrix. Go to Step 2.

As one might expect, the computationally difficult part of the Hungarian method lies in finding the minimum number of covering lines in Step 2. For details on this procedure, consult Kuhn (1955) and Kuhn (1956).

The running time of the Hungarian method for an $n \times n$ assignment problem is $O(n^3)$ (Jungnickel 1999). Many others have developed $O(n^3)$ assignment algorithms, including: Engquist (1982), Hung (1983), Balinski (1985), Goldbarb (1985), and Balinski (1986). For performance results with a C implementation of the Hungarian method, see Section 5.2.

3.2.3 Duality

Fundamental to the theory of linear programming is the concept of duality. Given a maximum LP problem, it is possible to construct a related minimum LP problem (and vice versa). The original problem is called the *primal* and the related problem is called the *dual* (Gale 1960):

Definition 3.5 (Dual of a Maximum Problem for a Linear Program) : The *dual* of a maximum problem over a linear program (Definition 3.3) is that of finding an n-vector y such that:

$$yb \text{ is a minimum}$$

subject to

$$Ay \geq c.$$

The dual of a minimum problem can be defined analogously. Duality is particularly important because of the Duality Theorem, which was originally asserted by von Neumann (1947) and first rigorously proved by Gale, Kuhn & Tucker (1951). This theorem shows that the primal and dual problems are actually equivalent (Gale 1960):

Theorem 3.1 (Duality Theorem for Linear Programs) : If both a problem and its dual are feasible then both have optimal solutions and the values of the optimal solutions are the same. If either is infeasible, then neither has an optimal solution.

The dual of the MRTA problem can be stated as follows: find n integers u_i and n integers v_j that minimize:

$$\sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j$$

subject to:

$$u_i + v_j \geq U_{ij}, \ \forall i, j.$$

Thus, given an instance of the MRTA problem, we can trivially construct its dual, then solve whichever form is more computationally convenient; there exist algorithms for each. Furthermore, the dual of the MRTA problem has an important economic interpretation, in which the values u_i and v_j represent the maximum profits to be made by selfish individuals interacting in a price-based market. This idea is explored in the next section.

3.3 Economics and game theory

There has long been a strong connection between (micro)economics and game theory (von Neumann & Morgenstern 1964), and game-theoretic techniques are often used to analyze and synthesize rules for economic systems. In the context of a game, a problem is usually given as a collection of autonomous participants and a set of rules that govern their interactions. Over time, the participants must make decisions in accordance with the rules. It is assumed that the participants are selfish, in that each participant makes utility-maximizing decisions. It was during the early stages of linear programming that von Neumann (1947) first noticed the relationship between LP extremum problems and certain economic / game-theoretic problems (Dantzig 1982).

3.3.1 MRTA as an economic game

The MRTA problem can be posed as an economic game in the following way (Shapely & Shubik 1972). Construct a price-based *task market*, in which tasks are sold by brokers to robots. Each task j is for sale

by a broker, which places a value c_j on the task. Also, robot i places a value h_{ij} on task j. The problem then is to establish task *prices* p_j, which will in turn determine the allocation of tasks to robots. To be *feasible*, the price p_j for task j must be greater than or equal to the broker's valuation c_j; otherwise, the broker would refuse to sell. Assuming that the robots are acting selfishly, each robot i will elect to buy a task t_i for which its profit is maximized:

$$t_i \in \underset{j}{\operatorname{argmax}} \{h_{ij} - p_j\}$$

Such a market is said to be at *equilibrium* when prices are such that no two robots select the same task.

At equilibrium, each individual's profit in this market is maximized. Furthermore, the profits made by the robots and the profits made by the brokers form an optimal solution to the dual of the MRTA problem (see Section 3.2.3):

$$\begin{aligned} u_i &= h_{it_i} - p_{t_i}, \ \forall i \\ v_j &= p_j - c_j, \ \forall j. \end{aligned}$$

Thus, the allocation produced by the market at equilibrium is optimal (Gale 1960).

In the MRTA problem, we are not given separate valuations in this manner, but only combined utility estimates for robot-task pairs. However, we can easily define task valuations for the robots and brokers:

$$\begin{aligned} h_{ij} &= \alpha_{ij} \\ c_j &= 0. \end{aligned}$$

The solution to the corresponding dual problem then becomes:

$$\begin{aligned} u_i &= \alpha_{it_i} - p_{t_i} \\ v_j &= p_j. \end{aligned}$$

Note that by setting c_j to 0, we are implicitly stating that the brokers always prefer to sell their tasks, regardless of how much they are paid. In other words, it is always better to execute a task than not execute it, regardless of the expected performance. In economic terminology, we have *lexicographic* preferences with regard to the tasks[3] (Pearce 1999). Such preferences violate important assumptions concerning the nature of utility values that are made when building or analyzing general economic systems. Fortunately, in constructing the market corresponding to the MRTA problem, no assumptions are made concerning the robots' preferences, and so lexicographic preferences do not present a problem. On the other hand, the behavior of more complex, long-lived economies (such as the markets suggested by Dias & Stentz (2001) and Gerkey & Matarić (2002*a*)) may depend strongly the nature of the robots' preferences, especially if the synthetic economies are meant to interact with the human economy.

3.3.2 Auction algorithms

A number of algorithms are available for solving this economic form of the MRTA problem. In most of the approaches, prices are determined by some kind of *auction* in which the participants make *bids* on items of interest (McAfee & McMillan 1987). One of the best-known auction algorithms is due to Bertsekas (1990). For the purposes of this algorithm, ϵ is a positive scalar and a robot i is said to be "happy" if and only if it is assigned to a task t_i for which its profit is ϵ-maximized, given current prices:

$$\alpha_{it_i} - p_{t_i} \geq \max_j \{\alpha_{ij} - p_j\} - \epsilon.$$

[3] I thank Herbert Dawid for pointing this out.

Algorithm 3.2 (Bertsekas's Auction algorithm) :

1. Randomly assign tasks to robots and prices to tasks.
2. Randomly select a robot i that is not happy; if no such robot exists, then the market has reached equilibrium; quit.
3. Find a task t_i that maximizes profit for robot i:
$$t_i \in \operatorname*{argmax}_j \{\alpha_{ij} - p_j\}.$$
4. Swap tasks between robot i and the robot that is currently assigned task t_i.
5. Increment the price of t_i by:
$$\gamma_i = v_i - w_i + \epsilon$$
where v_i is the expected profit to robot i from its most preferred task:
$$v_i = \max_j \{\alpha_{ij} - p_j\}$$
and w_i is the expected profit to robot i from its second-most preferred task:
$$w_i = \max_{j \neq t_i} \{\alpha_{ij} - p_j\}.$$
6. Return to Step 2.

This auction algorithm is guaranteed to terminate. At termination it produces an assignment for which the overall utility is within $n\epsilon$ of the optimal utility; if the utilities α_{ij} are integral and if $\epsilon < \frac{1}{n}$, then the assignment is optimal (Bertsekas 1990).

Others have developed similar auction-based assignment algorithms, including Gale & Shapley (1962), Crawford & Knoer (1981), and Demange, Gale & Sotomayor (1986). Massively parallel auctions have also been explored (Wein & Zenios 1991, Bertsekas & Castañon 1991). For performance results with a C implementation of the Auction algorithm, see Section 5.2.

3.4 Scheduling

A scheduling problem is formulated as follows: given a set of *machines*, a set of *jobs* to be processed, and a performance criterion, construct a *schedule* of jobs for each machine such that performance is maximized. Following Brucker's (1998) terminology, a scheduling problem is formally defined by three variables:

machine environment (α), job characteristics (β), and optimality criterion (γ). Traditionally, a problem's classification is given as a triple:

$$\alpha \mid \beta \mid \gamma.$$

The machine environment (α) determines how many machines are available, which jobs each machine can process, and how fast each machine can be expected to process a given job. For our purposes, α will generally be either P, to indicate an arbitrary number of *identical parallel machines* that all process each job at the same rate; or R, to indicate an arbitrary number of *unrelated parallel machines* that potentially process each job at a different rate. The job characteristics (β) determine what relationships and constraints (e.g., precedence) exist between the jobs. For our purposes, β will generally be empty, to indicate that the individual jobs are not related to or dependent upon each other. The optimality criterion (γ) determines the goal of the scheduling problem and is usually some aspect of the time taken to process the jobs, such as finishing time or lateness. For our purposes, γ will generally be C_{max}, to indicate that we aim to minimize the maximum time taken to complete any job; $\sum C_j$, to indicate that we aim to minimize the sum of processing times over all jobs; or $\sum w_j C_j$, to indicate that we aim to minimize the *weighted* sum of processing times over all jobs.

3.4.1 MRTA as a scheduling problem

The MRTA problem is in the class of scheduling problems described by:

$$R \mid\mid \sum w_j C_j. \tag{3.4}$$

That is, the system is composed of unrelated parallel machines and overall performance is computed as a weighted sum of the processing times for the individual tasks (Gerkey & Matarić 2003*b*). In this case, we define the processing time C_j as K_{ij}, the cost expected from the execution of task j by the robot i to which j is assigned.

The problem class (3.4) is a superset of the simpler case of identical parallel machines:

$$P \mid\mid \sum w_j C_j \tag{3.5}$$

Problems in the class (3.5) are known to be strongly $\mathcal{NP}$-hard (Brucker 1998), and thus so are problems in the class (3.4).

Fortunately, we can simplify our problem by making two domain-specific observations. First, we recognize that MRTA is a degenerate scheduling problem. Whereas in scheduling one must assign tasks to machines over time, in MRTA we consider only a single time-slot. Second, we can incorporate the task weights directly into the cost estimates if we make the reasonable assumption that the task weights are known to the robots and can be used in cost estimation. Given a cost estimate K_{ij} for robot i and task j and a scalar task weight w_j, we can define a new weighted cost estimate:

$$K'_{ij} = w_j K_{ij}.$$

For a single time-slot, we can trivially make this change of variables for each of the mn utilities K_{ij} in $O(mn)$ time. Thus, our particular problem reduces to an unweighted scheduling problem, becoming an instance of the class:

$$R \mid\mid \sum C_j \tag{3.6}$$

Problems in the class (3.6) are known to be polynomially solvable, for example by Bruno, Coffman & Sethi's (1974) job scheduling algorithm, which runs in $O(mn^3)$ time.

3.4.2 Preemption

In some scheduling problems, the job characteristics β contain the special token *pmtn* to indicate that preemption, or job splitting, is allowed. If preemption is allowed, then a job can be interrupted during execution, moved to a different machine, and then resumed. On the second machine, the remaining execution

cost (e.g., processing time) for the job will be reduced by the cost incurred from partial execution on the first machine. That is, work is conserved and transferable. In general, a job can be interrupted and moved multiple times.

Preemption is a powerful tool in scheduling problems, but a large class of MRTA problems cannot be meaningfully modeled to allow preemption, for two reasons. First, robots by definition engage in physical tasks, and the significant physical state associated with such tasks can be difficult or even impossible to transfer from one robot to another. As an extreme example, imagine a robot that, like Sisyphus, is tasked to roll a heavy stone up a hill. Handing the task off to another robot, without letting the stone roll back down the hill, is likely to be infeasible, or at least extremely costly. Second, many robot tasks permit only binary progress: the task is either achieved or not achieved, and work *toward* achievement is not transferable. Any task that primarily consists of physically moving to a goal location has this property. If a robot is reassigned after moving halfway to the goal, there is no way to leverage that robot's sunk costs. Having said that, there are some continuous monitoring-style tasks, such as object-tracking (see Section 5.2), for which preemption can be advantageous.

3.5 Network flows

In a general network flow problem, we are given a weighted graph, with maximum capacities for the edges. Nodes in the graph that can provide a common resource are called *sources*; nodes that require the common resource are called *sinks*. Example problems that can be defined over such networks include: find a shortest path, find the maximum flow, and find the minimum cost flow. Of particular relevance to the current discussion is the minimum cost flow problem (Ahuja et al. 1993):

Definition 3.6 (Minimum Cost Network Flow Problem) : Given are a directed graph $G = (V, E)$ with $n = |E|$, nonnegative edge costs $c_e,\ \forall e \in E$, and edge capacities $k_e,\ \forall e \in E$. Also given are vertex demands $b_v,\ \forall v \in V$. If $b_v < 0$, then v is a sink; if $b_v > 0$, then v is a source. For any vertex $v \in V$, let $\delta(v)$ and $\gamma(v)$ denote the set of edges entering and leaving v, respectively. Find n flows f_e that minimize:

$$\sum_{e \in E} f_e c_e$$

subject to:

$$\begin{aligned} \sum_{e \in \gamma(v)} f_e - \sum_{e \in \delta(v)} f_e &= b_v, \forall v \in V \\ f_e &\leq k_e, \forall e \in E. \end{aligned}$$

That is, find the cheapest way to satisfy demands, subject to source and capacity constraints. This problem arises in many situations, and can be used to formulate and solve many other problems, including transportation problems, multicommodity flow problems, and assignment problems.

3.5.1 MRTA as a network flow problem

The MRTA problem can be stated as a network flow problem as follows (Ahuja et al. 1993). Construct a bipartite graph $G = (R \cup T, E)$, with $n = |R| = |T|$; R is the set of robots, T is the set of tasks, and E is the set of possible robot-task pairs. The cost c_e associated with an edge is defined to be the cost estimate of the underlying robot-task pair. Now add to G a source node p with demand $b_p = n$ and a sink node q with demand $b_q = -n$. For each robot $r_i \in R$, add to G a zero-cost edge linking p and r_i. Likewise, for each task $t_i \in T$, add to G a zero-cost edge linking t_i and q. The capacities k_e for all edges in the graph are 1.

The MRTA problem can then be solved by finding the minimum cost integral flow in the augmented graph. One way to solve this flow problem is by finding successive shortest paths from the source p to the sink q. By implementing this search with Dijkstra's algorithm on Fibonacci heaps, an $m \times n$ MRTA problem can be solved in $O(mn + n^2 \log n)$ time; this is the fastest known algorithm for the OAP (Korte & Vygen 2000). Orlin & Lee (1993) propose an algorithm that uses heuristics to accelerate the search for shortest paths; they suggest (but do not prove) that this algorithm can be expected to run in $O(n)$ time on sparse OAPs (i.e., those problems in which the corresponding bipartite graph is far from fully connected).

3.6 Combinatorial optimization

The field of combinatorial optimization provides a set-theoretic framework, based on *subset systems* (called *independence systems* by Korte & Vygen (2000)), for describing optimization problems such as the OAP (Ahuja et al. 1993):

Definition 3.7 (Subset System) : A *subset system* (E, F) is a finite set of objects E and a nonempty collection F of subsets, called *independent sets*, of E that satisfies the property that if $X \in F$ and $Y \subseteq X$ then $Y \in F$.

That is, any subset of an independent set is also an independent set. We can define a general maximization problem in the following way:

Definition 3.8 (Maximization Problem over a Subset System) : Given a a subset system (E, F) and a utility function $u : E \to \mathbb{R}_+$, find an $X \in F$ that maximizes the total utility:

$$u(X) = \sum_{e \in X} u(e) \tag{3.7}$$

The elements of F are usually not given directly, or at least are inconvenient to represent explicitly. Instead, it is assumed that an *oracle* is available that, given a candidate set X, can decide whether $X \in F$. That is, the job of such an oracle, given a proposed solution, is to verify the feasibility of that solution. For many problems, this verification is computationally trivial when compared to the complexity of the optimization problem.

Given a maximization problem over a subset system, we can define algorithms that attempt to solve it. Of particular interest is the canonical *Greedy algorithm* (Ahuja et al. 1993):

Algorithm 3.3 (The Greedy algorithm) :

1. Reorder the elements of $E = \{e_1, e_2, \ldots, e_n\}$ such that $u(e_1) \geq u(e_2) \geq \ldots \geq u(e_n)$.
2. Set $X := \emptyset$.
3. For $j = 1$ to n:
   if $X \cup \{e_j\} \in F$ then $X = X \cup \{e_j\}$

This algorithm is an abstraction of the familiar and intuitive greedy algorithm for solving a problem: repeatedly take the best valid option. While the Greedy algorithm performs well on some optimization

problems, it can perform quite poorly on others. In particular, it performs well on certain subset systems that can be further classified as *matroids*:

Definition 3.9 (Matroid) : A subset system (E, F) is a *matroid* if, for each $X, Y \in F$ with $|X| > |Y|$, there exists an $x \in X \setminus Y$ such that $Y \cup \{x\} \in F$.

That is, given two independent sets X and Y with X larger than Y, we can always "grow" Y by adding to it some element from X. With respect to the current discussion, an equivalent definition of a matroid, originally due to Rado (1957) and Edmonds (1971), is that a subset system (E, F) is a matroid if and only if the Greedy algorithm finds an optimal solution (Korte & Vygen 2000). In the parlance of algorithmic analysis, matroids satisfy the *greedy-choice property*, which is a prerequisite for a greedy algorithm to produce an optimal solution (Cormen, Leiserson & Rivest 1997). Matroids are of particular interest precisely because their associated optimization problems are amenable to greedy solution.

3.6.1 MRTA as a combinatorial optimization problem

The MRTA problem can be stated as combinatorial optimization problem in the following way. Given is a subset system (E, F), where E is the set of all possible robot-tasks pairs and F is the set of all possible assignments. Also given is a function $u(e)$ that returns the utility of each robot-task pair $e \in E$. The objective is to find an assignment $X \in F$ such that the utility sum (3.7) is maximized.

Unfortunately, this optimization problem is *not* a matroid and is *not* guaranteed to be optimally solved by the Greedy algorithm, as can be seen from the following example. Consider the task allocation represented by the following matrix, giving the utilities (already weighted by task priority) of robots A and B for tasks x and y:

	x	y
A	100	99
B	99	1

(3.8)

If we execute the Greedy algorithm on this problem, task x will be assigned to robot A, leaving task y for robot B. The resultant benefit of 101 is little more than half the optimal benefit of 198.

In fact, MRTA is a weighted intersection of two matroids, the result of which is not a matroid. The maximization problem over a weighted matroid intersection can be stated as follows: given matroids (E, F_1) and (E, F_2), and a utility function $u : E \rightarrow \mathbb{R}_+$; find a set $X \in F_1 \cap F_2$ whose weight $u(X)$ is maximum. Such problems can be solved by a polynomial time algorithm, due to Frank (1981), in $O(|E|^4 + |E|^3\theta)$, where θ is the maximum complexity of the two required independence oracles (Korte & Vygen 2000).

While the Greedy algorithm does not *optimally* solve MRTA, it is useful to know how *bad* the greedy solution can be. For such purposes it is common to report a *competitive factor* for the sub-optimal algorithm (Sleator & Tarjan 1985). For a maximization problem, an algorithm is called α*-competitive* if, for any input, it finds a solution whose benefit is never less than $\frac{1}{\alpha}$ of the optimal benefit. For the OAP, it is known that the Greedy algorithm is 2-competitive: in the worst case the Greedy algorithm will produce a solution whose benefit is $\frac{1}{2}$ of the optimal benefit (Avis 1983). I take up the question of greedy performance further in Chapter 5.

3.7 Summary

In this chapter, I have given an overview of some organizational theory that is relevant to the study of MRTA problems. I have shown how a simple case of the MRTA problem can be seen from five disciplines: operations research / linear programming, economics / game theory, scheduling, network flows, and combinatorial optimization.

In the next chapter, I describe an auction-based MRTA architecture, called MURDOCH, and present validation experiments with physical robots. I use the theory developed in this chapter to explain how and why MURDOCH functions.

Chapter 4

An economic solution

As explained in Section 3.2.3, there is a natural economic interpretation of some multi-robot task allocation (MRTA) problems. Furthermore, as explained in Section 3.3.2, certain auction algorithms are guaranteed to produce optimal allocations, although they can incur significant communication overhead and take many rounds to converge to equilibrium. In this chapter, I describe MURDOCH,[1] an auction-based MRTA architecture that that I have developed and extensively validated through experimentation with physical robots. MURDOCH is a variant of the well-known Contract Net Protocol, or CNP (Smith 1980, Davis & Smith 1983). As such, this architecture provides, in a distributed manner, a tradeoff between communication overhead and solution quality, producing greedy solutions at a low cost. Further analysis of MURDOCH, including algorithmic characteristics, is provided in the next chapter.

The MRTA domain for which MURDOCH is designed can be characterized with the following assumptions:

- The system is composed of physically embodied robots.
- The robots are heterogeneous, possessing different skills.
- The robots can communicate, but messages may be lost.
- The robots are honest and cooperative.

[1] So named because it controls the flow of information, as does Rupert Murdoch.

- The robots (or parts of them) can fail at any time.
- A robot may not be aware of its own (partial) failure.
- The robots are multi-purpose, not task-specific.
- No model is available to describe the sequence in which tasks are generated.
- If a robot is to oversee a task, it can determine its own progress and completion of the task.

Together, these assumptions define a realistic environment in which task allocation can be explored. In fact, all but the last three assumptions arise directly from the unavoidable characteristics and constraints of working with physical robots. The three self-imposed constraints represent my view of what will be required of multi-robot systems for them to be generally useful.

In a model-free environment, with no expectation of what future task allocation requirements might be, the goal is to allocate each incoming task as efficiently as possible based on what the system is trying to optimize. In approaching task allocation, I strive to minimize three aspects of the system:

- Resource usage;
- Task completion time;
- Communication overhead.

These criteria are not orthogonal; rather they interact strongly. For instance, a task allocation that minimizes resource usage might take longer to complete than another assignment which minimizes completion time. Likewise, having found an inadequate task allocation, one might choose to expend additional bandwidth in search of a better allocation. The details of these trade-offs and the way in which they are combined into a combined utility measure (see Section 3.1.1) are application-specific, but the overall optimization goal remains the same.

The MRTA problem that I have just described differs somewhat from the simpler problem defined in Chapter 3. The primary difference is that in the current problem, the tasks arrive over time, rather than all being available at one instant (this distinction is developed further in Section 5.2).

To my knowledge, MURDOCH is the first auction-based approach to a MRTA problem to be validated with teams of physical robots.[2] Thus in this chapter I answer the challenge posed by Parker (1998):

> "[Negotiation-based] solutions have not been adequately demonstrated in *situated* agent (i.e., robotic) teams, which have to live in, and react to, dynamic and uncertain environments amidst noisy sensors and effectors, frequent agent failures, and a limited bandwidth, noisy communication mechanism."

4.1 Methodology

In this section I motivate the communication model, task representation, and auction model that I employ to solve the task allocation problem given in the previous section. The implementation details of the approach are given in the next section.

4.1.1 Anonymous communication

Traditionally, communication in computer networks uses point-to-point unicast message delivery. That is, when Computer A wants to send a message to Computer B, A addresses the message *by name* to B and hands it off to the network. This communication model is problematic in robot systems for two reasons.

First, compared to broadcast messaging, unicast is extremely inefficient for delivery of messages to multiple recipients. Using broadcast, a message can be sent once and it will be received by everyone, yielding a potentially large savings in bandwidth, which is an important consideration for multi-robot systems. Second, the systems under study are fluid. Robots can move in and out of communication range, the communication system itself can drop messages, and the robots can experience many different failures. In order to tolerate these dynamics, the communication layer should never address robots by name. Instead robots should communicate *anonymously* through broadcast means.

[2] The earliest example that I have found of an economically-inspired solution to a MRTA problem is Botelho & Alami's (1999) M+ system, which was demonstrated in simulation. Dias & Stentz (2000) also demonstrated the use of auctions in simulation, and have since validated their approach with physical robots (Zlot, Stentz, Dias & Thayer 2002, Dias & Stentz 2002).

4.1.2 Hierarchical task structure

To effectively describe the tasks that are to be allocated, a representation must be chosen that is both powerful enough to handle a wide variety of tasks and simple enough for a human to use. For this purpose, I have developed a flexible hierarchical task structure. A task is simply a tree which can contain other tasks. That is, the root node of one task tree can be an intermediate node in another task tree. In these trees, a parent-child relationship means that the parent task is responsible for allocating (and subsequently monitoring) the child task. Since in this work I am investigating task allocation, not task decomposition, I delegate to the designer the work of defining the task trees. Alternatively, an off-line planner, such as that described by Erol, Hendler & Nau (1994), endowed with knowledge of the preconditions, postconditions, and interdependences of the tasks involved could be employed. Note that the planner would not be deriving specific motion sequences, but rather overall task structure; the robots themselves generate *in situ* trajectories.

4.1.3 Auctions

Auctions, in one form or another, have been used in societies throughout history to allocate scarce resources among individuals and groups. The auction, as used by humans, has proven a powerful tool for achieving efficient resource allocations, especially in large-scale environments in which the acquisition of consistent global state information is difficult or impossible. As a consequence, auctions have garnered a great deal of attention from economists, and a rich body of theory exists regarding their properties, especially their ability to deal with uncertainty. For an overview of the common auction types, consult McAfee & McMillan (1987). Auctions have also been studied theoretically by researchers in many fields, including game theory, operations research, and multi-agent systems. Many algorithms for holding and deciding auctions have been proposed and analyzed, both from the perspective of resource allocation and task allocation. As a result some auction systems are now known to produce optimal solutions to resource allocation problems.

I decided upon auctions because they are particularly well-suited to the distributed robotic domain: auctions are scalable, allow for significant compartmentalization of data and control, and are efficient in computation and communication. Furthermore, as discussed in Section 3.3.2, certain auction algorithms are guaranteed to produce optimal allocations. However, the auction system that I developed, described in Section 4.2.2, is *not* guaranteed to produce optimal allocations. Prizing communication efficiency over optimality, I chose to use simple, first-price auctions to allocate tasks in a greedy manner. Though it can produce sub-optimal solutions, this approach has a known worst-case performance bound (see Section 5.2.2), and tends to work very well in practice, a claim borne out by the experimental results given later in this chapter.

It may seem counter-intuitive to employ a competitive mechanism (i.e., the auction) in order to elicit cooperative behavior. In this case the difference between competition and cooperation reduces to a question of motivation: are the robots selfish? Philosophical arguments aside, I believe that the answer is no. Having programmed them myself, I can state unequivocally that the robots' most basic motivation is to do useful work (i.e., accomplish tasks). They are neither greedy nor dishonest, and are only self-interested in so far as their limited resources impact their ability to do work. For example, as described in Section 4.3.2.1, when its battery level is sufficiently low a robot will refuse to take on future tasks and will seek a charging station. Except in such special circumstances, the robots cooperate with each other to the greatest possible extent.

4.2 Murdoch

In this section, I explain in detail the protocols and algorithms that underlie MURDOCH, which was originally described by Gerkey (1998), Gerkey & Matarić (2000), Gerkey & Matarić (2001), and Gerkey & Matarić (2002*c*). In addition to the work described in this chapter, MURDOCH has been used in a broader study of large-scale human-system interaction (Tews, Matarić, Sukhatme & Gerkey 2002, Tews, Matarić & Sukhatme 2003).

4.2.1 Publish/subscribe messaging

In implementing distributed control systems for teams of robots, researchers typically resort to *ad hoc* communication strategies. These specialized strategies are often implemented as hand-crafted, task-specific communication graphs. For example, in the work of Parker (1998) and Werger & Matarić (2000), communication channels are explicitly created among individual behaviors on the different robots. While this model is well-suited to the design of special-purpose, single-task systems, it has not been demonstrated for the general case of controlling a large groups, in which the members dynamically form teams to accomplish different tasks as they are presented to the system. As an alternative to such special-purpose systems, I propose a principled communication model based on *publish/subscribe messaging*, an instance of anonymous communication.

4.2.1.1 Background

The unifying concept of publish/subscribe systems is that *messages are addressed by content rather than by destination.* This idea, often called *subject-based addressing*, is used to divide the network into a loosely-coupled association of anonymous data producers and data consumers. A data producer simply tags a message with a subject describing its content, and "publishes" it onto the network; any data consumers who have registered interest in that subject by "subscribing" will automatically receive the message. Data producers need not have any knowledge of which consumers, if any, are receiving their messages, and vice versa. This kind of communication represents a fundamental departure from the traditional unicast model. I have tailored this idea slightly so that when a message is published, it is addressed to a set of subjects, rather than just one. A data consumer will receive a message if the subjects in the message comprise any subset of the consumer's current subscription list. Although I have not optimized the subset matching algorithm in my implementation, others have investigated the topic (Aguilera, Strom, Sturman, Astley & Chandra 1999, Pereira, Fabret, Llirbat & Shasha 2000).

4.2.1.2 Subject namespace

The first step in creating a publish/subscribe system is designating the semantics of the subject namespace. Analogous to deciding the layout of a database, the interpretation of subjects will heavily influence the rest of the system. In MURDOCH, since I am allocating tasks among a group of potentially heterogeneous robots, I use subjects to represent their "resources."[3] Resources can be physical devices (e.g., `camera`, `gripper`, `sonar`), higher-level capabilities (e.g., `mobile`, `door-opener`) or abstracted notions of current state (e.g., `idle`, `have-puck`, `currently-pushing-box`). Thus, given a task that involves going to and observing some location, it is possible to reach the capable robots (who are otherwise unengaged) by addressing a message to the set {`mobile camera idle`}. Since messages are addressed to subjects and subjects represent resources, all inter-robot communication will necessarily be resource-centric (or possibly task-centric). The robots never interact with each other by name and in fact have no explicit knowledge of each others' existence; rather they only communicate about tasks and all messages are addressed in terms of the resources required to perform those tasks.

4.2.2 Auction protocol

At the heart of MURDOCH lies a simple distributed negotiation protocol that allocates tasks via a sequence of first-price one-round auctions (McAfee & McMillan 1987). This auction-based negotiation system is a variant of the well-known Contract Net Protocol (CNP) (Smith 1980, Davis & Smith 1983). The process is triggered by the introduction of a task to the system. The task could be introduced in many ways, including a human user, a `cron`-style alarm, or an already ongoing task. In every case, the auction proceeds in five steps:

Algorithm 4.1 (MURDOCH Auction Protocol) :

1. **Task announcement** - An agent (working on behalf of a user, alarm, or task) acts as "auctioneer" by publishing an `announcement` message for the task. The message contains details about the task,

[3] I could have used *tasks* instead of *resources* in constructing the subject namespace. Since the robots must agree on a common task vocabulary, few changes would be required in MURDOCH. Further, the vocabulary of tasks is smaller than the vocabulary of resources, and adhering to a task-level abstraction would easily accommodate more heterogeneous systems in which different robots execute the same task using different resources.

such as its name, length, and a new subject on which to negotiate it. The message is addressed to the set of subjects which represent the resources required to execute the task; thus only those robots currently capable of performing the task will receive the message.

2. **Metric evaluation** - Since there may be more than one capable robot available for a given task, method is required to decide among them. This decision process is the very basis of achieving effective task allocation; the goal is to allocate each task to the most fit robot. Thus the `announcement` also includes *metric(s)* with which each candidate robot can determine its fitness (see Section 4.2.2.1 for details on metrics).

3. **Bid submission** - After evaluating the appropriate metric(s), each candidate robot publishes its resulting task-specific fitness "score" as a `bid` message.

4. **Close of auction** - After sufficient time has passed[4], the auctioneer processes the `bid`s, determines the winner, and notifies the bidders with a `close` message. The winner is awarded a time-limited contract to execute the task and the losers return to listening for new tasks.

5. **Progress monitoring / contract renewal** - While the winner executes the task, the auctioneer monitors task progress. Assuming sufficient progress is being made, the auctioneer periodically sends a `renewal` message to the winner, which responds with an `acknowledgment` message, until the task is completed.

The fact that the winner's contract is time-limited is an essential part of MURDOCH's fault-tolerant capabilities. Recalling the argument for "soft state" (Clark 1988), time-limited contracts provide a built-in timeout that can trigger fault detection and recovery. For example, if the auctioneer fails to receive an acknowledgment after sending a renewal, it can assume that the robot previously executing the task has failed and so it can reassign the task. Similarly, the task can be reassigned if the auctioneer finds that insufficient progress has been made. In either case, the previous winner will terminate task execution when its contract has expired without renewal.

Since each task will always be claimed by the most capable robot available at the time, MURDOCH acts as an instantaneous greedy task scheduler. Thus it suffers from the well-known problems of greedy algorithms; they are manifested in this domain as situations in which, although sufficient resources exist to achieve a given set of tasks, the order in which the tasks are presented causes resources to be exploited in a non-optimal manner such that not all the tasks are actually achieved. Centralized broker and matchmaker systems sometimes avoid this pitfall by analyzing concurrent tasks before allocating them. That kind of planning, however, will not help in the class of domains where individual tasks are input stochastically

[4] In the current implementation, the timeout for an auction is 1.5 seconds.

from some outside source, such as a human user. The performance of greedy algorithms in general, and MURDOCH in particular, is discussed in Chapter 5.

4.2.2.1 Metrics

In MURDOCH, utilities are computed by *metrics*. Metrics can take many forms, with the restriction that each one, when evaluated in the context of a specific robot, should return a scalar value representing that robot's fitness for the task. A metric is usually defined as some function of the robot's current state, although in general, a metric could perform any arbitrary computation, including inter-robot communication. As an example, if the task under consideration is to go to a certain location and pick up an object, one possible metric is to compute the Cartesian distance from the robot's current position to the goal position, with a shorter distance scoring better[5]. Multiple metrics can be defined for a single task, with the final score being some combination of the individual scores; I have experimented with combining metrics through both simple sums and weighted sums that allow for a prioritization of metrics. It is important to note that metrics, being functions of each robot's own sensor data, may not accurately represent the true utility of the robots, possibly resulting in a non-optimal allocation of the task.

4.3 Experimental validation

I have validated MURDOCH through a series of experiments designed to test the task allocation and fault-tolerant capabilities of the system. In this section I detail those experiments, then present and discuss the results. These experiments were originally described by Gerkey & Matarić (2002*b*).

[5]By using such metrics to compare different robots, it is assumed that the scalar utilities produced by the metrics are indeed comparable. For example, if distance is a metric then, although they may use different methods for measuring distance (e.g., odometry versus IMU), the robots must all report distance in the same units.

4.3.1 Platform

I used as my experimental test bed a team of seven ActivMedia Pioneer 2-DX mobile robots (see Section 6.2). Low-level robotic device control was handled by Player (see Section 6.1.2). Inter-robot communication in this case is provided by way of wireless Ethernet (802.11), with an effective shared bandwidth of approximately 1.9Mbps. The topology is such that every machine on the network can communicate freely with every other machine.

4.3.2 Experimental design

I have claimed that MURDOCH's negotiation-style task allocation is applicable to a variety of problems and that it is robust to environmental dynamics, such as robot failures. To substantiate the first claim, I have evaluated MURDOCH in two very different domains: a long-term scenario consisting of many loosely-coupled single-robot tasks, and a cooperative box-pushing task requiring tight coordination among the robots. To demonstrate fault-tolerance, I explored several robot failure modes in the box-pushing task. Video footage of these experiments is available at:

`http://robotics.usc.edu/~agents/projects/auctions.html`.

4.3.2.1 Loosely-coupled task allocation

In the pursuit of long-term autonomy, I designed a scenario (see Figure 4.1) in which a large heterogeneous group was required to self-organize to accomplish a randomly generated sequence of tasks.

Task description The team comprised the following machines:

- 3 robots with cameras
- 3 robots with cameras and laser range-finders
- 1 robot with a camera and a tactile bumper array
- 1 desktop computer with an overhead camera

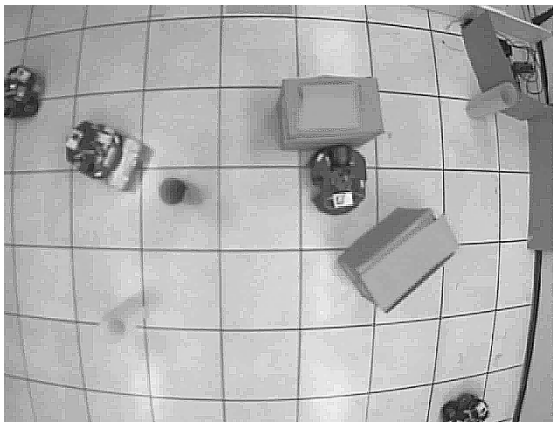

Figure 4.1: *Overhead snapshot of the long-term loosely-coupled scenario. The robot in the center of the image is performing the* **cleanup** *task while the robot to its immediate left is performing the* **object-tracking** *task. In the upper right corner is the charging station.*

The team was exercised with four different single-robot tasks:

- **object-tracking**
- **sentry-duty**
- **cleanup**
- **monitor-object**

The first, **object-tracking**, requires the `camera` and `mobile` resources. The task is to find and track from a safe distance a certain colored object. The second, **sentry-duty**, requires `camera`, `laser`, and `mobile`. The task is to go to a target location (marked by a colored object), then turn about and remain still, watching for any motion with the laser and setting off an intruder alarm if motion is detected. The third task, **cleanup**, requires `camera`, `bumpers`, and `mobile`. The task is to find each small box of a certain color and use tactile bumpers to push the box to the edge of the room, thereby cleaning the room. The final task, **monitor-object**, requires only `overhead-camera`. The task is to monitor the positions of various colored objects, such as boxes and robots, from an overhead view, and log the information for later review.

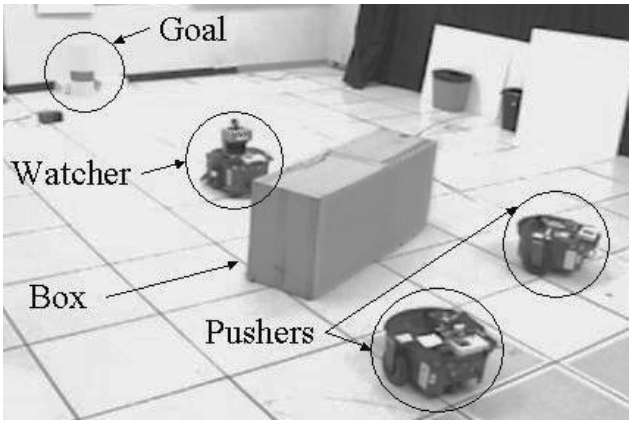

Figure 4.2: *The experimental box-pushing setup. The task is for the* `pushers` *to move the box to the goal with the help of the* `watcher`. *(Image 1 of 3 taken from an experimental trial; see Figures 4.4 & 4.5)*

Each robot also runs a battery-monitoring behavior that checks the current charge whenever the robot is idle, between tasks. At that time, if the battery is low enough, the robot will unsubscribe from all subjects (thereby removing itself from consideration for future tasks) and go to a clearly marked charging station. After charging for some time, the robot is freed to reenter the experiment.

Experiment I ran the team over a long period of time (approximately three hours), periodically injecting random tasks of random lengths into the system. Each new task was auctioned (and subsequently monitored) by an automated tasking agent that was executing on a separate desktop computer. Each machine was controlled by a copy of the same program; this program simply queried its host for the list of currently available devices, then made the proper resource subscriptions. For example, the robots equipped with both cameras and lasers subscribed to {`camera laser mobile`}, while the desktop computer only subscribed to {`overhead-camera`}.

4.3.2.2 Box-pushing

While the previous experimental domain demonstrates MURDOCH's ability to handle heterogeneous multi-robot systems, the tasks themselves require little coordination beyond the initial task assignment. To be generally useful as a tool for control of robot teams, a system must also be capable of allocating and

coordinating tasks that require tightly-coupled cooperation among the robots, and it must do so in a fault-tolerant manner. I chose as an example of this tightly-coupled task domain the long-studied *box-pushing* problem.

Task description The experimental box-pushing setup is shown in Figure 4.2. The task I address is to cooperatively move a box from some initial location to some observable goal location. In solving this problem, I took inspiration from human coordination commonly observed when people move large pieces of furniture. If the people who are pushing or carrying cannot see where they are going, another person stands between the carried object and the goal and periodically directs them. This "watcher" can see both the current position of the object and the goal, and thus can compute an error signal, perhaps in the form of a correction angle, that can be communicated to the "pushers"[6].

I define this problem with a set of constraints. First, both the box and the goal are observable, and there is an obstacle-free path between them that is wide enough for the box and robots to pass (i.e., I do not consider negotiating obstacles in a coordinated fashion, only as part of individual low-level control). Second, the box is large compared to the size of the robots[7]. Third, the robots can only move the box by pushing through frictional contact. Finally, the pushing robots cannot directly perceive the goal due to the size of the box.

I construct a natural task hierarchy in which high-level watching tasks are auctioned to watcher robots, who then direct the actions of pusher robots by auctioning appropriate low-level pushing tasks. In these experiments, as shown in Figure 4.2, two robots act as `pushers`, and a third performs the `watcher` role. The `pushers` can see the box, and the `watcher` can see the goal. In addition, the `watcher`, while servoing on the goal, can accurately perceive (using a laser range-finder) the angular error of the box's orientation with respect to the path from the box to the goal. The aim, then, is to rotate the box until that angular error is zero (i.e., the box is orthogonal to the path to the goal), while simultaneously translating

[6] Actually, the watcher likely will not communicate the raw angle, but rather some higher-level command, such as "push more on the right"; the robots do the same (see Section 4.1.1).

[7] Specifically, the intended contact surfaces should allow at least two robots to be pushing simultaneously.

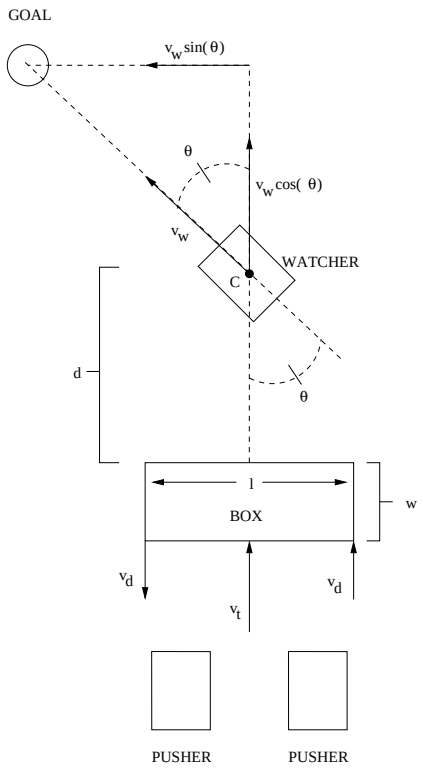

Figure 4.3: *The model used to derive the pushing velocities for moving the box along the desired trajectory.*

it toward the goal. I call this approach to box-pushing `pusher-watcher`, which I describe in the next section.

Algorithm In order to achieve the goal described in the previous section, I must first determine the `pushers`' velocities. To this end, as shown in Figure 4.3, I model the box and `watcher` together as a rigid body; I attach an imaginary link between the center of rotation of the watcher, C, and the center of the side of the box, orthogonal to the box. The box and link together rotate freely about C. The `pushers` are assumed to be point velocities that act on this rigid body (for simplicity, I disregard mass and acceleration). At each point in time, the `watcher` is rotated away from the normal to the box by an angle θ, and is moving toward the goal with a velocity v_w. The distance between the `watcher` and box, d, will change with the difference of the box's translational velocity, v_t, and the projection of v_w along the normal to the box:

$$\dot{d} = v_w \cos\theta - v_t$$

In order to maintain the rigid body constraint, the distance d must be constant; that is, $\dot{d} = 0$. Thus:

$$v_t = v_w \cos\theta$$

Now, assuming that the box rotates about its center, it can be rotated so as to minimize the angular disparity θ by setting the projection of the box's rotational velocity equal to that of the `watcher`:

$$(d + w)\dot{\theta} = v_w \sin\theta$$

The rotational velocity $\dot{\theta}$ can be distributed differentially to the two pushing points:

$$v_d = \frac{l}{2}\frac{v_w}{(d + w)} \sin\theta$$

Now, compute the two `pusher` velocities:

$$v_p = v_t \pm v_d$$

The velocities given by v_p, if applied continuously at either end of the box, will ensure that a constant distance is maintained between the box and the `watcher` and that the angular error θ tends toward zero. The actual trajectory will be a curve that approaches the path to the goal with the box's orientation tending toward orthogonality with respect to that path.

Implementation details As is the case with any choice of robots, the decision to implement `pusher-watcher` on this particular platform added extra constraints to the problem. I account for these constraints by implementing not the exact algorithm given in Section 4.3.2.2, but rather a suitable approximation. For example, although the algorithm can result in negative pushing velocities, the robots can only push, not pull, the box. Thus I bound the pushing velocities below by zero, which has the effect of increasing the

minimum turning radius of the box (see Section 4.4). Further, it turns out that some pairs of pushing velocities, especially those with large differences, are extremely difficult to execute robustly in practice. This difficulty is due mostly to the non-holonomicity of the robots, which cannot move laterally; if a robot slips off the end of the box, it cannot re-acquire the box without performing a sort of "parallel-parking" maneuver, which is challenging in a dynamic environment. For this reason, I discretize the box's orientation space into bins (currently five) for which the resultant pushing velocities are practical.

The box-pushing team comprises the following machines:

- 1 robot with a camera and laser range-finder
- 2 robots with cameras

A user poses a `relocate-box` task to MURDOCH; this task is hierarchical and is composed of a `watch-box` task that has two children `push-box` tasks. The `watch-box` task is auctioned first, with the resource list {`mobile laser camera`}. The one robot with those resources claims the task, becoming the `watcher`. It begins executing the `watch-box` task[8], which consists of a simple algorithm:

Algorithm 4.2 (`watcher` Algorithm) :

1. Servo toward the goal (with the camera).
2. Determine the angular error of the box (with the laser range-finder) and calculate appropriate left and right pushing velocities.
3. If new velocities are approximately equal to the previously assigned velocities, then renew existing contracts and go to Step 1.
4. Sequentially auction new left and right `push-box` tasks with resource lists {`mobile camera`}, in order of decreasing velocity.
5. Go to Step 1.

The metric for the `push-box` tasks is based on the color camera input and is a measure of how well the robot is positioned for pushing on a particular end of the box. For example, when the task is to push on

[8] As with all tasks, the `watch-box` task is time-limited and is monitored, in this case by an agent on behalf of the user. If the `watcher` fails in some way, then the `watch-box` task will be automatically reallocated if a new `watcher` can be found; otherwise the user is notified of the problem.

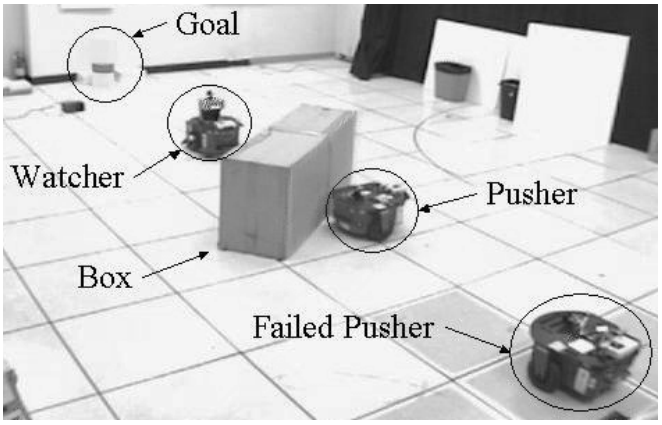

Figure 4.4: *Fault-tolerance in action: after a single robot failure was induced, the remaining robot is left to push on its own. (Image 2 of 3 taken from an experimental trial; see Figures 4.2 & 4.5)*

the right end, the metric reflects whether the box is offset to the left in the robot's camera image. Because of the dynamic nature of the system, each `push-box` task is 3 seconds in length.

In a typical run of `pusher-watcher`, the `watcher` initially auctions left and right `push-box` tasks with proper velocities, and lets them push until the box's orientation changes sufficiently to warrant different pushing velocities and thus new tasks. At that point, the current contracts are allowed to expire, and new ones are formed. This reactivity to world conditions is the feature that enables MURDOCH to dynamically reassign tasks in the face of robot failure. For example, when only a single robot is available, the `watcher` will actually try to allocate two pushing tasks as usual, but only one (the one that has the higher velocity and is auctioned first) will be claimed. That single contract is renewed and the robot pushes on one end of the box until the orientation changes enough that the it is more important to push the other end, at which point the robot will simply switch sides. When another robot is introduced, it will claim the next available pushing task and the two robots will cooperate at pushing the box.

Experiments In order to evaluate MURDOCH in the box-pushing domain, I performed five sets of experiments on the group of Pioneer robots, as described below. During the experiments, I measured two quantities: success/failure and elapsed time. Success is defined as the situation in which the `watcher`

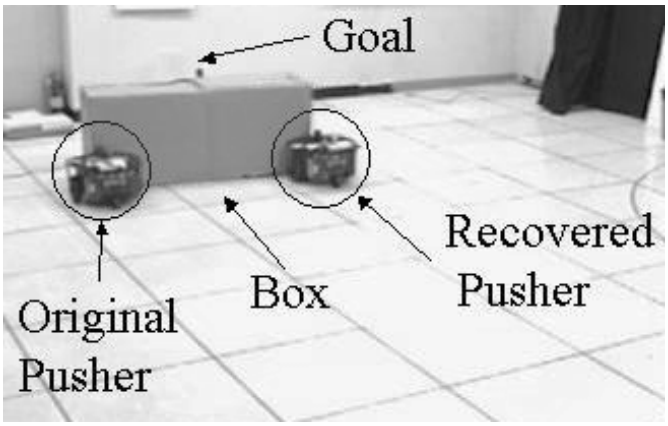

Figure 4.5: *After the failed robot was "revived", it was reintegrated into the team, and all completed the task together. (Image 3 of 3 taken from an experimental trial; see Figures 4.2 & 4.4)*

declares that the task is terminated, and the center of the box is positioned within 0.5 meters of the target location (see Figure 4.5); no target orientation is specified for the box. Conversely, a trial is a failure if either the `watcher` declares termination when the box is not close enough to the goal, or the box is rotated so far that the `watcher` can no longer perceive it using its laser range-finder (this threshold is approximately $70°$). For ground-truth I used an external metrology system consisting of multiple laser range-finders and beacons to track the positions of the box and robots throughout the experiments (plots are based on that data).

As a control, Experiment Set 1 involved the simplest scenario. Two `pusher` robots had to move the box along a straight-line path for approximately 3 meters (90% of the length of the lab), and no faults were introduced. In Experiment Set 2, I tested the system's tolerance to an individual robot failure. The setup is the same as in Experiment Set 1, with two `pushers`, but, after they pushed the box approximately 1.2 meters, I simulated a robot failure by seizing one `pusher` and shutting it off. As a result, the remaining `pusher` was left to push the box by itself, alternating sides under the direction of the `watcher` (see Figure 4.4). In Experiment Set 3, I tested MURDOCH's progress monitoring capability by introducing a partial robot fault. As with Experiment Set 2, two `pushers` began the task together; I then simulated a

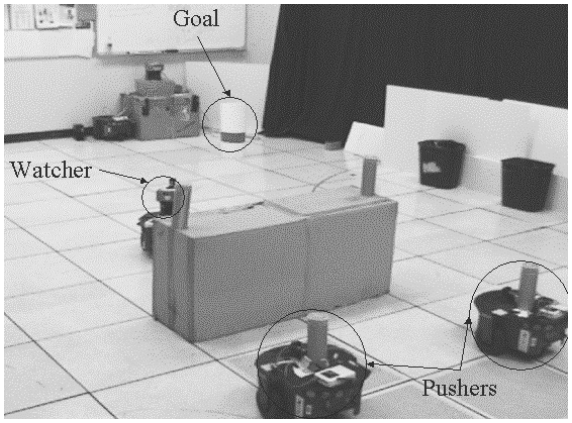

Figure 4.6: *The setup for Experiment Set 5. The goal is approximately* 35° *off the center line, requiring the robots to push the box along a curved trajectory.*

robot becoming stuck (e.g., in sand) by disabling its motor power. This failed robot was still in communication with the team and claimed to be fit for pushing tasks, but was immobile. In order to complete the task, the `watcher` had to detect the lack of progress on the part of the stuck robot and exclude it from future task offerings. In Experiment Set 4, I tested the system's dynamic response by inducing both failure and recovery. I first let them both push approximately 0.6 meters, then seized one `pusher` to simulate complete failure. After the remaining `pusher` had single-handedly pushed the box another 1.2 meters, I reintroduced the failed `pusher`, at which point the two had to finish the task together (see Figure 4.5).

While Experiment Sets 1–4 showcase the ability of the `pusher-watcher` system to cooperatively move a box along a straight line, it is important to be able to follow more general paths. In Experiment Set 5, I tested the ability of the system to execute curved trajectories by placing the goal marker approximately 35° off the center line and 2.75 meters away (see Figure 4.6). In order to follow this path, the robots had to behave in a tightly coordinated fashion, making a series of rotational and translational adjustments.

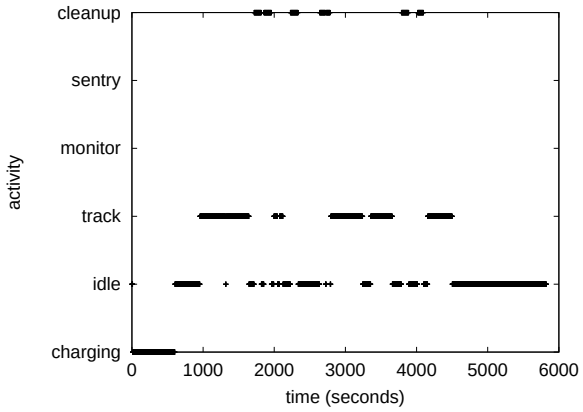

Figure 4.7: *An example plot of task execution during the long-term experiment. Shown here is the activity, over the first half of the experiment, of the robot equipped with a camera, laser range-finder, and bumpers.*

4.4 Results

In this section, I present and discuss the results from experiments performed with MURDOCH in the two task domains previously described.

4.4.1 Loosely-coupled task allocation

In the loosely-coupled scenario, I observed the following system behavior. Over the course of three hours, the group (seven robots and one desktop computer) successfully executed 49 tasks. They returned to charge their batteries twelve times. Some of the randomly generated tasks were unachievable due to a lack of resources, because all the capable robots were either charging or otherwise engaged. In these situations, an error was returned, suggesting that the task be reintroduced later. I have not automated the task reintroduction process, but a variety of solutions are immediately apparent (e.g., the exponential back-off algorithm used by Ethernet to resolve bus contention).

The same control program executed on each robot for the length of the experiment, with robots periodically idle (only executing passive collision avoidance), executing some task, or charging. Shown in Figure 4.7 is a plot of the task execution over time of one of the robots. The plot is from the robot equipped

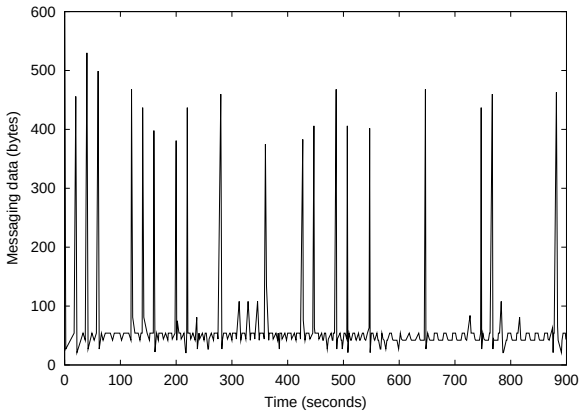

Figure 4.8: *The communication overhead incurred by* MURDOCH. *Shown here is the total amount of data transferred among the robots during the first 15 minutes of the long-term experiment. Each spike corresponds to an auction, which causes a brief flurry of activity.*

with a camera and tactile bumper array. The robot begins at the charging station, then is periodically idle and engaged in various tasks. Given its resources, this robot was only capable of the **track-object** and **cleanup** tasks, and so those are the tasks that it accepted and executed.

Whenever sufficient resources were available for a task, the task was allocated and executed. Given the assumption of truly random task injection, with no look-ahead, the robot assigned to each task was the best available robot for that task at the time of assignment. Finally, MURDOCH is efficient with regard to communication. Over the entire experiment, 3791 messages were sent, with an average size of 36.06 bytes (see Figure 4.8). The average total bandwidth used by MURDOCH was 0.66Kbps, with a maximum burst of 4.74Kbps. This usage, which is the entire communication overhead incurred by MURDOCH, is a tiny fraction of the bandwidth available on modern radio networks (1 – 11Mbps), suggesting that the system will scale well with larger numbers of robots and/or tasks.

4.4.2 Box-pushing

I performed 10 trials each from Experiments Sets 1–4. In the 40 trials, there were a total of four failures, one occurring in each set. Three failures were due to over-rotation of the box, and the fourth was due

Set	Description	μ	σ
1	No failure (straight path)	31.22	0.44
2	Pusher failure	132.75	26.94
3	Partial pusher failure	260.89	37.79
4	Pusher failure & recovery	116.44	37.72

Table 4.1: *Mean (μ) and standard deviation (σ) of the elapsed time (in seconds) for the successful pushing trials in each of the four box-pushing Experiment Sets.*

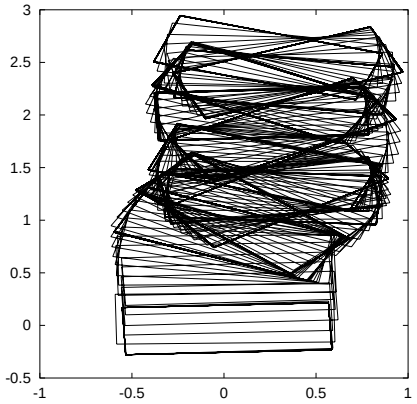

Figure 4.9: *Path of the box in an example trial from Experiment Set 3 (units are meters). The right* `pusher` *fails, leaving the other* `pusher` *to push either end of the box in turn.*

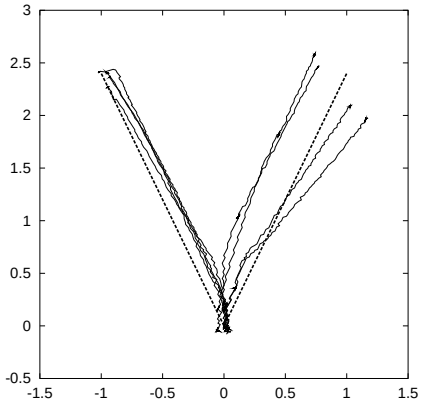

Figure 4.10: *Path of the box in trials from Experiment Set 5 (units are meters). Shown here is the trajectory of the center of the box for four successful trials in either direction. For comparison, the dashed lines represent the "ideal" paths.*

to premature termination on the part of the `watcher`, presumably because of sensor noise. With 36 successes in 40 trials, the two-sided 95% binomial confidence interval for the overall success rate of the system is: $p \in (0.76, 0.97)$. I also analyzed the time elapsed during the successful trials, as a measure of relative efficiency among the different experiments. The results are shown in Table 4.1. In Experiment Set 1, with no failure, the two `pushers` almost always executed the task in one continuous movement, yielding extremely similar (and short) completion times across trials. When one robot failed the completion time rose considerably because the remaining robot was left to push either side of the box in turn, which is rather inefficient compared to the two-robot cooperative case.

Experiment Set 3 (see Figure 4.9) took longer still because, before the healthy `pusher` could take over the task on its own, the `watcher` had to recognize that the other robot was "stuck" and declare it unfit to participate. Exactly how long the `watcher` should wait is a system parameter that represents a trade-off between good dynamic response and the chance of falsely declaring a fault. The average completion time for Experiment Set 4 was less than for the other two failure modes, which suggests that the overhead required for coordination is outweighed by the improvement in performance from the robot's reintegration to the team. However, the large standard deviations preclude stronger comparisons. The magnitude of the standard deviations in the failure cases is explained intuitively by the complexity of the system; as the situation becomes more complicated, the exact behavior of the robots is less repeatable, due to the numerous interacting dynamic processes (e.g., variable torque output from motors, friction between the box and the floor, etc.).

Note that, in Experiment Set 4, after the failed robot recovered and was reintroduced, the `pushers` switched sides when appropriate, which was in half the trials. The appropriateness of switching was determined by the configuration of the box and the remaining `pusher` at the time of reintroduction, and this configuration was in turn a result of the complex system dynamics mentioned above. However, the fact that the pushers automatically switched sides at the right times, with no detriment to performance, demonstrates that the task allocation system performs as specified.

In addition to verifying the *validity* of the task assignments made by MURDOCH, I also want to know their *quality*. I evaluated the `pusher-watcher` system in this respect by comparing the measured trajectories of the box to an "ideal" trajectory. For this comparison, the ideal trajectory is defined as the one which causes the center of the box to follow the shortest path to the goal. This path is simply the straight line from the initial location of the box to the goal. However, this trajectory is unrealistic because it would require that the box be rotated in place at the start, a feat of which the physical robots are not capable. Shown in Figure 4.10 are measured and ideal box trajectories for trials in Experiment Set 5. It can be seen from this data that the `pusher-watcher` system generates efficient paths that are close to the (unachievable) ideal.

4.5 Related Work

The multi-robot coordination problems addressed by MURDOCH touches on many established research areas. In this section, I discuss the relevant work in task allocation (both physically embodied and otherwise) and box-pushing.

4.5.1 Unembodied task allocation

Beginning with the seminal Contract Net Protocol (CNP) (Smith 1980, Davis & Smith 1983), automated coordination schemes for multi-agent systems have been applied to a wide variety of problem domains. In fact, the CNP itself has been widely studied and extended (Sandholm 1993, Ramos 1996, Golfarelli, Maio & Rizzi 1997). A number of other coordination approaches have also been proposed; to date, two of the more mature systems are the Open Agent Architecture (Martin, Cheyer & Moran 1999) and RETSINA (Sycara, Decker, Pannu, Williamson et al. 1996). Both architectures focus on providing a maximally general environment in which a diverse population of agents, such as user interfaces and legacy database management systems, can interact and coordinate. Since we are concerned specifically with task allocation for physical mobile robots, we do not require the overhead (such as ontology specifications) that allows

these systems to be so general. There is a component of task allocation among their agents, although the tasks are purely informational, rather than physical. Both architectures accomplish task allocation through a broker (called *facilitator* and *matchmaker*, respectively) which matches new task requests with agents that have previously advertised relevant capabilities. In contrast, MURDOCH completely distributes the matchmaking process and thus does away with the centralized broker (Gerkey & Matarić 2001).

A more situated (but still not *embodied* (Brooks 1991)) distributed coordination system is the process scheduler *Condor* (Litzkow, Livny & Mutka 1988), which attempts to maximize processor usage in a network by remote execution of background jobs on idle workstations. More recently, the same process scheduling problem has been approached from an agent perspective with *Challenger* (Chavez, Moukas & Maes 1997). The scheduling mechanism in *Challenger* requires agents throughout the network to bid for available jobs in a manner not unlike the auctions used in MURDOCH.

4.5.2 Embodied task allocation

Compared to software agent systems, task allocation for physically embodied robots has received less (but still significant) attention. Perhaps the best-known example is ALLIANCE (Parker 1998), an architecture for coordinating behavior-based robots (Arkin 1998, Matarić 1997) that must cooperate to achieve some task. Every robot knows the entire task to be completed; the robots use *motivational behaviors* to compete for the various task components over time. Each robot tracks both its own and its teammates' fitness and progress, incorporating this performance information into local measures of *impatience* and *acquiescence*. If a robot becomes sufficiently impatient, it may seize a task from a sufficiently acquiescent robot, thereby providing fault-tolerance. ALLIANCE has been demonstrated on multi-robot foraging (Parker 1998), box-pushing (Parker 1994*a*), and target-tracking (CMOMMT) (Parker 1999*a*).

A similar but more minimalist approach to robot task allocation is taken by BLE (Werger & Matarić 2000), a modern distributed derivative of the Subsumption Architecture (Brooks 1986) that has been demonstrated on a weighted version of the CMOMMT target-tracking task. In the BLE framework, robots

have no commitment to their task; the relevant behaviors themselves continuously communicate to decide who is best fit for each job. Since the robots may switch tasks at any moment, the system has a good dynamic response to environmental changes, at the expense of added communication required for the continuous fitness broadcasts.

Importantly, both BLE and ALLIANCE assume behavior-based control of the robots in the team. They implement task allocation through behavior inhibition; each robot has some desire to execute each task, and the robots suppress each others' activity directly at the behavior level. No such assumptions are made in MURDOCH, requiring only that each robot be able to start and stop a task execution module (be it behavior-based, hybrid, or deliberative) on demand.

Previous work has also been done on using auctions for MRTA problems. To my knowledge, the first example is M+ (Botelho & Alami 1999), which is an implementation of the CNP. The M+ system was used to coordinate a simulated multi-robot system, and has since been extended (Botelho & Alami 2000, Botelho & Alami 2001).

Another economically-inspired approach is Stentz & Dias's (1999) market-based MRTA system, which also uses first-price auctions, recalling the CNP. This approach has been demonstrated in the control of simulated robots performing an area coverage task (Dias & Stentz 2000), and in an exploration task with both simulated (Thayer, Digney, Dias, Stentz et al. 2000) and physical (Zlot et al. 2002) robots. The same approach has been used as a task allocator in a more comprehensive multi-robot coordination architectureSimmons02. An interesting addition to this market-based system is the capability to make use of "leaders" to dynamically form teams for multi-robot tasks (Dias & Stentz 2002). I discuss the implications of this mechanism in the context of coalition formation in Section 5.4.

4.5.3 Box-pushing

Box-pushing has long been one of the canonical task domains for mobile robot researchers, and a natural one for cooperative robotics. Several (but not many) systems have been demonstrated; relevant ones are highlighted here.

At one extreme, Kube & Zhang (1992) and Kube & Zhang (1996) describe a swarm-like method for moving a large box with many small, locally controlled robots; the system could fairly be described as *emergent*. In stark contrast is the planner-based master-slave pushing system described by Noreils (1993). A similarly deliberative approach is taken by Donald et al. (1997*a*), who focus on the analysis of various two-robot pushing protocols with regard to information requirements. Some middle ground is found by the two-robot behavior-based approach presented by Matarić, Nilsson & Simsarian (1995), with an emphasis on the robots' learning policies to enable effective cooperation. Parker (1994*a*) develops a fault-tolerant two-robot box-pushing system and gives proof-of-concept demonstrations. More recently, a method for single-robot box-pushing through an obstacle field (in the context of robot soccer) is given by Emery & Balch (2001).

With regard to the pushing control system itself, the work presented in Section 4.3.2.2 is most similar to the `pusher-steerer` protocol of Donald et al. (1997*a*) and, to a lesser extent, the master-slave system of Noreils (1993). However, neither of these architectures made any provision for robot failures. Of the other three multi-robot systems, only Matarić et al. (1995) is goal-directed, and in that case, both pushing robots could directly perceive the goal, somewhat reducing the need for cooperation.

4.6 Summary

In this chapter, I have presented a novel method of dynamic task allocation for multi-robot systems, based on the Contract Net Protocol (Smith 1980, Davis & Smith 1983). To evaluate this approach I have implemented the task allocation system MURDOCH, based on a principled publish/subscribe messaging model. In this model, all inter-robot communication is necessarily anonymous and resource-centric. I tested MURDOCH on physical robots in both a long-term loosely-coupled task domain and a short-term tightly-coupled box-pushing task. I demonstrated that the system is extremely reactive to changes in the environment, including abrupt failures of robots and random introduction of new tasks. The primary contribution of this

work is the empirical demonstration that distributed negotiation mechanisms such as MURDOCH are effective in coordinating physical multi-robot systems. Such systems are, as a rule, complex and difficult to coordinate. MURDOCH simplifies this problem by automating task allocation in a resource-efficient manner. Other, more sophisticated, economic solutions are possible, such as those suggested by Golfarelli et al. (1997), Gerkey & Matarić (2002*a*), and Dias & Stentz (2002).

A remaining consideration is how well a system like MURDOCH can be expected to perform in general on MRTA problems. I take up this issue in the next chapter, finding that MURDOCH is in fact an implementation of the Greedy algorithm (see Section 3.6) for the online assignment problem (see Section 5.2.2), and thus is 3-competitive, which is the best one can do without a model of the task sequence and without reassignment of tasks.

Chapter 5

A taxonomy

In Chapter 3, I gave a definition (3.2) of the multi-robot task allocation (MRTA) problem that is modeled after the optimal assignment problem (OAP): instantaneously assign single-robot tasks to single-task robots so as to maximize expected performance. Of course, MRTA problems that are encountered in practice are not so simple. These "real" MRTA problems are complicated by a variety of factors, including multi-robot tasks, interrelated utilities, and the passage of time. Even the MRTA problem studied in Chapter 4, though similar to the ideal OAP, is complicated by the fact that tasks are randomly introduced over time; I address this particular problem below in Section 5.2.2.

In this chapter, I propose a taxonomy of MRTA problems based on axes laid out in Section 5.1. My goals are two-fold: to show how various MRTA problems can be positioned in the resulting problem space; and to explain how the organizational theory developed in Chapter 3 relates to those problems and to proposed solutions from the robotics literature. In some cases, it will be possible to construct provably optimal solutions, while in others only approximate solutions are available. There are also some difficult MRTA problems for which there do not currently exist good approximations. When designing a multi-robot system, it is essential to understand what kind of task allocation problem is present in order to solve it in a principled manner.

5.1 Axes of MRTA

I propose the following three axes for use in describing MRTA problems:

- **single-task robots (ST)** vs. **multi-task robots (MT)**: ST means that each robot is capable of executing as most one task at a time, while MT means that some robots can potentially execute multiple tasks simultaneously.

- **single-robot tasks (SR)** vs. **multi-robot tasks (MR)**: SR means that each task requires exactly one robot to achieve it, while MR means that some tasks require multiple robots.

- **instantaneous assignment (IA)** vs. **time-extended assignment (TE)**: IA means that the available information concerning the robots, the tasks, and the environment permits only an instantaneous allocation of tasks to robots, with no planning for future allocations. TE means that more information is available, such as the set of all tasks that will need to be assigned, or a model of how tasks are expected to arrive.

I denote a particular MRTA problem by a triple of two-letter abbreviations drawn from this list. For example, a problem in which multi-robot tasks must be allocated once to single-task robots is designated ST-MR-IA.

These axes are not meant to be exhaustive, but I suggest that they allow for a taxonomy that is both broad enough and detailed enough to meaningfully characterize many practical MRTA problems. Furthermore, this taxonomy will often allow for a prescription of solutions. In the following sections, I present the combinations allowed by these axes, discussing for each which MRTA problem(s) it represents and what organizational theory pertains. In Section 5.9, I treat some important MRTA problems that are not captured by this taxonomy.

5.2 ST-SR-IA: Single-task robots, single-robot tasks, instantaneous assignment

This problem is the simplest, as it is actually an instance of the OAP: given a group of robots and a group of tasks, assign at most one task to each robot. As described in Chapter 3, there exist both centralized and distributed solutions to this problem. If the robots' utilities can be collected at one machine (or distributed to all machines), then a centralized linear programming approach (e.g., Kuhn's (1955) Hungarian method) will find the optimal allocation in $O(n^3)$ time. Alternatively, a distributed auction-based approach (e.g., Bertsekas's (1990) Auction algorithm) will find the optimal allocation, usually requiring time proportional to the maximum utility and inversely proportional to the minimum bidding increment.

The two approaches represent a tradeoff between solution time and communication overhead. Centralized approaches generally run faster than distributed approaches, but incur a higher communication overhead. In order to explore this tradeoff, I have implemented, in ANSI C, the Hungarian method and the Auction algorithm (freely available from: `http://robotics.usc.edu/~gerkey`). As a simplification for testing, my implementation of the Auction algorithm is contained within a single process, and thus is not truly distributed. The communication overhead of this implementation, as measured by the number of required bid messages, is unaffected. However, the solution times reported here for the Auction algorithm should be considered lower bounds, because they do not include message transmission delays that would be seen in practice with a distributed implementation.

I conducted performance tests on my implementations of the Hungarian method and Auction algorithm on a Pentium III-700MHz running Linux. Shown in Figures 5.1 & 5.2 are the mean computation and communication overheads, respectively, for $n \times n$ assignment problems with uniformly randomly distributed utilities.[1] By linear regression, I have determined the running time of my implementation of the Hungarian method to be approximately $O(n^{3.89})$ with a constant coefficient on the order of 10^{-10} (seconds), which

[1] Of course, real MRTA problems are unlikely to exhibit uniformly randomly distributed utilities. Regardless, these results are indicative of the running time one can expect from these algorithms applied to MRTA problems.

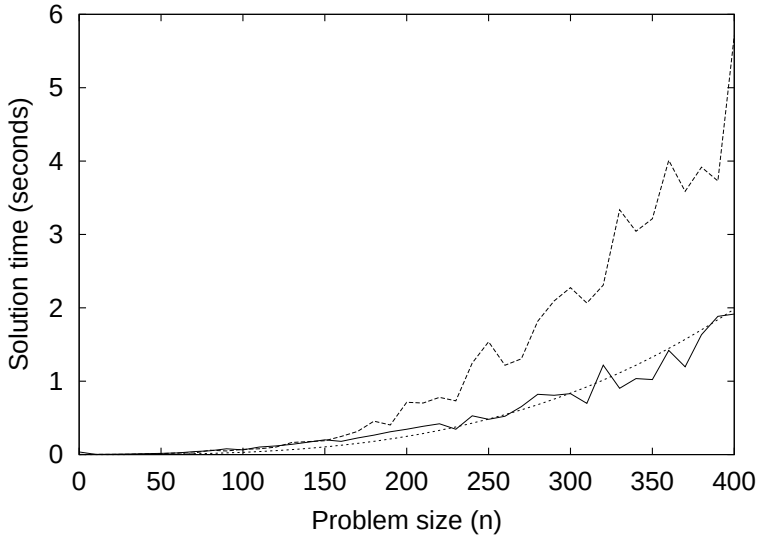

Figure 5.1: *Comparison of the computational overhead of assignment algorithms. The solid line and dashed line show the amount of time required by the Hungarian method and Auction algorithm, respectively, to solve randomly generated symmetric ($n \times n$) instances of the Optimal Assignment Problem (OAP) on a 700Mhz Pentium III running Linux. For comparison, the dotted line is $10^{-10} n^{3.89}$; the Hungarian method is known to exhibit a running time of $O(n^3)$.*

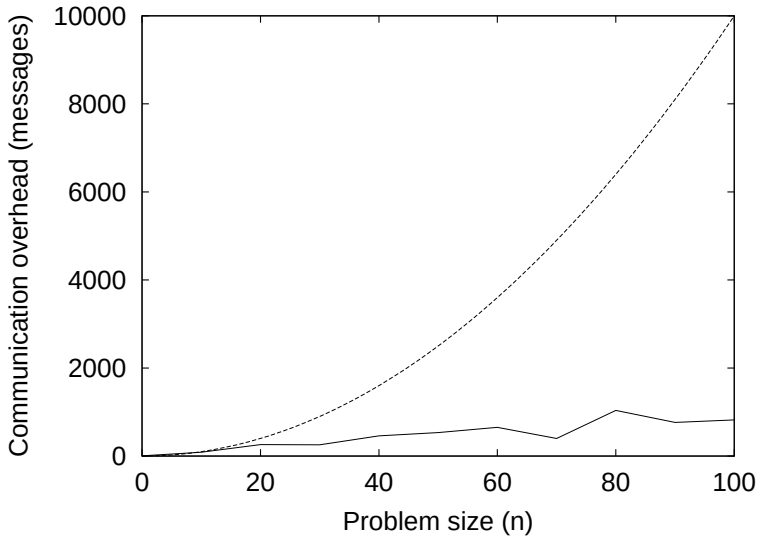

Figure 5.2: *Comparison of the communication overhead of assignment algorithms. The solid line and dashed line show the number of messages sent by the Hungarian method and Auction algorithm, respectively, when solving randomly generated symmetric ($n \times n$) instances of the Optimal Assignment Problem (OAP).*

roughly agrees with the theoretically predicted running time of $O(n^3)$. As Figure 5.1 shows, this algorithm is efficient enough to be used to solve MRTA problems with hundreds of robots and hundreds of tasks online.

To implement a centralized assignment algorithm, n^2 messages are required to transmit the utility of each robot for each task; an auction-based solution usually requires far fewer (sometimes fewer than n) messages to reach equilibrium, as shown in Figure 5.2. With the addition of simple optimizations, such as buffering multiple utility values and transmitting them in one message, this gap in communication overhead will only become apparent in large-scale systems. Furthermore, the time required to transmit a message cannot be ignored, especially in wireless networks, which can induce significant latency. Thus, for small- to medium-scale systems, say $n < 200$, a broadcast-based, centralized assignment solution is likely the better choice. Not surprisingly, many MRTA architectures implement some form of this approach (Parker 1998, Werger & Matarić 2000, Castelpietra, Iocchi, Nardi, Piaggio, Scalzo & Sgorbissa 2001, Weigel, Auerback, Dietl, Dümler, Gutmann, Marko, Müller, Nebel, Szerbakowski & Thiel 2001, Østergård, Matarić & Sukhatme 2001).

5.2.1 Variant: iterated assignment

Few MRTA problems exhibit exactly the above one-time assignment structure. However, many problems can be framed as *iterated* instances of ST-SR-IA. Consider the cooperative multi-object tracking problem CMOMMT, studied by Parker (1999*b*) and Werger & Matarić (2000), which consists of coordinating robots to observe multiple unpredictably moving targets. When presented with new sensor inputs (e.g., camera images) and consequent utility values (e.g., perceived distance to each target), we must decide which robot should track which target.

Werger & Matarić's (2000) MRTA architecture BLE solves this iterated assignment problem using the following algorithm:

Algorithm 5.1 (BLE assignment algorithm) :

1. If any robot remains unassigned, find the robot-task pair (i, j) with the highest utility. Otherwise, quit.
2. Assign robot i to task j and remove them from consideration.
3. Go to step 1.

This algorithm is equivalent to the canonical Greedy algorithm (Algorithm 3.3). The Greedy algorithm is known to be 2-competitive for the OAP (see Section 3.6.1), and thus so is BLE. Exactly this algorithm, operating on a global blackboard, has been used in a study of the impact of communication and coordination on MRTA (Østergård et al. 2001). An extremely similar assignment algorithm is also used by Botelho & Alami's (1999) MRTA architecture M+, which, like MURDOCH, is based on the Contract Net Protocol (CNP) (Smith 1980, Davis & Smith 1983).

Parker's (1998) MRTA architecture L-ALLIANCE, which can also perform iterated allocation, learns its assignment algorithm from experience. The resulting algorithm is similar to, but potentially more sophisticated than, the Greedy algorithm. If well-trained, the L-ALLIANCE assignment algorithm can perform better than the Greedy algorithm (Parker 1994*b*), but is not guaranteed to be optimal.

Another domain in which the iterated OAP arises is robot soccer (Asada et al. 1999). Since many of the robots are interchangeable, it is often advantageous to allow any player to take on any role within the team, according to the current situation in the game. The resulting coordination problem can be cast as an iterated assignment problem in which the robots' roles are periodically reevaluated, usually at a frequency on the order of 10Hz. This utility-based dynamic role assignment problem and has been studied by many (Stone & Veloso 1999, Weigel et al. 2001, Castelpietra et al. 2001, Brusey, Makies, Padgham, Woodvine & Fantone 2001, Emery, Sikorski & Balch 2002, Vail & Veloso 2003).

It is common in the robot soccer domain for each robot to calculate its utility for each role and periodically broadcast these values to its teammates. The robots can then execute, in parallel, some centralized assignment algorithm. For example, Castelpietra et al.'s (2001) assignment algorithm consists of ordering the roles in descending priority and then assigning each to the available robot with the highest utility. This algorithm is yet another instance of the Greedy algorithm. Vail & Veloso (2003) also employ the Greedy algorithm with fixed priority roles. Weigel et al. (2001) employ a similar but slightly more sophisticated

algorithm that tries to address the problem of excessive role-swapping by imposing stricter prerequisites for reassignment. Among other things, the algorithm requires that both robots "want" to exchange roles in order to maximize their respective utilities, recalling the conditions for equilibrium in markets (see Section 3.3). However, Weigel et al.'s (2001) algorithm is not guaranteed to produce optimal assignments of roles, a fact that can easily be shown by counterexample.

Since the number of robots involved in many iterated MRTA problems is small ($n \leq 11$ for robot soccer, which is more than for most multi-robot systems), $O(n^3)$ *optimal* assignment algorithms could easily replace the suboptimal *ad hoc* assignment algorithms that are usually used. As the performance results in the previous section show, the Hungarian method can be used to solve problems of this size in less than 1ms per iteration with the moderately powerful computers that can be found on today's robots.

Since there is some additional cost for running an optimal algorithm (if only in the work involved in the implementation), one might ask whether the optimal solution provides a sufficient benefit. For example, we know that for arbitrary assignment problems, the Greedy algorithm's *worst-case* behavior is to produce a solution with half of the optimal utility. However, we do not know how the algorithm can be expected to perform on typical MRTA problems, which exhibit some some structure and are unlikely to present pathological utility combinations. Empirical evidence suggests that the Greedy algorithm works extremely well on such problems. An interesting avenue of research would be to analytically determine how well the Greedy algorithm will perform on the kinds of utility landscapes that are encountered in MRTA problems.

5.2.2 Variant: online assignment

In some MRTA problems, the set of tasks is not revealed at once, but rather the tasks are introduced one at a time. If robots that have already been assigned cannot be reassigned, then this problem is a variant of SR-ST-IA known as *online assignment* (Khuller, Mitchell & Vazirani 1994). Instead of being initially given, the robot-task utility matrix is revealed one column (or row) at a time. If previously assigned robots

can be reassigned, then the problem reduces to an instance of the iterated SR-ST-IA problem, which can be optimally solved with standard assignment algorithms.

The MRTA problems presented in Chapter 4, in which tasks are randomly injected into the system over time, are instances of the online assignment problem. The MURDOCH assignment algorithm that was used to solve those problems can be stated as follows:

Algorithm 5.2 (MURDOCH assignment algorithm) :

1. When a new task is introduced, assign it to the most fit robot that is currently available.

This simple algorithm is known in the context of network flows as the *Farthest Neighbor* algorithm and is 3-competitive with respect to the optimal *post hoc* offline solution. Furthermore, this performance bound is the best possible for any online assignment algorithm (Kalyanasundaram & Pruhs 1993). Thus, without a model of the tasks that are to be introduced, and without the option of reassigning robots that have already been assigned, it is impossible to construct a better task allocator than MURDOCH.

5.2.3 Analysis of some existing approaches

Presumably because it is the simplest case of MRTA, the ST-SR-IA problem has received the most attention from the research community. Having developed a formal framework in which to study to this MRTA problem, it is possible to apply that framework in an analysis of some of the key task allocation architectures from the literature. In this section I examine six approaches to MRTA, focusing on three characteristics (this analysis was originally presented by Gerkey & Matarić (2003*b*)):

- computation requirements (Cormen et al. 1997)
- communication requirements (Kushilevitz & Nisan 1997)
- solution quality (Sleator & Tarjan 1985)

In part because of trends in the research community that stress the importance of experimental validation with physical robots, such theoretical aspects of multi-robot coordination mechanisms have been

largely ignored. However, they are vitally important to the study, comparison, and objective evaluation of the mechanisms. The large-scale and long-term behavior of the system will be strongly determined by the fundamental characteristics of the underlying algorithm(s). Thus I endeavor to derive and explicate those characteristics here. First, however, it will be necessary to explain the methodology that is used in the analysis.

5.2.3.1 Methodology

I determine computational requirements, or running time, in the usual way, as the number of times that some dominant operation is repeated. For the MRTA domain that operation is usually either a calculation or comparison of utility, and running time is stated as a function of n and m, the numbers of robots and tasks, respectively. Since modern robots have significant processing capabilities on-board and can easily work in parallel, I assume that the computational load is evenly distributed over the robots, and state the running time as it is *for each robot*. For example, if I need to find for each robot the task with the highest utility, then the running time is $O(m)$, because each robot performs m comparisons, in parallel. Note that I do *not* measure or consider the actual running time of the utility calculation, in large part because that information is not generally reported. Rather I operate under the assumption that the utility calculations are computationally similar enough to be meaningfully compared.

Communication requirements are determined as the total number of inter-robot messages sent over the network. I do not consider message sizes, on the assumption that they are generally small (e.g., single scalar utility values) and approximately the same for different algorithms. I also assume that a perfect shared broadcast communication medium is in use and that messages are always broadcast, rather than unicast. So if, for example, each robot must tell every other robot its own highest utility value then the overhead is $O(n)$, because each robot makes a single broadcast.

Solution quality is reported in terms of a competitive factor (see Section 3.6.1).

Name	Comp. / iter.	Comm. / iter.	Solution quality
ALLIANCE[2] (Parker 1998)	$O(mn)$	$O(m)$	at least 2-competitive
BLE (Werger & Matarić 2000)	$O(mn)$	$O(mn)$	2-competitive
M+ (Botelho & Alami 1999)	$O(mn)$	$O(mn)$	2-competitive

Table 5.1: *Summary of selected iterated assignment architectures for MRTA. Shown here for each architecture are the computational and communication requirements, as well as solution quality.*

Name	Comp. / task	Comm. / task	Solution quality
MURDOCH (Gerkey & Matarić 2002*c*)	$O(1)$ / bidder $O(n)$ / auctioneer	$O(n)$	3-competitive
First-price auctions (Dias & Stentz 2001)	$O(1)$ / bidder $O(n)$ / auctioneer	$O(n)$	at least 3-competitive
Dynamic role assignment (Chaimowicz et al. 2002)	$O(1)$ / bidder $O(n)$ / auctioneer	$O(n)$	at least 3-competitive

Table 5.2: *Summary of selected online assignment architectures for MRTA. Shown here for each architecture are the computational and communication requirements, as well as solution quality.*

5.2.3.2 Results & discussion

I have analyzed six MRTA architectures that have been validated on either physical or simulated robots. My choices are somewhat subjective, for there are a great many more architectures in the literature. However, I believe that I have gathered a set of approaches that is fairly representative of the work to date.

Most details of the analysis have been presented in the previous sections, and I do not repeat them here. Rather I refer the reader to Tables 5.1 & 5.2, in which are summarized the results for the iterated assignment architectures and online assignment architectures, respectively. Perhaps the most significant trend in those results is how similar the architectures look when examined within this framework. For example, the iterated architectures listed in Table 5.1, which assign all available tasks simultaneously, exhibit almost identical algorithmic characteristics. Only the ALLIANCE architecture (Parker 1998) shows any

[2] In addition to solving the ST-SR-IA problem, the ALLIANCE architecture is also capable of building time-extended task *schedules* in order to solve a form of the ST-SR-TE problem (see Section 5.3.1).

difference; in this case the decrease in communication overhead is achieved by having each robot internally model the fitness of the other robots, thereby effectively distributing the utility calculations. More striking are the results in Table 5.2, which lists architectures that assign tasks in a sequential manner: with respect to computational and communication requirements, these architectures are *identical*. In terms of solution quality, Dias & Stentz's (2001) and Chaimowicz et al.'s (2002) approaches, which allow reassignment of tasks, can potentially perform better than MURDOCH.

These results are particularly interesting because they suggest that there is some common methodology underlying many existing approaches to MRTA. This trend is difficult or impossible to discern from simply reading the technical papers describing the work, as each researcher tells a different "story" regarding his or her architecture, validates the architecture in a different task domain, and explains it in different terms. However, seen through the lens of a formal framework, fundamental similarities of the various architectures are immediately obvious. These similarities are encouraging because they suggest that, regardless of the details of the robots or tasks in use, we are all studying a common, deep problem in autonomous coordination. As a corollary, there is some reason to believe that these *ad hoc* architectures may in fact have properties that allow them to be generalized and applied widely.

Without such analysis, it is impossible to objectively compare proposed solutions to robotics problems. Of course I have not captured all relevant aspects of the systems under study. For example, in the ALLIANCE architecture the robots' computational load is increased to handle modeling of other robots, but we do not consider that extra load in our analysis. Such details, which are currently not widely discussed in the literature, will likely become more important as the field demands better cross-evaluation of solutions.

In addition to enabling evaluation, this kind of analysis can be used to explain *why* certain solutions work in practice. For example, the online assignment architectures listed in Table 5.2 are all economically-inspired, built around task *auctions*. While the designers of such architectures generally justify their approach with a loose analogy to the efficiency of the free market as it used by humans, it is possible to gain a clearer understanding of what is happening. In fact, it is not surprising that auction-based allocation

methods work in practice, for it is well known that synthetic economic systems can be used to solve a variety of optimization problems. In fact, an appropriately constructed price-based market (which the previously described architectures approximate to varying degrees) can optimally solve assignment problems, as discussed in Section 3.3.

5.3 ST-SR-TE: Single-task robots, single-robot tasks, time-extended assignment

If we are given more tasks than robots (or have a model of how tasks will arrive) and can, with some accuracy, predict robots' *future* utilities for the tasks, then we have an instance of ST-SR-TE. This problem is one of building a time-extended *schedule* of tasks for each robot, with the goal of minimizing total weighted cost. The problem is an instance of the class of scheduling problems:

$$R \mid\mid \sum w_j C_j$$

As mentioned in Section 3.4.1, problems in this class are strongly $\mathcal{NP}$-hard (Bruno et al. 1974). Even for a relatively small problem, the exponential space of possible schedules precludes an enumerative solution.

One way to attack ST-SR-TE is to ignore the time-extended component and approximate the problem as an instance of iterated ST-SR-IA, described in Section 5.2.1. For example, given m robots and n tasks, with $m > n$, then we can do the following:

Algorithm 5.3 (ST-SR-TE approximation algorithm) :

1. Optimally solve the initial $m \times n$ assignment problem.
2. Use the Greedy algorithm to assign the remaining tasks as the robots become available.

The performance of this algorithm is bounded below by the normal Greedy algorithm, which is 3-competitive. Clearly, the more tasks that are assigned in the first step, the better this algorithm will perform. As the difference between the number of robots and the number of tasks shrinks (i.e., $(n-m) \rightarrow 0$), performance

approaches optimality, wherein all tasks are assigned in one step. Thus, although it is not guaranteed to produce optimal solutions, Algorithm 5.3 should work well in practice, especially for ST-SR-TE problems with short time horizons.

Another way to approach this problem is to employ an iterative task allocation system, such as Dias & Stentz's (2001) free market. The robots would opportunistically exchange tasks over time, thereby modifying their schedules. This idea is demonstrated by the multi-robot exploration system described by Zlot et al. (2002). Within the multi-agent community, several biologically-inspired approaches to this problem have been proposed (e.g., Cicirello & Smith (2001), Nouyan (2002)).

5.3.1 Variant: ALLIANCE Efficiency Problem

Parker (1995) formulated a related MRTA problem called the ALLIANCE Efficiency Problem (AEP).[3] In the AEP, given is a set of tasks making up a mission, and the objective is to allocate a subset of these tasks to each robot so as to minimize the maximum time taken by a robot to serially execute its allocated tasks. Thus in order to solve the AEP, one must construct a time-extended schedule of tasks for each robot. This problem is an instance of the class of scheduling problems:

$$R \mid\mid C_{max}$$

Problems in this class are known to be strongly $\mathcal{NP}$-hard (Garey & Johnson 1978). Parker (1995) arrived at the same conclusion regarding the AEP, by reduction from the $\mathcal{NP}$-complete problem PARTITION.

To attack the AEP, Parker (1998) used a learning approach, in which the robots learn both their utility estimates and their scheduling algorithms from experience. When trained for a particular task domain, this system has the potential to outperform Algorithm 5.3.

[3] Parker (1997) also refers to the AEP as the Heterogeneous Robot Action Selection Problem (HRASP).

5.4 ST-MR-IA: Single-task robots, multi-robot tasks, instantaneous assignment

Many MRTA problems involve tasks that require the combined effort of multiple robots. In this case we must consider the *combined* utilities of groups of robots. These combined utility values are in general *not* sums over individual utilities; rather utility may be defined arbitrarily for each potential group. For example, if a task requires a particular skill or device, then any group of robots without that skill or device has zero utility with respect to that task, regardless of the capabilities of the other robots in the group. This kind of problem is significantly more difficult than the previous MRTA problems, which were restricted to single-robot tasks.

In the multi-agent community, the ST-MR-IA problem is referred to as *coalition formation*, and has been studied by many, including Ketchpel (1994), Sandholm & Lesser (1997), Sandholm, Larson, Andersson, Shehory & Tohmé (1999), Shehory & Kraus (1998), Lerman & Shehory (2000), and Ortiz, Hsu, desJardins, Rauenbusch, Grosz, Yadgar & Kraus (2001).

It is natural to think of the ST-MR-IA problem as splitting the set of robots into task-specific coalitions. A relevant concept from set theory is that of a set partition. A family X is a *partition* of a set E if and only if the elements of X are mutually disjoint and their union is E:

$$\bigcap_{x \in X} = \emptyset$$

$$\bigcup_{x \in X} = E.$$

With the idea of partitions in mind, there is a well-known problem in combinatorial optimization called the (maximum utility) Set Partitioning Problem, or SPP (Balas & Padberg 1976):

Definition 5.1 (Set Partitioning Problem (SPP)) : Given a finite set E, a family F of acceptable subsets of E, and a utility function $u : F \rightarrow \mathbb{R}_+$, find a maximum-utility family X of elements in F such that X is a partition of E.

The ST-MR-IA problem can be cast as an instance of SPP, with E as the set of robots, F as the set of all feasible coalition-task pairs, and u as the utility estimate for each such pair.

Unfortunately, the SPP is strongly $\mathcal{NP}$-hard (Garey & Johnson 1978). Fortunately, the problem has been studied in depth (e.g., Balas & Padberg (1976), Fisher & Kedia (1990), Atamtürk, Nemhauser & Savelsbergh (1995)), especially in the context of solving crew scheduling problems[4] for airlines (e.g., Marsten & Shepardson (1981), Hoffman & Padberg (1993), Wedelin (1995), Chu & Beasley (1995)). As a result, many heuristic SPP algorithms, both approximate and exact, have been developed.

It remains to be seen whether such heuristic algorithms are applicable to MRTA problems. Some approximation algorithms, including those of Hoffman & Padberg (1993) and Atamtürk et al. (1995), have been shown to produce high-quality solutions to many instances of SPP. Even with hundreds of rows/columns and using mid-1990s workstation-class machines, these algorithms require at most a few tens of seconds to arrive at a solution. Especially in light of the ever-increasing computational power that is available on robots, it seems plausible that SPP approximation algorithms could be used to solve small- and medium-scale instance of the ST-MR-IA problem. To this end, a potentially important question is whether and how these algorithms can be parallelized. Shehory & Kraus (1998) showed how to implement such a parallel SPP algorithm for coalition formation in multi-agent systems. Another important point is that, in order to apply some SPP algorithms to ST-MR-IA problems, it may be necessary to enumerate a set of feasible coalition-task combinations. In the case that the space of such combinations is very large, there will be a need to prune the feasible set; this pruning can be done in many cases with metrics such as distance.

[4] In such problems, it is not the set of employees that is being partitioned into crews. Rather, given predetermined crews, the task is to partition the set of flight legs into multi-leg routes. Regardless, the underlying problem, SPP, remains the same.

5.5 ST-MR-TE: Single-task robots, multi-robot tasks, time-extended assignment

The ST-MR-TE class of problems includes both coalition formation component and scheduling. To produce an optimal solution, we would have to consider all possible schedules for all possible coalitions. This problem is clearly $\mathcal{NP}$-hard. If the coalitions are given, with no more than one coalition allowed for each task, then we have an instance of a multiprocessor scheduling problem:

$$MPTm \mid\mid \sum w_j C_j.$$

Even with two processors ($MPT2 \mid\mid \sum w_j C_j$), this problem is strongly $\mathcal{NP}$-hard (Hoogeveen, van del Velde & Veltman 1994), as is the unweighted version ($MPT2 \mid\mid \sum C_j$) (Cai, Lee & Li 1998). With three processors, the maximum finishing time version ($MPT3 \mid\mid C_{max}$) is also strongly $\mathcal{NP}$-hard (Błażewicz, Dell'Olmo, Drozdowski & Speranza 1992, Hoogeveen et al. 1994).

One way to attack ST-MR-TE is to ignore the time-extended component and approximate the problem as an instance of iterated ST-MR-IA. One can imagine a greedy approximation algorithm akin to Algorithm 5.3. Unfortunately, the quality of such an approximation is difficult to determine. Another way to approach this problem is to employ a leader-based mechanism to dynamically form coalitions and build task schedules for them, as described by Dias & Stentz (2002).

5.6 MT-SR-IA & MT-SR-TE: Multi-task robots, single-robot tasks

The MT-SR-IA and MT-SR-TE problems are uncommon, for they assume robots that can each concurrently execute multiple tasks. Today's mobile robots are generally actuator-poor, with the only actuators in many cases being the wheelmotors. Since a robot's ability to effect its environment is limited to changing its position, it is relatively rare that a robot can execute more than one task at a time. However, there are sensory and computational tasks that may fit the MT-SR-IA or MT-SR-TE models.

Regardless, solving the MT-SR-IA problem is equivalent to solving the ST-MR-IA problem (see Section 5.4), with the robots and tasks interchanged in the SPP formulation. Likewise, the MT-SR-TE problem is equivalent to the ST-MR-TE problem (see Section 5.5). Thus the analysis and algorithms provided for the multi-robot task problems also apply here to the multi-task robot problems.

5.7 MT-MR-IA: Multi-task robots, multi-robot tasks, instantaneous assignment

When both multi-task robots and multi-robot tasks are allowed, we have an instance of the MT-MR-IA problem. A relevant concept from set theory is the set cover. A family X is a *cover* of a set E if and only if the union of elements of X is E:

$$\bigcup_{x \in X} = E.$$

As compared with a partition (see Section 5.4), the subsets in a cover need not be disjoint. There is a well-known problem in combinatorial optimization called the (minimum cost) Set Covering Problem, or SCP (Balas & Padberg 1972):

Definition 5.2 (Set Covering Problem (SCP)) : Given a finite set E, a family F of acceptable subsets of E, and a cost function $c : F \rightarrow \mathbb{R}_+$, find a minimum-cost family X of elements in F such that X is a cover of E.

The MT-MR-IA problem can be cast as an instance of the SCP, with E as the set of robots, F as the set of all feasible (and possibly overlapping) coalition-task pairs, and c as the cost estimate for each such pair.

Though superficially similar to the SPP, the SCP is in fact a "distant relative," with the solution space of the SCP being far less constrained (Balas & Padberg 1976). The two problems are similar in that the SCP is also strongly $\mathcal{NP}$-hard (Korte & Vygen 2000).

Based on earlier work by Johnson (1974) and Lovász (1975), Chvátal (1979) developed a greedy approximation algorithm for the SCP. The competitive factor for this algorithm is logarithmic in the size of the largest feasible subset (i.e., $\mathrm{argmax}_{f \in F} |f|$), and the running time is polynomial in the number of

feasible subsets (i.e., $|F|$). Bar-Yehuda & Even (1981) present another heuristic set covering algorithm. The competitive factor of this algorithm is the maximum number of subsets to which any element belongs (i.e., $\max_{e \in E} |\{f \in F : e \in f\}|$), and the running time is the sum of the sizes of the feasible subsets (i.e., $\sum_{f \in F} |f|$) (Korte & Vygen 2000).

The important trend to note is that these heuristic algorithms perform well when the space of feasible subsets is limited, and that they perform poorly in the most general case of the SCP, with all subsets allowed. For MRTA, this result suggests that such algorithms would best be applied in environments in which the space of possible coalitions is naturally limited, as is the case with heterogeneous and/or physically distantly separated robots. In the case of equally-skilled collocated robots, these algorithms would tend to run slowly and produce poor-quality solutions.

To my knowledge set covering algorithms have not been applied to MRTA problems, and it is an open question as to whether such an application would be beneficial. However, the fact that Shehory & Kraus (1996) successfully adapted and distributed Chvátal's (1979) approximation algorithm for use in multi-agent systems suggests that SCP algorithms may indeed be viable for MRTA problems.

5.8 MT-MR-TE: Multi-task robots, multi-robot tasks, time-extended assignment

The MT-MR-TE problem is an instance of scheduling problem with multiprocessor tasks and multipurpose machines:

$$MPTmMPMn \mid\mid \sum w_j C_j.$$

This problem is strongly $\mathcal{NP}$-hard, because it includes as a special case the strongly $\mathcal{NP}$-hard scheduling problem $MPT2 \mid\mid \sum w_j C_j$. I am not aware of any heuristic or approximation algorithms for this difficult problem.

5.9 Other problems

Although the taxonomy given in the previous sections covers many MRTA domains, several potentially important problems are excluded. In this section I describe some problem domains that are not captured by the taxonomy.

5.9.1 Interrelated utilities

Consider the problem of assigning target points to a team of robots that are cooperatively exploring an unknown environment. Many targets (e.g., the so-called "frontier points" of Burgard, Moors, Fox, Simmons & Thrun (2000)) may be known at one time, and so the possibility exists to build a schedule of targets for each robot. Unfortunately, this problem is not an instance of ST-SR-TE, for the cost for a robot to visit target C will depend on whether that robots first visits target A or target B. This problem is actually an instance of the multiple traveling salesperson problem, or MTSP; even in the restricted case of one salesperson, this problem is strongly $\mathcal{NP}$-hard (Korte & Vygen 2000). If, as is often the case with exploration, it is possible to discover *new* targets over time, then we have an instance of the dynamic MTSP, or the Marco Polo problem.[5]

Given the difficulty of the multi-robot exploration problem, it is not surprising that researchers have not attempted to solve it directly or exactly. A heuristic approximation is offered by Zlot et al. (2002), who use TSP heuristics to build target schedules and derive costs that are used in Dias & Stentz's (2001) market-based task allocation architecture. When a robot discovers a new target, that robot inserts the new target into its schedule, but retains the option of later auctioning the target off to another, closer robot.

The multi-robot exploration problem is an example of a larger class of problems, in which a robot's utility for a task may depend on which other tasks that robot executes. These problems in turn form part of another, more general class of problems in which a robot's utility for a task may depend on which other

[5]The name *Marco Polo problem* is due to Andrew Howard, in reference to the idea of Marco Polo and his companions planning trips to various unexplored villages, with the chance of someone in each village informing the explorers of the existence and locations of other unexplored villages.

tasks *any* robot executes. That is, each robot-task utility can depend on the overall allocation of tasks to robots. Such interrelated utilities can sometimes be tractably captured with factored Partially Observable Markov Decision Processes (POMDPs), assuming that a world model is available (Guestrin et al. 2001).

For mobile robots, this situation can arise any time that physical interference contributes significantly to task performance. For example, consider a multi-robot resource transportation problem in which each robot must choose which of a predetermined number of source-sink roads to travel. The decision of which road to travel should take into account the congestion caused by other robots. Taking the position that interference effects are difficult or impossible to adequately model *a priori*, Dahl, Matarić & Sukhatme (2002) developed a reinforcement learning approach to the multi-robot resource transportation problem, and showed that the approach produces good solutions. The robots do not communicate with each other directly, but rather through physical interactions, with each robot maintaining and updating an estimate of the utility for each available road.

5.9.2 Task constraints

In addition to an assumption of independent utilities, the taxonomy given in this chapter assumes independent tasks. If instead there are constraints between the tasks, such as sequential or parallel execution, then this taxonomy will not suffice. Although the topic of job constraints is addressed by the scheduling literature (Brucker 1998), the addition of such constraints generally increases problem difficulty, and tractable algorithms exist for only the simplest kinds of constraints. A possible way to approach this problem is with the dynamic constraints satisfaction method described by Modi et al. (2001).

5.10 Summary

In this chapter, I presented a candidate taxonomy for studying problems of multi-robot task allocation (MRTA). Based on organizational theory developed in a variety of fields, this taxonomy provides a way to

classify many existing MRTA problems, as well as analyze proposed solutions. However, this taxonomy is not exhaustive, as the problems described in Section 5.9 show by counterexample.

In the next chapter, I describe the software and hardware infrastructure that I have developed and used in the validation experiments described in Chapter 4. The goal of providing and distributing such standard infrastructure is to enable empirical evaluation of proposed MRTA architectures, as a complement to the analytical framework described in this chapter.

Chapter 6

Enabling infrastructure

In the study of multi-robot task allocation (MRTA) problems, such as those described in the previous chapter, formal analysis is only one component of a comprehensive research agenda. Sensor-actuator systems such as robots present a rich, complex problem domain that can exhibit significant levels of noise and uncertainty. Thus one must implement and experimentally validate proposed MRTA algorithms. This kind of empirical work is a natural complement to formal analysis, and can serve the crucial role of suggesting modifications to the formal model that is analyzed.

Such experiments require substantial supporting infrastructure, both software and hardware. This infrastructure must be *embodied*, either in the physical world or in realistic simulation. In this chapter I motivate and describe the underlying software facilities that I developed for experimental use in this dissertation, as well as for more general use. In particular, I discuss the Player/Stage project, which produces high-quality Open Source software to support robotics research. One goal of this project is to provide a standard platform for mobile robot experiments that would allow for direct empirical comparison of competing control and coordination algorithms.

As a founding member of the Player/Stage project, I have led development on the robot device server Player and aided in the development of the robot simulator Stage. Stage development is led by Richard T. Vaughan. Andrew Howard contributes significantly to the development of both Player and Stage. Together,

we maintain the project, which is hosted at SourceForge.net:

`http://playerstage.sourceforge.net`.

6.1 Player & Stage

Programming robots is complicated and time-consuming. Working with multiple and distributed robot systems is further complicated by more robots and the difficulties of network programming. The *Player/Stage Project* provides Open Source tools that simplify controller development, particularly for multiple-robot, distributed-robot, and sensor network systems.

The Player/Stage project began at the USC Robotics Research Lab in 1999 to address an internal need for interfacing and simulation for multi-robot systems. It has since been adopted, modified and extended by researchers around the world. For many applications, particularly in multi-robot systems, Player and Stage offer a combination of transparency, flexibility and speed that makes it the most useful robot development environment available.

This section provides an overview of the Player/Stage tools and their application to multi-robot systems. I describe the tools and review published multi-robot work using Player/Stage, as well as describe some of the under-explored research opportunities opened up by this infrastructure.

6.1.1 The software

The Player/Stage project provides the *Player* robot device server and the *Stage* multiple robot simulator, plus supporting tools and libraries.

Running on a robot, Player provides a clean and simple interface to the robot's sensors and actuators over a network. Client control programs talk to Player over a TCP socket, reading data from sensors, writing commands to actuators, and configuring devices on the fly. Player supports a variety of robot hardware and provides implementations of sophisticated sensing and control algorithms, such as landmark tracking and probabilistic localization.

Stage provides a population of simulated robots and sensors operating in a two-dimensional bitmapped environment. The devices are accessed through Player, as if they were real hardware. Stage aims to be efficient and configurable rather than highly accurate. In practice this means that Stage can simulate tens or hundreds of robots on a desktop PC, and that controllers commonly work similarly on simulated and real robots.

Player and Stage run on many UNIX-like platforms, are released as Free Software under the GNU General Public License (Free Software Foundation 1991) and are available at:

`http://playerstage.sourceforge.net`.

6.1.2 Player goals and design

The Player architecture was originally described by Gerkey, Støy & Vaughan (2000) and Gerkey, Vaughan, Støy, Howard, Sukhtame & Matarić (2001). This section reports on the current state of Player, focusing on those aspects of the design that facilitate the investigation of novel distributed sensing and control algorithms and the empirical comparison of proposed algorithms. Some of these ideas were originally explored by Gerkey, Vaughan & Howard (2003).

6.1.2.1 Client interface

Player is a socket-based device server that allows control of a wide variety of robotic sensors and actuators, and was influenced by experience with the TRIP server (Jennings 1998). Player executes on a machine that is physically connected to a collection of such devices and offers a TCP socket interface to clients that wish to control them. Clients connect to Player and communicate with the devices by exchanging messages with Player over a TCP socket. In this way, Player is similar to other device servers, such as the standard UNIX printer daemon `lpd`. Like those servers, Player can support multiple clients concurrently, each on a different socket.

Because Player's external interface is simply a TCP socket, client programs can be written in any programming language that provides socket support, and almost every language does. Client libraries, which

encapsulate the details of the Player message protocol and facilitate the development of control programs, are currently available in: C, C++, Tcl, Python, Java, and Common LISP. With language neutrality comes platform neutrality; control programs written in Tcl, Python, and Java can run on almost any modern system, even those running Windows. In addition, the C++ client library has been ported to the Win32 environment.

More importantly, the socket abstraction allows *location* neutrality. Regardless of the physical location of a collection of robotic devices, a client program can exert control over them from any machine to which there is network connectivity. When combined with Player's ability to support multiple clients concurrently, this location neutrality provides new opportunities for building distributed sensing and control systems. I take up this idea further in Section 6.1.3.

Device model In order to provide a uniform abstraction for a variety of devices, Player follows the UNIX model of treating devices as files (Ritchie & Thompson 1974), which was in turn influenced by Multics (Feiertag & Organick 1971). Thus the familiar file semantics hold for Player devices. For example, to begin receiving sensor readings, the client opens the appropriate device with `read` access; likewise, before controlling an actuator, the client must open the appropriate device with `write` access. In addition to the asynchronous data and command streams, there is a request/reply mechanism, akin to `ioctl()`, that clients can use to get and set configuration information for Player devices. As this model has served UNIX-like operating systems well for decades, it can be expected suit Player devices well into the future.

Player does not implement any device locking, so when multiple clients are connected to a Player server, they can both issue commands to the same device. In general, there is no locking of devices or queuing of commands, and each new command will overwrite the old one. Locking is not implemented in order to provide maximal power and flexibility to the client programs. If multiple clients are concurrently controlling a single device, such as a robot's wheels, then those clients are probably cooperative, in which case they should implement their own arbitration mechanism at a higher level than Player. If the clients are

not cooperative, then the subject of research is presumably the interaction of competitive agents, in which case device locking would be a hindrance.

Player borrows further from classic operating system design in the way that deice interfaces are separated from device drivers. For example, in an operating system there is a joystick interface that defines the API for interacting with joysticks, and there are joystick drivers that allow the programmer to control various joysticks through that same API. Similarly, in Player, a device *interface* is a specification of data, command, and configuration formats, and a device *driver* is a module that controls a device and provides a standard interface to it.

Probably the most commonly used Player interface is the `position` interface, which is used to control a mobile robot base. This interface specifies a command format that includes velocity and/or position targets and a data format that includes velocity and position status. One implementation of the `position` interface is Player's `p2os` driver, which controls research robots made by ActivMedia, including the popular Pioneer 2-DX (see Figure 6.2). Other drivers that control other kinds of robots also support the `position` interface, which means that they all accept commands and return data in the same format. In general, multiple drivers can support the same interface, and a driver can support multiple interfaces. Some advantages of this design are discussed in Section 6.1.3.

6.1.2.2 Stage goals and design

Stage simulates a population of mobile robots, sensors and inanimate objects. It has two original purposes: (i) to enable rapid development of controllers that will eventually drive real robots; and (ii) to enable robot experiments without access to the real hardware and environments. Since Stage's inception, the sensor and actuator models have been extended and generalized beyond the limits of any available hardware, adding another purpose: (iii) to enable hypothetical experiments with novel devices that do not (yet) exist. An eventual goal along this path is to used Stage as a tool to determine the likely benefits of developing one type of sensor over another. This idea is discussed further in Section 6.1.3.6.

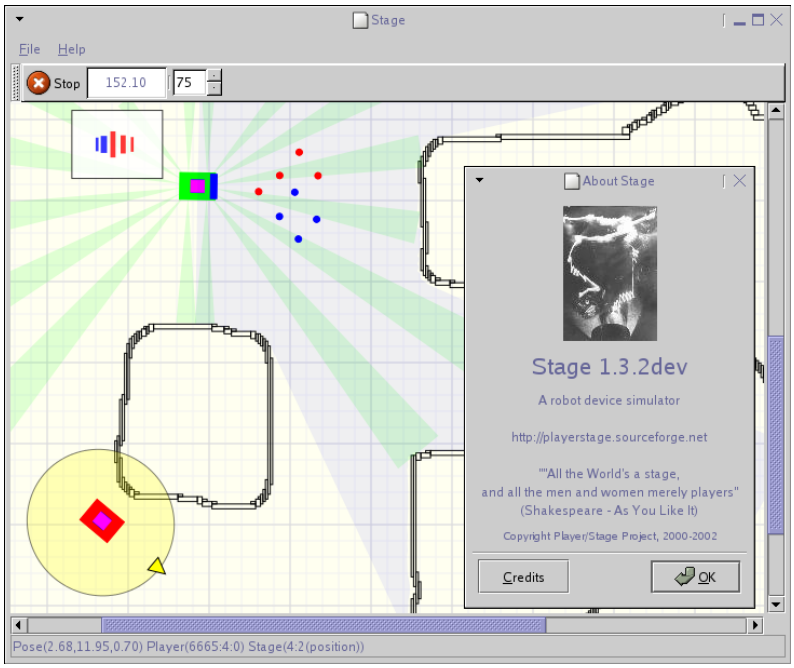

Figure 6.1: *Stage screenshot showing two robots (solid rectangles) with visualization of the top robot's laser range scanner, sonar and color blob-finder data. Stage's modular architecture allows multiple GUIs; this is GNOME2.*

Stage was specifically designed to support research into multi-robot systems. When programming and experimenting with many robots the benefits of rapid development are multiplied, and Stage enables experiments with large populations of robots that would be prohibitively expensive to buy and maintain. There are several aspects of Stage's design that make it suitable for multi-robot systems:

- **Good *enough* fidelity**: Stage provides fairly simple, computationally cheap models of lots of devices rather than attempting to emulate any device with great fidelity. Low fidelity simulation can actually be an advantage when designing robot controllers that must run on real robots, as it encourages the use of robust control techniques (Jakobi 1997). Low computational demands mean that Stage can simulate many devices on commodity hardware.

- **Linear scaling with population:** All sensor models use algorithms that are independent of population size. Thus Stage's computational requirements grow linearly with population[1].

[1] Some future devices may not follow this rule, as some algorithms that scale as a power of the population size can be convenient to implement and will perform well with small populations.

- **Configurable, composable device models:** Various sensors and actuators are provided, including sonars, scanning laser rangefinders, visual color segmenters, fiducial detectors, and a versatile mobile robot base with odometry. The models are often more general and flexible than any specific piece of hardware, so each model is configured to approximate the (real or imagined) target device. The manual (Vaughan, Howard & Gerkey 2002) provides a complete list of devices and their properties.

- **Player interface:** All sensor and actuator models are available through Player's standard interfaces. Typically, clients cannot tell the difference between the real robot devices and their simulated Stage equivalents (unless they try very hard). Thus Stage inherits the flexibility of Player's non-locking, platform- and language-neutral interface for all its devices.

Devices, populations and performance Devices can be composed in tree structures to build up complex robots. For example, most users base their robots on the `position` model with a selection of sensor models on top. Several users (Howard, Matarić & Sukhatme 2003, Jung & Sukhatme 2002, Howard et al. 2003, Makarenko, Williams, Bourgault & Durrant-Whyte 2002, Vaughan, Støy, Howard, Sukhatme & Matarić 2002) have simulated the Pioneer 2DX pictured in Figure 6.2 with a `position` model carrying a `sonar` array and `laser` scanning rangefinder. The Stage distribution includes some commonly-used configurations, such as the geometry of the Pioneer's sixteen sonar transducers.

By default, Stage attempts to run in real-time. Models are updated at a fixed (configurable) interval. If the update takes longer than the suggested interval, simulations will run slower than real time. Device models vary greatly in computational demands; for reference, a 600MHz Pentium III Linux PC can simluate 200 sonar-guided robots (`position` & `sonar` models) or 15 laser-guided robots (`position` & `laser` models) in real time at spatial and temporal resolutions of 0.02m and 100ms, respectively.

In the optional "fast mode", Stage does not wait for the real-time clock. Simple simulations will run much faster than real time, which is useful for long or batch experiments (e.g., Dahl et al. (2002)). Time-sensitive clients use Player's internal clock to avoid time-warping issues.

Validity There is no guarantee that experiments in Stage are directly comparable with those using real-world robots. However, users have found that clients developed using Stage will work with little or no modification with the real robots and vice versa (Jung & Sukhatme 2002, Makarenko et al. 2002, Vaughan, Støy, Howard, Sukhatme & Matarić 2002). As the number of transfers between Stage and real robots increases, users have an increasingly powerful argument to support the real-world validity of Stage-only experiments. This is a major advantage of using a well-known simulator instead of home-grown, project-specific code. Also, Stage's Open Source license allows peer review of the simulation code, and encourages sharing of models, configurations and environments. Just as Player facilitates code re-use and sharing, Stage enables experiment sharing. It is expected that standard test scenarios will emerge, in which users can compare their controllers. Already several papers describe experiments done using Stage's "hospital" environment, though with different robot configurations (e.g., Howard, Matarić & Sukhatme (2002*a*)).

6.1.3 Opportunities for research

Player makes it very easy for clients to read data from and send commands to any device on the network, as well as send arbitrary messages to one another. Stage allows for convenient and rapid evaluation of many clients. Both programs constrain the design of clients very little; they aim to provide transparent infrastructure for multi-robot research. In this section I describe some of the research opportunities enabled by the design decisions described above.

6.1.3.1 Embedded systems

The design of Player has been guided in part by a desire to maximize its utility and applicability by keeping it small and fast. Thus the server core, which provides sophisticated client services, is actually quite simple and has become more so over time as, for example, most of its functionality has been collapsed into a single thread of execution. The device driver system is modular, allowing the system designer to include only those drivers that are necessary for a particular application[2]. Because it is small and fast, Player is equally

[2]Without any device drivers, the Player server binary, as of version 1.3, is only about 64KB.

well-suited to run on low-power embedded systems and high-power workstations. Player is currently in use on embedded PPC/Linux and ARM/Linux systems.

6.1.3.2 Sophisticated devices

In addition to "regular" device drivers that provide an almost transparent control interface to a piece of hardware, Player's extensible device model allows sophisticated sensing and control algorithms to be encapsulated in drivers. These "abstract" drivers can perform arbitrary computation, ranging from signal processing to closed-loop control. For example, Player includes both a `waveaudio` driver that delivers raw audio data and a `fixedtones` driver that performs a Fast Fourier Transform on the raw data and reports the frequencies and amplitudes of the highest peaks in the frequency domain. Similarly, Player contains a collection of `fiducial` detectors, each designed to find a different kind of landmark by processing data from various sensors. One such detector fuses information from a laser range-finder and a camera image and incorporates control of a pan-tilt-zoom unit to find landmarks. Recently, we have added drivers that implement different forms of the widely-used Monte Carlo localization scheme. When included as drivers in the server, these algorithms become standard services that any client can exploit, even without knowledge of how they work. In this way, Player becomes a common code repository that is open to the research community.

6.1.3.3 Common device interfaces

As mentioned in Section 6.1.2.1, Player's device model permits drivers that control different hardware or implement different algorithms to present the same interface to the client. As a result, control programs can largely ignore the details of the underlying hardware or algorithm, treating the system as a collection of generic devices. For example, the Player drivers that control mobile robot bases made by ActivMedia, RWI, and K-Team all present the same `position` interface. Thus a Player client program can control any of those robots, with little or no changes required to move between platforms. This idea is explored at some length by Vaughan, Gerkey & Howard (2003). Several IMU drivers also present the `position` interface

and so appear to be immobile bases. Similarly, the landmark detectors mentioned above all present the `fiducial` interface, so, for example, a client program that builds a landmark-based map can employ the fiducials that are most appropriate to a given environment but largely ignore which detector is in use.

6.1.3.4 Novel control systems

As a result of its innovative network-centric architecture, Player permits any client, located anywhere on the network, to access any device; a robot can effectively "see" through its teammates' eyes. Using Player as a substrate, novel distributed sensing and control systems that were previously unrealizable can now be constructed quite easily. This feature was exploited in recent work on concurrent control (Gerkey, Matarić & Sukhatme 2002), in which approximately fifty independent agents were simultaneously controlling a single robot's motors through Player. Similarly, Player allowed a team of robots engaged in cooperative localization (Howard et al. 2003) to directly access each others' sensors, thereby facilitating sensor fusion. In building such systems, the designer is free to choose the most appropriate programming language and computing platform to implement each component.

6.1.3.5 Comparing controllers and performance metrics

As a result of its flexibility and the Open Source development model, Player and Stage are becoming widely adopted. Thus there arises the potential for an open *standard* test platform that would encourage objective evaluation and comparison of robot control and coordination algorithms. If an algorithm is implemented to interact with Player's standard device interfaces, then that implementation can be tested against the wide variety of devices, both physical and simulated, that are supported by Player[3].

There are currently relatively few practical metrics or other characterizations of robot behavior, yet there is a great deal of interest in this area. As such metrics are developed, they can be embedded in Stage, allowing the simulator to automatically "score" a given algorithm, and offering the possibility of operating

[3]Some care may be required to write truly portable robot code. For example, a sonar-based motion controller would need to account for the geometry of the particular sonar array in use. Such geometry information is available at run time by querying Player.

an always-on Stage server that allows anyone with a connection to the Internet to test his or her algorithm in a variety of benchmark environments. Such evaluation will be required for the field to transition from a primarily *ad hoc* experimental science to a more principled discipline.

6.1.3.6 Fantastic sensors

Stage is normally used to simulate existing robot devices, as users test the feasibility of their ideas for controlling real robots. Stage can also be used as a "what if" tool, to explore robot controllers that use devices that do not exist. This is useful as a conventional design tool, allowing investigations such as: "How would my localization algorithm perform with a device that performs half-way between a sonar and a laser?"; "What are the trade-offs between robot speed and battery life, or sensor update rates and resolution?"; or "What novel algorithms could exploit an ultra-wide-band radar that could detect walls *and* the objects behind them?". Robotics projects that are developing a new sensor can experiment with controllers in simulation before their hardware is ready.

Stage's modular architecture makes it easy to add entirely new models in order to explore less common ground: "What could I do if my robots could change color at will, or visually express some internal states to their colleagues, or quickly recognize and categorize each other (Vaughan, Støy, Sukhatme & Matarić 2000)?" Exploring the use of devices that are not currently feasible opens up a new field of study; robotics as it *could be*. Freedom from practical constraints distinguishes science from engineering; having a means to perform experiments distinguishes science from science fiction.

6.1.3.7 Challenges in scaling sensor-based simulation

Stage simulates multiple devices and scales to a few hundred devices, but is not currently useful for simulating massive populations, say on the scale of an ant colony, which could have a significant impact on some multi-robot research. It would be interesting to distribute Stage's computational load over a cluster of computers to support very large populations (tens of thousands) in real time. This is a non-trivial technical challenge that poses unsolved problems in representation and synchronization. Stage would also benefit

from advanced spatial representation to improve speed and memory efficiency in large and/or sparsely populated environments.

6.1.4 Usage

To date, software from the Player/Stage project has been downloaded over 3000 times. Player and Stage are currently in use in more than 20 major academic and industrial research labs around the world, and are also used in teaching undergraduate and graduate classes. Given the modest size of the target audience (i.e., robotics researchers and students), the project can be considered to be a significant success. An important factor in that success has been the Open Source model, which encourages inclusive, collaborative software development. As the developer team and user base have grown, major enhancements have been made to the software. Because of their modular designs, Player and Stage are easily extended by, for example, encapsulating a sophisticated control algorithm into the server or adding a model of an unavailable but interesting sensor to the simulator.

Since it is the collective experience of the users that drives development, I now briefly review a few projects in which Player and/or Stage have been used. One such project is concerned with robotic sensor networks (Sibley, Rahimi & Sukhatme 2002), which are characterized by extremely large collections of inexpensive mobile robots. Given the practical limitations on fielding even tens of physical robots, the ability to simulate hundreds or thousands of robots in Stage is invaluable to researchers wishing to test communication and coordination algorithms on large-scale systems. A similar project (Grocholsky, Makarenko & Durrant-Whyte 2003) aims to extend the scalability and modularity of the Decentralized Data Fusion (DDF) architecture to active sensor networks in which some or all of the network components have actuators. The practical implementation will be able to utilize the resources of a heterogeneous and dynamic team of sensing platforms to find and track stationary and moving features in an indoor environment. Player is used as a common hardware abstraction layer throughout the diverse set of software modules and Stage is used extensively for code validation and initial performance assessment.

Another project (Jung & Sukhatme 2002) studies resource allocation for target-tracking in sensor-actuator networks, using a region-based approach to control deployment of mobile robots. A multi-resolution task assignment architecture allows the system to handle significant environmental occlusion. Because the same Player interface is used with both physical and simulated devices, the tracking system that was developed in Stage was trivially transitioned to real robots. Resource allocation is also investigated from a learning perspective (Dahl et al. 2002), with the goal of developing general adaptive capabilities in robots and multi-robot systems. Focusing on spatio-temporal adaptivity, this project uses reinforcement learning to allow robots to dynamically adjust their behavior to any given environment while performing a set task. Of particular use in this project was Stage's ability to run simulation trials faster than real time, and thereby generate the substantial amount of data required to determine the run-time characteristics and performance impact of the learning system.

Player is also used in an investigation of novel multi-robot task allocation algorithms (Gerkey & Matarić 2002*c*), providing a unified interface to a group of heterogeneous robots. In this case, an economically-inspired task auctioning system is developed and validated on multi-robot teams, ranging in size from 3 to 7 robots, engaged in a variety of tasks. The resulting task allocation system, MURDOCH (see Chapter 4), is also used in a broader study of large-scale human-system interaction (Tews et al. 2002, Tews et al. 2003). The coordination infrastructure was originally developed and tested in Stage with a large group of simulated robots and was then validated on a smaller team of physical robots. This simulate-validate approach has also been successfully employed in many other projects, including recent work on multi-robot resource transportation (Vaughan, Støy, Howard, Sukhatme & Matarić 2002).

6.1.5 Related work

The distinguishing features of tools developed by the Player/Stage project are: (i) they seek to constrain the design of the controlling client program as little as possible; and (ii) they are efficiently implemented around a custom network server.

By minimizing constraints on the control program, Player and Stage offer a uniquely flexible robot development environment compared to others such as Saphira (Konolige 1997), Mission Lab (MacKenzie, Arkin & Cameron 1997), TeamBots (Balch 1998), Ayllu (Werger 2000), DCA (Petersson, Austin & Christensen 2001), ARIA (ActivMedia Robotics, Inc. 2002), and CARMEN (Thrun, Fox, Burgard & Dellaert 2001). As a tradeoff for providing support for a particular control or coordination philosophy, these systems all restrict the end-user's choice of programming language and/or structure. While such constraints can be useful in guiding the user in a particular paradigm, implementing the constraints at a low level is unsuitable for a general-purpose system; the programmer should have the choice to build any kind of control system while still enjoying device abstraction and encapsulation. Thus in Player there is a clear distinction between the *programming interface* and the *control structure*, with a preference for a highly general programming interface. Thus users are allowed to develop their own tools, including sophisticated architectures like those mentioned above.

Because it is designed explicitly for robotic device control, Player is efficient for this purpose; the primary limitation on its performance is currently the speed and fidelity and the underlying operating system's scheduler. By using on a custom protocol and server instead of adhering to a "generic" communications standard, such as CORBA (Object Management Group, Inc. 2002) or Jini (Waldo 1999), Player is free from the computational and programmatic overhead that is generally associated with the practical application of such a standard. Robot interfaces that do rely on these standards, such as Mobility (iRobot Corporation 2003), OROCOS (OROCOS 2003), and others (Burchard & Feddema 1997, Holness, Karuppiah, Uppala & Grupen 2001), benefit from readily-available client-side libraries that hide many of the communication details. However, as demonstrated by the proliferation of similar client-side libraries for Player (currently available in 6 languages), the custom message protocol is simple and easy to implement.

6.1.6 Acknowledgements

I thank my fellow maintainers of the Player/Stage project: Richard T. Vaughan and Andrew Howard. I also thank the many developers and users who have contributed so much to the success of the project,

Figure 6.2: *A typical Pioneer 2-DX robot. Visible in this image are the two drive wheels, front sonar ring, compass (small cube), Ethernet modem (antennae protruding vertically), and laser range-finder (looks like a coffee-maker). At the rear of the robot is another ring of sonars.*

especially: Maxim Batalin, Josh Bers, Brendan Burns, Jason Douglas, Jakob Fredslund, Kim Jinsuck, Boyoon Jung, Alex Makarenko, Andy Martignoni III, Nik Melchior, Dave Naffin, Esben Østergård, Gabe Sibley, Kasper Støy, John Sweeney and Doug Vail. Thanks also to SourceForge.net for project hosting.

6.2 Hardware

I use as my mobile experimental test-bed a team of ActivMedia Pioneer 2-DX robots (ActivMedia Robotics, Inc. 1999). The Pioneer 2-DX, shown in Figure 6.2, is a $44cm$ x $38cm$ x $22cm$ non-holonomic two-wheeled base that is differentially steered (a passive caster provides balance). This versatile and currently widely-used research robot can be configured with many different peripherals, including front and rear sonar arrays, compasses, pan-tilt-zoom cameras, and laser range-finders.

Each robot houses a Pentium-based single-board computer running Linux. Player executes on this embedded computer and handles low-level sensor and actuator control. Inter-robot communication for these robots is provided by way of wireless Ethernet, with an effective shared bandwidth of 1 – 11Mbps.

6.3 Summary

In this chapter, I have described the software and hardware infrastructure that has played an integral role in my (and others') experiments with physical and simulated robots. I have suggested how this infrastructure can be used to support empirical comparison of multi-robot coordination algorithms, which complements the analytical taxonomy presented in Chapter 5. In the next chapter, I conclude this dissertation with a discussion of the limits of the taxonomy and suggestions for how to extend it.

Chapter 7

Conclusion

In the field of mobile robotics, the study of multi-robot systems is now of paramount importance. Having solved some of the basic problems concerning single-robot control, many researchers have shifted their focus to the study of multi-robot coordination. There are by now a plethora of examples of demonstrated coordinated behavior in multi-robot systems, and just as many coordination architectures. However, despite more than a decade of research, the field lacks a theoretical foundation that can explain or predict the behavior of a multi-robot system. The goal of this dissertation has been to provide a candidate framework for studying such systems.

The word "coordination" is somewhat imprecise, and it has different meanings for different people.[1] In order to be exact about the problem with which I am concerned, I have chosen to carve out a smaller problem: multi-robot task allocation (MRTA). That is, given some robots and some tasks, which robot(s) should execute which task(s)? I suggest that this restricted problem is both theoretically and practically important, and my choice it is supported by the significant body of existing work that focuses on MRTA, in one form or another.

To date, the majority of research in MRTA has been *ad hoc* and experimental in nature. The standard procedure, which has been repeated numerous times, is to construct a novel MRTA architecture and then

[1] The words "coordination," "cooperation," and "collaboration" are often thrown about when discussing multi-robot systems, but their relationships are rarely established. Some researchers (e.g., Emery et al. (2002)) assert that each word represents a unique concept, while some dictionaries (e.g., Barnhart (1952)) define at least one of the three words in terms of the other two.

validate it in one or more application domains. This proof-of-concept method has led to the proposal of many MRTA architectures, each of which has been experimentally validated to a greater or lesser extent, sometimes in simulation and sometimes with physical robots. These research efforts are undeniably useful, as they each demonstrate that successful multi-robot coordination is indeed possible, even in relatively complex environments. However, can something more be said about the stack of existing MRTA architectures? Beyond providing numerous existence proofs, does the work to date allow for general conclusions to be made regarding the underlying problems? Is it possible to establish a prescriptive strategy that would dictate how to achieve task allocation in a given multi-robot system?

Robotics has for decades been an empirical venture, and I suggest that the time has come to undertake the formal analysis of multi-robot problems and their solutions, in order that we might one day (soon) answer the above questions in the affirmative. Importantly, I do *not* advocate a return to Good Old-Fashioned Artificial Intelligence (Russell & Norvig 1995), which held that all AI problems, including those in robotics, could be solved by building a world model and then reasoning with that model from first principles. Rather, I propose to follow in the footsteps of the natural sciences: accrue experimental evidence regarding some system or phenomenon until one can propose a plausible descriptive model. I believe that sufficient experimental evidence now exists within the robotics literature to begin proposing such models.

There are many ways in which one could build this kind of model. I view MRTA problems as fundamentally organizational in nature, in that the goal is to allocate limited resources in such a way as to efficiently achieve some task(s). Thus in this dissertation I have shown how MRTA problems can be studied in a formal manner by adapting for use in robotics some of the rich theory developed in various disciplines that study organizational and optimization problems. Throughout, I have borrowed techniques and results from operations research, economics, scheduling, network flows, and combinatorial optimization. Each of these disciplines provides its own perspective on organizational problems such as those encountered in MRTA domains. In Chapter 3, I illustrated these different perspectives by describing how

each discipline views the Optimal Assignment Problem (OAP), and showing how a relatively simple (and common) MRTA problem can be seen as an instance of the OAP.

Theory from economics and operations research suggests that one way to solve the OAP is with the use of a synthetic price-based market. As explained in Section 3.3, certain markets are known to produce optimal solutions to such problems. In Chapter 4, I showed that this prediction holds true for physical multi-robot systems by describing my own auction-based MRTA architecture, MURDOCH, which represents a tradeoff between optimality and efficiency, producing a greedy solution at low communication cost. It is common to justify such a use of market-based techniques for distributed coordination by reference to the efficacy of the free market as it used by humans. This view of the supremacy of free-market capitalism was perhaps voiced best by Hayek (1945):

> "If we can agree that the economic problem of society is mainly one of rapid adaptation to changes in the particular circumstances of time and place, it would seem to follow that the ultimate decisions must be left to the people who are familiar with these circumstances, who know directly of the relevant changes and of the resources immediately available to meet them."

Though not without a certain romantic appeal, the invocation of this justification for the success of market methods in synthetic systems (e.g., Dias & Stentz (2001), Gerkey & Matarić (2002*a*)) is far less satisfying than that provided by operations research: the OAP is a maximum problem for a linear program; any such problem can be transformed into a *dual* minimum problem; and the solution to such a dual is the same as the prices that result from the evolution of a certain price-based market system (see Section 3.2.3). In this light, the fact that market-based techniques result in efficient allocations of tasks is far from surprising. Analogous theoretical grounding should always be given when methods are borrowed from other fields, including physics, biology, and ethology. The mere fact that a particular technique appears to work well or accurately describe the behavior of a natural system is an insufficient justification for (and likely a shallow explanation of) the adaptation of that technique for use in multi-robot systems. The burden is on the researcher to clearly explain three things: which problem he or she is trying solve, which method is being used to solve the problem, and *why* the method works or does not work. Using the framework of the

OAP to study MRTA, I was able to make precise statements on these three issues, including exactly why market-oriented techniques should be expected to produce good solutions.

Of course, not all MRTA problems can be cast as assignment problems. For example, whenever multi-robot tasks are allowed, the OAP is an insufficient model, for it does not permit nonlinear combinations of individual utility values. Fortunately, the organizational disciplines have also studied many of these more complex problems that are encountered in robotics. In Chapter 5, I proposed a taxonomy of MRTA problems, based on criteria such as whether multi-robot tasks and/or multi-task robots are permitted. I showed where many common MRTA problems fall within this taxonomy and suggested how optimization algorithms can be adapted for use in the robotics domains. Furthermore, I analyzed several existing MRTA architectures, deriving important characteristics about their underlying algorithms, such as solution quality and communication overhead. This kind of analysis is rare, if not unique, within the field of robotics, and it offers the opportunity to objectively compare and evaluate proposed solutions to robotics problems. In particular, by stripping away the "story" in which each researcher wraps his or her creation, it is possible to appreciate the fundamental characteristics of a MRTA architecture, and even predict its performance in unseen domains.

When dealing with embodied systems such as teams of mobile robots, theoretical analysis is not the entire story. It is of paramount importance to verify analytical predictions through experimentation with multi-robot systems, either in simulation or physical hardware. Such experimentation requires significant supporting software infrastructure, the importance of which is often underestimated. In Chapter 6, I described the Player/Stage project, the goal of which is to produce high-quality, Open Source software to enable and facilitate experimental work with mobile robots. Software from the Player/Stage project has been downloaded over 3000 times and is used in more than 20 academic and industrial research labs around the world, as well as some classrooms. As the user base has grown and as the device server Player has come to control a wide array of different robot platforms and peripherals, the possibility has arisen that Player could become a standard, generic interface to robot systems. By defining the semantics and formats of allowable sensor / actuator interactions, Player constitutes a virtual machine that anyone (everyone?)

could target when writing a robot control program. As a result, it is conceivable that competing multi-robot coordination algorithms, each having been implemented to interact with Player, could be compared *directly* in experimental scenarios. The multi-robot simulator Stage could be exploited to significant advantage for the purpose of such evaluation, in that common environments and tasks could be defined (and enforced) by an always-on, long-lived "Stage server" that is modeled in part after modern real-time strategy (RTS) games. Users could connect to the server from anywhere in the world and test their favorite algorithms against others in a controlled environment. A key component of such a system would be the definition of system metrics within the simulator, so that it could automatically score implemented algorithms. For example, if the task is to forage for pucks, then reasonable metrics include: total pucks collected in a given period of time, time to collect a given percentage of pucks, and total robot travel distance. It is an open, domain-dependent question as to how to select such metrics.

7.1 Adding domain information

The MRTA formalism presented in this dissertation is very general, in that it relies only on domain-independent theory and techniques. Thus, for example, the taxonomy given in Chapter 5 should apply equally well in multi-agent and multi-robot systems. However, in exchange for such generality, this formalism is only capable of providing coarse characterizations of MRTA problems and their proposed solutions. Consider the analysis showing that MURDOCH, as an implementation of the canonical Greedy algorithm, is 3-competitive for the online assignment problem (see Section 5.2.2). This kind of competitive factor gives an algorithm's *worst-case* behavior, which may be quite different from its *average-case* behavior. In this respect, the bounds established for existing MRTA architectures, in terms of computational overhead, communication overhead, and solution quality, are relatively loose.

One way to tighten these bounds is to add domain-specific information to the formalism. By capturing and embedding models of how real MRTA domains behave and evolve over time, it should be possible to make more accurate predictions about an algorithm's performance. For example, while the classical theory

of the OAP makes no assumptions about the nature of the utility matrices that form the input, MRTA problems are likely to exhibit significant structure in their utility values. Far from randomly generated, utility values generally follow one of a few common models, determined primarily by the kind of sensor data that is used in estimating utility. If only "local" sensor information is used (e.g., can the robot currently see a particular target, and if so, how close is it?), then utility estimates tend to be strongly bifurcated (e.g., a robot will have very high utility for those targets that it can see, and zero utility for all others). On the other hand, if "global" sensor information is available (e.g., how close is the robot to a goal location?), then utility estimates tend to be smoother (e.g., utility will fall off smoothly in space away from the goal).

A promising avenue for future research would be to characterize this "utility landscape" as it is encountered in MRTA domains. It might then be possible to classify different MRTA problems according to the shapes of their landscapes, and make predictions about, for example, how well a greedy assignment algorithm should be expected to work, as opposed to a more costly optimal assignment algorithm. Anecdotal evidence suggests that for many MRTA domains, greedy and other heuristic algorithms produce solutions that are extremely close (or even equal) in quality the optimal solution; it would be interesting to make a more definite statement to this effect by leveraging relevant theory. Perhaps tools from the field of matrix analysis (Bhatia 1997), such as spectral radius or matrix entropy, could be used to derive the structure of utility matrices.

Furthermore, for all its elegance, organizational theory generally ignores implementation details that can prove crucial in embodied systems, such as limited communication bandwidth, communication failures (or worse, errors), and robot failures. Contingency mechanisms for these events must be present in any implemented MRTA system (e.g., consider the progress monitoring / contract renewal step in the MURDOCH auction protocol; Section 4.2.2), but they do not appear in the current formalism. As with any model, if important characteristics of the system are not accounted for, then predictions of overall performance can be arbitrarily bad. For those MRTA domains in which issues such as robot failures contribute significantly to system behavior, it may be possible to explicitly incorporate them into the formalism in a probabilistic fashion, as has been done for other problems in various ways, including Monte Carlo and Markovian

methods (e.g., Guestrin et al. (2001), Thrun et al. (2001), Howard, Matarić & Sukhatme (2002*b*), Roy & Gordon (2002), Goldberg & Matarić (2003)).

Finally, the formalism that I have presented in this dissertation is primarily *descriptive*, in that it enables *analysis* of problems and proposed solutions. A logical next step is to create a *prescriptive* model that contributes to *synthesis* of new solutions. Such a model would function as a tool for algorithmic design by suggesting to the researcher how to go about solving a given MRTA problem. Again, this extension would rely heavily on the judicious embedding of domain-specific characteristics, a task that is far from trivial.

Reference List

ActivMedia Robotics, Inc. (1999), *Pioneer 2 Mobile Robot - Operations Manual*. v4.

ActivMedia Robotics, Inc. (2002), *Aria Reference Manual*. 1.1.10.

Agassounon, W. & Martinoli, A. (2002), A Macroscopic Model of an Aggregation Experiment using Embodied Agents in Groups of Time-Varying Sizes, *in* 'Proc. of the IEEE Conf. on System, Man and Cybernetics (SMC)', Hammamet, Tunisia, pp. 250–255.

Aguilera, M. K., Strom, R. E., Sturman, D. C., Astley, M. & Chandra, T. D. (1999), Matching Events in a Content-based Subscription System, *in* 'Proc. of the ACM Symp. on Principles of Distributed Computing', Atlanta, Georgia, pp. 53–61.

Ahuja, R. K., Magnanti, T. L. & Orlin, J. B. (1993), *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, Upper Saddle River, New Jersey.

Arkin, R. C. (1987), Motor schema based navigation for a mobile robot: An approach to programming by behavior, *in* 'Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)', Raleigh, NC, pp. 264–271.

Arkin, R. C. (1998), *Behavior-Based Robotics*, MIT Press, Cambridge, Massachusetts.

Arkin, R. C., Balch, T. & Nitz, E. (1993), Communication of behavorial state in multi-agent retrieval tasks, *in* 'Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)', Atlanta, Georgia, pp. 588–594.

Asada, M., Kitano, H., Noda, I. & Veloso, M. (1999), 'RoboCup: Today and tomorrow – What we have have learned', *Artificial Intelligence* **110**, 193–214.

Atamtürk, A., Nemhauser, G. & Savelsbergh, M. (1995), 'A Combined Lagrangian, Linear Programming and Implication Heuristic for Large-Scale Set Partitioning Problems', *J. of Heuristics* **1**, 247–259.

Avis, D. (1983), 'A Survey of Heuristics for the Weighted Matching Problem', *Networks* **13**, 475–493.

Balas, E. & Padberg, M. W. (1972), 'On the Set-Covering Problem', *Operations Research* **20**(6), 1152–1161.

Balas, E. & Padberg, M. W. (1976), 'Set Partitioning: A Survey', *SIAM Review* **18**(4), 710–760.

Balch, T. (1998), Behavioral Diversity in Learning Robot Teams, PhD thesis, College of Computing, Georgia Institute of Technology.

Balinski, M. (1985), 'Signature Methods for the Assignment Problem', *Operations Research* **33**(3), 527–536.

Balinski, M. (1986), 'A competitive (dual) simplex method for the assignment problem', *Mathematical Programming* **34**, 125–141.

Bar-Yehuda, R. & Even, S. (1981), 'A linear-time approximation algorithm for the weighted vertex cover problem', *J. of Algorithms* **2**, 198–203.

Barnhart, C. L., ed. (1952), *The American College Dictionary*, Random House, New York.

Bertsekas, D. P. (1990), 'The Auction Algorithm for Assignment and Other Network Flow Problems: A Tutorial', *Interfaces* **20**(4), 133–149.

Bertsekas, D. P. & Castañon, D. A. (1991), 'Parallel synchronous and asynchronous implementations of the auction algorithm', *Parallel Computing* **17**, 707–732.

Bhatia, R. (1997), *Matrix Analysis*, Springer-Verlag, New York.

Błażewicz, J., Dell'Olmo, P., Drozdowski, M. & Speranza, M. (1992), 'Scheduling multiprocessor tasks on three dedicated processors', *Information Processing Letters* **41**, 275–280. Corrigendum (1994): *Information Processing Letters* **49**, 269–270.

Borgwardt, K.-H. (1982), 'Some distribution-independent results about the asymptotic order of the average number of pivot steps of the simplex algorithm', *Mathematics of Operations Research* **7**, 441–461.

Botelho, S. C. & Alami, R. (1999), M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement, *in* 'Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)', Detroit, Michigan, pp. 1234–1239.

Botelho, S. C. & Alami, R. (2000), A multi-robot cooperative task achievement system, *in* 'Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)', San Francisco, California, pp. 2716–2721.

Botelho, S. C. & Alami, R. (2001), Multi-robot cooperation through the common use of "mechanisms", *in* 'Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)', Wailea, Hawaii, pp. 375–380.

Bratman, M., Israel, D. & Pollack, M. (1988), 'Plans and resource-bounded practical reasoning', *Computational Intelligence* **4**, 349–355.

Brooks, R. A. (1986), 'A robust layered control system for a mobile robot', *IEEE J. of Robotics and Automation* **2**(1), 14–23.

Brooks, R. A. (1991), Intelligence Without Reason, *in* 'Proc. of the Intl. Joint Conf. on Artificial Intelligence (IJCAI)', Sydney, Australia, pp. 569–595.

Brucker, P. (1998), *Scheduling Algorithms*, 2^{nd} edn, Springer-Verlag, Berlin.

Bruno, J. L., Coffman, E. G. & Sethi, R. (1974), 'Scheduling Independent Tasks To Reduce Mean Finishing Time', *Communications of the ACM* **17**(7), 382–387.

Brusey, J., Makies, M., Padgham, L., Woodvine, B. & Fantone, K. (2001), RMIT United, *in* P. Stone, T. Balch & G. Kraetzschmar, eds, 'RoboCup 2000, LNAI 2019', Springer-Verlag, Berlin, pp. 563–566.

Burchard, R. L. & Feddema, J. T. (1997), Generic robotic and motion control API based on GISC-Kit technology and CORBA communications, *in* 'Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)', Minneapolis, Minnesota, pp. 712–717.

Burgard, W., Moors, M., Fox, D., Simmons, R. & Thrun, S. (2000), Collaborative Multi-Robot Exploration, *in* 'Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)', San Francisco, California, pp. 476–481.

Cai, X., Lee, C.-Y. & Li, C.-L. (1998), 'Minimizing Total Completion Time in Two-Processor Task Systems with Prespecified Processor Allocations', *Naval Research Logistics* **45**(2), 231–242.

Castelpietra, C., Iocchi, L., Nardi, D., Piaggio, M., Scalzo, A. & Sgorbissa, A. (2001), Communication and Coordination among heterogeneous Mid-size players: ART99, *in* P. Stone, T. Balch & G. Kraetzschmar, eds, 'RoboCup 2000, LNAI 2019', Springer-Verlag, Berlin, pp. 86–95.

Chaimowicz, L., Campos, M. F. M. & Kumar, V. (2002), Dynamic Role Assignment for Cooperative Robots, *in* 'Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)', Washington, DC, pp. 293–298.

Chavez, A., Moukas, A. & Maes, P. (1997), *Challenger*: A Multi-agent System for Distributed Resource Allocation, *in* 'Proc. of Autonomous Agents', Marina del Rey, CA, pp. 323–331.

Chien, S., Barrett, A., Estlin, T. & Rabideau, G. (2000), A comparison of coordinated planning methods for cooperating rovers, *in* 'Proc. of Autonomous Agents', Barcelona, Spain, pp. 100–101.

Chu, P. & Beasley, J. (1995), A Genetic Algorithm for the Set Partitioning Problem, Technical report, The Management School, Imperial College, London.

Chvátal, V. (1979), 'A greedy heuristic for the set cover problem', *Mathematics of Operations Research* **4**, 233–235.

Cicirello, V. A. & Smith, S. F. (2001), Wasp-like Agents for Distributed Factory Coordination, Technical Report CMU-RI-TR-01-39, The Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania.

Clark, D. D. (1988), 'The design philosophy of the DARPA Internet protocols', *Computer Communication Review* **18**(4), 106–114.

Clearwater, S. H., Huberman, B. A. & Hogg, T. (1991), 'Cooperative Solution of Constraint Satisfaction Problems', *Science* **254**, 1181–1183.

Cohen, P. R. & Levesque, H. J. (1991), 'Teamwork', *Nous* **25**, 487–512.

Cormen, T. H., Leiserson, C. E. & Rivest, R. L. (1997), *Introduction to Algorithms*, MIT Press, Cambridge, Massachusetts.

Crawford, V. P. & Knoer, E. M. (1981), 'Job Matching with Heterogeneous Firms and Workers', *Econometrica* **49**(2), 437–450.

Dahl, T. S., Matarić, M. J. & Sukhatme, G. S. (2002), Adaptive spatio-temporal organization in groups of robots, *in* 'Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)', Lausanne, Switzerland, pp. 1044–1049.

Dantzig, G. B. (1951), Maximization fo a linear function of variables subject to linear inequalities, *in* T. C. Koopmans, ed., 'Activity Analysis of Production and Allocation', Wiley, New York, pp. 339–347.

Dantzig, G. B. (1963), *Linear Programming and Extensions*, Princeton University Press, Princeton, New Jersey.

Dantzig, G. B. (1982), 'Reminiscences about the origins of linear programming', *Operations Research Letters* **1**(2), 43–48.

Dantzig, G. B., Orden, A. & Wolfe, P. (1955), 'The generalized simplex method for minimizing a linear form under linear inequality restraints', *Pacific J. of Mathematics* **5**, 183–195.

Davis, R. & Smith, R. G. (1983), 'Negotiation as a Metaphor for Distributed Problem Solving', *Artificial Intelligence* **20**(1), 63–109.

Demange, G., Gale, D. & Sotomayor, M. (1986), 'Multi-Item Auctions', *The J. of Political Economy* **94**(4), 863–872.

Deneubourg, J.-L., Theraulaz, G. & Beckers, R. (1991), Swarm-made architectures, *in* 'Proc. of the European. Conf. on Artificial Life (ECAL)', Paris, France, pp. 123–133.

Dertouzos, M. L. & Mok, A. K. (1983), 'Multiprocessor On-Line Scheduling of Hard-Real-Time Tasks', *IEEE Transactions on Software Engineering* **15**(12), 1497–1506.

Dias, M. B. & Stentz, A. (2000), A Free Market Architecture for Distributed Control of a Multirobot System, *in* 'Proc. of the Intl. Conf. on Intelligent Autonomous Systems (IAS)', Venice, Italy, pp. 115–122.

Dias, M. B. & Stentz, A. (2001), A Market Approach to Multirobot Coordination, Technical Report CMU-RI-TR-01-26, The Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania.

Dias, M. B. & Stentz, A. (2002), Opportunistic Optimization for Market-Based Multirobot Control, *in* 'Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)', Lausanne, Switzerland, pp. 2714–2720.

d'Inverno, M., Luck, M. & Wooldridge, M. (1997), Cooperation Structures, *in* 'Proc. of the Intl. Joint Conf. on Artificial Intelligence (IJCAI)', Nagoya, Japan, pp. 600–605.

Donald, B., Jennings, J. & Rus, D. (1997*a*), 'Information invariants for distributed manipulation', *The Intl. J. of Robotics Research* **16**(5), 673–702.

Donald, B., Jennings, J. & Rus, D. (1997*b*), 'Minimalism + distribution = supermodularity', *J. of Experimental and Theoretical Artifical Intelligence* **9**(2–3), 293–321.

Dorfman, R. (1984), 'The Discovery of Linear Programming', *Annals of the History of Computing* **6**(3), 283–295.

Dwyer, P. S. (1954), 'Solution of the Personnel Classification Problem with the Method of Optimal Regions', *Psychometrika* **18**(1), 11–26.

Edgeworth, F. Y. (1881), 'Mathematical Psychics: An Essay on the Application of Mathematics to the Moral Sciences'. Translated and reprinted, Augustus M. Kelley, New York, 1967.

Edmonds, J. (1971), 'Matroids and the greedy algorithm', *Mathematical Programming* **1**, 127–136.

Emery, R. & Balch, T. (2001), Behavior-based control of a non-holonomic robot in pushing tasks, *in* 'Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)', Seoul, South Korea, pp. 2381–2388.

Emery, R., Sikorski, K. & Balch, T. (2002), Protocols for Collaboration, Coordination, and Dynamic Role Assignment in a Robot Team, *in* 'Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)', Washington, DC, pp. 3008–3015.

Engquist, M. (1982), 'A successive shortest path algorithm for the assignment problem', *INFOR* **4**, 370–384.

Erol, K., Hendler, J. & Nau, D. S. (1994), UCMP: A sound and complete procedure for Hierarchical Task-Network planning, *in* 'Proc. of the Intl. Conf. on Artificial Intelligence Planning Systems', Chicago, IL, pp. 249–254.

Feiertag, R. & Organick, E. (1971), The Multics input/output systems, *in* 'Proc. of the Symp. on Operating Systems Principles', New York, pp. 35–41.

Fisher, M. L. & Kedia, P. (1990), 'Optimal solution of set covering/partitioning problems using dual heuristics', *Management Science* **36**(6), 674–688.

Frank, A. (1981), 'A weighted matroid intersection algorithm', *J. of Algorithms* **2**, 328–336.

Free Software Foundation (1991), *GNU General Public License*. Version 2.

Fukuda, T., Nakagawa, S., Kawauchi, Y. & Buss, M. (1988), Self organizing robots based on cell structures – CEBOT, *in* 'Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)', IEEE Computer Society Press, pp. 145–150.

Gale, D. (1960), *The Theory of Linear Economic Models*, McGraw-Hill Book Company, Inc., New York.

Gale, D., Kuhn, H. & Tucker, A. (1951), Linear programming and the theory of games, *in* T. C. Koopmans, ed., 'Activity Analysis of Production and Allocation', Wiley, New York, pp. 317–329.

Gale, D. & Shapley, L. S. (1962), 'College Admissions and the Stability of Marriage', *American Mathematical Monthly* **69**(1), 9–15.

Galstyan, A. & Lerman, K. (2002), 'Adaptive Boolean networks and minority games with time-dependent capacities', *Physical Review E* **66**(015103).

Garey, M. R. & Johnson, D. S. (1978), '"Strong" NP-Completeness Results: Motivation, Examples, and Implications', *J. of the ACM* **25**(3), 499–508.

Georgeff, M. & Lansky, A. (1987), Reactive reasoning and planning, *in* 'Proc. of the Natl. Conf. on Artificial Intelligence (AAAI)', Seattle, Washington, pp. 677–682.

Georgeff, M., Pollack, B. P. M., Tambe, M. & Wooldridge, M. (1999), The Belief-Desire-Intention Model of Agency, *in* J. Müller, M. Singh & A. Rao, eds, 'Intelligent Agents V: Agent Theories, Architectures, and Languages, LNAI 1555', Springer-Verlag, Berlin, pp. 1–10.

Gerkey, B. P. (1998), 'Task Allocation for Heterogeneous Robots'. Undergraduate honors thesis, Department of Electrical Engineering and Computer Science, Tulane University.

Gerkey, B. P. & Matarić, M. J. (2000), Murdoch: Publish/Subscribe Task Allocation for Heterogeneous Agents, *in* 'Proc. of Autonomous Agents', Barcelona, Spain, pp. 203–204.

Gerkey, B. P. & Matarić, M. J. (2001), Principled communication for dynamic multi-robot task allocation, *in* D. Rus & S. Singh, eds, 'Experimental Robotics VII, LNCIS 271', Springer-Verlag, Berlin, pp. 353–362.

Gerkey, B. P. & Matarić, M. J. (2002*a*), A market-based formulation of sensor-actuator network coordination, *in* 'Proc. of the AAAI Spring Symp. on Intelligent Embedded and Distributed Systems', Palo Alto, California, pp. 21–26.

Gerkey, B. P. & Matarić, M. J. (2002*b*), Pusher-watcher: An approach to fault-tolerant tightly-coupled robot coordination, *in* 'Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)', Washington D.C., pp. 464–469.

Gerkey, B. P. & Matarić, M. J. (2002*c*), 'Sold!: Auction methods for multi-robot coordination', *IEEE Transactions on Robotics and Automation* **18**(5), 758–768.

Gerkey, B. P. & Matarić, M. J. (2003*a*), A Framework for Studying Multi-Robot Task Allocation, *in* A. Schultz et al., eds, 'Multi-Robot Systems: From Swarms to Intelligent Automata, Volume II', Kluwer Academic Publishers, the Netherlands, pp. 15–26.

Gerkey, B. P. & Matarić, M. J. (2003*b*), Multi-Robot Task Allocation: Analyzing the Complexity and Optimality of Key Architectures, *in* 'Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)', Taipei, Taiwan. To appear.

Gerkey, B. P., Matarić, M. J. & Sukhatme, G. S. (2002), Exploiting physical dynamics for concurrent control of a mobile robot, *in* 'Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)', Washington D.C., pp. 3467–3472.

Gerkey, B. P., Støy, K. & Vaughan, R. T. (2000), Player robot server, Technical Report IRIS-00-392, Institute for Robotics and Intelligent Systems, School of Engineering, University of Southern California, Los Angeles, California.

Gerkey, B. P., Vaughan, R. T. & Howard, A. (2003), The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems, *in* 'Proc. of the Intl. Conf. on Advanced Robotics (ICAR)', Coimbra, Portugal. To appear.

Gerkey, B. P., Vaughan, R. T., Støy, K., Howard, A., Sukhtame, G. S. & Matarić, M. J. (2001), Most Valuable Player: A Robot Device Server for Distributed Control, *in* 'Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)', Wailea, Hawaii, pp. 1226–1231.

Goldbarb, D. (1985), 'Efficient dual simplex algorithms for the assignment problem', *Mathematical Programming* **33**, 187–203.

Goldberg, D. & Matarić, M. J. (2003), 'Maximizing Reward in a Non-Stationary Mobile Robot Environment', *Autonomous Agents and Multi Agent Systems* **6**(3), 287–316.

Golfarelli, M., Maio, D. & Rizzi, S. (1997), A Task-Swap Negotiation Protocol Based on the Contract Net Paradigm, Technical Report 005-97, DEIS, CSITE - Universitá di Bologna.

Grocholsky, B., Makarenko, A. & Durrant-Whyte, H. F. (2003), Information-Theoretic Coordinated Control of Multiple Sensor Platforms, *in* 'Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)', Taipei, Taiwan.

Grosz, B. & Kraus, S. (1993), Collaborative Plans for Group Activities, *in* 'Proc. of the Intl. Joint Conf. on Artificial Intelligence (IJCAI)', Chambery, France, pp. 367–373.

Guestrin, C., Koller, D. & Parr, R. (2001), Multiagent Planning with Factored MDPs, *in* 'Proc. of Advances in Neural Information Processing Systems (NIPS)', Vancouver, Canada, pp. 1523–1530.

Hayek, F. A. (1945), 'The Use of Knowledge in Society', *The American Economic Review* **35**(4), 519–530.

Hoffman, K. L. & Padberg, M. W. (1993), 'Solving Airline Crew Scheduling Problems by Branch-and-Cut', *Management Science* **39**(6), 657–682.

Holness, G., Karuppiah, D., Uppala, S. & Grupen, R. (2001), A Service Paradigm for Reconfigurable Agents, *in* 'Proc. of the 2^{nd} Intl. Workshop on Infrastructure for Agents, MAS, and Scalable MAS at Autonomous Agents 2001', Montreal, Canada.

Hoogeveen, J., van del Velde, S. & Veltman, B. (1994), 'Complexity of scheduling multiprocessor tasks with prespecified processor allocations', *Discrete Applied Mathematics* **55**, 259–272.

Howard, A., Matarić, M. J. & Sukhatme, G. S. (2002*a*), 'An Incremental Self-Deployment Algorithm for Mobile Sensor Networks', *Autonomous Robots* **13**(2), 113–126.

Howard, A., Matarić, M. J. & Sukhatme, G. S. (2002*b*), Localization for Mobile Robot Teams Using Maximum Likelihood Estimation, *in* 'Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)', Lausanne, Switzerland, pp. 434–459.

Howard, A., Matarić, M. J. & Sukhatme, G. S. (2003), Putting the 'I' in 'Team': An Ego-Centric Approach to Cooperative Localization, *in* 'Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)', Taipei, Taiwan.

Hung, M. S. (1983), 'A Polynomial Simplex Method for the Assignment Problem', *Operations Research* **31**(3), 595–600.

iRobot Corporation (2003), *Mobility Software*. www.irobot.com.

Jakobi, N. (1997), 'Evolutionary robotics and the radical envelope of noise hypothesis', *Adaptive Behavior* **6**(2), 325–368.

Jennings, J. S. (1998), Threaded servers enable thin robot clients, Technical report, Electrical Engineering and Computer Science Dept., Tulane University, New Orleans, Louisiana.

Jennings, J. S. & Kirkwood-Watts, C. (1998), Distributed Mobile Robotics by the Method of Dynamic Teams, *in* 'Proc. of the Intl. Symp. on Distributed Autonomous Robotic Systems (DARS)', Karlsruhe, Germany.

Jennings, N. (1995), 'Controlling cooperative problem solving in industrial multi-agent systems using joint intentions', *Artificial Intelligence* **75**, 195–240.

Johnson, D. S. (1974), 'Approximation algorithms for combinatorial problems', *J. of Computer and System Sciences* **9**, 256–278.

Jung, B. & Sukhatme, G. S. (2002), 'Tracking Targets using Multiple Robots: The Effect of Environment Occlusion', *Autonomous Robots* **13**(3), 191–205.

Jung, H. & Tambe, M. (2003), Performance models for large scale multiagent systems: Using distributed pomdp building blocks, *in* 'Proc. of Intl. Conf. on Autonomous Agents and Multi Agent Systems', Melbourne, Australia.

Jung, H., Tambe, M. & Kulkarni, S. (2001), Argumentation as distributed constraint satisfaction: applications and results, *in* 'Proc. of Autonomous Agents', Montreal, Canada, pp. 324–331.

Jungnickel, D. (1999), *Graphs, Networks, and Algorithms*, Springer-Verlag, Berlin.

Kalyanasundaram, B. & Pruhs, K. (1993), 'Online Weighted Matching', *J. of Algorithms* **14**, 478–488.

Ketchpel, S. (1994), Forming Coalitions in the Face of Uncertain Rewards, *in* 'Proc. of the Natl. Conf. on Artificial Intelligence (AAAI)', Seattle, WA, pp. 414–419.

Khuller, S., Mitchell, S. G. & Vazirani, V. V. (1994), 'On-line algorithms for weighted bipartite matching and stable marriages', *Theoretical Computer Science* **127**, 255–267.

Klavins, E. (2002*a*), Automatic synthesis of controllers for distributed assembly and formation forming, *in* 'Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)', Washington, DC, pp. 3296–3302.

Klavins, E. (2002*b*), Communication Complexity of Multi-Robot Systems, *in* 'Proc. of the Intl. Workshop on the Algorithmic Foundations of Robotics (WAFR)', Nice, France. To appear.

Klee, V. & Minty, G. (1972), How good is the simplex algorithm?, *in* O. Shisha, ed., 'Inequalities, III', Academic Press, New York, pp. 159–175.

Konolige, K. (1997), COLBERT: A Language for Reactive Control in Saphira, *in* 'Proc. of the German Conf. on Artificial Intellgence', Freiburg, Germany, pp. 31–52.

Korte, B. & Vygen, J. (2000), *Combinatorial Optimization: Theory and Algorithms*, Springer-Verlag, Berlin.

Kraus, S., Wilkenfeld, J. & Zlotkin, G. (1995), 'Multiagent negotiation under time constraints', *Artificial Intelligence* **75**(2), 297–345.

Kube, C. R. & Zhang, H. (1992), Collective robotic intelligence, *in* 'Proc. of the Intl. Conf. on Simulation of Adaptive Behavior (SAB)', pp. 460–468.

Kube, C. R. & Zhang, H. (1996), The use of perceptual cues in multi-robot box-pushing, *in* 'Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)', Minneapolis, Minnesota, pp. 2085–2090.

Kuhn, H. W. (1955), 'The Hungarian Method for the Assignment Problem', *Naval Research Logistics Quarterly* **2**(1), 83–97.

Kuhn, H. W. (1956), 'Variants of the Hungarian Method for Assignment Problems', *Naval Research Logistics Quarterly* **3**, 253–258.

Kushilevitz, E. & Nisan, N. (1997), *Communication Complexity*, Cambridge University Press, Cambridge.

Lerman, K. & Galstyan, A. (2002), 'Mathematical Model of Foraging in a Group of Robots: Effect of Interference', *Autonomous Robots* **13**(2), 127–141.

Lerman, K. & Shehory, O. (2000), Coalition Formation for Large-Scale Electronic Markets, *in* 'Proc. of the Intl. Conf. on Multi Agent Systems (ICMAS)', Boston, pp. 167–174.

Litzkow, M. J., Livny, M. & Mutka, M. W. (1988), Condor - A hunter of idle workstations, *in* 'Proc. of the Intl. Conf. on Distributed Computing Systems', Washington, DC, pp. 104–111.

Lovász, L. (1975), 'On the ratio of optimal integral and fractional covers', *Discrete Mathematics* **13**, 383–390.

Lynch, K. M. & Mason, M. T. (1995), Controllability of pushing, *in* 'Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)', Nagoya, Japan, pp. 112–119.

MacKenzie, D. C., Arkin, R. & Cameron, J. M. (1997), 'Multiagent Mission Specification and Execution', *Autonomous Robots* **4**(1), 29–52.

Makarenko, A., Williams, S., Bourgault, F. & Durrant-Whyte, H. F. (2002), An Experiment in Integrated Exploration, *in* 'Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)', Lausanne, Switzerland.

Marsten, R. E. & Shepardson, F. (1981), 'Exact Solution of Crew Scheduling Problems Using the Set Partitioning Model: Recent Successful Applications', *Networks* **11**, 165–177.

Martin, D. L., Cheyer, A. J. & Moran, D. B. (1999), 'The open agent architecture: A framework for building distributed software systems', *Applied Artificial Intelligence* **13**(1), 91–128.

Matarić, M. J. (1992), Designing Emergent Behaviors: From Local Interactions to Collective Intelligence, *in* J.-A. Meyer, H. Roitblat & S. Wilson, eds, 'From Animals to Animats 2, Second International Conference on Simulation of Adaptive Behavior (SAB-92)', MIT Press, pp. 432–441.

Matarić, M. J. (1995), 'Designing and Understanding Adaptive Group Behavior', *Adaptive Behavior* **4**(1), 51–80.

Matarić, M. J. (1997), 'Behavior-based control: Examples from navigation, learning, and group behavior', *J. of Experimental and Theoretical Artifical Intelligence* **9**(2–3), 323–336.

Matarić, M. J., Nilsson, M. & Simsarian, K. T. (1995), Cooperative multi-robot box-pushing, *in* 'Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)', Pittsburgh, Pennsylvania, pp. 556–561.

McAfee, R. P. & McMillan, J. (1987), 'Auctions and Bidding', *J. of Economic Literature* **25**(2), 699–738.

Milch, B. & Koller, D. (2000), Probabilistic models for agents' beliefs and decisions, *in* 'Proc. of the Conference on Uncertainty in AI (UAI)', Stanford, California, pp. 389–396.

Modi, J., Jung, H., Tambe, M., Shen, W.-M. & Kulkarni, S. (2001), A Dynamic Distributed Constraint Satisfaction Approach to Resource Allocation, *in* 'Proc. of the Intl. Conf. on Principles and Practices of Constraint Programming', Paphos, Cyprus.

Modi, P. J., Shen, W.-M., Tambe, M. & Yokoo, M. (2003), An Asynchronous Complete Method for Distributed Constraint Optimization, *in* 'Proc. of Intl. Conf. on Autonomous Agents and Multi Agent Systems', Melbourne, Australia.

Nair, R., Tambe, M. & Marsella, S. (2003), Role Allocation and Reallocation in Multiagent Teams: Towards a Practical Analysis, *in* 'Proc. of Intl. Conf. on Autonomous Agents and Multi Agent Systems', Melbourne, Australia.

Noreils, F. R. (1993), 'Toward a robot architecture integrating cooperation between mobile robots: Application to indoor environment', *The Intl. J. of Robotics Research* **12**(1), 79–98.

Nouyan, S. (2002), Agent-Based Approach to Dynamic Task Allocation, *in* M. Dorigo et al., eds, 'ANTS 2002, LNCS 2463', Springer-Verlag, Berlin, pp. 28–39.

Object Management Group, Inc. (2002), *The Common Object Request Broker: Architecture and Specification, Version 3.0*.

Orlin, J. B. & Lee, Y. (1993), 'QuickMatch: A Very Fast Algorithm for the Assignment Problem'. Working Paper #3547-93, MIT Sloan School of Management, Cambridge, Massachusetts.

OROCOS (2003), 'The OROCOS project'. http://www.orocos.org.

Ortiz, C. L., Hsu, E., desJardins, M., Rauenbusch, T., Grosz, B., Yadgar, O. & Kraus, S. (2001), Incremental Negotiation and Coalition Formation for Resource-bounded Agents: Preliminary report, *in* 'AAAI Fall Symp. on Negotiation Methods for Autonomous Cooperative Systems', Falmouth, Massachusetts.

Østergård, E. H., Matarić, M. J. & Sukhatme, G. S. (2001), Distributed multi-robot task allocation for emergency handling, *in* 'Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)', Wailea, Hawaii, pp. 821–826.

Parker, L. E. (1994*a*), ALLIANCE: An Architecture for Fault Tolerant, Cooperative Control of Heterogeneous Mobile Robots, *in* 'Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)', Munich, Germany, pp. 776–783.

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

Parker, L. E. (1995), L-ALLIANCE: A Mechanism for Adaptive Action Selection in Heterogeneous Multi-Robot Teams, Technical Report ORNL/TM-13000, Oak Ridge National Laboratory.

Parker, L. E. (1997), 'L-ALLIANCE: Task-Oriented Multi-Robot Learning in Behavior-Based Systems', *Advanced Robotics* **11**(4), 305–322.

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

Parker, L. E. (1999*a*), A case study for life-long learning and adaptation in cooperative robot teams, *in* 'Proc. of SPIE Sensor Fusion and Decentralized Control in Robotic Systems II', Vol. 3839, pp. 92–101.

Parker, L. E. (1999*b*), 'Cooperative Robotics for Multi-Target Observation', *Intelligent Automation and Soft Computing* **5**(1), 5–19.

Pearce, D. W., ed. (1999), *The MIT Dictionary of Modern Economics*, 4^{th} edn, The MIT Press, Cambridge, Massachusetts.

Pereira, J., Fabret, F., Llirbat, F. & Shasha, D. (2000), Efficient matching for web-based publish/subscribe systems, *in* 'Proc. of the Intl. Conf. on Cooperative Information Systems', Eiat, Israel, pp. 162–173.

Petersson, L., Austin, D. & Christensen, H. (2001), DCA: A Distributed Control Architecture for Robotics, *in* 'Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)', Wailea, Hawaii, pp. 2361–2368.

Pynadath, D. V. & Tambe, M. (2002), 'The Communicative Multiagent Team Decision Problem: Analyzing Teamwork Theories and Models', *J. of Artificial Intelligence Research* **16**, 389–423.

Rado, R. (1957), 'Note on independence functions', *Proc. of the London Mathematical Society* **7**, 300–320.

Ramos, C. (1996), A Holonic Approach for Task Scheduling in Manufacturing Systems, *in* 'Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)', Minneapolis, Minnesota, pp. 2511–2516.

Ritchie, D. M. & Thompson, K. (1974), 'The UNIX Time-Sharing System', *Communications of the ACM* **17**(7), 365–375.

Roy, N. & Gordon, G. (2002), Exponential Family PCA for Belief Compression in POMDPs, *in* 'Proc. of Advances in Neural Information Processing Systems (NIPS)', Vancouver, Canada.

Russell, S. & Norvig, P. (1995), *Artificial Intelligence: A Modern Approach*, Prentice Hall, Upper Saddle River, New Jersey.

Sandholm, T. (1993), An Implementation of the Contract Net Protocol Based on Marginal Cost Calculations, *in* 'Proc. of the Natl. Conf. on Artificial Intelligence (AAAI)', Washington, DC, pp. 256–262.

Sandholm, T., Larson, K., Andersson, M., Shehory, O. & Tohmé, F. (1999), 'Coalition structure generation with worst-case guarantees', *Artificial Intelligence* **111**(1–2), 209–238.

Sandholm, T. W. & Lesser, V. R. (1997), 'Coalitions among computationally bounded agents', *Artificial Intelligence* **94**(1), 99–137.

Scerri, P., Modi, J., Tambe, M. & Shen, W.-M. (2003), Are multiagent algorithms relevant for real hardware? A case study of distributed constraint algorithms, *in* 'Proc. of the ACM Symp. on Applied Computing (SAC)', Melbourne, Florida. To appear.

Schrijver, A. (1986), *Theory of Linear and Integer Programming*, John Wiley & Sons, Chichester, England.

Shapely, L. S. & Shubik, M. (1972), 'The Assignment Game I: The Core', *Intl. J. of Game Theory* **1**(2), 111–130.

Shehory, O. & Kraus, S. (1996), Formation of overlapping coalitions for precedence-ordered task-execution among autonomous agents, *in* 'Proc. of the Intl. Conf. on Multi Agent Systems (ICMAS)', Kyoto, Japan, pp. 330–337.

Shehory, O. & Kraus, S. (1998), 'Methods for task allocation via agent coalition formation', *Artificial Intelligence* **101**(1–2), 165–200.

Sibley, G. T., Rahimi, M. H. & Sukhatme, G. S. (2002), Robomote: A Tiny Mobile Robot Platform for Large-Scale Ad-hoc Sensor Networks, *in* 'Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)', Washington D.C., pp. 1143–1148.

Simon, H. A. (2001), *The Sciences of the Artificial*, 3^{rd} edn, MIT Press, Cambridge, Massachusetts.

Singh, S., Jaakkola, T. & Jordan, M. (1994), Learning Without State-Estimation in Partially Observable Markovian Decision Processes, *in* 'Proc. of the Intl. Conf. on Machine Learning', pp. 284–292.

Sleator, D. D. & Tarjan, R. E. (1985), 'Amortized Efficiency of List Update and Paging Rules', *Communications of the ACM* **28**(2), 202–208.

Smith, R. G. (1980), 'The Contract Net Protocol', *IEEE Tranactions on Computers* **29**(12), 1104–1113.

Sondik, E. (1978), 'The optimal control of partially observable markov processes over the infinite horizon: discounted case', *Operations Research* **26**, 282–304.

Spielman, D. A. & Teng, S.-H. (2001), Smoothed analysis: Why the simplex algorithm usually takes polynomial time, *in* 'Proc. of the ACM Symp. on Theory of Computing', Hersonissos, Greece, pp. 296–305.

Spletzer, J. R. & Taylor, C. J. (2001), A Framework for Sensor Planning and Control with Applications to Vision Guided Multi-robot Systems, *in* 'Proc. of Computer Vision and Pattern Recognition Conf. (CVPR)', Kauai, Hawaii, pp. 378–383.

Stentz, A. & Dias, M. B. (1999), A free market architecture for coordinating multiple robots, Technical Report CMU-RI-TR-99-42, The Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania.

Stone, P. & Veloso, M. (1999), 'Task Decomposition, Dynamic Role Assignment, and Low-Bandwidth Communication for Real-Time Strategic Teamwork', *Artificial Intelligence* **110**(2), 241–273.

Sycara, K., Decker, K., Pannu, A., Williamson, M. et al. (1996), 'Distributed intelligent agents', *IEEE Expert* **11**(6), 36–46.

Tambe, M. (1997), Agent architectures for flexible, practical teamwork, *in* 'Proc. of the Natl. Conf. on Artificial Intelligence (AAAI)', Providence, Rhode Island, pp. 22–28.

Tambe, M. & Zhang, W. (2000), 'Towards Flexible Teamwork in Persistent Teams', *Autonomous Agents and Multi Agent Systems* **3**, 159–183.

Tews, A., Matarić, M. J. & Sukhatme, G. S. (2003), A Scalable Approach to Human-Robot Interaction, *in* 'Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)', Taipei, Taiwan.

Tews, A., Matarić, M. J., Sukhatme, G. S. & Gerkey, B. P. (2002), G'day Mate. Let me Introduce you to Everyone: An Infrastructure for Scalable Human-System Interaction, Technical Report CRES-02-004, Center for Robotics and Embedded Systems, School of Engineering, University of Southern California.

Thayer, S., Digney, B., Dias, M. B., Stentz, A. T. et al. (2000), Distributed Robotic Mapping of Extreme Environments, *in* 'Proc. of SPIE Vol. 4195: Mobile Robots XV and Telemanipulator and Telepresence Technologies VII', Vol. 4195.

Thorndike, R. L. (1950), 'The Problem of Classification of Personnel', *Psychometrika* **15**(3), 215–235.

Thrun, S., Fox, D., Burgard, W. & Dellaert, F. (2001), 'Robust Monte Carlo Localization for Mobile Robots', *Artificial Intelligence* **128**(1–2), 99–141.

Vail, D. & Veloso, M. (2003), Dynamic Multi-Robot Coordination, *in* A. Schultz et al., eds, 'Multi-Robot Systems: From Swarms to Intelligent Automata, Volume II', Kluwer Academic Publishers, the Netherlands, pp. 87–98.

van Alstyne, M. (1997), 'The State of Network Organization: A Survey in Three Frameworks', *J. of Organizational Computing* **7**(3).

Vaughan, R. T., Gerkey, B. P. & Howard, A. (2003), On device abstractions for portable, reusable robot code, *in* 'Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)', Las Vegas, Nevada. To appear.

Vaughan, R. T., Howard, A. & Gerkey, B. P. (2002), *Stage User Manual 1.3*, Player/Stage Project, http://playerstage.sourceforge.net.

Vaughan, R. T., Støy, K., Howard, A., Sukhatme, G. & Matarić, M. J. (2002), 'LOST: Localization-Space Trails for Robot Teams', *IEEE Transactions on Robotics and Autonomous Systems* **18**(5), 796–812.

Vaughan, R. T., Støy, K., Sukhatme, G. S. & Matarić, M. J. (2000), Go ahead, make my day: Robot conflict resolution by aggressive competition, *in* 'Proc. of the Intl. Conf. on Simulation of Adaptive Behavior (SAB)', Paris, France, pp. 491–500.

von Neumann, J. (1947), 'Discussion of a maximum problem'. Unpublished working paper. Reprinted in A.H. Taub, ed., *John von Neumann, Collected Works Vol VI*, Pergamon Press, Oxford, 1963, pp. 89–95.

von Neumann, J. & Morgenstern, O. (1964), *Theory of Games and Economic Behavior*, 3^{rd} edn, J. Wiley, New York.

Waldo, J. (1999), 'The Jini Architecture for Network-Centric Computing', *Communications of the ACM* **42**(7), 76–82.

Wedelin, D. (1995), 'An algorithm for large scale 0-1 integer programming with application to airline crew scheduling', *Annals of Operations Research* **57**, 283–301.

Weigel, T., Auerback, W., Dietl, M., Dümler, B., Gutmann, J.-S., Marko, K., Müller, K., Nebel, B., Szerbakowski, B. & Thiel, M. (2001), CS Freiburg: Doing the Right Thing in a Group, *in* P. Stone, T. Balch & G. Kraetzschmar, eds, 'RoboCup 2000, LNAI 2019', Springer-Verlag, Berlin, pp. 52–63.

Wein, J. M. & Zenios, S. A. (1991), 'On the Massively Parallel Solution of the Assignment Problem', *J. of Parallel and Distributed Computing* **13**, 228–236.

Werger, B. B. (2000), Ayllu: Distributed port-arbitrated behavior-based control, *in* L. E. Parker, G. Bekey & J. Barhen, eds, 'Distributed Autonomous Robotic Systems 4', Springer-Verlag, Knoxville, Tennessee, pp. 25–34.

Werger, B. B. & Matarić, M. J. (2000), Broadcast of Local Eligibility for Multi-Target Observation, *in* L. E. Parker, G. Bekey & J. Barhen, eds, 'Distributed Autonomous Robotic Systems 4', Springer-Verlag, pp. 347–356.

Winston, W. L. (1991), *Introduction to Mathematical Programming: Applications and Algorithms*, PWS-KENT Publishing Company, Boston, Massachusetts.

Yokoo, M., Durfee, E., Ishida, T. & Kuwabara, K. (1998), 'The distributed constraint satisfaction problem: formalization and algorithms', *IEEE Transactions on Knowledge and Data Engineering* **10**(5), 673–685.

Zlot, R., Stentz, A., Dias, M. B. & Thayer, S. (2002), Multi-Robot Exploration Controlled by a Market Economy, *in* 'Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)', Washington, DC, pp. 3016–3023.