TRUST AND REPUTATION FOR FORMATION AND EVOLUTION OF MULTI-ROBOT TEAMS

A Dissertation
Presented to
The Academic Faculty

by

Charles Everett Pippin II

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
College of Computing

Georgia Institute of Technology
December 2013

TRUST AND REPUTATION FOR FORMATION AND EVOLUTION OF MULTI-ROBOT TEAMS

Approved by:

Professor Henrik Christensen, Advisor
Director, Center for Robotics and Intelligent Machines
School of Interactive Computing
College of Computing
Georgia Institute of Technology

Professor Magnus Egerstedt
School of Electrical and Computer Engineering
College of Engineering
Georgia Institute of Technology

Professor Tucker Balch
School of Interactive Computing
College of Computing
Georgia Institute of Technology

Professor Lynne Parker
Department of Electrical Engineering and Computer Science
University of Tennessee, Knoxville

Professor Frank Dellaert
School of Interactive Computing
College of Computing
Georgia Institute of Technology

Date Approved: November 8, 2013

Keep the earth below my feet,

For all my sweat, my blood runs weak,

Let me learn from where I have been,

Keep my eyes to serve, my hands to learn.

-Mumford & Sons.

To my sons, for whom the world awaits;

may you find it to be full of wonder.

ACKNOWLEDGEMENTS

On the desk in my office sits a TRS-80 Model III microcomputer, with 48K RAM, built circa 1980. The computer that I am typing this dissertation into has a processor that is about 1,000 times faster and has 200,000 times more RAM than the TRS-80. I keep it as a reminder of how far advances in computing have come in my lifetime, and as a reminder of the pace of technological change. When I was about eight years old, my mother took my brother and me to a summer course in which we learned to enter BASIC programs into these machines and watch them run. I still remember sitting in front of one, with its gleaming silver case and watching a pixelated space ship being plotted on the screen after I had entered a long series of commands. It was pure joy and it captured my imagination. A few years after this, we had our own IBM computer at home, and I would spend hours typing in, debugging and modifying more BASIC programs. Later, when I was in high school and contemplating career choices, it was my father who recommend a Computer Science degree to me. Of course, I would not be where I am today, without the help and guidance of my family. Just as the computer has been improved, I have grown as well, through the teaching, mentoring and care of my family, teachers and colleagues over the years. I am forever grateful to my mother and father for providing these opportunities and for nurturing my interest in this field. My father encouraged me to read science fiction, to always have a positive attitude and to always try to learn. My mother has always lent words of encouragement and has helped to move me forward by asking me how my research was going. My grandparents sacrificed, so that life would be better for future generations. Most importantly, I am forever grateful to my loving wife, Fernanda, who has graciously and patiently supported my goals over the years, not to mention being supportive of robotics experiments that have run late into the evening and on weekends. She has helped to soothe the sting of setbacks and greeted my successes with genuine excitement. Thank you for keeping me grounded and reminding me of what is important in life.

Figure 1: A TRS-80 Model III Microcomputer.

My first computer science job was as a co-op student at Intergraph Corporation, porting Unix applications to the Windows NT operating system, and I would like to thank Vince Smith for that opportunity. After graduating from the University of Alabama in Huntsville with a bachelor's degree, I found myself working as a software consultant and building large IT systems. This was fulfilling work, for a while, but I kept feeling the tug back to the scientific foundations in my field. Around this time, I was reading futurist works by Hans Moravec and Ray Kurzweil and became very excited about the pace of technological progress, in relation to Moore's law. I decided that if these predictions were even partly correct, that the future would be very exciting for computing and robotics. I felt that if this is something that was going to happen, I would rather be part of it, with an opportunity to effect positive outcomes, than just being a passive observer.

It was around this time, in 2003, that I enrolled in the Masters of Computer Science program at the Georgia Tech College of Computing, and I completed my master's degree the following year. I am grateful to my master's advisor, Dr. Charles Isbell for his guidance. When I was not accepted into the Ph.D. program, the following year he encouraged me to try again, and for that I am very appreciative. Around this time, I also began my employment with the Georgia Tech Research Institute (GTRI). I would like to offer special

thanks to my colleagues and mentors at GTRI, including Lora Weiss, Terry Hilderbrand and Rusty Roberts for giving me opportunities to perform robotics work and for supporting me in my research. Dr. Weiss has been an excellent mentor and has helped me to align my thesis with my research at GTRI. In addition, my research using UAV platforms would not be possible without the excellent team of specialists in UAV development and operations, including Dave Price, Gary Gray, and Warren Lee. I would like to thank Stephen Camp Jr. for his assistance with environmental setup and observation of some of my larger robot experiments. I would also like to thank my GTRI colleagues Alan Wagner, Zsolt Kira and Misha Novitzky for their advice and guidance with my research over the last few years. I will always be grateful to GTRI, for supporting my research through internal research and development projects and Ph.D. stipends, and for providing tuition benefits to GTRI employees.

In the Georgia Tech College of Computing, I would like to especially thank Dr. Tucker Balch, who took me under his wing when I first entered the Ph.D. program and gave me the opportunity to work on the DARPA LAGR robotics project. This project taught me many lessons about robotics and perception in unstructured environments, and allowed me to understand the practical challenges in mobile robotics research. For this opportunity, I will always be grateful. He has continued to provide guidance and I am again lucky to have him on my dissertation committee.

Later in my Ph.D. program when my wheels were slipping, Dr. Henrik Christensen helped me to relocalize and to get back on track toward my goals. This is after all one of his specialties in robotics, but he is good at orienting students as well. He is an excellent advisor and one of the best listeners that I know and is able to provide critical feedback effectively. His vast knowledge of robotics and computer science has been extremely helpful in my research and he has been kind enough to offer encouragement along the way. I am blessed to have him as my advisor and thankful for his guidance over the years. It was around the Spring of 2010 when we began to have discussions about ways in which robots could learn to cooperate better on multi-robot teams. We discussed metaphors in business wherein partners cooperate but must learn which of their potential and past partners can

bid accurately on proposals and deliver projects effectively, and this thesis was born. It was also in his office that he suggested I investigate local-only interactions between robots, when robots cannot communicate beyond their immediate partners, using John Bartholdi's bucket brigade algorithms from operations research. This dissertation is the fruit of those early discussions and many since.

I have also been blessed to have three other world class scientists on my committee. Drs. Magnus Egerstedt and Frank Dellaert have been helpful since my days on the LAGR project and were instrumental in giving feedback during my proposal and thesis work, resulting in a better research result. Dr. Lynne Parker is an expert on large multi-robot teams and robot performance and it has been an honor to receive her time, guidance and feedback as well. I am also grateful to many other colleagues, fellow students and friends who have provided friendship, advice, and feedback on my research, including David Millard, Mike Heiges, Tom Collins, Gary McMurray, Miles Thompson, Michael Matthews, Ai-Ping Hu, Stephen Balakirsky, Paul Robinette, John Rogers, Carlos Nieto, Tucker Hermans, David Portugal, Matt Powers, David Wooden, Richard Roberts, Jinhan Lee, Chip Mappus, Miriam Pierce, Michele Burts, Brian Harris, Rajeev Dayal, Reed MacTavish and Matthew Meyer.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

GLOSSARY

agency An organization, corporation or government entity that is represented by a robot, operates as the owner of the robot, and for whose benefit the robot peforms tasks., p. 210.

annotation A special type of marking that is placed onto a trust model for a given trust dimension that marks exceptions to a set of observations and can be used to affect the interpretation of the trust value., p. 128.

coalition A team of agents that is temporally and dynamically formed to perform a set of distinct task having a distinct description. (see Parker and Tang, 2006)., p. 24.

cooperator A robot peer that regularly cooperates, participates, performs or behaves as expected., p. 92.

defector A robot peer that regularly violates a dimension of trust or performs unexpectedly., p. 92.

federate Of a number of organizations or entities that form a single centralized unit, wherein each unit maintains some internal self government or operation., p. 52.

freeloader An agent or robot that benefits from the cooperation of others but does not substantially cooperate or contribute in return., p. 166.

good performer A robot or agent team member that is performing or cooperating at a level that is consistent with expectations for that robot., p. 92.

observation A single instance of behavior or interaction which was observed by another party and can be used to update a trust model., p. 92.

originator The originator of a task that needs to be performed. The originator may peform the task or rely on another robot to perform it on their behalf., p. 102.

peer Another robot with whom a particular robot may communicate with and cooperate with to perform tasks and may or may not yet be trusted., p. 91.

performer A robot that peforms tasks on behalf of other robots or the team., p. 102.

poor performer A robot or agent team member that is not performing or cooperating at a level that is consistent with expectations for that robot., p. 92.

reputation A measure of trust that is shared within a community, the social component of trust., p. 37.

role A behavioral set, group of actions or sensing capability that a robot is capable of performing, see [63]., p. 128.

shadow A trusted robot that closely follows and observes a robot teammate, to verify its performance., p. 93.

team The set of robots or agents that may cooperate together to perform tasks and are considered to be trusted., p. 92.

trust A degree of belief in the reliability of an entity (See Castelfanchi and Falcone, 1998)., p. 37.

trusted A robot or agent that has a sufficient level of trust, along with a specific level of confidence given the model., p. 92.

UAV Unmanned Aerial Vehicle, an aircraft that is remotely piloted or that flies using autonomous control., p. 3.

untrusted A robot or agent that has a low level of trust, and the confidence in this level is sufficiently high., p. 92.

SUMMARY

Agents in most types of societies use information about potential partners to determine whether to form mutually beneficial partnerships. We can say that when this information is used to decide to form a partnership that one agent trusts another, and when agents work together for mutual benefit in a partnership, we refer to this as a form of cooperation. Current multi-robot teams typically have the team's goals either explicitly or implicitly encoded into each robot's utility function and are expected to cooperate and perform as designed. However, there are many situations in which robots may not be interested in full cooperation, or may not be capable of performing as expected. In addition, the control strategy for robots may be fixed with no mechanism for modifying the team structure if teammate performance deteriorates. This dissertation investigates the application of trust to multi-robot teams. This research also addresses the problem of how cooperation can be enabled through the use of incentive mechanisms. We posit a framework wherein robot teams may be formed dynamically, using models of trust. These models are used to improve performance on the team, through evolution of the team dynamics. In this context, robots learn online which of their peers are capable and trustworthy to dynamically adjust their teaming strategy.

We apply this framework to multi-robot task allocation and patrolling domains and show that performance is improved when this approach is used on teams that may have poorly performing or untrustworthy members. The contributions of this dissertation include algorithms for applying performance characteristics of individual robots to task allocation, methods for monitoring performance of robot team members, and a framework for modeling trust of robot team members. This work also includes experimental results gathered using simulations and on a team of indoor mobile robots to show that the use of a trust model can improve performance on multi-robot teams in the patrolling task.

CHAPTER I

INTRODUCTION

1.1 Defining the Problem

Agents in most types of societies use information about potential partners to determine whether to form mutually beneficial partnerships. We can say that when this information is used to decide to form a partnership that one agent *trusts* another, and when agents work together for mutual benefit in a partnership, we refer to this as a form of cooperation. A large body of research exists on cooperation and trust in multi-agent systems for e-commerce, computer networks and social networks. In these domains, issues of trust and cooperation have been studied extensively. Examples of cooperation also exist in animal and even plant and microbial societies.

Multi-robot teams are advantageous because of the ability for a team of robots to operate more efficiently and be more robust to failure than a single robot. There is an abundance of research interest in cooperative multi-robot teams, and cooperative robot teams are used in real world environments today. However, many of the current approaches make assumptions about the teaming and control strategies that will not apply to dynamically formed and federate teams. In traditional multi-robot systems approaches, each team member explicitly operates as part of a team and has the team's goals either explicitly or implicitly encoded into a utility function or are expected to cooperate and perform as designed. In addition, the control strategy for these robots is also generally fixed and the teams have no mechanism for dynamically modifying the team structure and control strategy if teammate performance deteriorates.

Trust is an obvious mechanism for cooperation in human and business relationships, although we may not be able to definitively know if it is used in other natural societies. However, we would like to consider this question for robot teams: Should members of robot teams apply trust to each other, and would this help them to cooperate more effectively?

This research seeks to answer this question, and investigates the application of trust to the formation of multi-robot teams.

As a motivating example, consider a large rescue operation in which many responders from different organizations, agencies and countries arrive at varying times and each deploys one or more robots. Each of these organizations and robots has the ultimate goal of searching for survivors and would like to cooperate with other robots if this would make the rescue operation more efficient. However, the robots would likely have different capabilities and levels of reliability. To cooperate, they would need to dynamically form partnerships with other team members. In the presence of varying capabilities and changing performance characteristics, these robots should be able to decide which team members they will continue to cooperate with and have a flexible control strategy based on the trustworthiness of potential partners.

For smaller multi-robot teams that can be tightly controlled, explicit cooperation and control design may be sufficient. However, as the prevalence of robotic systems increases, there will be a need for robots to cooperate if economies of scale are to be achieved. Consider the number of UAV manufacturers and platforms that exist today, see Figure 2. Although these systems currently have lower levels of autonomy, as the levels of autonomy increases, there are efficiencies to be gained by having the vehicles cooperate with each other and across organizational boundaries.[1] In order to form dynamic partnerships, such future robot teams will need to discover which potential partners are trustworthy and adjust their team composition and tasking strategies accordingly.

Robotic systems can be classified by the degree of their autonomy and also the degree to which they are deployed in real world environments, as shown in Figure 3. The vertical axis reflects the level of autonomy present in a robotic platform placed in a dynamic environment, while the horizontal axis reflects the degree to which a platform has matured and has been deployed in relevant real world environments. For example, autonomous automobiles have recently been developed by several research labs and universities for participation in DARPA

[1]The current approach of having multiple operators and centralized control of individual vehicles does not scale well for dynamic environments.

Figure 2: Examples of the variety of Unmanned Aerial Vehicles in operation at the time of this publication. Similar systems are widely deployed, come from different manufacturers and are operated by various agencies. As the level of autonomy of such robotic systems increases, the ability to form dynamic teams will become important. Shown clockwise from the top left: General Atomics Predator (photo credit Leslie Pratt), AAI Shadow, AeroVironment Raven RQ-11, Sikorsky S-97, Lockheed Martin Sentinel (all photos courtesy of Wikimedia Commons).

grand challenge competitions [92]. These vehicles have a high level of autonomy, yet are not fully mature or deployed to end users. Similarly, collaborative research robots are used to perform experiments and demonstrations of robots working in teams. Some examples are of aerial and ground vehicle cooperation on a target detection task [54], and indoor robot cooperation on an intruder detection task [59]. While these vehicles require high levels of autonomy on the individual platforms and methods for cooperation, these are largely research platforms and are not mature and widely available. On the other hand, there are robot platforms that have wide deployment and levels of production, but have little autonomous capability. For instance, unmanned aerial vehicles (UAVs) are widely used in military, energy and homeland security domains [101], as are small unmanned ground vehicles (SUGVs), such as bomb disposal robots [164]. However, while the platforms are mature and mass produced, the vehicles themselves are generally remotely operated. Similarly, factory robots have been heavily used in factory automation for decades, but they work on statically defined problems and do not require the ability to reason in dynamic environments.

As the use of unmanned systems becomes more prevalent and robotic platforms become more autonomous, there will be a need for these systems to form dynamic partnerships. Robots with different sensors and capabilities should be able to form dynamic teams and

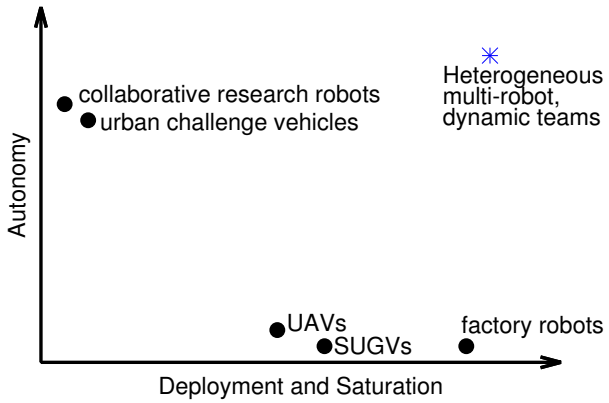


Figure 3: As the level of autonomy of deployed systems increases, there will be a need for these systems to dynamically form teams and collaborate.

cooperate on tasks. In situations where platforms have a high level of autonomous capability and are widely available and in use, the need will arise for robots to form heterogeneous, dynamic teams. In addition to search and rescue operations, other example scenarios include dynamic allocation of robot teams to disaster locations, security applications, and detection of forest fires.

While there are many examples of multi-robot systems working on a team, this research addresses robot teams that are both heterogeneous and dynamically formed. See Table 1 for a categorization of robot teams across the *homogenous vs. heterogeneous* and the *statically formed cooperation vs. dynamically formed cooperation* dimensions. As described above, statically formed robotic teams exist today to perform various cooperative tasks, such as soccer planing and target detection. However, the cooperation is generally explicitly encoded. There are also examples of dynamically formed cooperation with multi-agent swarms that are primarily reactive to the agents in their vicinity. Even in the case of homogenous robots, these systems will need an active approach for dealing with failed units [16]. This thesis aims to explore mechanisms to enable dynamically formed cooperative teams, with robots of different capabilities working together and with mechanisms for determining how well they can trust each other.

The explicit design approach requires that robot designers to build standards and capabilities into these systems at design time and to explicitly incorporate logic that enables

Table 1: Cooperation in dynamic, heterogeneous, multi-robot teams: This work explores ways in which teams of heterogeneous robots can use incentives and evaluation to dynamically form cooperative teams.

	Homogenous	Heterogeneous
Static Cooperation	soccer playing robots; distributed SLAM; traditional auction methods	air-ground collaborative target detection; fire-fighting robots
Dynamic Cooperation	swarming robots	*dynamic team formation via partner selection*

and encourages robots to cooperate with each other in future scenarios. However, this is not practical for several reasons: robot designers may not be able to predict all situations and scenarios in which robots should cooperate, full standards may be difficult to design in advance, such a design would allow for malicious designers and agents to exploit the system, and the large number of manufacturers makes this impractical. The very existence of computer viruses serves as an everyday example of exploitive uses of computers and networks that were not intended by the original system designers.[2]

The approach presented in this thesis is to design incentives for robots to cooperate multilaterally, to build robots with capabilities for recognizing situations in which cooperation would be effective, and for recognizing when other team members are not participating or performing as expected. Of course even this approach requires some degree of communication amongst the robots on a team and, this work will assume that the robots have the ability to exchange a set of standardized messages that allow them to perform task allocation. For them to be effective, the robots will need the flexibility to select from numerous partners and this also assumes that partners are uniquely identifiable and observable. In addition, robots will need the ability to evaluate and monitor partner performance, where the cost of doing so is less than the additional benefit gained from the resulting cooperation. The next section will present the research questions that are related to this approach.

[2]Early research into computer worms showed difficulties in controlling them through explicit design, as unintended consequences of the designs resulted in undesirable behaviors [142].

1.2 Research Questions

As a part of this research proposal, the following hypotheses are presented. The research to be performed will be in direct support of these claims.

Hypothesis 1

The use of trust and reputation mechanisms will provide incentives for cooperation on dynamically formed, heterogeneous robotic teams.

On robot teams that are dynamically formed and composed of self-interested entities, mechanisms and incentives will need to be in place to support cooperation. For a self-interested robot to cooperate, the benefits of doing so must outweigh the cost. Furthermore, trust and reputation mechanisms can support partner selection and punishment in the case of exploitation.

Hypothesis 2

The use of trust and reputation mechanisms will result in improved performance on dynamically formed, heterogeneous robotic teams.

If trust and reputation can be used to incentivize cooperation, then improved cooperation will result in better global performance for the team as well as the individual cooperators. Here, the performance metric considered will be the time taken to perform an assigned task. While other metrics (such as robustness, sensor detection probability, etc.) could be used on multi-robot teams, the use of time will provide for an easily accountable basis that can be easily compared across heterogeneous teams.

Addressing these claims as part of this thesis will lead to the following research question and set of supporting questions as shown below.

Primary Research Question

How can trust and reputation in dynamically formed, heterogeneous, multi-robot teams be used to improve task performance?

Corollary

How can heterogeneous, multi-robot teams utilize trust and reputation to evolve interactions with these partners?

Related to this research question are a number of supporting questions that must also be answered. This research will focus on the following four questions:

Supporting Question 1

How should the heterogeneous characteristics of each robot team member be included into the task assignment approach?

In dynamically formed, heterogeneous teams, robots will likely have different sensor characteristics. When these robots work together in teams, how can their different capabilities be considered? Investigation into this question is presented in Chapter 4, including the use of expected utility to predict probabilities for successful task performance, as part of an the auction based framework. This chapter also presents an approach for performing dynamic allocation on a multi-robot patrolling task.

Supporting Question 2

How can incentives be used to enable rational team members to cooperate?

On statically formed teams, cooperation is generally built into the system. However, on dynamically formed teams, incentives will be necessary to encourage cooperation. The use of incentives is presented in Chapter 7.

Supporting Question 3

How can team members use trust and reputation as a basis for selecting a cooperation and team formation strategy?

Dynamically formed teams will also need mechanisms to observe and evaluate the performance of their team members. This will require the use of mechanisms for building and scoring models of peer behavior. The robots can reason over these models and determine whether to continue cooperation with a given team member in the future or to select a strategy for cooperation based on the composition of the team members present. Work on this question is introduced in Chapter 6 and applied experimentally in Chapter 7.

Supporting Question 4

How can a team member monitor performance of other team members in an online fashion to update the models for trust and reputation?

Team members may begin with initial models and a priori information about other team members, but will need to update these models from observations of peer interactions. Monitoring costs and techniques should also be considered. Work on this question is also introduced in Chapter 5.

1.3 Preview of Contributions

There are several unique contributions that have resulted from this dissertation. They include:

- **Algorithms and methods for learning and applying performance characteristics of individual robots to task allocation (Chapter 4).** We demonstrated that task allocation mechanisms can incorporate robot sensor characteristics (Section 4.1). We have conducted an experiment in simulation with robots having three different sensor characteristics, and applied an expected utility formulation to the task allocation mechanism. We showed that considering individual sensor characteristics on a robot team can result in better detection performance than if sensors performance is not considered. In addition, we have shown that even robots having poor sensor detection rates can still be useful if the tasks to be performed by the poorly performing sensor are low cost, even if this results in multiple sensor tasks by different

robots to achieve the desired result. In addition, we demonstrated an approach for learning which robot team members accurately estimate costs for task allocation (Section 4.2). We also presented the bucket brigade algorithm, for robots with different known performance characteristics to dynamically adjust the task allocation approach in a multi-robot patrolling domain (Section 4.3).

- **Algorithms and methods for robots to monitor performance of team members (Chapter 5).** We presented an approach based on statistical methods from operations research for detecting when a robot was performing poorly on a task in relation to its team members (Section 5.4). We also demonstrated this approach using a central monitor for observing robot performance in a multi-robot patrolling task (Section 5.5).

- **A framework for modeling trust of robot team members (Chapter 6).** We presented an approach for modeling multiple dimensions of trust for individual robots, based on observation histories. We included mechanisms for sharing reputation with other team members. We also demonstrated this approach using a team of UAVs performing a patrolling task, with a single UAV performing observation and trust modeling (Section 6.3).

- **Demonstrations that using the trust model can improve performance on multi-robot teams in the patrolling task (Chapter 7).** We presented game-theoretic foundation for the application of the trust model and demonstrated that the use of a trust model can help to sustain cooperation on multi-robot teams (Sections 7.1, 7.2 and 7.3). We also demonstrated the use of the trust model on a team of eight robots performing a multi-robot patrolling task, and showed that the use of the trust model allowed for the robot team to dynamically reallocate tasks away from poorly performing robots (Section 7.4). While many examples exist for applying trust and reputation in multi-agent systems and online communities, our work serves as an early example of the application of trust models for cooperation and formation of multi-robot teams.

1.3.1 Dissertation Outline

Having described our research goals, we review background material and related work in Chapter 2, and describe our contributions in this context. Chapter 3 presents the problem domain and a taxonomy for robot cooperation and discusses and a contextual representation for the concepts used in this dissertation.

Chapter 4 provides algorithms and approaches for adjusting interaction strategies on multiple examples of robot task allocation, when the performance characteristics of individual robots are known or can be learned. Next, Chapter 5 discusses approaches to observation of robot teammates and presents algorithms and statistical methods from the field of Operations Research for monitoring teammate performance. Chapter 6 introduces the framework for modeling trust based on observation histories and Chapter 7 relates approaches for applying the trust model to adjust robot interaction strategies and partner selection in task allocation and multi-robot patrolling domains.

Chapter 8 presents a recipe for robot system designers to follow to adopt this methodology in other robot domains. Finally, Chapter 9 concludes the dissertation, reviews the contributions made and discusses future research directions. Where relevant, several chapters contain experiments in simulation and using real robots to validate our framework and methodology.

CHAPTER II

TRUST AND COOPERATIVE MULTI-ROBOT TEAMS, A REVIEW

In traditional multi-robot systems approaches, each team member explicitly operates as part of a team and has the team's goals either explicitly or implicitly encoded. Other approaches to multi-robot teaming include the careful design of utility functions and the use of negotiation strategies from game theory. Market based methods are used in many domains for exchanging goods among a selection of partners. However, they often assume the exchange of currency or explicit cooperation. Related to these approaches, partner selection and evaluation are necessary components for incentive based team formation.

Our research into this subject is related to work in diverse areas, including the multi-agent systems community, game theory and economics, and robotics. In this chapter, we first review relevant approaches to multi-agent and multi-robot systems that rely on explicit cooperation, and present the motivation for trust and reputation mechanisms.

There is a rich body of literature in the multi-agent systems community on trust and reputation in multi-agent systems. From the robotics community, our work has relevance to the formation of multi-robot teams as well as the game theoretic approaches to cooperation. One of the contributions of this thesis is the application of trust and reputation to the formation of multi-robot teams. In the sections that follow, we review the formative and recent work from others in each of these areas.

2.1 Explicit Cooperation

In classification of multi-robot teams, Cao et al. [27] considered the organization of multi-robot teams along dimensions of cooperation and relate the following:

> In almost all of the work in collective robotics so far, it has been assumed that cooperation is explicitly designed into the system. An interesting research

problem is to study how cooperation can arise without explicit human motivation among possibly selfish agents.

Also recognizing this problem, Brafman and Tennenholtz present an early distinction between two types of agents in a multi-agent system: controllable agents, which are controlled or explicitly designed by a system designer and uncontrollable agents, that are not under a designer's direct control [20]. As such, systems that contain both of these types are referred to as partially controlled multi-agent systems. This work further breaks down the uncontrollable agents into two groups which have different structural assumptions about their motivations: rational agents which seek to maximize expected utility and learning agents. Rational agents can be considered in the context of incentives and enforcement of social laws. Social laws and enforcement of these laws can place limits on how agents from different designers can work together. This work showed that through the use of a sufficient number of punishment agents, social norms can be enforced and incentives to deviate from the social laws can be reduced. Consequently, the punishment mechanism should not be needed and the number of punishers can be minimized. In the second case, with the underlying assumption that uncontrollable agents are learning agents, a teacher can be used to apply punishment and reward and guide the learning agent toward cooperative behavior. In this case, the teacher may have to temporarily sacrifice optimal behavior on their part in order to punish or guide the learner toward optimal behavior.

Howard, Parker and Sukhatme present the results of experiments involving a large heterogeneous team of heterogeneous robots performing mapping and exploration tasks [59]. The team consisted of approximately 80 robots consisting of two different types with varying capabilities. The robots were explicitly designed to cooperate in teams, with team leaders coordinating tasks of simpler, helper robots.

2.1.1 Social Laws

Early work in the multi-agent system community by Shoham and Tennenholtz [143], considers the use of social laws for agent societies, which are incorporated into a system at design time. The laws can be useful in constraining the set of actions in a multi-agent setting to

those that enable cooperative behavior and are beneficial to all agents. An example is that of traffic control, where for instance, robots can be designed to always drive on the right side of the road. A criticism of this approach is that there is nothing in the design that forces agents to follow the social laws, but rather that agents are assumed to follow the laws. Related work by the same authors [144] continues on the work of social rules as an off-line design tool and expands their use to the case in which a society of agents converges on rules over time. The authors describe a framework in which multi-robot teams develop social conventions through a stochastic game process. Agents interact with each other and make observations about the system. The system eventually converges on a set of social conventions. Their approach also uses techniques which are similar to reinforcement learning for updating agent behavior in a system. This example represents an early recognition in the field for an improvement to the assumption of explicit cooperation in agent societies, and provides a motivation for additional mechanisms that would enforce cooperation.

2.1.2 Biological Inspiration for Partner Selection

Representative examples from biology provide motivation for a mechanism that can elicit cooperation among agents that cannot be explicitly programmed. Recent work on cooperation between chimpanzees has shown that the animals will cooperate on a task, but only if the cooperation would result in greater reward for each partner than by working alone. In this task, both animals must simultaneously pull on a separate rope in order to bring a food tray toward them. If one of the animals does not pull, the rope slips away and they miss the opportunity for reward (this task is similar to the multi-robot box-pushing task described above). In addition, the experiments showed that the animals will learn from interactions with different partners and remember which partners are effective cooperators. Then, given a chance to select between two partners for completion of a task, they will choose the partner that has the best observed performance [88]. Elephants are also able to recognize the need for cooperation on the same rope pulling task. However, in experiments [122], one subject learned that she could merely step on her end of the rope and let her partner do all of the pulling, resulting in both animals receiving reward. Those experiments

did not allow for partner selection; however, so cooperation with an exploitive partner was the only option available for the other subject to gain the reward.

Cooperation in biological systems is not limited to animals. In one example involving plants and microbes, selection from a market of potential partners occurs in symbiotic relationships in which a single plant interacts with soil microbes [145]. In this relationship, nutrient exchange is critical to the success of both partners for the fixation of nitrogen. However, cheating occurs when ineffective strains of bacterial partners obtain benefits of nitrogen fixation with the plant, but do not exchange the resulting nutrients back to the plant. Plants have abilities to punish cheaters by removing nutrients and isolating those strains. In addition they have the ability to recognize the signature of the cheating strains. The work further suggests that partner choice can constrain cheating, but only if there is a selection of partners available to choose from, there is a mechanism for selecting from those partners (as well as for observing the cheating) and that the benefit derived from selecting a good partner outweighs the cost of monitoring partners.

Biological markets research [95] uses game-theoretic models to describe the formation of collaborating pairs of individuals for partner selection. A critical component of these models is that the decision to cooperate is dependent on the ability to select from various partners with different potential offers.

These examples from biology serve as additional motivation for this problem. Natural environments are often used for inspiration in multi-agent systems because of the dynamic, open and complex nature in which federate agents (plants and animals) with very simple behavioral sets can form partnerships.

2.2 Trust and Reputation Models

This work presents related work on trust and reputation models. This thesis presents the application of trust and reputation models to multi-robot teams operating in real world environments. However, recently there has been a significant amount of research in the multi-agent systems, networking and human-robot-interaction communities. We present relevant work from each of these areas below.

2.2.1 Trust and Reputation in Multi-Agent Systems

Pinyol and Sabater-Mir provide a comprehensive review of trust and reputation approaches in mult-agent systems [113]. They posit that these mechanisms are often a key design consideration for multi-agent systems, and that a main feature of open multi-agent systems is that agents have unknown intentions. They also present the recent growth in the number of cognitive modeling based approaches in the agent literature. In the multi-agent community, cognitive based approaches may be desirable because they are easier for humans comprehend than numerical or game-theoretic approaches. This may be particularly relevant when these mechanisms are applied in environments with high degrees of human interaction, such as in online social networks or e-commerce.

An approach for learning trust strategies is described by Fullam and Barber[46]. That work enumerates the types of decisions and strategy profiles that an agent can learn and compares reputation based strategy learning (based on indirect observations) with experience based learning (based on direct observations.) Other work investigates reputation with the concept of multi-dimensional trust[2]. Trust can be described by different characteristics, such as quality, reliability and availability. They show that modeling trust with multiple dimensions can lead to greater agent rewards.

Rosaci et al. consider the relation between trust and reliability in multi-agent systems [132]. In their model, they dynamically compute the weighting factor for reliability as part of an agent's reputation.

Partner selection in a multi-agent formulation can be implemented using graph models to describe trust relations between members on a team. For instance, in [43], distrust relations are used to build coalitions of mutually trusting agents.

Trust in terms of E-commerce based customer relationships is discussed in [87]. The authors present a vocabulary for discussing trust models, from various fields such as sociology and psychology. A social network model for trust is presented in [129], in which human participants have the ability to perform partner selection by breaking or maintaining links in a social network, based on interactions with other humans. That work showed that cooperation could be maintained through the rewiring of the social network. A review of

challenges for trust and reputation systems in online markets and communities is presented in [65]. Reputation learning in an e-commerce setting is presented in [157]. Buyers learn to trust which sellers are reputable through the use of a reinforcement learning algorithm. Agents learn to avoid purchasing low quality goods by maintaining models of reputable sellers. Each buyer has a mechanism for evaluating the quality of the item purchased. Buyers first seeks to trade with reputable sellers that offer a good at a maximum expected value. If there are no sellers that submit bids at that value, then the buyers seek to trade with non-reputable sellers. Also, buyers will perform exploration with probability p into set of non-reputable sellers. After a good is purchased, the buyer applies a quality function to determine a valuation for the good. If value of the good exceeds the expected value, the expected value function can be updated. Furthermore, if the quality is less than anticipated, the expected value is updated with a smaller value and may cause that seller to not be selected in a later round. In addition to the expected value function, the reputation rating of the seller is updated. If the reputation rating falls below a threshold, then that seller is moved to the list of non-reputable sellers. Finally, an excellent review of computational trust and reputation models is presented in [136].

2.2.2 Trust and Reputation in Communications Networks

In [84], trust is applied to the information fusion problem, by incorporating it into a Kalman filter process. Trust is describe as being multi-dimensional, based on the domain. For instance, in computer networks, trust can refer to the trustworthiness of a sensor (whether it has been compromised), the quality of data from the sensor, or the security of the link between sensors. The work shows that the fusion algorithm that uses trust can effectively avoid including information from untrusted sensors.

Yu, Singh and Sycara investigated approaches for exchanging reputation information in peer to peer networks [169]. They discuss a process for requesting reputation information from peers and for gathering information in the aggregate.

The use of trust metrics to plan a most trusted path through a network is described further in [11]. Here, the domain is that of autonomic networks, in which self-interested

agents come together to *dynamically* form a coalition because through collaboration they can do better than through working alone. The work assumes that a trust value is provided by some monitoring mechanism. The agents play repeated games to determine if the benefit of forming a link with a neighboring node outweighs the cost in repeated interactions. This work is related to the thesis in the use of game theoretic approaches to providing incentives for link formation, which is similar to a partner selection mechanism. In addition, it frames the formation of a network coalition from a viewpoint of self-interested agents seeking to maximize utility. Finally, the work shows the benefit of using such approaches, by being able to plan trustworthy paths through the network.

The evaluation of trust models in ad-hoc networks is also presented in [155]. Here, the concept of indirect trust (second hand evidence) is discussed more thoroughly, with nodes being able to "vouch" for the trustworthiness of other nodes. If a node is sufficiently trusted, then it is taken at its word as to the trustworthiness of a node that it vouches for. Furthermore, an interesting concept presented is that the desired path through a network might not always be the shortest path, but rather the most trusted path.

In another work on the use trust in Ad-hoc networks [171], the network is modeled using game theory, with payoff to each node being related to the trust relations in a collaboration graph. These payoffs are used to provide incentives for cooperation, to encourage mutual cooperation, rather than exploitation of the network by individual nodes. The results showed an improvement in network performance when a trust based approach was applied in Ad-hoc wireless networks.

This work is also related because of the modeling of the game using game theoretic methods and through the use of incentives to encourage cooperation. This serves as another example of the use of trust in ad-hoc networks, which has similarities to multi-robot collaboration. However, it is worth noting again that the multi-robot domain faces additional challenges related to mobility, communications, observations of behaviors, and heterogeneous capabilities.

In [58], the communication infrastructure is studied as a factor that affects the performance of a networked system. The work describes the interdependence of the communication infrastructure on collaboration. The agents in the system use a "gossip" like algorithm to observe collaboration styles of their neighbors in a learning algorithm, before playing a series of 2-player games. The learning algorithm attempts to classify neighbors into a type, which relates how they might play the game. Neighbors in the system share estimates of other agents cooperation as a stochastic matrix, and compute locally whether they should coordinate with other agents based on this matrix. After the games are played, a collaboration graph is formed to include all of the agents who collaborated during the interactions.

This work is related in using observations to model the collaboration of another player. It is also useful to treat these interactions as a 2-player game, for modeling purposes. The use of collaboration graphs is also related, in that this thesis considers that each player can build direction trust edges in a similar collaboration graph, as part of performing partner selection.

Another example of the use of trust for anomaly detection in computer networks is presented in [170], where trust is interpreted in terms of network security. That work presents a network with 2 layers: the regular sensor nodes and the monitoring nodes. The monitoring nodes are use to obtain observations about the performance of the regular sensor nodes and to incorporate the observations into a trust model.

In [85], the authors incorporate the use of trust into Kalman Filter for use in state estimation of power systems. In this domain, multiple separate agents exchange state estimates for the system with their neighbors. The inclusion of a trust model in this system improved the accuracy in the estimation for the state in 2 cases: 1) when the estimates were inaccurate as an interpretation of variance, and 2) when the estimates were unreliable as a measure of disagreement with neighbor estimates. Of particular interest is the notion that the use of trust can be used to protect against situations when a node has been compromised and is no longer reliable.

The accuracy and reliability interpretations of trust are related to this thesis; however,

the problem domain constrains the problem in a way that is not relevant to multi-robot systems. Unlike nodes in a network, robots are not fixed in position and may have heterogeneous characteristics. In addition the multi-robot domain presents additional challenges in observation, dissemination of reputation information and dynamic neighborhoods.

Similar to the work on trust based information fusion in a network, in [44], a distributed reputation system is used to improve odometry systems in collaborative robot systems. Social odometry methods perform information fusion on each robot's position estimation based on confidence levels. In that work, robots placed more trust on the more capable robots in the system. Those capabilities were estimated to be one of multiple categories, based on a priori knowledge of each robot as well as observations. That work presented experiments that showed the use of a reputation system improved social odometry performance.

2.2.3 Trust and Reputation in Human Robot Teams

There has also been considerable recent interest in trust relations for mixed human and robot teams. Ososky, et al. consider the implications of humans considering robots as possible teammates and present the importance of considering human mental models of trust when applying these models to human-machine relationships. Human mental models are related to behavioral, but also physical characteristics of robots. In these situations, it is important for humans to have correct and complete mental models of the capabilities of a robot.

Robinette, et al. allow for a robot to maintain the level of trust that humans have toward it, through an algorithm for selecting behaviors that maintain trust [131]. In that approach, a robot predicts a mental model for humans that it is assisting in an emergency evacuation and selects actions that might increase the human's confidence in the robot's capabilities.

Desai, et al. recognize the importance for robots to operate in dynamic and unstructured environments and that trust is an important component in human-robot interaction [35]. In real robot navigation experiments, the authors found that humans are more likely to allow the robots to navigate autonomously when they have a high level of trust in their reliability. As the robots' reliability decreased, the humans were more likely to take manual control.

The research on human to robot trust is related to our work in several ways. The focus on operation in dynamic environments, reliable operation and the recognition that robots will need to be trustworthy partners applies to multi-robot teams as well. However, modeling human trust requires higher level mental and cognitive modeling that is not necessary for robot models of trust. In addition, our research considers how these trust models could be applied to adjust partnering and task allocation approaches on multi-robot teams.

2.3 Incentives from Game Theory and Economics

Work by Groves from the field of economics on incentives in teams [55] deals with the problem of providing incentives for agents to behave as if they were working on a team. This work defines the team decision problem as a situation in which the individual team members make decisions with varying information but are motivated toward a common goal. Therefore, a team decision problem is one in which all of the team members share the goals and preferences of the organization's leader. However, if the individual units do not necessarily share the goals of the organization, but rather are motivated by compensation, then the decision problem can be represented as an n-person game, with the set of compensation rules defined as an incentive structure. Grove presents as an example the problem of a conglomerate that consists of a set of semi-autonomous units. The conglomerate's head organization wishes to have all units making optimal decisions. However, there may be situations in which the units could do better for themselves by acting in ways that are not optimal for the conglomerate. Similarly, different strategies are presented for incentives that can lead to optimal behavior by the head. In the first, if the conglomerate has full information about the performance of each unit, then the unit is compensated based on their performance and has incentive to perform well. in the second, the entire profit of the conglomerate is shared across all of the units. In this case, the conglomerate does not need to have full information about each unit's performance. In this strategy, there is nothing in place to prevent a unit from operating sub-optimally. In the final case, each unit's compensation includes a portion of the overall profit as well as a component that represents how well their contribution benefits other units. In that case, the goals for all units are aligned

and units have incentives to perform optimally, thereby maximizing the group reward. The significance of this work is that it provides an early example of using incentives to bring together semi-autonomous groups. With the proper use of incentives, the group shares a common goal. This work also allows for the head to operate without full knowledge of each unit's state. However, this does make the assumption that payments in the form of profits will be transferred between the central node and the units.

Monitoring and incentives can also be applied in a decision-theoretic approach to mitigate risk in agent interactions [23]. Models of trust can be maintained about potential team members based on repeated interactions and these models can be used to calculate expected utility decision trees for cooperation with other agents. Ahn, DeAngelis, and Barber further investigate reputation with the concept of multi-dimensional trust [2]. Trust can be described by different characteristics, such as quality, reliability and availability. They show that modeling trust with multiple dimensions can lead to greater agent rewards. Game theory approaches are used to perform dynamic team formation in network routing problems in [17], [62], [148] and [10]. Other work describes the use of incentives using the Tit-for-Tat strategy for improving robustness in the peer to peer file sharing network, BitTorrent [33].

Kandori presents the social norm strategy as an approach to the random matching game for situations when agents may not interact with the same partner repeatedly, but perform interactions within a society. That work shows that with the addition of a reputation mechanism, community enforcement of social norms provides sufficient incentives for cooperation [68]. Blanc et al. [17] applied Kandori's social norm to the peer-to-peer routing task.

McGrew and Shoham present the use of contracts to enforce behaviors in multi-agent teams [86]. In that approach, there exists a centralized node, the center, which performs three functions. First, the center presents contract arrangements to all players and collects their assent (signatures). Second, in the execution stage, the center can monitor the actions of all players that signed the contract and observe whether they followed the contract. Third, in the enforcement stage, the center enforces the contract by fining those agents that didn't follow the contract. The use of this central node provides incentives for agents to cooperate and when the agents follow the equilibrium strategy, the center does not need

to administer fines in the enforcement stage. The authors also show approaches that allow for there to be less burden on the center: if the agents are able to negotiate and sign the contract parameters in advance, then the center does not need to collect signatures. During the execution stage, the center's monitoring costs can be reduced if the agents report when team members did not follow the correct action. In this case, the center punishes deviators by a large amount, rewards correct complainers by a small amount, and punishes incorrect complainers by a small amount. This requires that the center have a monitoring mechanism for determining the true outcome of a complaint.

Arslan, Marden and Shamma describe game-theoretic approaches to distributed task assignment of a multi-vehicle team [7]. They investigate formulations for the local utility functions of self-interested agents that can be aligned with a global utility. This reduces the multi-robot task assignment problem to one of designing proper local utilities and negotiation mechanisms. One option is to set the local utility to the global utility; however this requires continuous sharing of global information across all vehicles, similar to centralized optimization approaches. Another approach is to set the local utility to the marginal contribution of utility generated by the vehicle's participation, but this requires an appropriate negotiation mechanism to eliminate inefficient assignments. The vehicles in these algorithms announce their preferred target assignments and calculate local utility based on their assignments and the announced assignments of other vehicles.

Similarly, Xiong, Christensen and Svensson [168] described a multi-agent negotiation mechanism using a sub-game perfect equilibrium strategy for performing target distribution. Each sensor was represented by an agent that negotiated on behalf of the sensor and interacted with opponents to receive more assignments and achieve better payoffs. Strategy profiles were created to allow agents to come to agreement with minimal delay. Kraus [72, 73] provides an excellent review of strategic, competitive negotiation mechanisms for multi-agent systems and discusses approaches from game theory and economics.

Game theory has also been used to model conflict and cooperation in evolutionary biology and behaviors [146], warfare [8], nuclear deterrence [138], economics, business and

politics [103]. The applications to evolutionary biology and behaviors are particularly interesting, because they represent agents in nature that exhibit conflict and cooperation that can be described using game theory. For instance, John Maynard Smith describes cases in animal conflict in which animal behaviors were previously thought to have evolved because the behaviors benefited the population as a whole, for the"greater good"; however, he showed that these behaviors could be described using self interested agents that employ evolutionary stable strategies [147, 146] . This result is useful in multi-robot applications: when robot designers assemble robot teams, rather than expecting the robots to behave for the greater good (and for robots from other designers to do so), robots that are self interested can be used to put bounds on the behavioral outcomes. Recent work on trust has also shown the importance for robots to be able to recognize when the use of trust models might be necessary [162].

2.3.1 Self interest vs. Team Interest

There is often a tradeoff between designing systems that optimize a local function for self-interested robots, and that optimize a global function for team-interested robots. Paruchuri, Tambe, et al. [109] present a formulation for explicitly modeling this tradeoff on teams that are partially competitive within a cooperative setting.

2.4 Dynamic Team Formation

A significant component of this thesis is the use of trust models to perform dynamic team formation. The related research in this area highlights the importance for heterogeneous teams to be able to form dynamically. The relevant work into dynamic team formation on multi-robot teams includes approaches using task and role descriptions, synergy, and altruism mechanisms.

2.4.1 Task Descriptions

Parker and Tang [107] describe a mechanism for multi-robot teams to perform tasks that are typically performed by single robots through the use of task representations that use schemas to represent the specific task to be performed. Single robots or teams of robots

with complementary characteristics are able to respond to tasks requests that meet the schema definition. In addition, Parker and Tang define a coalition as a temporary and short lived formation of a robot team that can be dynamically formed to meet the needs of a task description. This is another example of robots having capabilities to dynamically form teams in combinations not anticipated or explicitly created by the human designers.

One of the common tasks described in the multi-robot literature is box-pushing [37, 107]. This represents the type of task in which multiple robot cooperation is necessary to complete the task. The cost of cooperation for both robots is high in this case, but there is no alternative if the robots wish to collect the utility for completing the task. Other types of tasks, such as distributed localization, can be completed by a single robot but may benefit from assistance by another robot, with minimal effort required by the assisting robot. Additionally, there are tasks that can be referred to as transportation and assignment problems, where robots may benefit by exchanging tasks amongst themselves.

Regarding task characterization, Balch provides a taxonomy of robot tasks and an approach for applying metrics to robot tasks across various dimensions, including time, resources, movement required and the subject of the action [9].

2.4.2 Dynamic Teams

Dynamically formed 'pickup teams' are presented as being necessary for robot teams to dynamically meet and form into teams, without prior knowledge or explicit programming [63, 67]. Stone and Veloso presented a framework for placing robots into roles and formations, and membership in formations could be dynamic, based on the capabilities of the robot [150]. In situations where there is high agent turnover and agents may not have time to build models of partners, stereotypical trust can be used to judge the trustworthiness of potential partners, using visible features [22].

Jones et al. present the problem of 'forming pickup teams' of heterogeneous, cooperative robots to perform tasks using an auction framework. The ability to form dynamic teams has several advantages: robots may be expensive or scarce, and it makes sense to share

them across organizational boundaries; robots may need to be organized quickly into ad-hoc teams (such as at disaster locations), and robots should be easily replaced when they fail [63]. The auction framework presented allows for the specification of roles that are needed to perform a task. Robots that are added to the team are labeled with the roles that they can fulfill. Each task that is given to the system is announced to all team members, and they negotiate task assignments based on local bid estimates and the roles that they can perform. This assumes, of course, that the robots all negotiate using the same auction protocol, that they accurately define roles and calculate utility using the same basis and that they correctly perform tasks that they bid on.

2.4.3 Synergistic Teams

Liemhetcharat and Veloso have investigated an approach for modeling the synergy of potential multirobot teams and forming a team from those team members that have been observed to maximize the synergy function [78]. Their work is one of the most similar to our research, based on the idea that robots can consider the capabilities of their peers and to dynamically select those that they work well with to form a team. In our work, rather than a metric of synergy, we model the desirability of the relation using a trust model and allow for this relation to change over time. We present multiple implementations of this model and demonstrate its use in experiments with real robots. Nevertheless, the recent work in this area demonstrates the relevance to the multi-robot research community for the ability to perform dynamic formation of a multi-robot team that works efficiently and is robust to failures.

2.4.4 Altruistic Teams

Robots can learn to form altruistic models of trust for determining bidding rules in [94, 32]. Altruism as defined as the amount of cost (in terms of time) that a robot is willing to spend to perform a task for another. This approach relies on a control law to drive the level of altruism that $robot_i$ will allow for $robot_j$ to be the observed level of altruism displayed by $robot_j$. This level of altruism is used to determine whether a robot will bid on another's task, if the task cost is less than that amount.

In contrast, our research will investigate the use of multiple dimensions of trust, including whether other agents have bid, and the quality of the bid as well as completion of tasks bid. This research will also investigating approaches for sharing the knowledge of altruism/trust across a community. Furthermore, this work will show that incentives from game theory can be used to induce cooperation, and will consider the effects of noise on the reputation and observations.

2.5 Cooperative Task Allocation

There is a large body of literature on approaches to task allocation on multi-robot teams. In this thesis, we have selected task allocation and multi-robot patrolling domains to illustrate the application of trust and reputation mechanisms. Cooperative task allocation problems are well suited to a discussion of dynamic team formation in a dynamically changing environment, due to the difficulty in computing optimal solutions and the challenges inherent with heterogeneous, federate robot teams. Therefore, we provide a brief background of multi-robot task allocation mechanisms that are relevant to this thesis.

2.5.1 Task Assignment

The problem of assigning a number of agents to a number of tasks, with minimal cost, is well known as the optimal assignment problem from operations research, and can be solved using linear programming methods [74]. However, centralized approaches to the assignment problem can be a source for communications and processing bottlenecks in the system and allow for a single point of failure [41]. Also, in dynamic environments it may not be practical to keep central nodes up to date with the current state of the environment and of other agents. Furthermore, centralized approaches, while able to find optimal solutions, may not scale as easily as a distributed system and are less practical when changes in a dynamic environment require frequent re-planning. Conversely, distributed, multi-agent approaches can operate using local state information. They can work on tasks in parallel, perform distributed sensing and operate in multiple locations at once. Furthermore, a team of robots adds redundancy to the system. Unfortunately, a tradeoff is that these teams must communicate and work together and uncertainty can exist regarding robots'

intentions towards each other. For instance, a team member may have trouble cooperating due to communication errors, or because they are busy performing other tasks, or even because of conflicting goals [5].

2.5.2 Market-based Approaches

There are many different methods for performing distributed cooperation, including centralized optimization algorithms and game theoretic techniques described above. However, auction based algorithms generally have low communication requirements (agents coordinate tasks through bid messages), and therefore are well suited to environments with communication constraints. Auctions can perform computations in parallel and the methods take advantage of the local information known to each agent [36]. For instance, an unmanned aerial vehicle (UAV) would not need to communicate a low fuel state to the entire team for allocating tasks, but could implicitly include this knowledge in their own task selection through cost-based bidding. Finally, these approaches are also amenable to standardization and cooperation across teams, as heterogeneous teams that are dynamically formed need only implement the auction messages in order to participate in cooperative tasks.

The assignment of robots to tasks is known as the multi-robot task allocation (MRTA) problem [51]. Auction methods are a class of decentralized algorithms that solve this problem by splitting computation across multiple nodes and iteratively performing task assignments [15]. In a specific type of the MRTA problem, there are multiple robots and multiple sequential tasks, which are locations to be visited, with the goal being to assign a robot to each of the locations while minimizing the overall team cost. Gerkey et al. [50] showed that the MRTA problem can be reduced to the well known optimal assignment problem from operations research [74], which can be solved using linear programming methods.

The basic auction approaches to the task allocation problem assume that team members can be trusted and are explicitly designed with the goal of the team in mind (to reduce the overall cost) [70]. Zlot and Stentz et al. implemented a market based mechanism on a team of indoor robots to perform a multi-robot exploration task [173]. The robots negotiated new areas available for exploration and revenue was exchanged for information. The multi-robot

team showed a significant performance improvement over teams that did not negotiate. Several other researchers have investigated the use of market-based mechanisms for multi-robot coordination. Examples of robots explicitly formed into cooperative teams using auction approaches are seen in the multi-robot mapping [173], coordinated box-pushing [49], and Mars rovers [139] domains; as well as in simulated UUVs [137], and UAVs [134]. An excellent survey of market-based multi robot coordination is provided by Kalra et al. in [67].

In many of these market-based approaches, self-interested robots operate in a virtual economy and exchange goods (information, task performance, etc.) for virtual revenue, which is not necessarily exchanged. While each agent seeks to improve their virtual profit, the entire team benefits from the cooperation. There are alternative market-based schemes that use the actual exchange of virtual currency to provide incentives. An overview of incentives for cooperation in these types of systems is provided by [72]. Such incentives include the use of contracting through monetary schemes and the exchange of credits between systems and their owners, as well as through bartering. Currency exchange mechanisms require agents to share a common valuation, to keep accounting of interactions and to have a secure mechanism for performing the currency transfer. Bartering depends on agents needing assistance from each other and may not work well when one agent can provide help and does not need any help itself, or in situations when agents may not be available in the future. Auction based approaches express tasks and costs in terms of a common utility and do not require the actual exchange of money. However, the use of incentives in contracting can inform the use of incentives for cooperation in auctions. For instance, [72] shows that the use of monitoring a task's completion can improve an agent's utility when it is risk averse.

Work on auctions with untrusted bidders in an e-commerce setting is presented in [21]. The authors show that by constructing the allowable types of bids, an auctioneer can identify the submitted bids as belonging to trustworthy or untrustworthy agents. However, this requires that each agent know their type in advance. Furthermore, this approach may not result in improved social welfare.

2.5.3 Distributed Multi-Agent Teaming

The ALLIANCE multi-robot cooperative architecture [108] relies on a set of motivational behaviors which control the activation of tasks on individual robots in a task allocation problem. Each robot calculates a local function that determines whether a robot will perform a task that has been announced. The function is dependent on observation of whether other robots are performing the task, whether the robot is capable of performing the task and the amount of time that a robot has been inactive.

2.5.4 Decentralized Data Fusion

Decentralized Data Fusion architectures consist of a network of sensor nodes that do not require centralized coordination or control. Each node relies on the use of information filters to keep the state estimate up to date at each node. Task allocation for target tracking in Decentralized Data Fusion is referred to as sensor management [81, 53]. Each sensor communicates to neighboring nodes their preferred target and expected utility for being tasked to that target, as a function of information gain. Each sensor then maximizes the expected utility function to determine target assignments locally, with minimal communication after the initial preferences have been announced.

Related research by Cameron and Durrant-Whyte [26] investigated the use of statistical decision theory to determine optimal sensing locations for performing localization and recognition tasks. Their work investigated the use of decision theory for a single robot with a manipulator to more effectively manage sensor placement. Bourgalt, Furukawa and Durrant-Whyte [19] investigated the use of decentralized Bayesian methods for allocating a team of UAVs in a search mission. The agents in this system exchanged target probability information, and each agent sought to maximize their own utility. Similarly, in the work by Tisdale et al. [156], multiple UAVs exchanged PDFs to update a target location estimate.

2.6 Summary

In this chapter, we have reviewed the relevant work in the multi-agent systems and networking, game theory, and multi-robot communities. This chapter also provides an overview of

algorithms and design approaches to multi-robot cooperation that will inform this work. Following are the key design considerations, motivated by the work described in this section.

- **Market based methods** can be used to perform decentralized task allocation, and also provide a mechanism for partner selection based on the partners' promised goods.
- **Partner selection** is key to the use of incentives and dynamic team formation. Examples from other works in economics and biology were presented to illustrate the use of partner selection.
- **Community enforcement** can be used to enforce incentives in populations in which agents may not frequently interact with the same partners [68].
- **Multi-dimensional models** of trust and reputation can be also be used to inform partner selection [2], and can be used to bias expectations and task assignment functions. Models can be built from repeated observations.

This research presented in this thesis will differ from the related work in several ways, as presented below.

- **Trust and Reputation** models have been considered rigorously by the multi-agent systems community, for use in agent based communities and for applications in e-commerce and online social networks. We can leverage from the research into the application of these models. Computer networking research in particular is also similar to the problems of multi-robot systems in many ways: the individual nodes or robots may be federate, and have resource constraints. However, robotic teams present additional challenges. Robotics systems have physical mobility, varying communication constraints and heterogenous characteristics. In this work, we apply trust and reputation models to multi-robot teams and consider the challenges related to mobility and the varying characteristics of robotic platforms. To our knowledge, others have not yet performed an in depth investigation into the application of trust models to the formation of multi-robot teams.

- **Game Theory and Economics** research on cooperation, partner selection, and negotiation provide the necessary incentive based structure that enables cooperation through community enforcement. This research will differ from the related work in several ways. First, we are concerned with building a framework for cooperation in multi-robot, dynamic environments, rather than studying the effects of specific strategies in forced, repeated play. Secondly, we wish to build mechanisms for cooperation that do not require currency exchange or detailed accounting mechanisms. Finally, we wish to investigate the use of monitoring, models, and incentives to build robot teams that cooperate effectively and reliably. While many approaches to cooperation consider intentional defection, here we are primarily interested in detecting the relative performance levels of team members using models with multiple dimensions, and in adjusting the task assignment function according to the learned abilities of the team.

- **Dynamic Team Formation** research is closely related to our work. Approaches from the literature have considered game-theoretic, altruistic and ad-hoc based approaches to task exchange and cooperation. Ad-hoc teams have considered role or schema based exchanges for solving the composition constraints of a team. Recent work into synergistic teams has considered how to select a group of agents that have been observed to work well together. Our research is related to these approaches, in that we also place importance on the ability to form dynamic, multi-robot teams. However, we consider the team formation problem using robot-to-peer trust, with multiple dimensions of trust, and demonstrate the sharing of the trust information between team members.

Given the large and relevant body of work on trust and reputation from the multi-agent systems community and the recent interest in dynamic team formation, we believe that this dissertation will be of interest to the multi-agent and robotics communities and applicable to dynamically formed, multi-robot teams. In the next chapter, we present our research methodology and contextual framework for applying trust and reputation on robot teams. Following chapters will demonstrate these concepts experimentally on multi-robot teams.

CHAPTER III

METHODOLOGY

This chapter presents the methodology that is used in this dissertation. We begin with a discussion of taxonomies for cooperation in multi-robot systems. The next section presents how models for trust and reputation are applied in this dissertation. We then present our conceptual model for a multi-robot system that incorporates trust and reputation. Finally, we integrate our approach with ideas from game theory to show how incentives for cooperation on a multi-robot system relate to trust and reputation models and multi-robot task allocation.

3.1 Taxonomies of Cooperation

When evaluating the cooperative makeup of robots on a team, it is important to consider whether robots are benevolent or competitive, as described by Stone [151]. This spectrum is presented in Figure 4. At the far left side of the spectrum, robots are benevolent, and can be thought of as fully cooperative. These robots work together to achieve the global utility and goals of the team, either because of explicit design, programming or configuration. Many of the existing multi-robot teams approaches could be described as benevolent. On the other end of the spectrum are robots that are fully competitive, and operate in competitive environments. These robots seek to maximize local utility and view the environment and other agents as a zero-sum game. Therefore, they not only seek to maximize their own utility, but may also prevent other robots from achieving their goals. In between these two extremes, are self-interested robots, which seek to maximize their own utility, but are not opposed to working with other robots if it is also in their own interest to do so. There has been a lot of interest in the Game Theory community on providing incentives for self-interested agents to cooperate to the benefit of the team, as described in Section 2.3.

At this point, a discussion of robot design choices is warranted. In order to highlight design tradeoffs, Dudek et al. present a taxonomy for multi-agent systems that classifies them

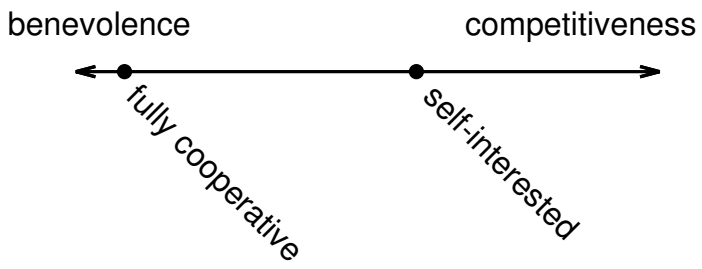


Figure 4: Cooperative multi-robot teams can span the spectrum from benevolence to competitiveness. Self interested agents can be cooperative under the correct incentives.

according to communication, coordination, computation and other dimensions [39]. Dudek posits the conditions over which a task should be solved using one robot or a collective:

- Tasks that require multiple agents (i.e. spatially separate or synchronized tasks, such as box-pushing.)
- Tasks that are traditionally multi-agent (i.e. agriculture and forestry, transportation and delivery).
- Tasks which are inherently single agent (some tasks such as a single task at a specific location that can only be occupied by one robot, would not benefit from multiple robots.)
- Tasks that may benefit from the use of multiple agents - this lies between the extremes. A collective might perform a task faster or more reliably than could a single agent.

Dudek classifies robot teams as a whole by using the following dimensions.

- **The size of the collective** - The number of autonomous robots in the environment.
- **The communication range** - The range is a function of the communications medium and distribution of robots in the environment. Along this dimension, robots may

not communicate at all, only communicate with others nearby, or communicate with everyone.

- **The communication topology** - Example topologies include broadcast, direct address, tree, and graph.
- **Communication bandwidth** - The bandwidth is expressed in relation to the cost of moving robot to different locations.
- **Collective reconfigurability** - Whether the physical arrangement of robots is static, coordinated or dynamic.
- **Processing ability** - The computational model used by the robots on the team.
- **Collective composition** - Whether the robots are heterogeneous or homogenous.

Cao et al. identify Traffic Control, Box Pushing and Foraging as typical multiple robot tasks, and they consider additional dimensions for organizing robots related to the generation of cooperative behavior as follows:

- **Group Architecture** - The architecture determines the overall design of the group system and determines whether the system is centralized or decentralized, homogenous or heterogeneous; it determines communications and sensing structures and capabilities for modeling other agents intents and beliefs.
- **Resource Conflict** - This dimension relates the degree to which conflict exists over shared resources, such as space, communications media and other resources in the environment.
- **The Origins of Cooperation** - This dimension describes whether cooperation is explicitly built in, is emergent or the result of interactions between selfish agents.
- **Learning** - It is desirable for multi-robot systems to learn control parameter values and be able to adjust to their environment.

- **Geometric Problems** - As robots navigate a physical world, they must be able to interact with each other. This dimension includes the ability to perform multi-robot path planning; as well as distributed formation and control.

Dudek's dimensions of collective reconfigurability and collective composition reflect the axes along which robots could dynamically organize themselves into teams, while Cao's origins of cooperation dimension would allow for the use of incentive mechanisms. Taking this further and considering team structure, the levels of cooperation in multi-robot teams is organized into a taxonomy relevant to this thesis, as shown in Figure 5. The classification of robots working on a team can be divided into the centralized and decentralized case. In the centralized case, a group of robots is guided or commanded by a centralized agent. In this case, remaining members on the team are explicitly designed to follow the commands from the central agent and can be modeled as extension of the central agent [151]. In contrast, decentralized multi-robot teams spread the decision making capabilities among the different agents on the team. Teams that are enabled with explicit cooperation are fully benevolent and execute the goals of their designer. They do not have to decide whether to cooperate, to do so is built into the structure of the system. Teams that exhibit emergent cooperation cooperate as a side effect of some local behavior. An example of this type of cooperation is seen in multi-agent swarms, in which team members compute a local function that relates how close they should move in relation to their neighbors. Finally, deliberating cooperative agents are the subject of interest. These agents process information about their neighbors and goals and decide if and how to cooperate with team members. Team-interested robots seek to maximize a team utility, while self-interested robots are concerned with a local utility maximization. Self-interested robots may simply always maximize a local utility, they may calculate incentives for cooperation, or they may be competitive.

3.2 Trust and Reputation Model

As presented above, cooperative agents interact with their team members in order to jointly perform tasks, and to perform them more efficiently than is possible without cooperation.

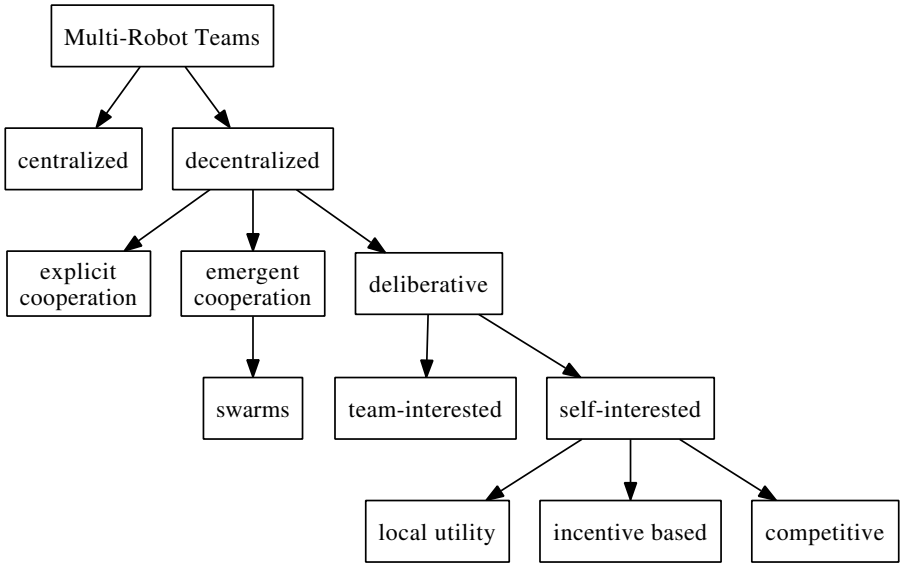


Figure 5: A taxonomy of cooperation in multi-robot teams.

Therefore, cooperation implies a dependence relationship on other team members. Castelfranchi and Falcone describe *trust* as the mental state by which an agent may decide to rely on another agent [28]. For the dependence relationship to exist, an agent first must have a goal that needs to be achieved. Secondly, an agent must have a belief about a team member as to whether that team member is reliable and capable of completing the goal. With both of these conditions: the need for a goal to be completed and the belief that another agent is capable of delivering the goal, an agent can decide to delegate the goal to another agent. Therefore, Castelfranchi and Falcone present *trust* as the "mental counter-part of delegation". In most cases, the mental state of trust is necessary for delegation, and the corollary is often true: delegation implies trust.[1] However, as described above, cooperation and thereby, trust is often explicitly built into multi-robot teams. On deliberative teams, team interested agents consider the global utility of the team and implicitly cooperate with all team members, while self-interested agents cooperate when it improves their utility or there is some incentive to do so. We can also view utilities and incentives in this trust framework: an incentive to have a goal completed, as well as the belief that it will be completed are necessary conditions for cooperation. Therefore, in this work, we build upon

[1]Castelfranchi and Falcone are careful to note that delegation may occur counter to trust (coercive delegation) or without trust (blind delegation).

Castelfranchi's idea that trust implies a decision to rely on another. The mental state of a robot will contain a model of trust for other team members and will be used to decide whether to rely on or cooperate with other team members.

While the terms *trust* and *reputation* are often used interchangeably, we adapt the common definitions [113]. Trust is defined as above to be the mental state that leads to a decision to cooperate. Reputation is the social component of trust, and is the information that is shared between agents about the trustworthiness of an agent. It can be used to calculate trust.

Pinyol and Sabater review the three levels of approaches to trust and reputation to solve the problem of uncertainty of potential behavior in agent communities [113]. These approaches are similar to those used in human societies and are listed below.

- **Security Approach** - At this level, basic properties of identity are considered. This includes approaches from the fields of computer and network security. The systems and communities are secured using identity, authentication, authorization using established theory from cryptography, employing digital signatures and certificate authorities. At this level, systems can be used to control access to a community and guarantee identity, but the *quality* of the information (or the performance of individual agents) is not considered.

- **Institutional Approach** - This approach uses a central hub or authority to monitor, control and enforce behavior in the system. Improper behavior, if noticed by the central authority, can be punished. The enforcement is focused on the set of allowed actions and behaviors. However, each individual agent may have subjective views of quality that are difficult to enforce centrally and uniformly.

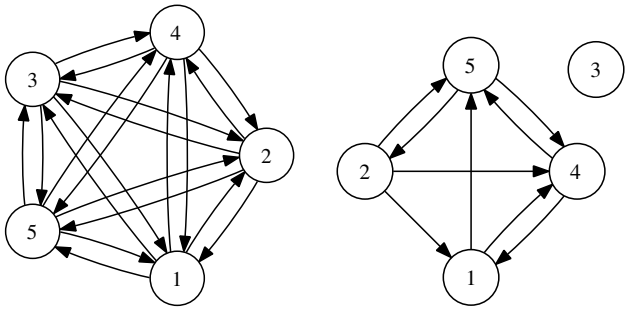
- **Social Approach** - This level uses reputation and trust mechanisms to enforce social norms. Individual agents are capable of monitoring behaviors and punishing non-desirable behaviors of others in the community. Possible punishment may include not selecting partners for future cooperation. This level requires models of trust and reputation and mechanisms for sharing and communicating reputation information.

Each of these approaches covers a different area of trust and reputation in multi-agent systems and all three are active research areas. In this thesis, we are primarily concerned with the social approach, relating how communities of robots can use trust and reputation mechanisms to enforce cooperation and form teams. In later chapters, we relate this social aspect to Kandori's social norm [68] from the game theory literature and consider how teams of robots can monitor each other to build peer models for partner selection. The institutional approach is also of tangential interest, and we consider an approach in which a central trust authority could be used to monitor a team of robots, and contrast that with a fully distributed approach. Finally, practical solutions may contain a mixture of these approaches. For instance, the monitoring and partner selection components of the social approach could be combined with a centralized trust authority.

3.3 Conceptual Model

This section will present a conceptual model for robot teams that work together to perform a set of tasks. Let $R = (r_1, r_2, \ldots, r_n)$ be the full set of available robots. The team selection structure is represented as digraph, with robots as nodes, and directed edges representing a cooperative relationship. The edge, $E\{r_i, r_j\}$ in the graph denotes that robot r_i considers r_j a team member and also that r_i is willing to cooperate with r_j. If the team structure is fixed such that the robots cannot perform partner selection, then the team is represented by the complete digraph, as shown in Figure 6(a). Also, let $R_i = (r_{i1}, r_{i2}, \ldots, r_{ik})$, where $R_i \in R$, denotes a subset of the available robots that a given robot has locally selected as partners. This consists of the edge set $E = \{\{r_i, r_1\}, \{r_i, r_2\}, \ldots, \{r_i, r_k\}\}$ in the team selection graph. As such, each robot may select partners from all available robots, however; the partner selection may not be reciprocal. Therefore, when robots are able to perform partner selection, the set of directed edges in this teaming graph will not form a complete directed graph. An example of a set of robots with each robot performing partner selection is shown in Figure 6(b).

In this work, the tasks are defined to be a set of surveillance tasks, where for each task, a robot must visit an (x, y) position in the environment and perform a sensor reading to



(a) Fixed Team (b) Team Formation

Figure 6: The team selection structure for a set of robots. a) In the fixed structure, each robot considers each peer to be a team member. b) With dynamic team formation, each robot can select the partners with which it will interact using a trust model.

detect the presence of a known target. However, various other tasks could apply to this model. Let T be the full set of tasks to be assigned across all robots, $T = (T_1, T_2, ..., T_m)$. Note that here, the number to tasks may not be the same as the number of robots. Applying similar notation as described in [69], each robot to task assignment is represented by the set $A = (A_1, A_2, ..., A_m)$ where for all tasks, $A_i =< r_i, T_j >$ reflects that a task, T_j is to be executed by a specific robot, r_i. Also, if a subset of the tasks are initially assigned to a robot, denote that initial assignment as $T_{ri} = (T_k, T_{k+1}, ..., T_{k+l})$ for robot r_i.

Multi-Robot System

The multi-robot *system* is defined as the set of architectural components (Robots, Cost Functions, Utility Functions and Communication Architecture) that will be applied to the set of tasks T.

$$S_R = (R, C, U, \zeta)$$

The number of robots on this team could be fixed to include all robots in the initial set R or it could be updated dynamically. Each robot on the team uses a cost function, C, to compute the cost for performing a task. The cost can be based on several factors, including time, distance, resource consumption or risk. Each robot may have a separate cost function, but it is necessary for every robot on the same team to use the same units of currency (such as time or energy necessary to complete a task) for cost calculations.

The multi-robot system also includes a utility function for each robot, which encodes

the probability that the robot will complete the task and collect a payoff. This function is specific to each robot on the team, $U = (U_1, U_2, ..., U_n)$. This function maps the utility awarded to a robot for completing a given task and quantifies the payoff for robots that are self-interested. In the target detection task, the utility is the probability that the robot will detect the target, $P(T)$. The inclusion of the utility function specifies how each team member approaches cooperation.

Another important consideration on multi-robot teams is the communication architecture [39, 151]. Communicating agents can explicitly negotiate over tasks and request assistance. Related to the ability to communicate are the practical limitations of the communications medium: the cost of communication, in terms of utility or resources, protocols, range and reliability. In terms of the robot team model, let $\zeta = (\zeta_1, \zeta_2, ..., \zeta_n)$ represent the communication strategy of each member of the team. This also describes the topology for communication between robots on the team. This includes the range of communication between pairs of robots, whether the robots have the ability to communicate using broadcast protocols, peer-to-peer messaging, mesh networking. The approach to communication affects the ability to assign tasks, perform monitoring and status updates, and to coordinate robots on shared tasks.

Strategy

The strategy architecture, K, captures how a multi-robot system relates assignment algorithms, negotiation, trust and team composition. The task assignment strategy, Γ, determines how tasks are distributed across the robot team, and should also be considered as part of the model. The robot team will require a mechanism for assigning tasks among its members, either in a centralized or decentralized manner. Example assignment strategies include linear programming, currency exchange methods, and multi-agent auctions, as described in Section 2.5.1.

The assignment strategy can also include components for negotiation, trust and reputation peer models, and team composition. Related to communication is the negotiation strategy for each robot, $\Theta = (\Theta_1, \Theta_2, ..., \Theta_n)$. Explicitly cooperative robots have simple

negotiation strategies, while competitive robots may not negotiate at all. Negotiation could also be built upon observation of actions, rather than explicit communication. Negotiation strategies could also include auction protocols and multiple offer negotiation based approaches.

If the robots maintain a model of their peers, they can use that model to reason over the probabilities of behavior of the team members. Models of reliability in multi-robot teams are discussed in detail in [106]. Trust can be used as a metric for the belief held by one robot that another will complete a task. Reputation can be models of trust that are learned over time and shared across the team. If a robot team incorporates such models for reasoning about each team members, it can be included in the robot team model. Let $\Phi = (\Phi_1, \Phi_2, ..., \Phi_n)$ represent the trust and reputation model that each robot on the team employs. Additionally, let the set of observation histories for all robots' actions be

$$O = (O_{11}, O_{12}, ..., O_{1j}, ..., O_{ij})$$

denoting the observation histories of robot j as known by robot i. Each robot on the team can use the models and observation histories of team members' actions to reason about whether to cooperate with other individuals and the community as a whole. Let the full team composition strategy be represented by $\Psi = (\Psi_1, \Psi_2, ..., \Psi_n)$, with each robot having a strategy for partner selection, Ψ_i. This strategy can be used to describe whether each robot is initially part of a team, and how they dynamically select or deselect others to be on their own team, R_i. As shown in Figure 6, robots that explicitly cooperate may have static team structures throughout their experimental lifetimes. However, self-interested robots working in dynamically formed teams will have opportunities for partner selection.

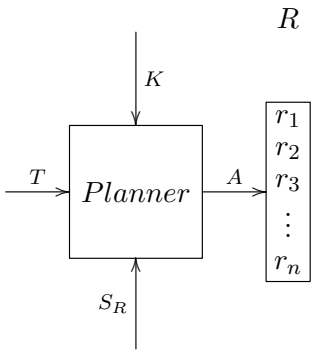


Figure 7: The multi-robot system, tasks, assignment strategy and planner. In the general case, the planner takes as input an assignment strategy, K; the full set of tasks, T; and the robot system architecture, S_R; and generates assignments, A, to the robots on the full team, R.

Putting this all together, the full assignment strategy consists of the negotiation strategy, the peer modeling architecture and the team composition strategy.

$$K = (\Gamma, \Theta, \Phi, \Psi)$$

Planner

The multi-robot planner takes as input the set of tasks, the assignment strategy and the multi-robot system to perform task assignments.

$$A = Planner(T, K, S_R)$$

The planner seeks to maximize the utility of the system (minimize the cost) by using the negotiation architecture, peer modeling, and team composition approach of the task assignment strategy. The general case for this process is shown in Figure 7, for which the task assignments can be performed by any number of methods. For instance the planner could be centralized and task assignments could be performed using integer programming methods.

There are numerous variations to this model that are outside of the scope of this study. For that reason, the following specializations to this model will be considered in this work, as summarized in Table 2. The task assignment strategy will use market based auction algorithms, with the cost function for performing a task being a function of the amount of

Table 2: The Conceptual Model approach used in this work.

S_R : multi-robot system	
R_i	$Robot_i$ team
C_i	Cost function
U	Utility based on time
ζ	Communication Strategy
K : strategy architecture	
Γ	Task assignment strategy
Θ	Evolving negotiation and control strategy
Φ	Trust models from observation
Ψ	Team formation strategy
$Planner$	
T_{Ri}	Tasks are owned by robots, and can be exchanged

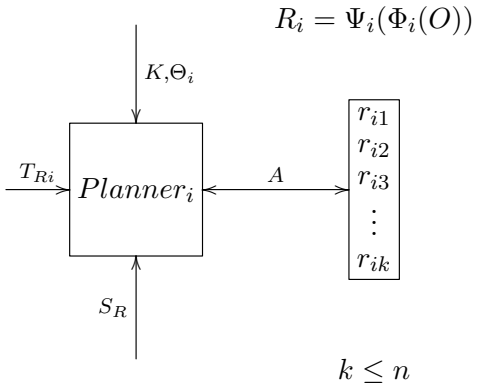


Figure 8: The multi-robot system, tasks, assignment strategy and planner for decentralized cooperation on dynamically formed teams. Each robot runs its own planner, takes as input the set of tasks initially owned by that robot T_{Ri}; an assignment strategy, K; a local negotiation strategy, Θ_i and the set of observations, O. Robots can dynamically adjust their team, R_i using the composition strategy, Ψ_i; peer models, Φ_i; and observations. Finally, robots can request task assignments, A, from others as well as receive task assignments from others through the negotiation and communication mechanisms.

time needed for a robot to complete the task, as well as simple task exchanges. These robots do not exchange virtual currency or tokens upon task completion. The example tasks will be for a robot to visit a 2D location and perform a sensor reading (or to visit a location repeatedly during patrols). It is assumed that each robot is capable of performing the task (although the performance characteristics may vary), and that the tasks are unimodal. It is also assumed that each robot understands a simple protocol for exchanging auction messages. The communication topology is a mesh network, with each robot being able to send messages to others but only within a limited range. Each robot may apply a negotiation strategy to the decisions for cooperation, including tit-for-tat, fully naive and fully distrustful strategies, as well as reputation and community enforced strategies. In addition, each robot can choose to employ evaluations, trust mechanisms and shared reputation data to build models. The planner is localized to each robot and attempts to maximize the robot's localized utility. This more specific conceptual model is shown in Figure 8. It should be noted that each robot can negotiate with others to re-assign their tasks as well. Finally, the robots each can maintain a list of robots with which they will cooperate, applying dynamic partner selection and thus forming their local robot team.

3.4 Cooperative Task Allocation using Incentives

In this section, we present incentive mechanisms from the game theory literature that can be used to elicit cooperation. We include these incentive mechanisms as part of our methodology, and illustrate it with a basic cooperative task exchange example. We also make the observation that unlike some of the assumptions from classical game theory discussions of repeated play, robot teams are not *required* to participate in repeated interactions. That is to say, robot architectures can include decision mechanisms for partner selection (or have the ability to select no partners at all.) We motivate this discussion with the theory from Vanberg and Congleton [159] on the ability to exit from repeated play. In later chapters, we show how this can be combined with Kandori's social norm [68] to punish robots that do not cooperate through the use of trust and reputation mechanisms. This will form the basis for our approach to team formation in multi-robot cooperative task allocation.

3.4.1 Prisoner's Dilemma

The well known prisoner's dilemma (PD) two player game has been used to describe a number of social situations [8]. In particular, the game illustrates the dilemma of a situation in which two players would benefit from mutual cooperation but the rational choice is for each player to not cooperate. An example payoff table for the game is shown in Figure 3, with the dominant strategy for each player being to *defect* (D), regardless of what the other player does. However, Axelrod presented strategies that have been shown to sustain cooperation in repeated play [8], including tit-for-tat.

Table 3: Payoff Matrix for the general Prisoner's Dilemma

		R2 C	R2 D
R1	C	R, R	S, T
	D	T, S	P, P

Prisoner's Dilemma with Exit

Much of the analysis of the prisoner's dilemma assumes that the players are forced to play the game. However, there are many situations in which players are free to choose whether to interact with each other. This version of the game is called the prisoner's dilemma with exit [159] and has been studied widely in various fields, including sociology [102], psychology [57], conflict resolution [140], economics [56], and multi-agent systems [93]. An example payoff table for the updated game, including the exit option is shown in Figure 4. The ability to model PD games with an exit option is interesting in the context of modeling multi-robot interactions, because robots may not be forced to interact with others on the team, but rather could choose partners. If a particular partner is determined to be unreliable, then a robot can choose to refuse interaction and operate alone or seek a matching with a more desirable partner.

Table 4: Payoff Matrix for the Prisoner's Dilemma with exit option

		R2		
		C	D	E
R1	C	R, R	S, T	N, N
	D	T, S	P, P	N, N
	E	N, N	N, N	N, N

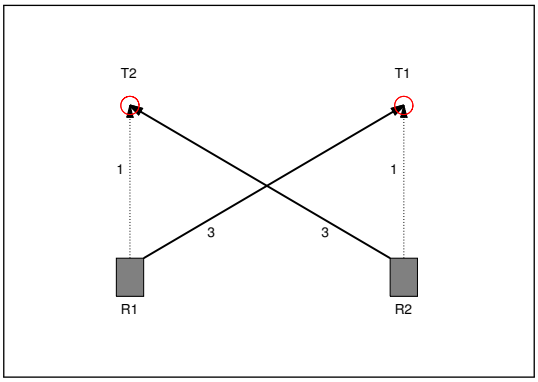


Figure 9: There are situations in which mutual cooperation can result in lower costs for task completion.

3.4.2 When to Cooperate

In task assignment problems, there may be opportunities for agents to exchange tasks with each other, when the exchange would result in each agent having their task completed at a lower cost. Consider a simple example, shown in Figure 10. Each robot has predefined task that must be completed. For instance, robot R1 must complete task T1 and Robot R2 must complete task T2, each at a cost of 3 units. The robots are better positioned to complete each other's tasks, with a cost of 1. However, to exchange tasks, the robots will need to communicate with each other, discover the opportunity for improvement, negotiate the exchange and have an expectation that the other will complete the task.

Assume that when a task is completed, a reward of 2 units is given to the agent that

initially owned the task. This matching can be represented as a 2-player prisoner's dilemma game, with the payoffs shown in Table 5. If the players are able to negotiate the task exchange, where each player *cooperates* with the other and performs the task, then each player receives the payoff of 1. However, if one of the players *defects* by not performing the agreed to task, while the other *cooperates*, then this means that the player that was defected against has to complete both tasks and receives a payoff of -2, while the defector receives a payoff of 2. If both players defect, then each must complete their own task with a payoff of -1. As discussed in Section 3.4.1, the dominant strategy for each player is to defect, resulting in no cooperation. However, cooperation can be enabled if the game is set up as a repeated 2-player game with an infinite (or unknown) number of matches.

Table 5: Payoff Matrix for the Prisoner's Dilemma with rewards for the task exchange example.

		R2: C	R2: D
R1	C	1, 1	-2, 2
R1	D	2, -2	-1, -1

However, in the multi-robot problem setup, there is no requirement that 2 robots interact with each other, they can in fact choose not to interact. In this view, the set of robots in the environment can be thought of as members of a multi-robot team, with dynamic partner selection. At any time, a robot can choose to not interact with another robot in the environment, effectively forming a local subset of the team. This is modeled as a 2-player prisoner's dilemma with exit game, with the payoffs shown in Table 6. If either player takes the exit option, this results in a payoff of N to both players. The value of N can vary; for values of $N \rightarrow 0$, the payoff represents that the player chose not to play with the other player and will likely select another partner, with little loss of resources and continuity. For values of $N \rightarrow -1$, this represents that the player may choose not to play with the other player and would rather perform the task on its own.

Table 6: Example Payoff Matrix for the Prisoner's Dilemma *with exit* and rewards for the task exchange example.

		R2 C	R2 D	R2 E
R1	C	1, 1	-2, 2	N, N
R1	D	2, -2	-1, -1	N, N
R1	E	N, N	N, N	N, N

$0 \geq N \geq -1$

Consider the experiment from Vanberg and Congleton [159], in which repeated play of the prisoner's dilemma with an exit option is simulated. The experiment is reproduced here using Matlab to illustrate the benefit of applying the exit option. The simulation is performed with four types of players, one of each type: AC: always cooperate, AD: always defect, TFT: Tit-for-Tat, and PR: Prudent. Every player is given the chance to play the PDE game against every other player for a number of rounds, using the payoffs shown in Table 6. Each player keeps a history of actions in the previous match for all other players. The AC players and AD players play the same strategy regardless of what the other players have done in the past. The TFT player starts with cooperation, and then does whatever the opponent did on the last time they two met. The PR strategy only plays with other players who have never previously defected, and always cooperates when it plays another player, but exits otherwise. The results are shown in Figure 10. The TFT players quickly switch to defection against the AD players, but are still required to play the game and collect the payoff, -1, in each such meeting. The PR players, after the first round, choose to exit when matched against AD players, and therefore only engage with other cooperative players, resulting in higher cumulative scores. Note that there is a cost with being initially cooperative and learning from the actions of other players, but that this is made up after 7

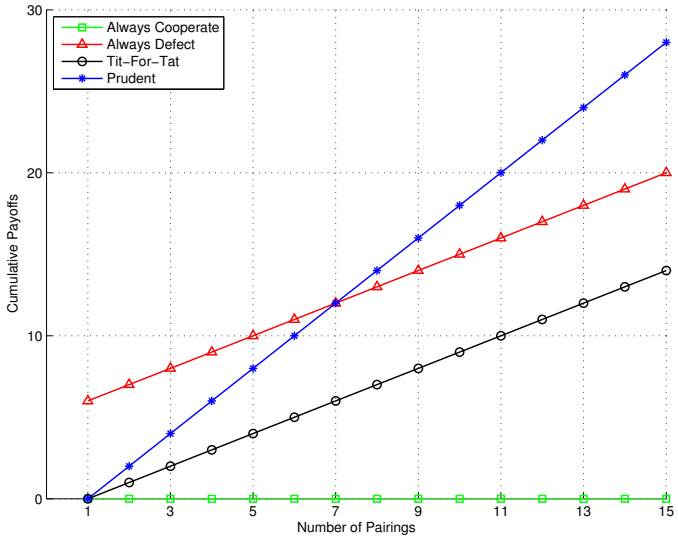
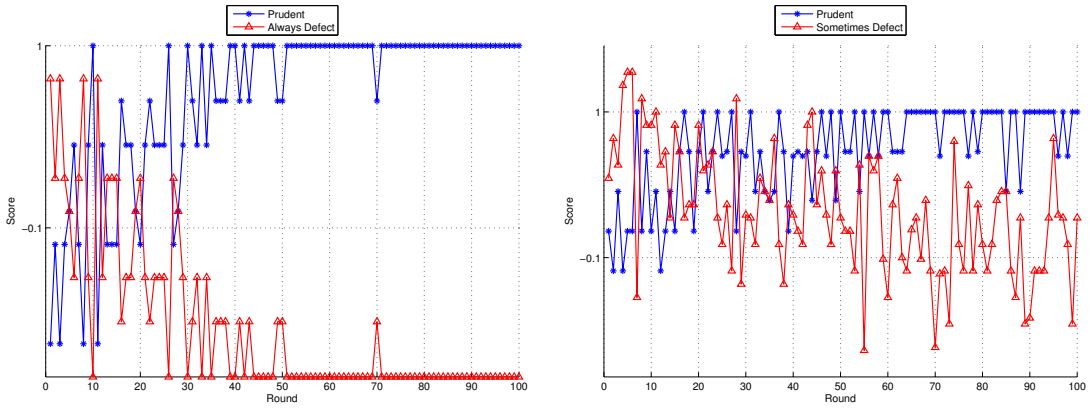


Figure 10: Vanberg and Congleton's experiment illustrating the *prudent* strategy in a repeated Prisoner's Dilemma *with exit* game.

pairings.

We performed additional experiments as part of this thesis in which the prudent agents are matched in repeated rounds against agents that regularly defect, with equal numbers of each type, to illustrate the motivation for the use of the *exit* option as part of the incentive mechanism. In the first experiment, prudent players are matched against players that always play defect. In the second experiment, the defectors sometimes defect with probability 0.5. At each round, every player is randomly matched with another player. If a player exits, they are re-matched until all players have an agreeable match or have refused to be matched multiple times. Next, each player plays the game, receives the payoffs listed and keeps a history of the opponent's action. This repeats for 100 rounds of play, and the results are shown in Figure 11. During the initial rounds, there is a period of learning in which the prudent players learn the histories of the defectors, but eventually have experience with all defectors and effectively isolate them from future play, resulting in higher scores for the prudent players. The defectors that only defect sometimes are more difficult to detect and it takes more observations for the prudent players to detect them.

In the third set of experiments, the problem setup is similar to the above, but the

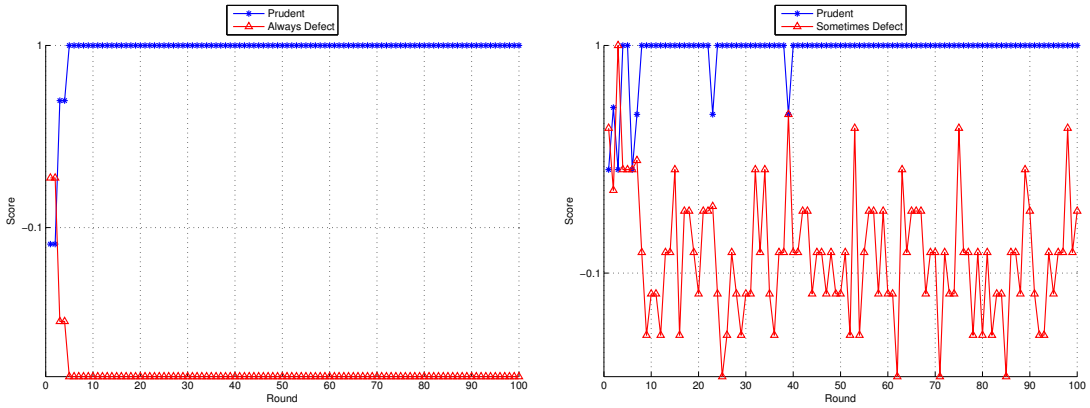


(a) Against agents that *always defect* (b) Against agents that *sometimes defect*

Figure 11: The repeated Prisoner's Dilemma with an exit option. The prudent strategy plays against the a) *always defect* strategies with randomized partner selection. After being exploited by the defectors, the prudent players refuse to select them for future interactions, resulting in higher scores for the prudent players. b) When playing against the *sometimes defect* strategies, it requires more interactions before they are isolated.

histories of player interaction are shared globally with all players using a *reputation* mechanism. After each encounter the with a player, the global history for that player is updated with the result of the last interaction. The results are shown in Figure 12. The prudent players rely on observations from other players rather than direct experience, and are able to quickly isolate the defectors from future interactions. This approach of course assumes perfect communications between all players and that players truthfully report interaction histories; however, it reflects the usefulness of monitoring peer performance and sharing reputation information between players.

These experiments provide very basic examples of interaction between multiple robots performing partner selection for cooperative tasks. Nevertheless, they are instructive because they serve to illustrate the methodology that we apply in this dissertation. When robots are able to *observe* and *model* peer behaviors, share their observations using a *reputation mechanism*, and *perform partner selection* by choosing which team members to cooperate with, they can improving their performance when non-cooperating robots are present, while simultaneously providing *incentives for cooperation*. We further illustrate this methodology in additional experiments and examples using multi-robot teams, in the chapters that follow.



(a) Against agents that *always defect*

(b) Against agents that *sometimes defect*

Figure 12: The repeated Prisoner's Dilemma with an exit option. The prudent strategy plays against the a) *always defect* and b) *sometimes defect* strategies with randomized partner selection and community enforcement. Using the shared reputation information, the prudent players are able to isolate the defectors very quickly.

CHAPTER IV

APPLYING TEAMMATE PERFORMANCE CHARACTERISTICS TO MULTI-ROBOT TASK ALLOCATION

Up to this point, we have discussed the importance for heterogeneous robots to form teams and deliberate over levels of cooperation based on the peer models that are created. However, before we discuss approaches to observation and trust modeling it is important to consider how robots with varying performance characteristics might evolve their teaming and interaction strategies.

When a robot's operators observe that the robot is not performing as well as expected, the first impulse might be to replace that robot with one that performs better or to pause operations and repair the robot before continuing. However, this may not always be possible or cost effective. Consider robots the are operating in remote locations (a mission on the Moon, or on the ocean floor) and cannot be repaired. Further, as described in this dissertation, robots may be operating as part of a federate team, with several different owners (consider a large rescue operation). Finally, the time or constraints may not allow for robots to be replaced or removed from the team, and the existing team should perform as efficiently as possible under these conditions. This chapter presents three examples where robots can adjust their task assignment strategies, based on known performance characteristics of their peers. In later chapters, we consider situations in which the peer performance characteristics are unknown.

In section 4.1, we consider a task assignment function where a team of Unmanned Aerial Vehicles (UAVs) operates with sensors of varying quality and performs task assignment, taking each peer's sensor characteristics as part of the task assignment function. In section 4.2, we again consider an auction based task assignment, where robots underestimate the time for performing tasks and a learning method is used to discover the correct cost function to

use for specific team members. In section 4.3, we present an approach for dynamically allocating robots based on each robot's performance characteristics, in a multi-robot patrolling domain, using the bucket brigades algorithm from the field of operations research.

4.1 A Bayesian Formulation for Task Allocation

This section describes the use of expected utility for including target detection probabilities into an auction based framework for performing task allocation across a heterogeneous multi-robot team. Consider a scenario with multiple robots, each carrying a single sensor. The tasks in this case are to simply visit a location and detect a target. The sensors are of varying quality, with some having a higher probability of target detection. The robots use knowledge of their environment to submit cost-based bids for performing each task and an auction is used to perform the task allocation. The auctioneer assigns task to the robots based on an estimated utility formulation. Analysis and results of experiments with multiple air systems performing distributed target detection are also presented.[1]

4.1.1 Motivation and Problem Statement

The need for assigning tasks to teams of agents with different capabilities applies to many domains; however, the search and rescue domain presents an interesting case in which multiple assets are combined. In a lost persons scenario, there are multiple assets that have different probabilities of detection of the victim, such as helicopters, searching dogs and people spaced out in a grid search [153]. In many cases, the probability of detection by an asset increases with repeated visits to a probable target location. With helicopters, the victim might be obscured by vegetation and may later be visible. Searching dogs can be highly effective, but their effectiveness varies based on terrain, weather and other factors. Teams of people that are sweeping an area with a specific spacing may miss a victim on the first pass and still spot them on the second or third. Of course, combinations of assets and multiple assets are possible. Search organizers can use known probabilities of detection (POD) for assets and asset combinations and consider the effect of multiple passes by sets

[1]The experiments in this section appear in [115].

of assets. For instance, a team of searchers spread out with only 20 feet between them, performing a single pass, are about as effective as a smaller team of searchers that is spread out at 60 feet, but that performs two passes [153]. The problem for search organizers then, is to consider the search assets available, the probabilities of detection for each of those assets and the probabilities that a target will exist in a given location. However, in such a search and rescue setup, there may be only a few victims and therefore detection of false positives are not a big concern, as victims can easily be identified once detected. Furthermore, these assets are suited to a centralized organization model.

In robotics and unmanned systems domains, events may happen more quickly and the environment may be more dynamic in nature. As such, task allocation is better suited for a decentralized and dynamic strategy. Yet, the problem is similar in many ways; the ideal team would leverage all assets available, even if some are more accurate than others. This may mean that some assets make multiple passes of a target to attain the same detection confidence that another might make in only one pass. In the search and rescue domain, false negatives are very important. That is, searchers do not want to miss a target that exists due to sensor error. In other domains, false positives are more of a concern. In a target detection scenario, it is important that a target's existence be verified with confidence, before additional (perhaps more expensive and dangerous) assets are deployed.

Problem Statement

In the basic problem setup there are multiple agents, each carrying a single sensor, varying in quality of detection. An auction mechanism is used to divide tasks among the agents, and the tasks in this case are to simply visit a location and perform an observation with the sensor. Targets are static (non-moving) and will be present only at some of the locations. Also, targets are assumed to be independent of each other. The auctioneer could in principle be any of the agents, but this work will assume a static, external auctioneer that periodically auctions new tasks (locations to visit.) This work also assumes perfect communications between the agents and the agents will exchange bid information with the auctioneer.

A basic auction mechanism that performs distributed task allocation might only consider

Table 7: Probabilities of detection for example sensors

		target_present	no_target
S_1	sensed_target	0.80	0.10
	not_found	0.20	0.90
S_2	sensed_target	0.70	0.20
	not_found	0.30	0.80
S_3	sensed_target	0.95	0.01
	not_found	0.05	0.99

the cost of executing a chosen task (i.e., visit a location and acquire a sensor reading). In such a model it is implicitly assumed that tasks execute and acquire perfect information. Thus there is no integration of sensor characteristics.

In the target detection task, there are two cases that need to be predicted with reasonable certainty: whether a target exists at a given location, and whether it does not exist at the location. When noisy sensors are considered, these cases correspond to the true positive case (the case that a target exists and is sensed) and the true negative case (the case that a target does not exist and is not sensed). Expressed as prior probabilities, these are $P(T|S)$, the probability that a target exists given that it was sensed, and $P(\hat{T}|\hat{S})$, the probability that a target does not exist, given that it was not sensed.

This work assumes that sensors have prior estimates for probability of detection and that these estimates are known in advance. Consider for example, a multi-agent team that consists of three different sensor types, (S_1, S_2, S_3). These sensors return a binary detection value ($positive, negative$). Also, assume that the probability that a target will exist at a given search location is $P(Target) = 0.25$. The Sensor-Target Probabilities for P(S) are given in Table 7, and the probabilities vary for each of the 3 sensor types. S_1 is considered reasonably accurate, S_2 has the least accuracy, with a high false-positive rate, and S_3 is very accurate. Given the prior probabilities, $P(T)$ and $P(S)$, Bayes' rule can be used to find the posterior, $P(T|S)$ as shown in Equation 19. As one might expect, using Sensor $S3$ leads to a very high probability (or confidence) that a target exists if the sensor returns a positive detection.

$$
\begin{aligned}
P(T|S) &= \frac{P(S|T)P(T)}{P(S)} \\
P(T|S_1) &= \frac{(0.80)(0.25)}{(0.80)(0.25)+(0.10)(0.75)} = 0.727 \\
P(T|S_2) &= \frac{(0.70)(0.25)}{(0.70)(0.25)+(0.20)(0.75)} = 0.538 \\
P(T|S_3) &= \frac{(0.95)(0.25)}{(0.95)(0.25)+(0.01)(0.75)} = 0.969
\end{aligned} \tag{1}
$$

4.1.2 Approach

Auction-Based Task Allocation

In the basic auction algorithm, the problem is to assign a number of tasks to agents (ex., UAVs with different sensor characteristics). The tasks in this case are to visit a target location and perform a sensor reading, resulting in a binary detection result ($detection$, $no-detection$). In the auction framework, each robot is a bidder and the items to be auctioned are the visit tasks. For ease of analysis, this work assumes that a central auctioneer exists to allocate the task, however; any of the agents in the system could serve as an auctioneer in a fully distributed implementation. This approach can easily be used on teams with different robot characteristics: each robot knows their own location and cost function and submits cost based bids to the auctioneer. While costs and rewards use the same basis for calculation, no revenue is actually exchanged.

In the auction mechanism, the auctioneer periodically auctions new tasks to each member of team. The agents each maintain a current task list and compute the incremental cost to complete a proposed task. This incremental cost is known as the cheapest insertion heuristic: for each pair of tasks in the current task list, the agent compares the additional Euclidian distance based cost for inserting the new task, and selects the insertion that minimizes that cost. This insertion cost forms the agent's bid. The auctioneer selects the lowest cost bid as the winner of that auction and performs the task assignment. When an agent wins and is assigned a new task, the task is inserted into the agent's task list, again using the cheapest insertion heuristic. Auctions that proceed with a single item being auctioned at a time in this manner are referred to as Sequential Single-Item (SSI) Auctions [70]. SSI Auctions provide reasonable theoretical performance guarantees for the sum of

travel distances and communications, even when using the cheapest insertion heuristic [75].

Multiple Sensor Observations

After an observation has been performed and the posterior probability of the target, $P(T|S)$, is calculated, the auctioneer can determine if the confidence threshold has been met. If the threshold is not met, then the system will require another observation to verify the target, increasing the overall team cost. However, additional observations would lead to increasingly more accurate estimates for $P(T|S)$. For instance, it may be worthwhile to request a sensor to sweep a target location multiple times, or perhaps send separate platforms, each with less accurate sensors if the platform with the more reliable sensor is not available or too costly. The equation for binomial probabilities, shown in Equation 2, can be used to compute the probability that a sensor returned k detections out of n target visits. Joint observations from different sensors are combined using Bayes' rule to get the probability that the target exists, given one or more detections from one or more sensors. This provides a framework for evaluating the probability that a target exists by combining sensor readings from multiple, different sensor visits of a target location. The system designer can place an acceptable detection threshold to determine the number of times a location should be visited.

$$p(k) = \binom{n}{k} p^k (1-p)^{n-k}; k = 0, 1, ..., n \tag{2}$$

As an example, the probability of S_2 returning 1 true detection out of 2 trials as well as the probability of 1 false detection out of 2 trials is given by:

$$\begin{aligned} p(k=1)_{S_2 true\oplus} &= \binom{2}{1}(0.7)^1(1-0.7)^1 = 0.42 \\ p(k=1)_{S_2 false\oplus} &= \binom{2}{1}(0.2)^1(1-0.2)^1 = 0.32 \end{aligned} \tag{3}$$

Also, the probabilities of S_1 returning 1 true detection out of 1 trial, as well as 1 false detection is given by:

$$\begin{aligned} p(k=1)_{S_1 true\oplus} &= \binom{1}{1}(0.8)^1(1-0.8)^1 = 0.8 \\ p(k=1)_{S_1 false\oplus} &= \binom{1}{1}(0.1)^1(1-0.1)^1 = 0.1 \end{aligned} \tag{4}$$

Finally, assume that each of the observations are independent and combine their joint probabilities using Bayes' rule to evaluate the probability that the target exists, given 2 positive detections out of 3 target visits from 2 different sensors in (5):

$$\begin{aligned} P(T|S_1^1,...,S_m^n) &= \frac{P(S_1^1,...,S_m^n|T)P(T)}{P(S_1^1,...,S_m^n)} \\ P(T|S_1^1,S_2^1,S_2^2) &= \frac{(0.80)(0.42)(0.25)}{(0.80)(0.42)(0.25)+(0.10)(0.32)(0.75)} = 0.778 \end{aligned} \tag{5}$$

Expected Utility

A detailed discussion of statistical decision theory applied to analyzing decisions under uncertainty is given by Raiffa[128]. In particular, the notion of expected utility, Eu, is useful for estimating future reward in an uncertain environment. Expected utilities of dynamic, uncertain environments can be modeled as a decision tree, as shown in Figure 13. Each circular node is a chance node, and the possible outcomes or branches of that node are show along with the probabilities. The expected utility is the sum of the probabilities times the reward of each branch. An example using the decision tree in Figure 13 is shown below. For each target location, a target will exist with probability $P(T)$. Upon successful completion, each task delivers a reward, R, and each agent's bid consists of a cost, C, to perform the task. A sensor detection is correct if the target exists and is reported by the sensor or if it does not exist and is not reported by the sensor. Rewards are assigned when the sensor detection is correct for the task and the detection likelihood is greater than the target detection threshold, α.

$$\begin{aligned} Eu(Sn) &= P(T)(P(S|T)(R-C)+P(\hat{S}|T)(0-C)) \\ &\quad +P(\hat{T})(P(S|\hat{T})(0-C)+P(\hat{S}|\hat{T})(R-C)) \end{aligned} \tag{6}$$

This reflects the probabilities at each decision point and the awarded utilities. As an example, assume that a constant Utility of 100 *units* is awarded whenever an agent successfully detects a target and that 0 *units* are awarded otherwise. Now, assume that each sensor's agent calculated their individual cost, per the basic auction framework, to deliver the associated sensor to the target location as: $S_1 = 20$ *units*; $S_2 = 10$ *units*;

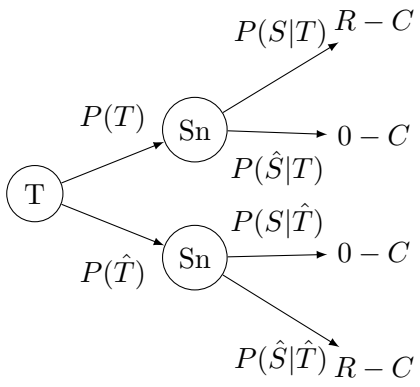


Figure 13: The Expected Utility Decision Tree.

$S_3 = 30$ *units*. Following each path in this tree, the expected utility, E_u, is calculated as shown in Equation 7. In this example, even though the more accurate sensor has a higher POD, the less accurate sensors are closer to the target location, having lower cost, and therefore the Eu values are approximately equivalent.

$$\begin{aligned} E_u(a_1) &= (0.75)((0.90)(80) + (0.10)(-20)) + (0.25)((0.80)(80) + (0.20)(-20)) = 67.5 \\ E_u(a_2) &= (0.75)((0.80)(90) + (0.20)(-10)) + (0.25)((0.70)(90) + (0.30)(-10)) = 67.5 \\ E_u(a_3) &= (0.75)((0.99)(70) + (0.01)(-30)) + (0.25)((0.95)(70) + (0.05)(-30)) = 68 \end{aligned} \tag{7}$$

This expression of cost, incorporating Expected Utility, could be used to inform a bid in the auction framework. As such, the auctioneer selects as the winning bid the agent sensor combination with the *maximum expected utility* for performing that task. The cost for the agent to perform the task is propagated through each branch of the tree to determine the expected utility, E_u. In selecting the best sensor for performing the task, the auctioneer can apply the above Eu calculation to the agent's cost bid, using the known sensor model, to arrive at the Eu of assigning the task to that agent. The auctioneer then simply assigns the task to the sensor that maximizes the Eu for performing the task.

Sequential Analysis

In addition to modeling the expected utility the decision tree can also model the effect of multiple, sequential sensor visits. This can be used to calculate a more accurate expectation in the case of multiple observations. The expected utility as described above can result in

infinite recursion if the depth of the decision tree is not limited. In those situations in which sensors oscillate between correct and incorrect detections (albeit with decreasing probabilities), the likelihood of a target may not reach the desired threshold and the expectation can be analyzed further in the future. Rather than setting an arbitrary depth on the expectation, it would be better to inform the decision process with an understanding of the amount of utility that additional information would provide.

Sequential Analysis, as described by Wald [163], is a technique from decision theory that provides a framework for analyzing the expected utility of repeated decisions. Sequential Analysis is often used by decision makers in business to determine whether to seek more information about a process or to stop sampling and make a decision. In the distributed task allocation problem, it is important for the auction algorithm to assign the most effective sensor combinations to each task by predicting those assignments in advance, with consideration that some sensors may perform multiple target visits if the detection threshold is sufficiently high. However, in practice, the system will observe the outcome of each trial (sensor task) and decide whether that task is complete (stop sampling) or if more information is needed (continue sampling by assigning the task to another sensor.)

In this framework the expected utility tree is performing an estimate of the expected cost for performing the task, including the cost for additional sensor visits if the detection likelihood threshold is not met. The task of visiting a target and performing a sensor detection is viewed as testing the hypothesis that the target exists, given the sensor observations. Each time that a sensor task is complete, a decision is to be made. The choices are to accept the hypothesis and stop the decision process, reject the hypothesis and stop the decision process or perform additional observations. The approach using sequential analysis allows for the hypothesis to be accepted or rejected when the likelihood falls above or below stated thresholds. In the last case, the decision whether to keep sampling is determined by the additional expected utility that further sampling would gain, noted the Expected Net Gain from Sampling (ENGS)[166].

The $ENGS$ is simply the difference between the expected utility with the current observations and the expected utility if an additional sample was to be taken. With this

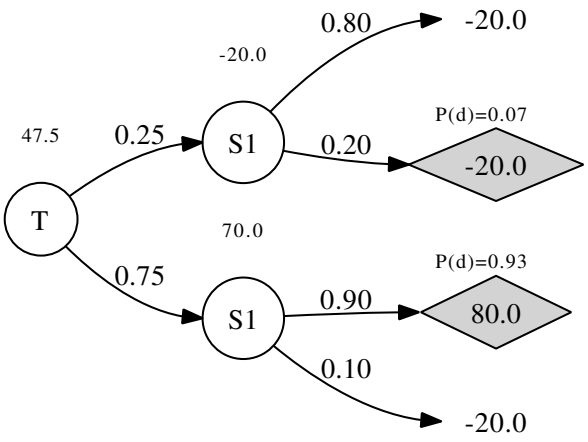
approach, the sequential decision process continues, calculating the likelihoods and $ENGS$ for different sample sizes of n, until the value for n that maximizes the $ENGS$ is found. This is the value that will be used to approximate the Eu for task assignment. For example, with reward and cost values of $R = 100$ and $C = 20$, the related sequential decision trees are shown in Figure 14. After all payoffs are considered (using the Eu), the $ENGS$ of the additional sample is computed as shown in Equation 8.

$$\begin{aligned} ENGS_1 &= Eu(S_1)_{1sample} - Eu(S_1)_{0samples} = 47.5 - 0 = 47.5 \\ ENGS_2 &= Eu(S_1)_{2samples} - Eu(S_1)_{1sample} = 58 - 47.5 = 10.5 \end{aligned} \tag{8}$$

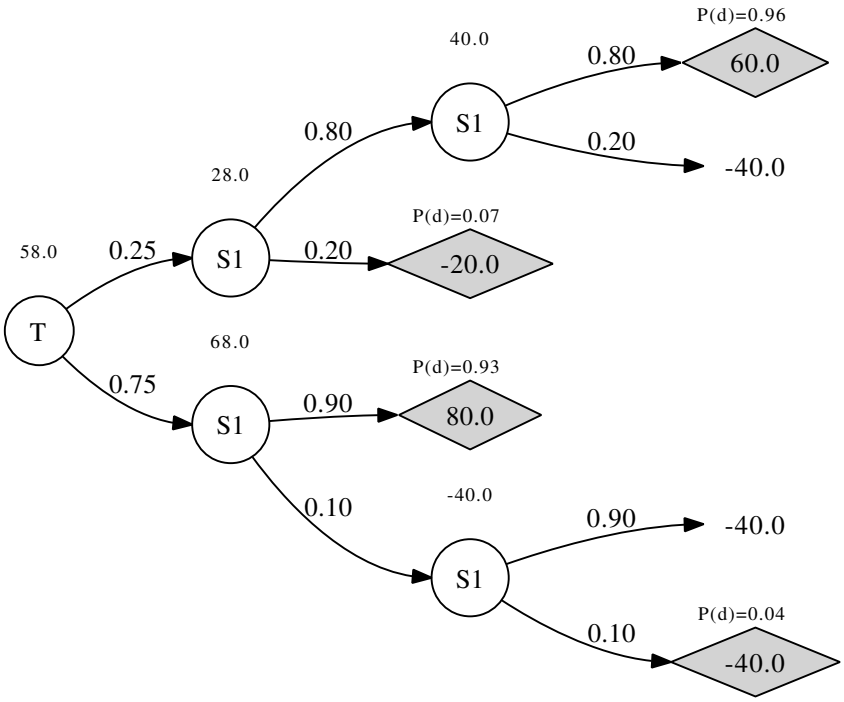
In the above example, the Eu is greater with $n = 2$, but the $ENGS$ is maximized with $n = 1$. Therefore, the appropriate sample depth for the Eu tree is 1. Intuitively, this reflects that it is not worth the resources to continue sampling for little added gain. In other cases, the value for n that maximizes the $ENGS$ value leads to a deeper Eu tree, as shown in Figure 15(c). With the sequential analysis approach, reward values are only granted to the leaf nodes if the detection threshold, α is met. For instance, if an initial sensor visit did not provide the desired confidence, then another observation would be required. If sequential analysis is not used, then the expected utility tree can either be optimistic (often over-estimating the utility) as shown in Figure 15(a) or pessimistic, possibly resulting in negative utilities. The sequential analysis approach therefore provides for a more refined estimated utility.

Discounted Reward and Cost Factor

An issue with relying too heavily on the expected utility approach alone is that it tends to favor the vehicles with the more accurate sensors. This would cause the other sensors to be underutilized. To address this issue, a discounted reward can be used to perform task assignment, as described in [139]. A discount factor, γ, between 0 and 1, is applied to the reward t time steps into the future, $\gamma^t R$. This results in a more even distribution of tasks. The discounted reward factor rewards tasks that are performed sooner, rather than later. Furthermore, this forces some tasks to be re-bid later when the vehicles have fewer tasks in their schedules. This results in a more equitable distribution of tasks as the items are

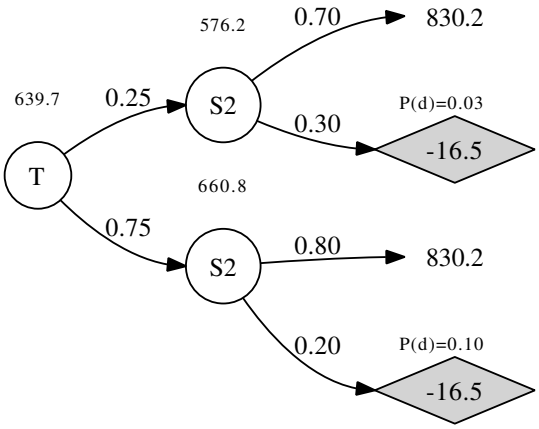


(a) $Eu(S_1)_{1sample}$

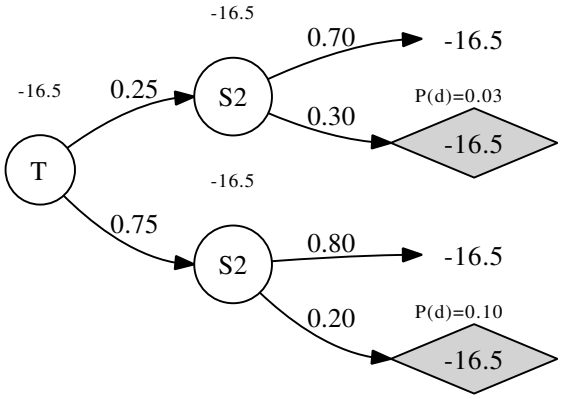


(b) $Eu(S_1)_{2samples}$

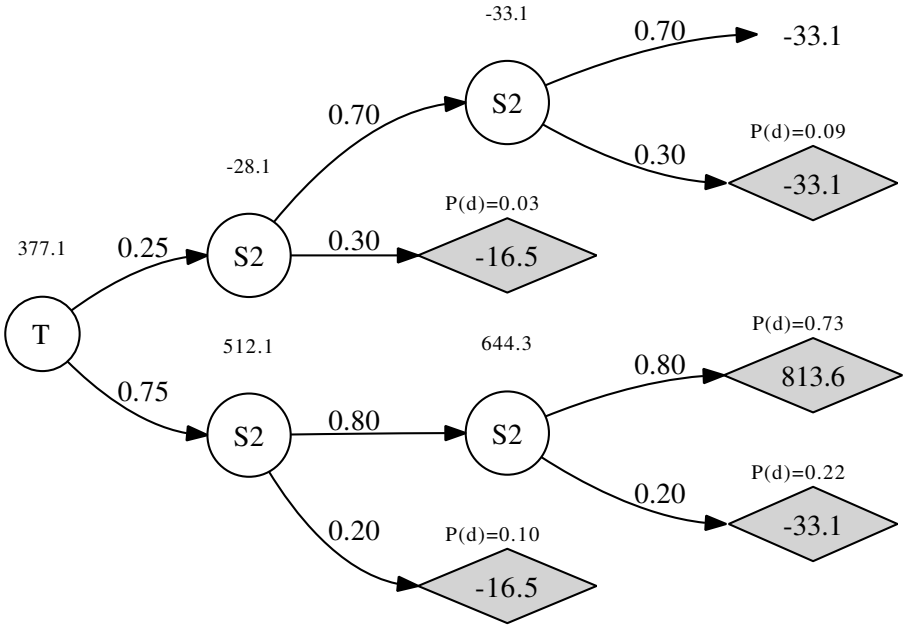
Figure 14: Sequential Analysis: different sample sizes can result in varying estimates for the expected utility for the task assignment. Final leaf nodes are outcomes that met the given probability threshold and are shown as diamond shapes. The cost is propagated to all leaf nodes and rewards are applied at final leaf nodes only.



(a) 1 sample, optimistic



(b) 1 sample, SA



(c) 2 samples, SA

Figure 15: Sequential Analysis vs Optimistic Look-ahead: a) If sequential analysis (SA) is not used, the expected utility can be overly optimistic. b) The sequential sampling approach provides for a more refined estimate of Eu. c) In this example, when using the SA method, the $ENGS$ is maximized when n=2.

re-auctioned until they are won (only bids resulting in $Eu > 0$ will be awarded). At the time they are awarded and assigned, the agents may be closer or have a smaller task list. All of the auction methods in these experiments used the discounted reward factor to ensure that any one agent was not overloaded with tasks.

4.1.3 Experimental Results

Experimental Setup

The MASON multi-agent simulation framework [80], further described in Appendix A.1, is used to perform simulations of multiple UAVs performing distributed task allocation using an auction based mechanism. The simulation consists of multiple simulated UAVs, modeled as points in a 2D plane. Vehicle dynamics and attitude are not modeled in these simulations. The simulation environment was 600x600 units, with target locations randomly distributed. During an experiment, the simulation engine executes for a number of time steps until all tasks are complete. The simulation also includes a centrally located auctioneer which introduces new tasks to the system every n time steps by announcing a new auction. At each time step, each UAV evaluates their position and adjusts their heading toward the next task. The vehicles' velocities are held constant at 1 unit per time step. The UAVs also evaluate their current task list and respond to auction messages with bids.

A UAV's task is considered *complete* when the UAV visits the task location and notifies the auctioneer. The auctioneer calculates the posterior probability that each task was successfully completed by comparing the target detection likelihood, given all sensor visits, $P(d|S)$. If the threshold is not yet met, the task is re-auctioned after r steps for assignment to UAV. Rewards are only granted to the UAVs that successfully complete a task. The simulation keeps a running count of each UAVs cost (expressed as the number of time steps) and reward. UAVs that have no task still accumulate cost at a constant rate.

Detection Threshold Experiments

Experiments were performed in which the following three auction methods were compared against each other while varying the required detection threshold for each task. In the *Basic Auction* method, the cost function considers only the cost for the UAV to visit the

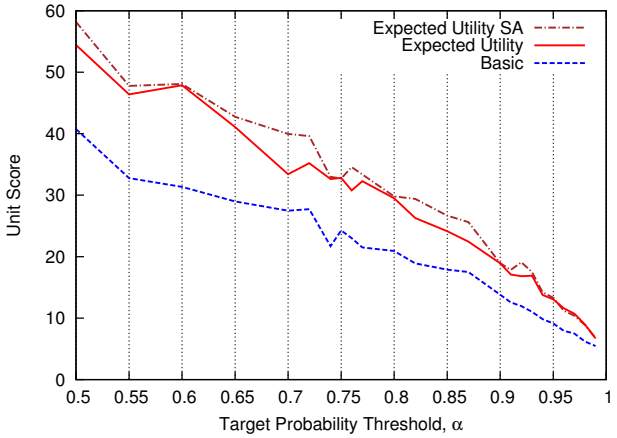
target location and does not consider the sensor qualities. The *Expected Utility* method applied the Eu to the agents bid with the expected utility tree pruned at the first level with optimistic look-ahead. The *Expected Utility SA* method used the ENGS stopping rule to prune the Eu tree.

In the *Expected Utility SA* method, the sequential analysis for multiple sensor visits to a single target models a future visit to the target location, re-using the same sensor type in the decision tree. In theory, any of the sensors could be assigned the visit task and the expectation could be taken over all available sensors. Another option is to cluster each sensor's task locations and assign the closest sensor. The cost bid for future visits could be modeled similarly, or by taking an even distribution of sensors. However, in practice these assumptions work well and can be though of as placeholders for a sensor with similar characteristics and costs.

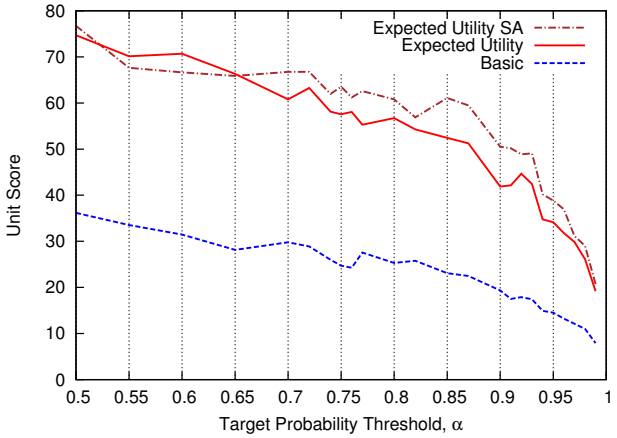
Sets of experiments were performed using teams of 2 and 6 UAVs. The 2-UAV team consisted only of the first two, less accurate, sensor types in Table 7 and the 6-UAV team consisted of two of each sensor type in Table 7. For each target probability threshold value the experiments were performed 30 times with random UAV starting locations. In each experiment, the auctioneer allocated 200 tasks, using SSI auctions. Tasks that did not meet the detection threshold upon completion were later re-auctioned, resulting in additional team cost. The results of each experiment were averaged over all of the runs.

4.1.4 Summary

The results of these experiments are plotted in Figure 16. In both sets of experiments, the global team cost for the Expected Utility methods performed better than when using the *Basic Auction* algorithm. The Eu methods are able to more efficiently allocate sensors to tasks because they explicitly consider the sensor detection probabilities as part of the task assignment. The *Expected Utility SA* method performed slightly better than the *Expected Utility* method with optimistic lookahead. The additional expectation provided by the sequential analysis likely resulted in a more efficient allocation of sensors to tasks by performing a more efficient lookahead into the expected utility of the task assignments.



(a) 2 UAVs



(b) 6 UAVs

Figure 16: Experimental Results: The results of multiple experiments with different values for the target detection threshold, α. The unit score is the total team's reward/cost over the entire experiment. The *Eu* methods for task assignment perform better than the *Basic Auction* method which does not explicitly account for sensor characteristics. The *Expected Utility SA* method using sequential analysis performs better than the *Expected Utility* method using optimistic look-ahead.

Multi-agent teams may consist of agents with different sensor qualities and characteristics. In many domains, multi-agent teams may need to perform cooperatively assign tasks across the team in order to maximize the overall team efficiency. In teams with different sensor capabilities, the sensor characteristics should be accounted for explicitly in when performing the task assignment. When the quality of the sensors on the team varies, it may be better in some cases to assign multiple, less accurate sensors to the detection task, rather than overload a more accurate sensor. In adopting this approach, the auctioneer will sequentially re-auction tasks until the detection threshold is met.

This section described an approach for applying an estimated utility to the task assignment function, along with an approach for calculating the number of future samples to consider when performing an estimate. The experiments show that when expected utility for performing a task is applied to the agent's bid, the overall performance of the team is improved over a basic auction mechanism. Furthermore, we showed that using sequential analysis to maximize the expected gain of additional samples could be used to approximate the appropriate tree depth when calculating the expected utility. Experiments with this approach result in slight performance improvements over an optimistic expected utility calculation, when detection confidence requirements are high.

4.2 Learning Performance in Task Allocation

This section describes an approach for learning which team members perform tasks at costs that accurately reflect the estimated costs. This approach can be used to more effectively perform auction based task allocation by using a reinforcement learning algorithm to adjust a cost factor that is applied to each team member's bid estimates.[2]

4.2.1 Motivation

In the previous section, robots had varying sensor characteristics and the sensor performance characteristics were known in advance. In this section, the performance variance is related to a robot's ability to estimate tasks.

[2]The experiments in this section appear in [117].

An example of learning opportunity costs in auctions was performed in simulation of Martian rovers [139]. The appropriate opportunity costs allow for the specialized robots to avoid becoming underutilized. Over multiple simulations, the different types of robots adjusted their opportunity costs such that neither was underutilized.

Agents learned what valuation to bid by direct observations of similar other agents in [100]. The approach in that work is for the agent to learn to adapt their valuation (and the resulting bid) to market conditions in a simulated, electronic market, using a reinforcement learning algorithm. The market domain was inspired by real world electronic commerce applications in which physical resources, such as trucks, and workers, competed to win tasks. The related problem of learning whether an agent should submit a bid is useful in domains in which computing a bid can be expensive because communication and computation costs can be considerable [25].

Jones, Dias and Stentz investigated techniques for learning proper task cost estimates in oversubscribed domains, using auction algorithms [64]. In that work, each robot attempted to learn their own bid estimates, and had full knowledge of their own state vectors, including their own schedule. In this section, we are interested in learning whether bids accurately match their estimated values, but from the viewpoint of the auctioneer. The auctioneer has less visibility into the state features that can be used to estimate a bid and relies on the estimated vs. actual cost to apply a cost adjustment to future bids.

This section uses a learning approach that is very similar to that used in Kohl and Stone for learning fast gaits on quadrupedal robots [71]. They applied a policy gradient reinforcement learning algorithm to learn control parameters for leg motions. The policy gradient algorithm was also applied by Mitsunaga et. al. to adapt robot behaviors to human partners [91]. This section will apply the policy gradient learner to the task of learning a cost factor for other robots' cost estimates in market-based auction algorithms.

4.2.2 Approach

In this work, the agents each maintain a current task list and locally compute their bid to complete the proposed task. The bid consists of the time-based cost to perform the task.

A potential source of error in task estimation is in the use of an insertion heuristic for calculating the marginal cost to perform a task, in addition to those tasks already assigned. In this section, each robot plans to visit the targets in the order in which they were assigned (using the $O1$ assignment rule from [41]). For each auction announcement received, each robot calculates their bid as the amount of time required to complete the task in addition to those on the current task list. When the winning *bidder* is assigned a new task, the task is appended to the robot's assigned task list.

Learning the Cost Factor

The learning method used in this work is the *policy gradient reinforcement learning* (PGRL) algorithm. This is a reinforcement learning method that is used to estimate the policy gradient when the true value function is not known. The PGRL algorithm is presented in detail by Baxter and Bartlett in [13], and it is shown that this approach converges towards a local optimum.

The PGRL Algorithm

The pseudocode for the PGRL algorithm, adapted from [71] and [91], is shown in Figure 17. At the beginning of the algorithm, the policy vector, Θ, is initialized. In this section, we are using the algorithm to learn a single parameter, θ, which is the cost factor to apply to a robot's task estimation. Therefore, we initialize $\theta = 1$, reflecting the belief that each robot perfectly estimates tasks that they will perform. In the main loop, the algorithm generates a set of random permutations for the policy by adding either $+\epsilon$, 0 or $-\epsilon$ to the policy.

Next, each of these permutations is evaluated by the system and the resulting reward is received. Averages of the rewards are maintained for the permutations of each type. After all of the permutations have been evaluated, the gradient is approximated by calculating the adjustment, a_j, related to each parameter in the policy. Each parameter's step distance, ϵ_j, and the global step distance, η, are applied to a_j and the values are normalized. Finally, the policy is updated with the adjustment.

An example of learning a cost factor, θ, using the PGRL algorithm is shown in Figure 20(a). In this example, the unknown cost factor is 2, and θ is initialized to 1. After about

100 evaluations, the learned policy begins to converge near the true value.

Learning in Auctions

The PGRL is applied to multi-robot auctions by applying the learned cost factor to each robot's bid, as shown in Figure 18. Each auctioneer maintains a PGRL learner for each known team member. After auctioning a task, the auctioneer will receive a set of bids from team members, where each bid represents the time-based cost for the bidder to complete that task. The auctioneer then queries the learner to get the cost factor, θ, related to that agent. This cost factor is multiplied by the original bid in line 3 to get an updated estimate for the agent to perform the task. The resulting bid in the set with the minimum cost is then awarded the task.

When a task is completed by an agent, the auctioneer that assigned the task is sent a message with the completed task information. The auctioneer can then compare the updated estimated cost for the task with the actual cost for completion, as shown in Figure 40. The ratio of these costs is used to determine the reward signal used by the reinforcement learner. The PGRL reward function, shown in Figure 20(a), calculates the scalar reward signal from this value. The reward is used by the PGRL algorithm, as described above, to calculate adjustments to the cost factor.

4.2.3 Experimental Results

Experimental Setup

A set of experiments were performed in simulation to test the cost factor learning approach in a multi-agent auction environment. In these experiments, each robot has 50 tasks that arrive at regular intervals and are sequentially auctioned by that robot's auctioneer. As part of the auction process, they also bid on their own tasks. The robots in the simulation have a limited communications range and can therefore only perform auctions with a subset of the other team members at a given time.

Rewards are given for task completion to the robot that originated the task. Each robot submits bids that represent the time-based cost for completing a task. Specifically, the bid represents the number of time steps until the task could be completed. Once a robot

1: $\Theta \leftarrow$ initial policy vector of size N
2: **while** NOT done **do**
3: $\Theta^T \leftarrow$ t random permuations of Θ
4: **for** $i = 1 \rightarrow T$ **do**
5: Run system using parameter set Θ^t
6: Evaluate reward
7: **end for**
8: **for** $n = 1 \rightarrow N$ **do**
9: $Avg_{+\epsilon,n} \leftarrow$ average reward for all Θ^t that have a positive perturbation in dimension n.
10: $Avg_{0,n} \leftarrow$ average reward for all Θ^t that have zero perturbation in dimension n.
11: $Avg_{-\epsilon,n} \leftarrow$ average reward for all Θ^t that have a negative perturbation in dimension n.
12: **if** ($Avg_{0,n} > Avg_{+\epsilon,n}$ AND $Avg_{0,n} > Avg_{-\epsilon,n}$) **then**
13: $a_j \leftarrow 0$
14: **else**
15: $a_j \leftarrow (Avg_{+\epsilon,n} - Avg_{-\epsilon,n})$
16: **end if**
17: **end for**
18: $A \leftarrow \frac{A}{|A|} * \eta$
19: $a_j \leftarrow a_j * \epsilon_j, \forall j$
20: $\Theta \leftarrow \Theta + A$
21: **end while**

Figure 17: The Policy gradient reinforcement learning algorithm pseudocode, with N policy parameters. During each iteration, t random policies are sampled near the current policy for evaluation. The resulting reward from each sample is used to estimate the gradient and move by a small amount in the correct direction.

Input: The set of posted bids, B.
Input: The set of bidders, A.

1: **for all** $a : A$ **do**
2: $\theta \leftarrow GetTheta(Learner_a)$
3: $B_a^* \leftarrow \theta * B_a$
4: $EstCost_{B_a} \leftarrow Cost_{B_a^*}$
5: **end for**
6: $winner \leftarrow Min(B_a^*)$
7: $AnnounceWinner(winner, a)$

Figure 18: HandleBids() pseudocode. The policy gradient reinforcement learning method learns the cost parameter, θ, for each team member. This cost factor is applied to future bids for that agent.

1: $ActualCost \leftarrow (CompleteTime - StartTime)$
2: $CostFactor_{task} \leftarrow ActualCost/EstCost_{B_a}$
3: $UpdateRL(Learner_a, CostFactor_{task})$

Figure 19: TaskComplete() pseudocode. The estimated vs. actual task completion time is used to evaluate the value of θ in the reinforcement learning algorithm.

finishes all tasks in their list, they no longer accumulate costs in the simulation. The initial locations of the robots and the tasks are randomly chosen for each iteration.

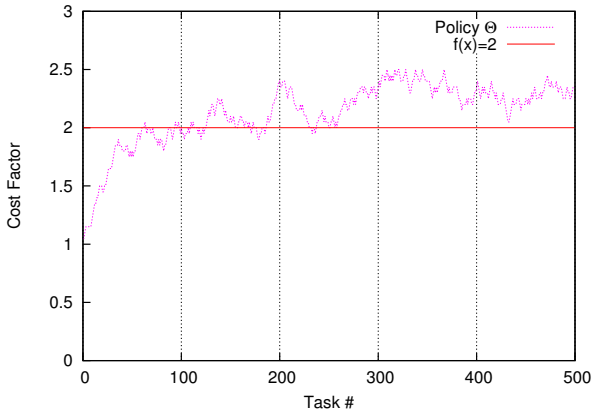
Task Estimation

In this section, the source for estimation error is assumed to be due to poor performing robot having an incorrect model of its own performance capabilities. To simulate robots that bid poorly, a percentage of robots on the team are modeled as *poor performers* by randomly assigning a cost factor at the start of the experiment, using a normal distribution with $\mu = 2$ or $\mu = 3$ and $\sigma = 0.1$. When a *poor performer* bids on a task, the unknown cost factor is drawn from this distribution and is applied to the robot's task performance to simulate error in estimation and execution.

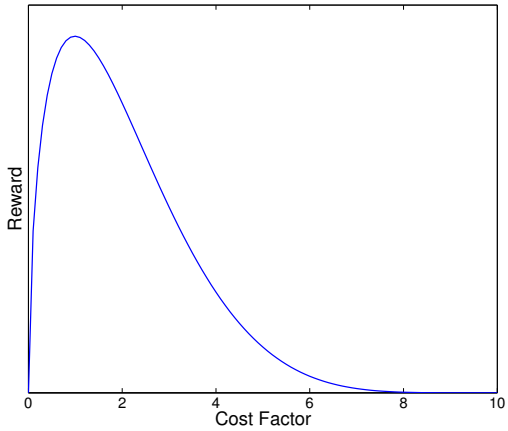
Learning and Applying the Policy

During the learning phase, 1000 auctions were performed with the PGRL algorithm running with a varying number of poor performers on the team in order to learn the cost factors. These experiments used a centralized learner to share the results across each agent's auctioneer.

After the cost factors were learned, they were loaded and a set of experiments were performed to perform auctions using the learned cost factor as the PGRL initial policy. The algorithm continued to learn online, but the step distance, ϵ_j was reduced to minimize exploration. This *policy learner* method is compared against a *naive* auction method which does not consider performance; a *known state* method that has access to the true cost factor for each team member; and a *no cooperation* method in which each team member performs all of their tasks without the benefits of cooperation. For each experiment, the average global score represents the score (*average task reward / average cost*) for the team. For



(a) Learning the Cost Factor



(b) Cost Factor Reward Function

Figure 20: PGRL approach to learning the cost factor: (a) The policy gradient reinforcement learning method learns the cost parameter, θ. In this example, the agent's unknown true cost factor is 2. (b) The reward function used by the PGRL algorithm converts an observed cost factor to a reward value.

each set of experiments, results were averaged over 100 runs.

Results

The results of these experiments are shown in Figure 21. The average global score using each strategy is plotted against teams with 2, 3 and 4 *poor performers* on a team of 6 robots. The error bars represent 1 standard deviation.

The *known state* strategy represents the best score on average that a team could achieve given the number of *poor performers* on the team. This strategy has access to an oracle with knowledge of each team member's hidden state and can calculate the true cost factor

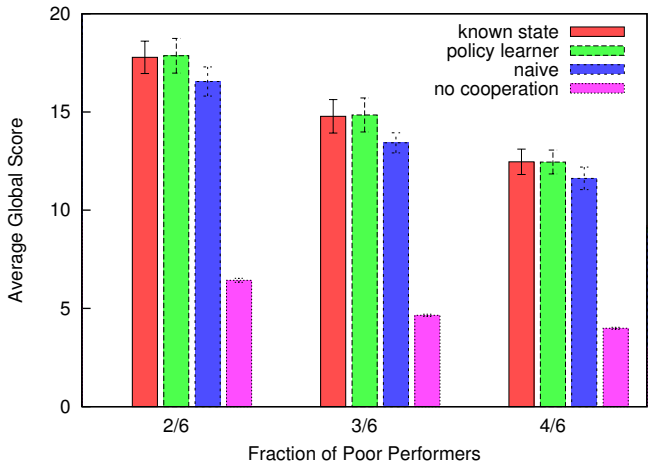


Figure 21: Task Performance: The learning method is compared with a method that knows the state of each agent in advance, a naive auction method that doesn't consider performance and with the case of no cooperation.

to apply to each bid. The scores for the *policy learner* strategy approach the scores of the *known state* strategy. The *policy learner* also scores better than the *naive* strategy which only uses a basic auction and does not consider bid estimation accuracy of each team member. Finally, the *no cooperation* strategy scores the worst, demonstrating that it is better to have poor performers on the team than to work on tasks in isolation.

The results indicate that the *policy learner* strategy can result in up to a 10% improvement in team performance than over the basic auction approach alone. Intuitively, we expected that the gap between the *policy learner* and the *naive* approaches would be larger. However, the auction method distributes task effectively as the *poor performers* get behind. That is, as the number of tasks begins to back up for the *poor performers*, the costs to add a task to the end of their schedule increases, and they win fewer auction assignments. This occurs even though their costs are underestimated. We intend to explore additional bidding and task insertion heuristics using this technique in future work. Nevertheless, these experiments demonstrate that the application of a learner to the cost estimates can be more effective than methods that do not consider estimation accuracy.

4.2.4 Summary

This section presented a reinforcement learning method for recognizing which agents are more likely to submit bids that accurately reflect the true cost for performing tasks. The above experiments showed that a learning mechanism can be effective for detecting poorly-performing team members in auctions, when compared to the naive approach. This may prove useful in situations in which auction based teams are dynamically formed and not all team members are likely to estimate costs correctly. The algorithm learned the cost factor to apply to each team member's bid estimate in a multi-robot auction. The results show that by learning the performance characteristics of individual robots, tasks can be allocated more efficiently.

The next section presents a decentralized coordination mechanism for a team of robots to cooperatively patrol an environment. The robots dynamically allocate tasks, using knowledge about the relative performance characteristics of each teammate. The multi-robot patrolling task has practical relevance in surveillance, search and rescue, and security applications. In this task, a team of robots must repeatedly visit areas in the environment, minimizing the time in-between visits to each. This task can be performed more efficiently by a team of robots, but challenges remain related to team formation and task assignment. In this work, we apply a particular form of task partitioning, a bucket brigade, from the field of operations research to the multi-robot patrolling problem. Using a bucket brigade protocol, individual robots can dynamically partition the work tasks in the environment, requiring little or no communication between robots. A set of realistic simulations using ROS is presented and the results show that a team of robots can effectively use this approach to dynamically partition the patrol tasks.

4.3 Dynamic, Performance Based Multi-Robot Patrolling using Bucket Brigades

It may be desirable to dynamically form teams in some situations (such as after a natural disaster) and approaches to cooperation should incorporate team members with heterogeneous capabilities. In addition, in practice, robot performance may be dynamic in nature:

robots may learn from experience, sensors may be affected by changes in the environment and batteries can run low. Multi robot patrolling approaches should also address these considerations.

The multi-robot patrolling problem is a surveillance task that uses multiple robots to visit every important location in a known environment [31]. Each location in the environment must be visited repeatedly, with the problem being to minimize the time in-between visits. This problem is interesting from a multi-robot research perspective, because it presents challenges in optimization and task assignment, cooperation, communication and reliability. This problem also has useful implications for real world scenarios, including those in the surveillance, security, and search and rescue domains. Cooperation is important in this task, as it is necessary for the robots to work together to improve the efficiency of the system as a whole. An effective multi-robot patrol team should be able to visit points more efficiently and with greater reliability than a single robot.

The bucket brigade protocol is an approach from the operations research field for dynamically partitioning tasks across workers in a production line, based on the relative productivity of each worker. Bucket brigades allow for a fully decentralized task partition and can adjust to dynamic environments. The bucket brigade protocol consists of a simple set of rules that are to be followed by each worker in the system, and results in a balanced production line.

This section presents an approach for applying a bucket brigade protocol to the multi-robot patrolling problem to perform dynamic and decentralized partitioning. The bucket brigade approach allows the robots to order themselves according to relative performance, and requires little or no communication between robots. In addition, the protocol allows for reconfiguration of the patrol partitions if robot performance characteristics change.

4.3.1 Background and Motivation

Bartholdi and Eisenstein introduced the mathematical model and key concepts behind bucket brigades as a dynamical system [12]. Anderson et al. relate the use of bucket brigades for task partitioning in insect colonies [3]. Bucket brigades are used in real world situations

in which new team members are introduced to a production line, and the work is partitioned based on individual performance.

The ability to form dynamic teams has several advantages: robots may be expensive or scarce, and it makes sense to share them across organizational boundaries; robots may need to be organized quickly into ad-hoc teams (such as at disaster locations), and robots should be easily replaced when they fail [63]. Jones et al. present the problem of 'forming pickup teams' of heterogeneous, cooperative robots to perform tasks [63] in dynamic situations.

Regarding cooperative, multi-robot teams, the bucket brigade approach has recently been applied to the multi-robot foraging problem [105, 141, 76]. A simple approach using bucket brigades [105] demonstrated that the robots could cooperate using only local sensing. The use of bucket brigades in this domain also reduces the amount of interference between robots. However, in the foraging task, robots explore a given search area and return items to the home base. In the partition based patrolling task, each robot repeatedly patrols a section of a linear patrol path.

The problem of cooperative patrolling by a multi-robot team has received considerable attention recently in the robotics literature [42, 60, 152, 61, 123]. This problem is similar to the well known 'Art Gallery' problem [90], in which each location in an art gallery must be viewable by a guard. However, here the tasks are to repeatedly visit the locations in the environment and to minimize the time in-between visits.

A theoretical analysis of the patrolling problem is provided by Chevaleyre [31]. The results showed that the problem could be solved with a Traveling Salesman Problem (TSP) approach. This is extended to the multi-robot case by spacing each of the robots evenly along the path [42, 31]. Chevaleyre also showed that it makes sense to partition the graph in some cases, particularly when there are long corridors or edges separating clusters of nodes. Other authors performed experiments in simulation using existing real robot architectures and realistic simulation environments [61, 123]. Recent work on patrolling a harbor using a local method and neighbor interaction is presented in [82].

This section presents a decentralized approach to the patrolling problem, by allowing

each robot along the patrol path to dynamically adjust its partition, based on the performance of its neighbors. We apply a simple bucket brigade mechanism to implement this partition, allowing each robot to rely on local observations.

Multi Robot Patrolling

Many recent approaches to the patrolling task represent areas in the environment with a topological map (a graph) [124]. The nodes in a graph represent areas of interest in the environment, and edges in the graph represent traversable paths between two locations. Applying the notation from the literature, we can refer to the graph as $G(V, E)$, where $V = 1 \ldots n$ is the set of nodes and E is the set of edges. A weight is associated with each edge, $e_{i,j}$, representing the distance between each edge. The graph is assumed to be metric and undirected. Let $R = 1 \ldots r$ be the team of robots to assign the set of nodes in each of the r graph partitions. When the patrol task begins, there is an initial startup time for all robots to navigate to their assigned starting nodes in the graph and to begin patrolling. Robots patrol simultaneously and repeatedly along the graph, visiting their assigned patrol nodes, according to a given strategy [31]. The performance criterion considered in this section is the *refresh time*, which is the time gap between any two visits to the same location[3]. The maximum refresh time reflects the bounds on the effectiveness of a robot team in detecting events in the environment [112].

Chevaleyre presents two main classes of patrolling strategies, the cyclic strategy and partition based strategies [31]. In the cyclic based strategies, a single closed path, s, is generated that visits all of the nodes in the graph at least once. In the single robot case, a robot travels this closed path indefinitely. In the worst case, the amount of time for a robot to visit a node twice while following this strategy is equal to the length of s. Calculating the closed path is known to be *np-hard*, and this problem is closely related to the *Travelling Salesman Problem* [31].

In the multi-robot case, the simplest approach is to space the robots along the closed path such that during the patrol they maintain a constant distance between them [31, 42].

[3]In the literature, this is also referred to as the *idle time* of a node.

Cyclic strategies have known optimality bounds and are preferred when the graph does not contain long edges that connect clusters of nodes [31]. In addition, these strategies have a deterministic behavior and this may not be desirable for security application [123]. From a reliability perspective, when one robot malfunctions, the remaining $(r-1)$team members can simply space themselves evenly over the patrol cycle and continue patrolling. However, there are situations in which robots may have degraded performance, but continue to function. In these situations it would be desirable to allow the poorly performing robot to continue to perform a subset of its original patrol path.

Graph partition approaches divide the graph into subsets of nodes and assign these nodes to individual robots on the team. Pasqualetti et al. present optimality bounds for three major types of partition based patrol graphs: cycles, trees, and chains, and remark that the selection of the roadmap may not be unique for an environment and that the performance can vary based on the choice of the graph structure [112]. For the partitioning case, a cyclic graph can be transformed into an acyclic roadmap using min-max path cover approaches or a chain partition approach. For acyclic graphs, a tree based approach can be used. For the purposes of these experiments, we convert a cyclic roadmap of the environment into an open tour using the approach described in [111].

Tour Creation

We start with the bitmap image file which represents a map of the environment generated from an offline mapping process and use the EVG-THIN software from Beeson [14] to generate a Voroni graph. This is the same approach used by Portugal and Rocha [124]. Next, we perform additional pre-processing on the graph to prune short leaf nodes and to merge nodes that are close together.

From this graph, we calculate the minimum spanning tree (MST) to remove cycles in the graph. Next, we compute an open tour of the edges that visits all of the vertices (nodes) in the graph by starting with a leaf in the MST and visiting each branch of the tree, shortest edges first. The open tour for our test environment is shown in Figure 24.

4.3.2 Bucket Brigades

The bucket brigade protocol for task partitioning is widely used in the operations research and supply chain communities to optimize production lines in a factory or distribution center [12]. The authors showed that this approach works well because it is self-organizing, with a set of simple rules that each worker should follow, resulting in a balanced distribution of work, and yielding optimal throughput. A challenge in the layout of a factory assembly line is to reduce the bottlenecks in the system. Similarly, in the multi-robot patrolling domain, the challenge is to reduce the maximum amount of time in between visits to a node; a single slow robot that increases the max visit time for the patrol results in a bottleneck. Central approaches to distributing work in the assembly line require knowledge of each component in the system and can also be expensive to compute. Furthermore, when the productivity of individual workers changes, the work assignments should be re-allocated. This can be difficult to perform in practice, as it could require significant computation or it may be difficult to fully observe each worker.

In the general bucket brigade, workers maintain a sequence along the production line from slowest to fastest. Each worker moves a product along the line and performs additional steps towards its completion. Without this ordering, it is possible for a faster, more efficient worker to become blocked by a slower worker.

When the last worker completes the product that he is working on, he moves in the reverse direction down the production line until he meets the next slowest worker and takes over the work on their product. This happens recursively back down the line until a new product is started by the slowest worker. The faster workers set the pace of the production line by "pulling" products from their slower neighbor, toward completion.

The bucket brigade approach has the advantage that workers dynamically adjust their partitions along the line in a decentralized manner, without intervention from a central process. In addition, little or no communication is required between the workers, once the process has begun. Once the workers have been sorted along the line, a simple protocol is used to transfer unfinished product to the next faster worker. Finally, the process is simple to implement and requires only a small amount of computation at each node.

By analogy, we consider the bucket brigade production line to be the open tour that must be partitioned for the multi-robot patrolling problem. In this analogy, the robots are workers, ordered from slowest to fastest and the work to be performed is to visit the patrol nodes along each section of the line. However, we further consider the case in which robots are not initially ordered along the path. Armbruster et al. present a bucket brigade production system in which workers learn and improve their performance over time [6]. In a practical example, when a new employee is added to a production line, he starts at the slowest end; however, as a the worker's performance improves, the ordering of the workers along the line may need to shift. Armbruster showed that if the bucket brigade system allows for re-ordering of its workers, the system will rebalance itself without central intervention. This ability to self organize is desirable in a multi-robot team as individual robot performance may change over time: batteries may discharge at different rates or sensors may have different capabilities. Furthermore, it may be desirable to be able to add new robots to a team during production and have the team adjust itself to an optimal configuration. In the next section, we apply the dynamic bucket brigade approach to a robotic team, using an example from nature as a motivating example.

4.3.3 Approach

An example of bucket brigades in nature was observed in ant colony behavior [3], in which the insects organized themselves for collecting food, without centralized control. Rather, coordination behavior emerges through a simple set of rules. The ants are not initially ordered as in the general bucket brigade case, but enter and leave the production line dynamically. However, the ants follow a basic protocol that allows for the exchange of tasks, and resulting division of the production line, through the use of an externally visible label, namely the size of the ant. In this case, the production line is a path from the ants' home nest to a food source, such as a location containing seeds that should be carried back to the nest. This process is illustrated in Figure 22, with the ants being replaced by robots to adopt this analogy to the multi-robot patrolling scenario.

Each ant begins their work independently, by heading along the path toward the food

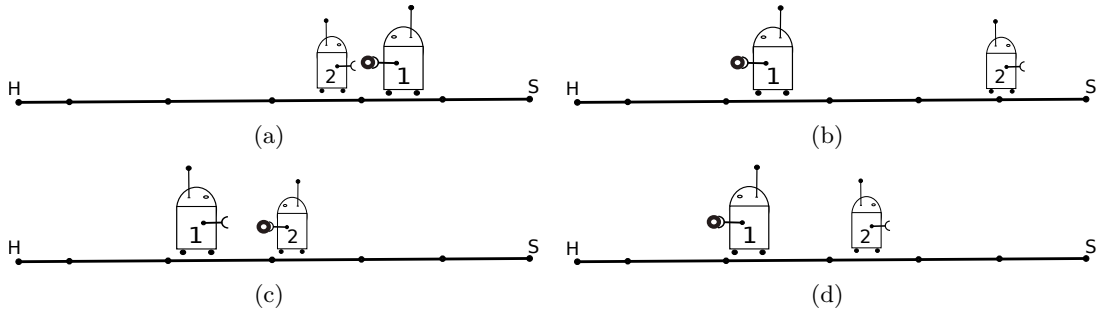


Figure 22: An example of patrol node partitioning using a bucket brigade with two robots. The virtual food source is shown on the right side of the path with the home on the left. (a) Robot 1 returns from the source with food, and (b) continues home after meeting a slower robot 2. (c) The robots later meet again as robot 2 returns from the source with food. (d) The faster Robot 1 takes the food and returns back to the nest, while robot 2 returns to the source, resulting in the robots covering different areas of the path based on their relative patrol velocities. The robots order themselves by speed, with the slowest being closest to the food source.

source. If it encounters no other ants along the way, it picks up a seed and returns to the nest. It important to note that the ants could each carry out this process independently and in parallel. However, ants have also been observed working together by performing a bucket brigade protocol which divides up the path to the food source [3]. The larger ants are able to carry a seed a farther distance in the same amount of time as a smaller ant, and are organized at the end of the process (near the home nest.) As an ant is returning to the nest, if it encounters a larger ant heading in the opposite direction, the larger ant wrestles the seed away, and returns to the nest, while the smaller ant must return to the food source. In this manner, the production line becomes divided, and the workers become sorted by size (which correlates to their speed), after the initial organization period.

At this point, we can describe in more detail the bucket brigade protocol, applied to the multi-robot patrol chain. A team of robots are available to perform the tour, and the size of the team can change dynamically. We note that is not necessary for the robots to explicitly communicate; they can instead rely on a publicly visible label for each robot that displays a robot's speed, the direction of motion along the line (forward or backward) and whether the robot has a product that needs to be completed. We further assume that when a robot is added to the team, it begins patrolling from the start of the chain, the *home* node location using the insect analogy, and begins traveling in the direction of the last node along the

chain, the *food source* node in the insects example. In addition, for illustration we employ the use of the virtual food item, although no tokens are actually exchanged. When a robot encounters another robot along the tour, the following protocol is observed, refer to Alg. 1.

The robots follow the patrol tour, described in Section 4.3.1, to visit the nodes. Each robot is constrained to travel along the same path, and two robots traveling along the path in opposite directions will eventually meet. When they do, each robot inspects the other robot's label. The label provides the information about the robot's current production speed, the direction that the robot is heading, and whether the robot is returning with virtual food. The production speed, $\dot{x}_r$, for robot r represents the maximum forward velocity of the robot as it navigates in the environment.

If a robot is heading away from the end of the production line (the *home* node), toward the *source* node, and is patrolling at a production speed greater than or equal to the other robot, it performs a *handoff* (lines 7-8) by taking taking the virtual food and reversing direction. Likewise, the slower robot submits to the *handoff* (lines 12-13) by reversing course and returning toward the *source* node.

When each robot in the environment follows this protocol, the robots will eventually become sorted by production speed, with the faster robots being located closer to the home node, and covering more patrol nodes per unit time. This approach also includes the addition of the passing rule to allow a faster robot to move in front of a slower one along the tour. Therefore, the order in which robots begin patrolling is not important, as they will order themselves along the tour. When a faster robot encounters a slower robot headed in the same direction, the faster robot is able to pass; however, if the faster robot meets a slower robot returning with food, then the slower robot must return to the food source. In addition, if a robot's speed changes or if robots are added or removed from the team, then the robots will dynamically reconfigure their positions along the the patrol chain.

Observation Only

The approach described above relies on the externally observable label that each robot displays[4]. When a robot meets another robot along the patrol tour, it inspect the other's label and simply follows the bucket brigade protocol. However, this leads to some inefficiencies in the cumulative patrol paths. In the multi-robot patrolling problem, the refresh time of each node is the performance metric and the traversable edges do not have to be covered by a robot path. This leads us to an improvement over the basic approach: the bucket brigade direction reversals can be made to occur at node locations if we introduce communication.

Explicit Communication

In the basic bucket brigade approach, robots handoff work or *reverse* direction, according to the protocol in Algorithm 1, when they *meet* by coming into close contact with each other, possibly along an edge in the graph. However, it is more efficient to reverse direction when robots visit adjacent nodes. Therefore, If robots have the ability to send messages to each other, even if the range is limited, the algorithm can be improved to reduce unnecessary edge traversals during bucket brigade handoffs. Each robot periodically sends a message to each of its neighbors, including the label information, the identifier of the next node being visited, and the estimated time of arrival at the next node. We can amend Algorithm 1 with the following cases for determining how robots meet. In all three situations, the robots are heading toward each other from opposite directions, with one of the robots returning from the food source.

robots are approaching the same node The robots are approaching the same node. In this case, the robot that will reach the node first continues to the node and reverses directions after visiting the node. The other robot reverses direction immediately.

robots are approaching the same edge The robots each are approaching nodes that are joined by an edge along the tour. In this case, each robot plans to reverse directions after visiting the node.

[4]The label information could also be implemented with explicit communication or through access to a central monitor.

robots are on the same edge The robots each are approaching each other along the same edge in the tour. In this case, each robot reverses directions immediately.

For partitioning to be effective, it is important to remove long edges in the graph between nodes in different partitions [31]. Therefore, the ideal communication radius should be greater than the longest edge in the graph. If the range is shorter than the longest edge, then the last case will allow the robots to reverse direction once they become within range.

```
   input  : An open tour N of nodes
 1 while Patrolling do
 2 |   move in direction along N;
 3 |   if meet other robot, R_j then
 4 |   |   read label R_j;
 5 |   |   if  ẋ_i ≥ ẋ_j then
 6 |   |   |   if direction == S and hasFood(R_j) then
 7 |   |   |   |   take food from R_i;
 8 |   |   |   |   reverse(direction);
 9 |   |   |   end
10 |   |   else
11 |   |   |   if direction == H and hasFood(R_i) then
12 |   |   |   |   submit food to R_j;
13 |   |   |   |   reverse(direction);
14 |   |   |   end
15 |   |   end
16 |   end
17 |   if n == H ∈ N then
18 |   |   drop food;
19 |   |   reverse(direction);
20 |   else if n == S ∈ N then
21 |   |   pickup food;
22 |   |   reverse(direction);
23 |   end
24 end
```

Algorithm 1: Bucket Brigade Protocol for Patrolling

4.3.4 Experimental Results

To demonstrate this approach in a multi-robot patrolling scenario, we implemented the protocol in a robot simulation using the Stage multi-robot simulation environment [160], shown in Figure 23. This section describes a set of experiments in simulation using a team of robots. Our results show that the the bucket brigade protocol can be effective in a low

communication setting, when robots have different performance characteristics.

Simulation Setup

The open-source Robot Operating System (ROS) architecture [127] was used to implement the robot messaging, low level control, and behaviors. Each robot uses the ROS navigation stack for navigation, localization, and obstacle avoidance. Each robot also runs a custom *Patrol* behavior which implements the bucket patrol protocol, and repeatedly navigates to the nodes in the robot's patrol path. We allow the robots to access the true localization information. Messages between robots are sent using broadcast messages over the local network.

Each robot is given a map of the test environment with 5cm resolution, as shown in Figure 24. A topological graph of the environment is generated from this map in advance and is also provided to each of the robots. The nodes in this graph represent the areas that are to be visited during the patrol. At startup, each of the r robots on the team locally calculates an open tour from the MST of the graph, as described in Section 4.3.1.

Simulation of Bucket Brigade

In the first experiment a team of three robots performs a multi-robot patrol using the bucket brigade protocol, using Alg. 1. This protocol relies on the observation of *labels* by robots on the team. However, we implemented this in the simulation, using a basic *label* message that each robot broadcasts over the network. The robots in the simulation can access these messages when they are within one meter of a teammate.

Table 8: The production speed for each robot, in ms^{-1}.

$\dot{x}_1$	$\dot{x}_2$	$\dot{x}_3$
0.25	0.20	0.15

This experiment was run for two simulated hours. Each robot was limited to a different production speed, which corresponds to the maximum allowable forward velocity for the robot's controller, shown in Table 8. The *home* and *source* locations are represented by the first and last nodes on the open tour, respectively. The robots were started from an initial

location in the environment, ordered from fastest to slowest. Each robot had an initial startup time to navigate to the first node on the tour and begin patrolling.

After the first visit to all nodes along the tour, the robots reached the *source* location and returned home. On the second trip back to the *source*, the faster robots each encountered a slower robot returning from the source and performed the bucket brigade protocol to perform a handoff, and reversed directions. From this point on, the robots maintained the ordering from slowest to fastest (starting from the *source*). The trajectories of each of the robots over this experiment, after the initial ordering occurred, are shown in Figure 25. The trajectories show a partitioning of the robots into different sections of the environment, with overlap on the partitions. As expected, the slowest robot covers a shorter section of the tour, with the next fastest robots covering more area. A different view of the trajectories is shown in Figure 37(a). Here, the cumulative distance along the tour is plotted against the time since the beginning of the patrol. The handoffs during the patrol are visible in the trajectories plot at the locations where two trajectories intersect and then reverse directions. Observe that the partitions are not statically preserved along the tour. Rather, to use the bucket brigade metaphor, when the fastest robot returns to the *node*, it heads back along the production line, and seeks out the next robot to "pull" work down the line.

Simulation with Explicit Communication

In this experiment, the robots have the ability to send explicit bucket brigade messages to neighbors, as described in Section 4.3.3. The trajectories from this experiment are shown in Figure 37(b). In this case, the robots are able to avoid unnecessary edge traversals during the *handoff* phase of the bucket brigade protocol. This can be seen in the gaps between trajectories at the points where the robots reverse directions. By shortening the patrol tour for the robots, this has the effect of reducing the maximum refresh time of the nodes.

4.3.5 Summary

This section presented an approach for applying a dynamic task partitioning mechanism from operations research to the multi-robot patrolling problem. The key benefits of this approach are that it is fully decentralized and can be implemented locally with a set of

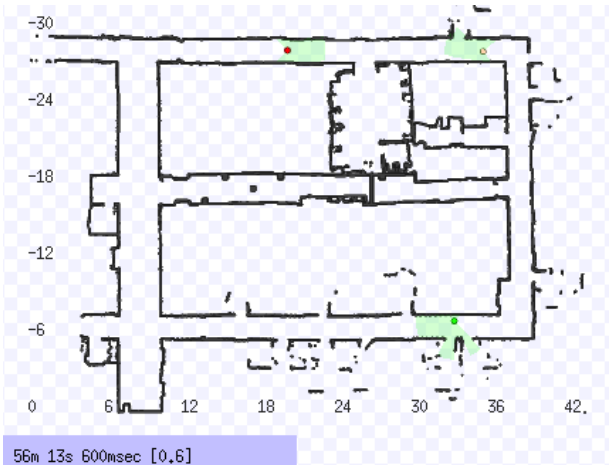


Figure 23: The Stage multi-robot simulation is shown with three robots using ROS.

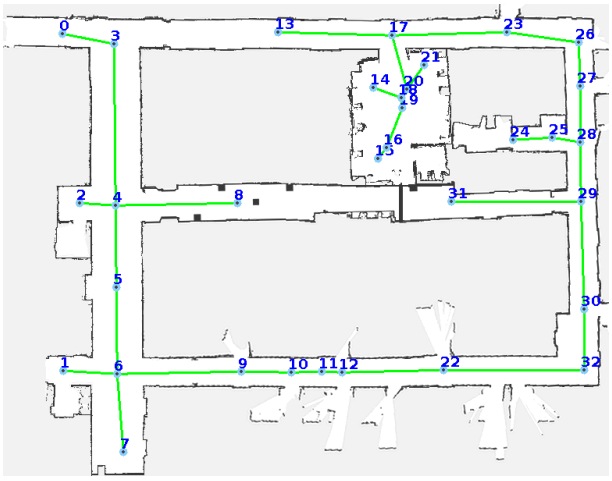


Figure 24: Starting with a cyclic graph of the environment, the MST is calculated to remove cycles and then the following open tour is generated for patrolling: [0, 3, 4, 2, 4, 8, 4, 5, 6, 1, 6, 7, 6, 9, 10, 11, 12, 22, 32, 30, 29, 31, 29, 28, 25, 24, 25, 28, 27, 26, 23, 17, 13, 17, 20, 21, 20, 18, 14, 18, 19, 16, 15].

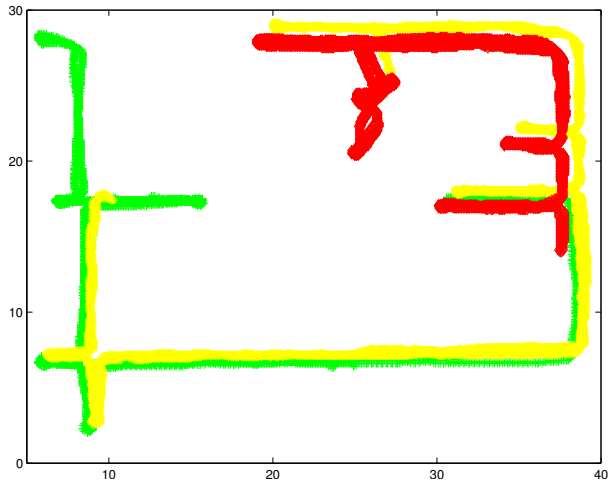
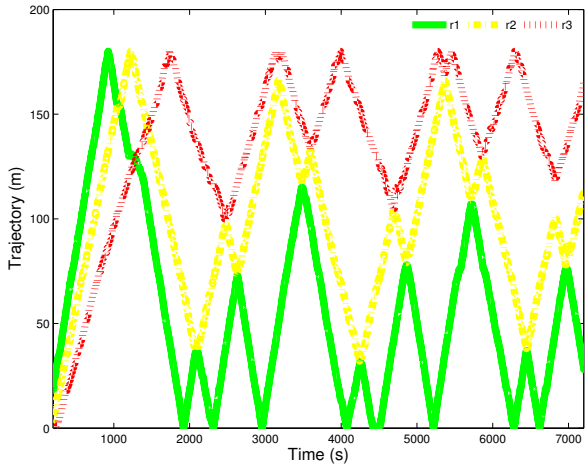


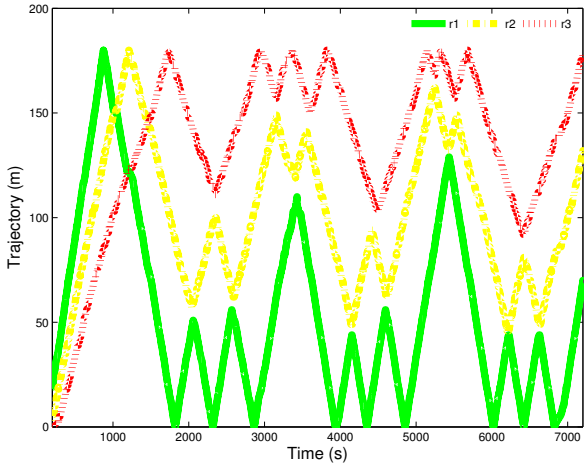
Figure 25: A set of trajectories is shown for a team of three robots that used the bucket brigade protocol to partition the patrol chain. After initial startup, the robots patrolled and dynamically partitioned the environment amongst themselves. (The trajectories are shifted slightly to make them more visible.)

simple rules. In addition, this approach requires little or no direct communication between robot team members. The protocol is responsive to changes in individual robot performance and will adjust partitions as robot performance changes. Finally, robots can be added or removed from the environment, and the task partitions will stabilize to a near-optimal partition. This may prove useful in situations in which multi-robot teams are dynamically formed or when not all team members are likely to perform effectively over time. The experimental results showed that a team of robots with varying performance characteristics could be introduced into a patrol environment and dynamically partition themselves over the patrol chain with no centralized coordination.

In the bucket brigade experiments, the performance level for each robot was given, using an observable label of the performance value for each robot. If the performance capabilities of a robot are unknown, this approach could also be applied using a trust model. In this case, each robot would query a trust authority to determine the established trust level for the peer robot, and use that trust value in the bucket brigade algorithm. We present our trust model approach in Chapter 6. Similarly, for the experiments using expected utility from Section 4.1, the quality levels of the sensors are given; however, in dynamically formed teams, the sensor characteristics for robot peers may be unknown. We consider such an



(a) Observation Only



(b) with Communication

Figure 26: The trajectories are plotted for the team of robots using the bucket brigade protocols: (a) using only observation and (b) using explicit communication. At startup, the faster robots reached the end of the tour first, but the robots later were ordered by speed.

example in Section 6.3, where the sensors with the same characteristics as those in Section 4.1 are used, but in this case, the characteristics are unknown and the relative reliability must be discovered through observation and the application of a trust model.

CHAPTER V

ROBOTS MONITORING TEAMMATE PERFORMANCE THROUGH OBSERVATIONS

In the previous chapter we considered some situations in which the robots' performance characteristics were given or could be estimated, and provided examples for using these characteristics to adjust the task assignment function based on each individual's characteristics. In this chapter, we consider situations in which a robot's performance characteristics are unknown, and could be changing or deteriorating over time. Traditional health monitoring techniques call attention to an operator or assume a binary classification of either success or failure. Robots that can identify poorly performing team members, as performance deteriorates, can adjust the task assignment process dynamically.

In section 5.1 we define some terms that describe the types of robot interactions that are to be monitored. In section 5.2, we discuss approaches that allow robots to monitor the performance of their peers. Section 5.3 presents the tradeoff involved with the cost of monitoring when there may be poor performers on the team. In section 5.4, we investigate the use of statistical process control charts from operations research as a tool for monitoring team member performance as part of a multi-robot task assignment framework. In section 5.5, we present an approach using control charts with a centralized monitor for robots performing a multi-robot patrol on an indoor robot platform. Our experimental results show that the monitor can detect poorly performing robots and reassign tasks to others that are known to be reliable.

5.1 The Purpose of Monitoring

The purpose of monitoring is to discover which peers or team members are performing as expected by a robot, robot team or central authority. A *peer* robot is one that is available for partnering and interaction, but may or may not be trusted. Before a robot can decide whether a peer can be trusted, it requires a series of observations from which to build a

model of that peer's behaviors. An *observation* is a single instance of behavior or interaction which was observed by another party and can be used to update a trust model. After a sufficient number of observations have been gathered using a monitoring process, then a peer can be classified based on this model. A robot or agent *defector* is one that regularly violates the trust of its peers by not performing or behaving as expected. We can also refer to this type as a *poor performer*. Conversely, a robot that behaves as expected by peers is a *cooperator*, and when it performs as expected, it is referred to as a *good performer*. We may consider the labels of cooperator, defector, performer and poor performer to be the unknown behavioral states from which robots derive their behavior, and these states cannot be observed directly. Finally, when we update a trust model from the observations gathered using a monitoring process, and use the model to decide whether to rely on other robots, we can classify a robot peer as being either *trusted* or *untrusted*, and the set of *trusted* robots forms a robot *team*. We discuss the trust model further in the next chapter.

5.2 Monitoring Approaches

Approaches to monitoring depend on the environment, but may include human observation, observation by other robots, computer vision based techniques, and RFID tags for logging visits to locations. Here, we only consider that a monitoring technique is available for use by the system and that it can reliably report when a particular robot visits each node. Each of these approaches is discussed further below.

- **Self Reporting** - Robots can self report their status and task completion. In this approach, we assume that robots are able to communicate with the monitoring mechanism, and can report status updates in a timely and truthful manner. If we are concerned with dimensions of robot performance rather than communication abilities or deception, then this may be sufficient. This approach has the benefits of being simple with a minimal cost for implementation.

- **Human Observation** - In some cases, human observers may be available and distributed throughout the environment for observing robots and reporting the successful completion of tasks. For instance, in a sensor observation task, a human can verify

the existence of an object that was reported by a robot, and may be easily verified. However, for some performance dimensions, this may be subjective. As an example, in a quickly executing environment, a human might not be able to verify the correct action for a robot to take. In addition, it would be difficult to place human observers sufficiently in large, remote or hostile environments.

- **Robot Peer Observation** - Robots themselves may be used to observe the behavior of their peers. In situations where robots observe and report on their peers, direct communication may not be desirable. Novitzky, et al. present an approach for underwater robots to monitor their team mate trajectories using onboard sensors and predict the behavioral state of a team member, when direct communication is not possible [96]. Similar to the human observation case, a robot with a trusted sensor can be used to verify observations made by one of its peers. We define a robot shadow as a trusted robot that can be used to verify the performance of team members through observation by following it through and environment and reporting on its actions. In Section 6.3, we present experiments using a UAVs in a sensing task, with a single UAV in the shadow role. This type of monitoring approach is also subjective, because it relies on information from a third party. However this can be mitigated by only selecting highly capable or trusted robots as observers, and by taking care to ensure that sensor modalities between two robots are compatible.

- **Sensor Networks** - A network of sensors can be placed throughout the environment to observe the operation of robots and report results to a monitor system. For instance, in indoor environments, a network of cameras could be used to observe the actions and trajectories of robots in the environment. Similarly, in the patrolling domain, a sensor, such as an RFID tag reader, could be used to verify that a robot visited a location. We assume that this type of monitor would return objective results. However, these results would be likely low level observation that would need to be processed by a central authority. For instance, they might work well for detecting the presence, trajectory and velocity of a robot, but may not be able to validate the sensor

reading of a robot or infer behavioral intent. The major drawback to this approach is that it may be expensive or even impossible to instrument large, dynamic or remote environments.

In practice a combination of these approaches can be used, with performance metrics that are specific to the problem domain. In cases where it may not be feasible to instrument the environment or to rely on human observation, allowing for robots to observe their peers is an alternative that provides the ability to reconfigure the monitoring network dynamically. This can be useful in environments that change quickly or are remote. The tradeoff is that monitoring using robots may come at a higher cost. However, in any of these cases, we may wish to consider the cost of monitoring as part of the team formation strategy.

5.3 The Cost of Monitoring

This section presents a set of experiments to illustrate the tradeoff in monitoring when monitoring has a cost. The tradeoff is between the cost of monitoring and the cost of failing to discover poor performers, illustrated in Figure 27. In an ideal scenario, all robots on a team perform well and cooperate effectively to complete tasks at a lower cost than if they each acted independently. In this case, then a monitoring approach is not needed. However, if one or more robots may perform poorly, and a monitor process can discover which are poor performers with minimal cost, then they can be isolated or handled differently than for the good performers. As the cost of failing to discover the poor performers increases, the benefit to monitoring increases. If the robots are operating in an environment with several poor performers and the cost of monitoring is high, it may be better for robots to act independently, rather than risk assigning tasks to poor performers who may not complete them or in spending precious resources determining which team members are poor performers. Another approach to consider is the use of probabilistic monitoring, which samples from a distribution weighted by the current trust level for a robot. This would serve to focus monitoring resources on robots that are more likely to become untrusted.

A set of experiments was performed to test the probabilistic monitoring strategy in a simulated task assignment domain using the Mason simulated multi-agent environment [80],

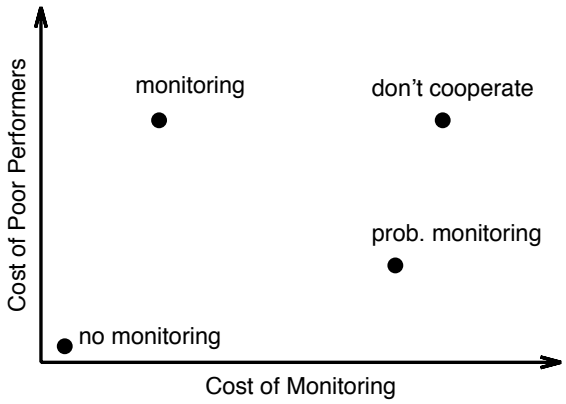


Figure 27: The cost of monitoring should be considered against the cost related to the existence of poor performers on the team. Recommended monitoring strategies are shown for various degrees of these costs.

further described in Section A.1. In these experiments, the robots are represented as Unmanned Aerial Vehicles (UAVs), and each UAV is assigned tasks to perform by an external, Poisson process. Each UAV can negotiate with its peers using an auction mechanism to exchange tasks with others, it they can be performed at a lower cost. Each UAV has an auctioneer and can auction their tasks to other UAVs, assigning the task to the UAV that submits the highest bid. The UAVs also bid on their own tasks. Rewards are given for task completion to the UAV that originated the task, and rewards decrease linearly with time until they reach 0. Each UAV submits bids that represent the surplus gain per unit time for performing the additional task. The UAVs in the simulation have a limited communications range and can therefore only perform auctions or exchange reputation information with a subset of the other team members at a given time.

In addition, each UAV periodically re-auctions the last n tasks to other agents in range. This allows tasks to be more optimally assigned by giving other agents a chance to bid on them if they were not in range during the initial auction. Once a UAV finishes all tasks in their list, they no longer accumulate costs in the simulation. Each experiment was performed using 10 UAVs, 2 of which are *defectors*. Experimental results are averaged over 100 iterations. Each UAV has 50 tasks that arrive at regular intervals and are sequentially auctioned. The initial locations of the UAVs and the tasks are randomly chosen for each iteration. The results show the average score for each of the *cooperator* agents as the

auctions are completed and rewards are assigned.

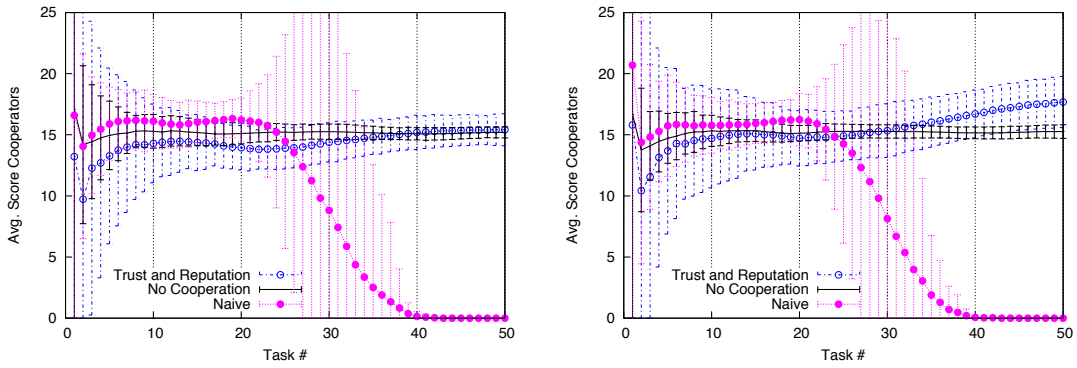
This set of experiments simulates the case in which the measure of cooperation is task completion. When a *defector* agent does not complete the tasks assigned to it by a *cooperator*, the *cooperator* loses the benefit of having that task performed. Therefore, it benefits the *cooperator* to identify those agents that do not complete the assigned tasks and remove them from consideration in future assignments. When an assigns a task, the agent can select whether that task should be monitored. However, monitoring incurs a fixed cost. When the cost of monitoring is high, it may not be worthwhile for an agent to monitor each task. At each time step, the agent can query the monitor to determine the status of the task, which is either *completed*, *failed* or *pending*. When a task is *completed* or *failed*, the trust model for the assignee is updated, and when an agent's trust value falls below a specific threshold the agent is removed from the team and further cooperation. The trust model specifics are presented in more detail in the next chapter.

In the experiments, the monitoring costs are high, equaling 4% of the possible reward. As a result, the cost of monitoring is approximately equal to the benefit gained from cooperation with the rest of the team, when there are 2 defectors present. The *defector* agents are occasionally successful in completing tasks assigned to them, but fail to complete their tasks most of the time. Defection is simulated by drawing from a normal distribution (with $\mu = 0.4, \sigma = 0.15$) and if the value is ≤ 0.5, the defector fails to perform the task. In the no-cooperation case, the UAVs perform all of their original tasks, and do not attempt to assign tasks to any team members.

In the first experiment, *Cooperators* that trust unconditionally, as shown in figure 52(a), perform worse in the later auction periods because a number of their assigned tasks get dropped and they do not receive the rewards associated with those tasks. The *cooperators* that use trust and reputation mechanisms perform better, but because of the high monitoring costs, the average scores are similar to the scores obtained through no cooperation.

In the second experiment, shown in Figure 52(b), *cooperators* selectively monitor other agents when assigning tasks, based on how much the assignee is trusted. For instance, if an agent is highly trusted, then it will be monitored with low probability and vice-versa. As

such, when a task is assigned, the probability that it will be monitored, given the assignee's trust value, x, is $P(monitor) = -x + 1$. This results in the monitoring costs being applied more to those agents that are untrusted. The agents that use the trust and reputation methods with probabilistic monitoring learn which team members are *defectors* and isolate them from future cooperation. This allows them to perform as well as the *Naive* and *No Cooperation* strategies in the beginning, but to also achieve higher scores near the end of the auction periods as the defectors are isolated.



(a) Full Monitoring (b) Probabilistic Monitoring

Figure 28: Task performance can be monitored, and agents that regularly fail to complete tasks can be removed from the team. (a)When monitoring costs are high, it may be better for agents to not cooperate at all. (b) Probabilistic monitoring allows for monitoring resources to be focused on those agents likely to defect and the trust strategies perform better than the no cooperation strategy.

The choice of the appropriate metric is also important. To this end, it is important to define the gradation by which we determine what defines a good performer. In some cases, it may be simple to verify whether a robot is cooperating or has completed a task. In others, due to challenges related to operating in dynamic environments, the definition of a reliable or well performing robot may be relative to the performance of its peers. For instance, if an outdoor environment is muddy, all of the robots might have trouble with wheel slipping and navigation. However, if only one robot is struggling and the others are performing much better in comparison, we might conclude that the single robot is a poor performer. In the next section, we consider a metric for performance in the completion of tasks and apply methods for operations research to determine when a robot is performing poorly as compared to its peers.

5.4 Performance Monitoring using Statistical Control Charts

This section describes approaches for using statistical control charts from the field of operations research as a quality control technique, applied to the task performance of team members in a multi-robot task assignment domain.[1]

Parker [106] describes two central issues[2] related to robot performance on multi-robot teams: 1) Can a robot detect when other team members are not performing tasks as expected, and 2) what actions should a robot take when poor performance is detected? In the first case, it is important to consider performance monitoring techniques from the viewpoint of the robot, rather than from that of the human operators. Future robot teams may be formed dynamically, and robots may choose team members based on observed performance. In the second case, the robot could choose to notify the operator or complete the task themselves [106], adjust the cost function based on performance characteristics [23], [115], attempt to provide aid [83], or perhaps choose to remove the poorly performing robot from the team [11]. This section will consider the first issue and present a model for monitoring performance of bid estimates vs. actual task completion times in an auction based multi-robot task allocation problem. In later chapters, we address the second issue, and present approaches for adjusting team formation in Sections 7.1 and 7.2, and the task allocation strategy in Section 7.4.

Control Charts

Statistical process control (SPC) is used in the operations research field to improve processes through monitoring and statistical analysis. A commonly used tool in SPC is the control chart [1]. Control charts are a widely used tool to monitor process quality and to detect when a measured process deviates beyond an acceptable level of performance. In this section, we apply control charts to the problem of detecting when a robot's performance exceeds an acceptable threshold. Control charts can be used to distinguish between acceptable noise in the process and abnormal operation. In this thesis, we apply control charts to the monitoring

[1] The approach described in this section appears in [119].

[2] Parker also describes a third issue related to performance, that being how or whether to diagnose the problem.

of robot performance in relation to its peers, to allow robots to detect when team members begin to perform poorly. If poor performance can be detected early, a robot could take steps to address the issue and improve performance. Control charts are commonly used to monitor industrial process performance but can be applied in many domains. Some examples include animal production systems [34], machine fault detection [172], and public health surveillance [167].

The ability to determine when a robot is not performing or functioning as expected can be used to re-assign tasks or call attention to an operator. Parker's L-Alliance framework addressed the problem of improving efficiency and fault tolerance in a multi-robot team [108]. The goal in that work was for robots to minimize the time to complete a task. Therefore, time was treated as a quality measure, wherein each robot on the team kept track of the average time, plus one standard deviation for that robot to perform a task. That approach is very similar to the use of control charts described herein; however, the robots relied on a behavioral framework as a mechanism for assigning tasks. This section will also treat time as a quality measure, as compared to the initial task estimate. In addition, this section further validates the running average approach by incorporating the control chart methods which have been heavily researched in other domains. Additional approaches to robot performance based metrics are also presented by Parker in [106]. This included a discussion of qualitative and quantitative metrics, such as mean time between failure or repair. The work also included the notion of effectiveness metrics, which seek to evaluate the success or failure of a task in retrospect. Additionally, the work related the use of statistical models to detect faults when a robot is in an inconsistent state as part of a sensing task. An open challenge mentioned in the work was the need for techniques that can infer the impact of robots that have partially failed, as well as approaches for handling the partial failures.

The control chart can be applied to situations when a process needs to be monitored over time against quality thresholds. Often, they are used as a graphical tool by managers of the process to detect and communicate instability. However, the time series data can be computed and used to monitor performance online. Control charts can monitor multiple

process quality characteristics in the multivariate case, or a single feature in the univariate case. Here we are interested in monitoring a single feature value. An example graphical control chart is shown in Figure 29. In general, to use the tool, samples are taken from a process over time and plotted, along with the known mean value for the process [1]. The known mean value is referred to as the center line (CL). Two other known limits are plotted, the upper control limit (UCL) and lower control limit (LCL).

The underlying concept of the control chart is that a process can have variation from two causes, common causes or assignable causes. Common causes are those that are due to small, unavoidable causes or noise in the production process. Assignable causes are those that are due to an unplanned, irregular behavior of the system. Assignable causes may be due to changes in the environment, a malfunction, or other unplanned change. When an assignable cause occurs, the system is said to be out of control [34] . The challenge, then is to separate the assignable causes from the common causes and seek to address the assignable causes before they significantly cause damage, increase cost or slow the production process.

The purpose of the control limits is to allow for the process to experience some natural variation before costly intervention occurs. Therefore, the selection of values for the control limits can be arbitrary and is dependent on the risk tolerance for the designers of the system and the tradeoff between the cost of false alarms and missed detections. Often, these limits are set to three times the standard error of the process [1]. When the measured process statistic exceeds the control limit, the process is determined to be out of control.

5.4.1 Applying Control Charts to a Multi-Robot Auction

While there are many mechanisms for performing multi-robot task assignment, here we focus on an application using a decentralized, market based task assignment approach. Market-based auction methods solve the multi-robot task allocation problem by splitting computation across multiple nodes and iteratively performing task assignments [15]. These algorithms generally do not explicitly consider individual team member performance when allocating tasks. However, there are situations in which the individual robots on the team may have varying levels of performance and task estimation accuracy. In order for tasks to

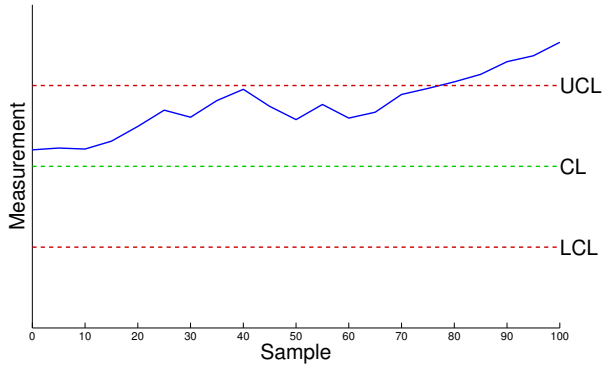


Figure 29: An example control chart. The control chart includes a mean for the process, the center line (CL); an upper control limit (UCL); and a lower control limit (LCL). When a statistical process exceeds one of the control limits, the process is considered to be out of control.

be allocated efficiently, it is important to be able to reliably trust that robots will perform their assigned tasks with costs that closely approximate their estimated costs and abilities. If a robot regularly exceeds its estimated cost for performing a task, the assignment algorithm should be able to detect this condition, and adjust the approach to task assignment.

In the basic multi-agent auction algorithm, the problem is to assign tasks to agents. In this case, the task is to visit a target location and perform an observation. In the auction framework for task exchange, each robot is a *bidder* and the items to be auctioned are the tasks. Each of the agents in the system also participates as an *auctioneer* and periodically auctions new task requests (it is assumed that the task requests are periodically provided to the agent by an external process, such as a human operator or other event). This approach can easily be used on teams with different robot characteristics: each robot knows their own location and cost function and submits cost based bids to the auctioneer. While costs and rewards use the same basis for calculation, no revenue is actually exchanged. Rather, an agent awards itself a utility value when one of its own tasks is completed.

In this work, the agents each maintain a current task list and locally compute their bid to complete the proposed task. The bid consists of the time-based cost to perform the task. A potential source of error in task estimation is in the use of an insertion heuristic for calculating the marginal cost to perform a task, in addition to those tasks already assigned. In this section, each robot plans to visit the targets in the order in which they were assigned

(using the $O1$ assignment rule from [41]). For each auction announcement received, each robot calculates its bid as the amount of time required to complete the task in addition to those on the current task list. When the winning *bidder* is assigned a new task, the task is appended to the robot's assigned task list.

Cost Factor Metric

At this point, we can define a *performer* as the robot that completes a task on behalf of another robot, the *originator*, who requested assistance.[3] As described above, the robots can expect better performance if they are able to exchange tasks with other team members that can complete them more efficiently. Upon task completion, the task *performer* notifies the *originator* that the task was completed.[4] When the task is completed, the *originator* calculates the actual time to complete the task, t_{actual}, and receives a reward according to the decreasing reward function from [41]: $f(t_{actual}) = (a - t_{actual}b)$, where a is the task reward and b is the decrease factor. In the linearly decreasing reward problem setup, it becomes even more important for tasks to be completed on time.

At this point, the *originator* can calculate the cost factor, $CostFactor_{B_a}$ from the ratio of the actual task completion time, t_{actual}, to the estimated task completion time, t_{est}, for the bid, B, by the *performer*, p.

$$CostFactor_{B_p} = \frac{t_{actual}}{t_{est}} \tag{9}$$

It is worth noting that a robot could miss the original cost estimate for a number of reasons: the cost function could be using an inaccurate insertion heuristic, the robot could be low on power and moving more slowly during actual task execution, or it could have unknown knowledge of its own internal state, for instance. Therefore, it would be useful to be able to allow for occasional variation in the process, but to recognize when a robot is performing beyond an acceptable degree of variation in relation to the team. To address this issue, we employ control charts as a threshold mechanism to the process of robot task estimation.

[3]The performer might also perform tasks on behalf of the team.
[4]The task completion notification could also come from an external monitor process.

This section presented a method for recognizing which robot team members are performing tasks as expected through the use of control charts. This quality control mechanism can be effective for detecting poorly-performing team members in a distributed task assignment domain. This may prove useful in situations in which multi-robot teams are dynamically formed and not all team members are likely to estimate costs correctly or when cost functions change over time. Although this section considered auction based task allocation, this approach for using control charts to allow for robots to monitor team members can be applied more generally. In Section 6.2, we revisit the use of control charts, and incorporate them into the use of a trust model.

The next section applies a performance metric to the multi-robot patrolling task to more efficiently distribute patrol areas among robot team members. The multi-robot patrolling task employs multiple robots to perform frequent visits to known areas in an environment, while minimizing the time between node visits. Conventional strategies for performing this task do not address situations in which some team members patrol inefficiently. This approach applies the control chart based approach for monitoring robot performance in a patrolling task and dynamically reassigning tasks from those team members that perform poorly. Experimental results from simulation and on a team of indoor robots demonstrate that in using this approach, tasks can be dynamically and more efficiently distributed in a multi-robot patrolling application.

5.5 Performance Monitoring in Multi-Robot Patrolling

Robot teams may need to consider which team members are reliable and dynamically adjust their teaming and task assignment strategies accordingly. Reliability is particularly important in security applications and if robots on a team do not perform as expected, the system should degrade gracefully. The performance criterion considered in this section is the *refresh time*, which is the time gap between any two visits to the same location[5]. The maximum refresh time reflects the bounds on the effectiveness of a robot team in detecting events in the environment [112]. If a robot fails to perform its assigned tasks or visits

[5]In the literature, this is also referred to as the *idle time* of a node.

locations too infrequently, this will affect the performance of the team. To mitigate such performance issues, other robots could be assigned some of these tasks, thereby decreasing the maximum refresh time. This section presents a dynamic approach for observing which team members patrol poorly, and can be used to more effectively perform patrol task allocation by reassigning those locations with the greatest refresh time to other team members using a bidding mechanism between the better performing team members.[6]

The problem of cooperative patrolling by a multi-robot team has received considerable attention recently in the robotics literature [112, 123, 61, 152]. This problem is similar to the well known 'Art Gallery' problem [90], in which each location in an art gallery must be viewable by a guard. However, here the tasks are to repeatedly visit the locations in the environment and to minimize the amount of time in-between visits. The patrol task is described in more detail, in Section 4.3.1.

An approach for reassigning tasks from poorly performing team members was presented in Parker's L-ALLIANCE framework, in which a robot monitored a peer robot and took over a task from when the time for completing the task exceeded a threshold [108]. Pippin and Christensen presented an approach to monitoring robot performance as compared to the performance of the team for determining when a robot's performance could be considered out of control [119]. Lewis and Weiss [77] developed a collection of metrics to measure the performance achieved when collaboration is allowed among vehicles, including the gain obtained over the base capability of the robots operating independently.

In other works, market-based auction methods were applied to the patrolling problem as an approach to the initial node assignment [60, 89]. Auction methods are a class of decentralized algorithms that solve the multi-robot task allocation problem by splitting computation across multiple nodes and iteratively performing task assignments [15]. These algorithms serve as a mechanism for distributed task allocation and generally do not explicitly consider individual team members' performance characteristics. However, individual robots on a team may have varying levels of performance. In this section, we update the multi-robot patrol task with a mechanism for monitoring task performance and reassigning

[6]Experiments in this section appeared in [120].

patrol locations using an auction based mechanism.

5.5.1 Approach

In determining our partitioning approach, we assume that the a map of the environment is provided in advance, and that further a topological map is generated from this map and provided to each of the robots. At startup, each of the r robots on the team partitions this graph and assigns the r^{th} partition to itself, calculating a closed cycle over the partition. There is an initial startup time for each robot to navigate from its starting location to the first node in its closed path. Once each robot begins patrolling, it is observed by a central monitor process which keeps track of the maximum refresh time of each robot's assigned nodes. If a robot's performance exceeds a threshold, based on the performance of its team members, one of the poorly performing robot's nodes is offered to the rest of the team and re-assigned using an auction based protocol.

Graph Partitioning

To obtain the initial graph representation of the environment we perform a series of pre-processing steps. We begin with the bitmap image file which represents a map of the environment generated from a mapping process and use the EVG-THIN software from Beeson [14] to generate a Voroni graph. This is the same approach used by Portugal and Rocha [124]. Next, we perform additional pre-processing on the graph to prune short leaf nodes and to merge nodes that are close together.

From this graph, we calculate the minimum spanning tree (MST) to remove cycles in the graph. Next, we compute an open tour of the edges that visits all of the vertices (nodes) in the graph by starting with a leaf in the MST and visiting each branch of the tree, shortest edges first. Finally, we use the chain partition algorithm to divide the path among the r team members. This determines the initial assignment of nodes to robots.

5.5.2 Performance Monitoring

In this set of experiments, we adopt the performance metric of *maximum refresh time*. When the refresh time of any robot's assigned nodes exceeds a threshold on this metric,

we seek to re-assign some of that team member's nodes to other team members. In our approach, we assume the existence of an external monitor that can fully observe the visits to each node. Each robot self reports node visits to the monitor which tracks the refresh time for each node. At each time step, the monitor can calculate the node with the maximum refresh time for each robot. We set the amount of time in between performance monitoring periods to be the expected maximum refresh time for the patrol partition.

The monitor compares each robot's maximum refresh time, defined as the maximum refresh time of all nodes assigned to that robot, against the average refresh times of all currently trusted team members. Specifically, we define a control threshold at one standard deviation, σ, above the average max refresh time for the team. When a robot exceeds this threshold, the monitor marks this robot as untrusted and performs a task reassignment.

The max refresh time for a robot is the maximum refresh time for all nodes assigned to robot r. Let I_k^r be the set of the refresh times at the previous k node visits for a robot, r. Let I_n^r denote the refresh time of a node visited by robot r and being the nth visit by r to any node assigned to it. The running max refresh time, $M_k^r = \max(I_{k..n}^r)$, is the observed maximum refresh time for a robot over the window $(n, n-1, \ldots, n-k)$, where $n > k > 0$. The threshold for the max refresh time, $\theta_{maxrefresh}$ is defined as the average running max refresh time over all robots, plus one standard deviation:

$$\theta_{maxrefresh} = \overline{M_k} + \sigma \tag{10}$$

We define a patrol period as the expected amount of time to perform a patrol of the maximum partition plus a constant factor. This factor is included to capture the additional time needed to navigate due to the non-holonomic motion of the robot and related to time spent navigating around obstacles. At the end of each patrol period, the monitor checks whether $M_k^r > \theta_{maxrefresh}$ for each robot. In that case, a robot is considered to be performing poorly and is marked as *untrusted*. The monitor then selects one node to be reassigned from all *poorly performing* robots, using the process described in the next section.

5.5.3 Auction Based Task Reassignment

We use a market-based auction algorithm to perform task reassignment from the *poorly performing* robots to the remaining robots on the team. This approach reassigns a single node from the set of *poorly performing* robots during each monitor period. The pseudocode for this process is shown in Figure 30. Recalling that the joint patrol partitions form a single patrol chain, we seek to exchange those nodes that are the ends of a robot's chain partition. These nodes comprise the leaf nodes of a partition. Let L denote the set of leaf nodes belonging to all known *poorly performing* robots. Then, our approach is to reassign a node from L to another team member that is currently performing well and considered to be a *trusted* performer.

This approach varies from the general auction approach in the literature in that tasks are initially assigned according to the graph partitioning algorithm described in Section **??**. Here, auctions are only used to reassign tasks. A central auctioneer performs the auction, announces the task winner and reassigns the task. The first step in the process is to calculate the set L over the partitions. A separate auction is announced to all team members for each node, $n \in L$, by sending an *Announce Auction* message with the node identifier.

Bid Calculation

Upon receiving the *Announce Auction* message, each robot calculates a bid for adding the new, candidate node to its patrol partition as follows. The candidate node is added to the list of the existing nodes in the robot's partition, along with any intermediate nodes along the minimal path to the candidate node. Next, the minimum spanning tree (MST) of the subgraph is calculated and an open tour that visits all of the nodes in the MST is generated by visiting each of the branches in the MST, shortest branches first, as described in [112]. The candidate max refresh time of the new partition, $c_{\max(i)}$, is calculated from the new tour by computing the path distance along the tour for a round trip: $c_{\max(i)} = PathDist(tour) * 2$. Finally, $C_{\max(i)}$ is submitted as the bid for this robot.

Winner Selection and Task Reassignment

The auction approach seeks to reassign a node from one of the poorly performing (*untrusted*) robots to a robot that is performing well by selecting from among the bidders that will result in the smallest candidate refresh time. Intuitively, this reassigns a node from a poorly performing robot to one that is comparatively underutilized by assigning it to the robot that would still have the smallest candidate path. Note that this is different from assigning the node to the robot with the smallest marginal cost for adding the candidate node. The latter could result in robots that disproportionately grow their patrol paths.

The auction algorithm selects the bid with the minimum max refresh time from the set of bids received from all trusted robots for all $n \in L$. To ensure iterative improvements, the monitor also keeps track of the max observed refresh time during the current patrol cycle and will not award a node to any candidate bid that exceeds this value. The winning bid is sent as an *Announce Winner* message to the winning bidder. The auctioneer waits to receive an acknowledgement message from the winning bidder before performing the task reassignment from the original robot. This is necessary to ensure that at least one robot is still including this node in its partition.

When a robot receives the *Reassign Task* message, it removes the node from its current partition, and recalculates the MST and open tour for the new partition. It is assumed that a robot will relinquish the node when this message is received. However, even if it does not, this will still result in an improved refresh time for the node because the winner robot will also cover that node.

5.5.4 Experimental Results

A set of experiments were performed to demonstrate that the use of the reassignment approach can improve the performance of the team by re-assigning tasks away from poorly performing team members, thereby reducing the overall max refresh time of nodes in the graph. Each of the r robots is given a map of the environment and the full patrol graph. At startup, each $robot_i$ locally partitions the graph into r separate partitions and assigns the i_{th} partition to itself. The central task monitor is available and has full visibility into

1: **for all** $p : PoorPerformers$ **do**
2: **for all** n : leaf nodes in $Partition_p$ **do**
3: $a \leftarrow AnnounceAuction(n, R)$
4: **end for**
5: $C \leftarrow ReceiveBids(A) \forall R \in Trusted$
6: $w \leftarrow Min(MaxRefresh(A))$
7: **end for**
8: $AnnounceWinner(w, n)$
9: **if** Receive ACK_w **then**
10: $ReassignTask(p, n)$
11: **end if**

Figure 30: ReassignTask() pseudocode.

Figure 31: Using the chain partition algorithm, the graph is initially partitioned into approximately equal tours for each robot.

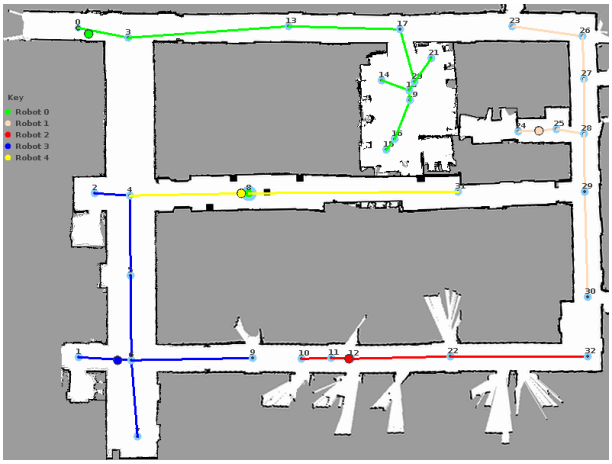


Figure 32: The partitions have been updated after several task reassignment operations as a result of poor performance by robot 2.

the arrival of robots at nodes. Robots send node visit messages to the monitor using the network interface. The monitor keeps track of the refresh time for each node, as well as the maximum refresh time for all nodes.

In these experiments, a subset of the robots are marked as *poor performers*. The performance for this type of robot is affected by randomly adjusting the maximum forward velocity of the robot after each visit to a patrol node. The robots that perform normally have a maximum speed of 0.25ms^{-1} and the maximum speed of the *poor performers* is determined by sampling from a normal distribution with $\mu = 0.15\text{ms}^{-1}$ and $\sigma = 0.10\text{ms}^{-1}$.

Experiments in Simulation

Three different experiment types were performed:

naive strategy The robots patrol the set of nodes in the initial partition. A subset (1 or 2) of the robots on the team are marked as *poor performers*.

auction strategy A central monitor observes the performance of the robots on the team and reassigns tasks using the auction based approach as described in Section 7.4.1. A subset (1 or 2) of the robots on the team are marked as *poor performers*.

all perform The robots patrol the set of nodes in the initial partition. None of the robots are *poor performers*.

Simulations were run with 3, 5, and 8 robots on a team. Each simulation ran for 2 hours of simulated time. For the 3 and 5 robot teams, 5 experiments of each type were performed, while 2 experiments were run for each 8 robot team type, resulting in over 80 hours of simulated patrols. The experiments used the Stage multi-robot simulation environment [160], shown in Figure 33. The open-source Robot Operating System (ROS) architecture [127] was used to implement the robot messaging, low level control and behaviors.[7] Each robot uses the ROS navigation stack for navigation, localization, and obstacle avoidance. Each robot also runs a custom *Patrol* behavior which implements the graph chain partition

[7]The setup was patterned after the ROS patrol simulation from: http://www.ros.org/wiki/isr-uc-ros-pkg#patrol.

algorithm, and repeatedly navigates to the nodes in the robot's patrol path. This behavior also implements the auction protocol and calculates the robot's bids. The simulation also includes a central monitor node which listens for task completion messages and includes the performance monitoring and task reassignment components. Robots communicated with the central monitor by sending messages using UDP broadcast over the local network.

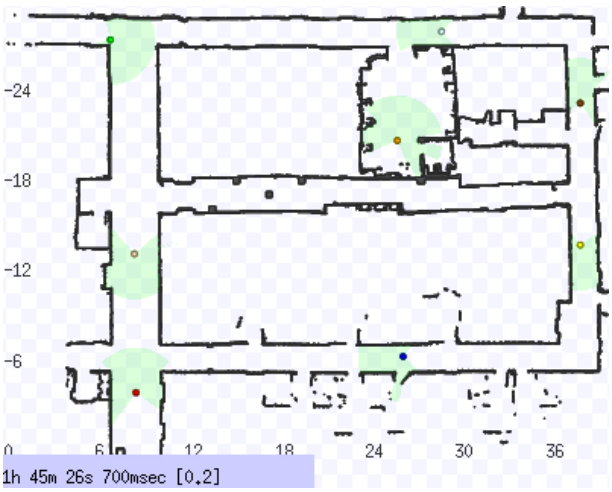


Figure 33: The Stage multi-robot simulator is shown with eight robots patrolling the environment. The robots in the simulation run the ROS navigation stack and participate in auctions for task reassignment.

For each experiment, we track the *running max refresh time* of the overall patrol. This value represents the maximum refresh time of any node in the environment, computed over a window of the last τ seconds. We set the value for τ to be greater than twice the expected time to complete a patrol, to prevent cycling of the value, due to the out and back nature of the open tour in each partition.

Robot Experiments

A second set of experiments was performed using the TurtleBot indoor mobile robot, described in Section A.2. The experiments were performed using a team of three TurtleBots in the same office environment, shown in Figure 34, that was mapped for use in the simulations. Three experimental runs were performed with a central monitor performing the auction strategy and a single *poor performer* on the team. In the first two runs, after observing multiple patrol cycles, the monitor observed the *poor performer* on the team and

used the auction-approach to reassign one of its nodes to other team members. Those runs were ended after the successful node reassignments.

The third run was executed for approximately 90 minutes, with all three robots patrolling continuously during that time. The robot trajectories during this run are shown in Figure 37. After several initial cycles, the monitor auctioned and reassigned nodes from the *poor performer*, *robot 0* to *robot 1*. Later, after several more patrol cycles, the monitor reassigned another set of nodes, this time from *robot 0* to *robot 2*. At this point, no more nodes were reassigned as the *running max refresh times* across the team were similar.

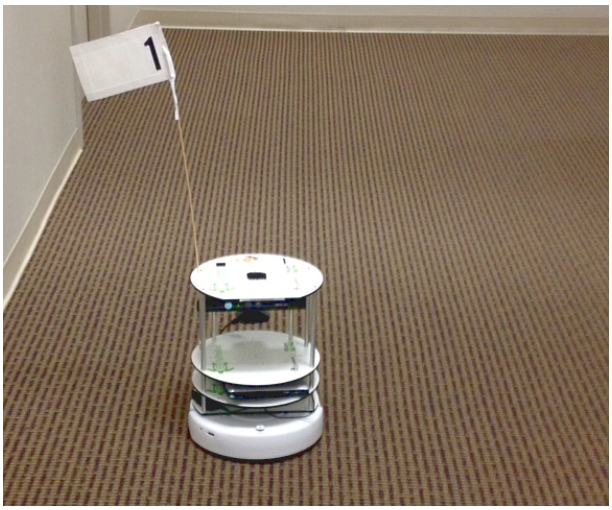


Figure 34: A TurtleBot shown patrolling in the hallways of an office building during the robot team experiments.

Results and Discussion

In our first result, we compare the *running max refresh time* for each of the three types over the full patrol time. An example result from experiments with three robots is shown in Figure 35. In the case where all robots perform as expected, the *running max refresh time* is approximately constant, and is close to the calculated value for the max partition patrol time (it is slightly greater than the calculated value, due to the non-holonomic motion and obstacle avoidance behaviors of the robot.) For both the naive and auction cases, the *running max refresh time* values vary, due to the random sampling of the velocity for the *poor performer* robot. However, after some initial task reassignment, the performance of the auction method improves over that of the naive approach. The auction approach reassigned

multiple nodes from the *poor performer* robot to robots with neighboring partitions. After the neighbors were assigned these tasks, the *poor performer* is well below maximum refresh time. Subsequent reassignments shifted tasks from the *poor performer's* neighbors to their neighbors.

The results for different team sizes were averaged over all of the experimental runs, and are shown in Figure 36. In all experiments, the auction based approach to task reassignment resulted in better performance than the naive approach to patrolling which does not consider individual robot performance. In the set of experiments with 1 *poor performer* out of 8 robots, the auction approach unexpectedly resulted in better performance than the all perform case. We attribute this result to the obstacle avoidance behavior of the robot. Here, there are fewer nodes for each robot to visit, and inefficiencies in robot motion are more noticeable. In this case, a robot that was modeled as a good performer had difficulty navigating a narrow doorway on the right side of the environment, and this caused the robot to slow down on this leg of its tour, resulting in unexpected poor performance and increasing the max refresh time for the entire patrol. The auction based method reassigned one of this robot's assigned nodes to a neighbor robot and this resulted in a decreased the maximum refresh time.

The experiments with the team of robots demonstrated the use of this approach in a real-time patrolling scenario in an office environment. Here the poor performance dimension of varying speed was artificially introduced. However, this approach could be applied more generally to other performance dimensions.

In this section, the tasks being reallocated are patrol areas. However, it should be noted that this approach to monitoring and task reallocation is not limited to graph partitions in the multi-robot patrolling problem. More generally, this approach can be applied to any domain that requires task allocation amongst multiple robots with the possibility of variance in performance across team members. It is assumed that the team members have the intent to perform tasks, but perform them poorly, due to errors in navigation, control or hardware. Once a robot's performance moved below the θ threshold, it was no longer considered trustworthy for assigning new tasks. However, we observed that in some

situations, an individual robot's performance decreases when it takes on additional tasks for the benefit of the team. An trust model should take this into account and not punish robots that assist others. This is addressed in Chapter 6. Furthermore, while the auction based task assignment mechanism is decentralized, in these experiments, the auctioneer and monitor were implemented on a centralized node. However, the monitor and auctioneers could also be distributed.

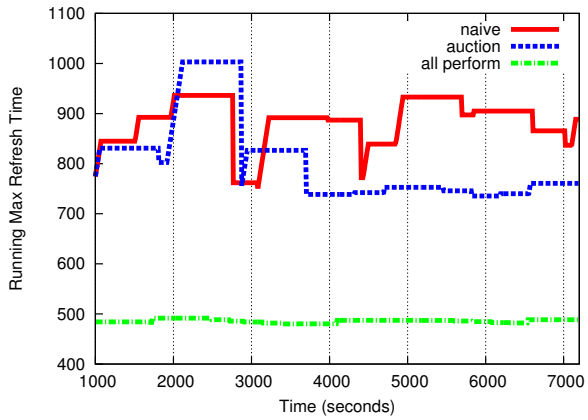


Figure 35: Results are shown for a team of 3 robots with 1 poor performer and compare Max Refresh Times for the naive and task re-assignment approaches and the case in which all robots perform as expected.

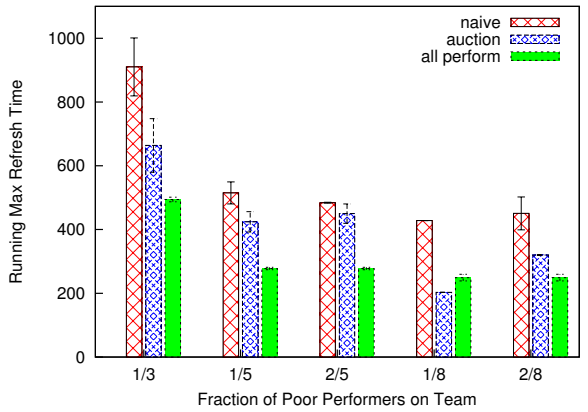
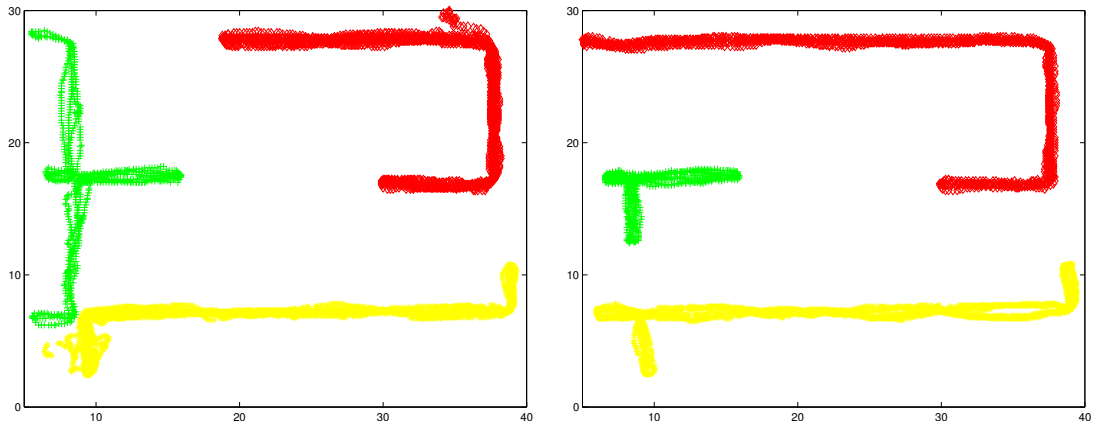


Figure 36: The auction based strategy is compared to the naive strategy when there are *poor performers* on the team and to the case where all robots perform as expected. The error bars represent one standard deviation.



(a) Initial Partition (b) After Task Reassignments

Figure 37: A set of trajectories is shown for a team of three TurtleBot indoor robots while patrolling an office environment, with 1 poor performer on the team (the leftmost, green trajectory). (a) After initial startup, the robots patrolled and partitioned the environment using the chain partition algorithm. (b) After observing multiple cycles, the central monitor auctioned and reassigned tasks away from the poor performer to others.

5.5.5 Summary

This section presented a method for recognizing which robots are performing poorly in a multi-robot patrolling task. Both simulated and robot experimental results using this approach were presented. The experiments showed that a monitoring approach can be effective for detecting poorly-performing team members. In addition, a task reassignment mechanism can be effective for more efficiently allocating patrol tasks, when compared to the naive approach which doesn't monitor individual robot performance. This may prove useful in situations in which multi-robot teams are dynamically formed or when not all team members are likely to perform effectively over time. The results show that by observing the performance characteristics of individual robots, tasks can be allocated more efficiently than the approaches which do not consider performance.

This chapter presented the purpose of monitoring on multi-robot teams for gathering observation histories that can be used to inform trust models. Multiple approaches to monitoring were presented, along with a discussion of the costs related to monitoring. A set of experiments demonstrated that for the use of monitoring to be effective, the cost of monitoring must be less than the additional benefits gained through cooperation.

We also presented experiments which demonstrated the use of monitoring using statistical control charts and for re-assigning tasks in a multi-robot patrolling domain. The next chapter will expand the use of control charts to use them as part of an overall trust framework. Further experiments will present monitoring in concert with a trust model to demonstrate that the addition of a trust model can allow for additional noise in the observations, can be used to allow for forgiveness, and can support multiple dimensions of trust.

CHAPTER VI

INCORPORATING A TRUST FRAMEWORK FOR MULTI-ROBOT TEAMS

In previous two chapters, we considered approaches for adjusting the task assignment function between team members, and for monitoring the performance characteristics of team members. We can now revisit our discussion of cooperation and trust from Chapter 3. Our definition of trust is the belief that another robot will reliably perform as expected. Once a robot has the performance characteristics of a peer, it can deliberate as to whether it should trust that peer to cooperate or complete a task. This chapter presents a framework for deliberating over the observed performance of team members using a trust model. We present our trust model formulation in section 6.1. In section 6.2, we continue our example using control charts and show how this monitoring technique can be combined with a trust model. In section 6.3, we present the results of modeling trust on a UAV platform in a sensing task, when the sensor characteristics are unknown.

6.1 Trust Model

The performance monitoring techniques presented in the previous chapter can be used to adjust task assignment and teaming strategies. However, these techniques alone present challenges in practical implementation, they are not tolerant to noise and do not allow forgiveness, there is no mechanism for sharing information gathered from different sources as reputation and it is not clear how multiple characteristics can be combined into a mental state for making a decision. These challenges form the desiderata by which we formulate our trust model. We will describe them further below.

There are many different approaches to trust modeling, including continuous and discrete numerical values, binary values, probabilities and entropy [84]. As Matei, Baras and Jiang relate, there is no single correct representation for trust model, but that the model should depend on the mission requirements and the environment [84]. A trust model can be

used on a multi-robot team to represent the trustworthiness or reliability of a robot team member across one or more dimensions.

Our approach relies on the use of a a probability based trust model, using the beta distribution from Teacy et al. [154] and Jøsang and Ismail [66]. We rely especially on the *TRAVOS* trust mechanism [154] for incorporating direct trust and reputation into a probabilistic formulation. The benefits of using a beta trust and reputation mechanism is that is is simple to implement, easy to combine trust and reputation from different sources and has a statistical foundation [66]. This probabilistic view of trust represents the belief that a robot has that another robot will perform a task as expected. Furthermore, this mechanism provides not only a trust belief about an agent, but also a confidence value. The approach can incorporate positive, α, and negative, β, histories to calculate the belief and confidence values. As the number of observations is increased, a more accurate estimate of the outcome is given by the model, and this leads to an increase in the confidence value for trust. Two example beta distributions are shown in figure 38, with different values for the number of α and β observations. When the number of α observations is equal to the number of β observations, the probabilistic belief about the trustworthiness of an agent, τ, is centered about 0.5, and with fewer observations the distribution approaches the uniform distribution. With more observations, the distribution becomes weighted more around the belief, and the confidence, γ, increases. Furthermore, multiple beta probability distributions can be combined into a single distribution, as shown in Figure 39.

The model can reside with one or more robots or be centrally located. The trust model maintains a set of α and β vectors that represent the histories of interactions with each team member. For a given team member, if the calculated trust value is less than the trust threshold, τ, and with confidence greater than γ, it is not trusted. However, a succession of positive observations (direct or indirect) can move an untrusted agent back to being trusted again. Furthermore, this approach is tolerant of noise as it can take multiple observations to move the value above or below the trust threshold. To better explain this model, the equations from [154] for calculating the trust value τ and confidence, γ, are included below.

When a trust authority receives new α and β updates for a dimension of trust, it can

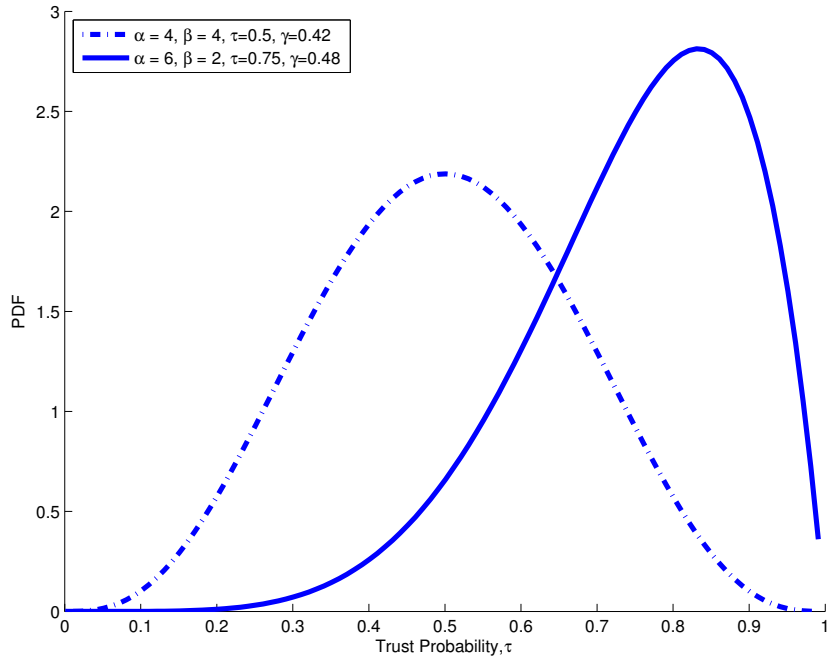


Figure 38: The Beta Trust Model. The model reflects a probability distribution over the trust probability value, τ. Two example trust models are shown. The dashed line reflects a trust model with an equal number of α (positive) and β (negative) observations; τ is centered about 0.5. The solid line reflects a trust model with more α observations and a higher value for τ.

calculate the Expected Value for trust using the trust model as follows.

$$E_{trust_{i,j}} = \frac{\alpha}{\alpha + \beta} \tag{11}$$

The value, $E_{trust_{i,j}}$, is the expected trust that $robot_i$ has toward $robot_j$, given a set of observations, O, from the start through time t. Therefore, the trust value, τ, is

$$\tau = [E_{trust_{i,j}} | O^{1:t}] \tag{12}$$

The confidence factor, γ, is calculated as the proportion of the beta distribution that is within ϵ of τ.

$$\gamma = \frac{\int_{\tau-\epsilon}^{\tau+\epsilon} X^{\alpha-1}(1-X)^{\beta-1}\,\mathrm{d}X}{\int_0^1 U^{\alpha-1}(1-U)^{\beta-1}\,\mathrm{d}U} \tag{13}$$

We define the set of *untrusted* robots, U, to include those with a trust score below the minimum trust threshold, $\tau < \theta_\tau$ and with confidence above the minimum confidence level, $\gamma > \theta_\gamma$. All other robots belong to the *trusted* set, T. The *Trust Authority* maintains the current sets T and U, and can be queried to determine the set membership for a robot.

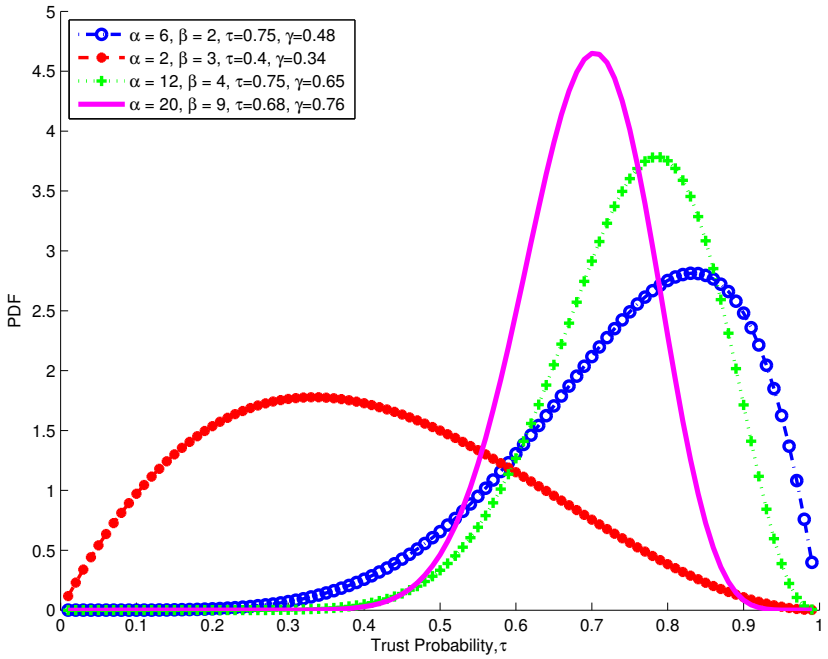


Figure 39: The Beta Trust Model. Multiple trust models from different robots are combined into a reputation model (solid magenta line), by incorporating the α and β observations from trusted teammates. The combined reputation model has a higher confidence value γ, related to the increased number of indirect observations.

6.1.1 Shared Reputation

In addition to the observations from direct interactions with other robots, this approach allows for the robots to incorporate indirect observations from other trusted team members, known as shared reputation information. The intent for sharing reputation information among team members is to quickly spread information about trusted or untrusted robots to the rest of the team. If a robot can rely on reputation information from other robots, it might be spared from negative direct interactions with uncooperative robots. However, the shared reputation information must be combined with the locally observed trust vectors.

In addition, robots only incorporate those updates from other **currently trusted** team members. These shared, direct, observation vectors are easily integrated into the local vectors and the scalar trust and confidence values are recalculated.

Rumor Propagation

If the reputation matrix is centrally located, then the robots can simply share their direct observations to the centralized reputation authority which consolidates them and shares them back to each robot, as discussed further in section 6.1.2 below. However, for distributed trust models, we consider how trust information can be shared using a distributed reputation authority. The are two general decentralized approaches in the literature [130], as follows:

Direct Observations - in this approach, robots only communicate the private information or direct observations resulting from direct experience with or monitoring of another robot.

Rumor Propagation - robots forward all observations, direct and indirect to neighboring team members. In subsequent exchanges they can propagate observations from neighbors of neighbors.

The rumor propagation approach can result in double counting of observation histories as information is forwarded and counted multiple times, without discerning direct observations from indirect observations. A possible improvement is to perform detailed accounting of the source of the message, but this requires additional overhead and the message size increases over time [130]. An additional approach is presented in [130] for propagating the observations relevant to the Dirchlet distribution with multiple trust dimensions. In social and e-commerce networks, it may be important to propagate reputation very quickly, and the increased accounting overhead and message sizes is an acceptable tradeoff. However, for multi-robot teams that have localized communication constraints, it is more likely that a robot will encounter another with direct experience of an unknown robot before it has to encounter it directly. In this dissertation, we apply the *direct observations* approach for distributing reputation information, as described below.

Sharing Direct Observations

Each robot maintains a set of α and β observations for every other known teammate.

$$N^r =< n^{\alpha}_{i,j}, n^{\beta}_{i,j} >; \forall i,j \in [1,n], r \in [1,m]; 1 \leq m \leq n \tag{14}$$

The *reputation* matrix, N^r of outcome observations known to robot r is given by:

$$N^r = \begin{pmatrix} n^{\alpha}_{1,1} & n^{\alpha}_{1,2} & \cdots & n^{\alpha}_{1,n} & n^{\beta}_{1,1} & n^{\beta}_{1,2} & \cdots & n^{\beta}_{1,n} \\ n^{\alpha}_{2,1} & n^{\alpha}_{2,2} & \cdots & n^{\alpha}_{2,n} & n^{\beta}_{2,1} & n^{\beta}_{2,2} & \cdots & n^{\beta}_{2,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ n^{\alpha}_{r,1} & n^{\alpha}_{r,2} & \cdots & n^{\alpha}_{r,n} & n^{\beta}_{r,1} & n^{\beta}_{r,2} & \cdots & n^{\beta}_{r,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ n^{\alpha}_{m,1} & n^{\alpha}_{m,2} & \cdots & n^{\alpha}_{m,n} & n^{\beta}_{m,1} & n^{\beta}_{m,2} & \cdots & n^{\beta}_{m,n} \end{pmatrix} \tag{15}$$

As such, the *reputation* matrix, N^r contains the set of all outcomes, from both direct and indirect observations that are known to robot r. The r^{th} row represents the set of direct observations, and the i^{th} row, where $i \neq r$ represents the set of indirect observations that are aggregated from trusted teammates. In the case that robot r has not received observations from teammates, then $m < n$. We assume that each robot teammate is uniquely identifiable and that there exists a mapping from each teammate's unique identifier to the indices used here.

To share reputation information, each robot shares the cumulative α and β direct observations for each team mate that it has interacted with. For a team of n robots, this results in a vector containing $2(n-1)$ outcomes that are exchanged. To illustrate, the vector containing direct observations by robot 1 on a team of five robots is given by:

$$N^1_d = < n^{\alpha}_{1,2}, n^{\alpha}_{1,3}, n^{\alpha}_{1,4}, n^{\alpha}_{1,5}, n^{\beta}_{1,2}, n^{\beta}_{1,3}, n^{\beta}_{1,4}, n^{\beta}_{1,5} > \tag{16}$$

and this vector would be exchanged periodically with all teammates that are within communications range. Each teammate then includes the observations from robot 1 into their set of observations.

When a robot r receives an observation vector from another robot k, it updates its own reputation matrix as follows. If robot k is trusted, then robot r incorporates the direct observation vector from robot k into the k^{th} row of its reputation matrix, N^r.

Robot r can then calculate the trust value for another robot j by first calculating the sums from the α and β columns (j^{th} and $(n+j)^{\text{th}}$ columns, respectively) from N_r, ignoring the values for $n^{\alpha}_{i,j}$ and $n^{\beta}_{i,j}$ where $i == j$. These sums represent the α and β values that are

used in equations 11 and 13 to calculate the trust τ and confidence γ for robot j.

An example for incorporating observations from three robots is shown in figure 39. The first three beta distributions represent the the α and β direct observations for a given robot k, from three different other robots. These observations can be combined in to a single beta distribution, the fourth distribution, which has more observations and a higher confidence value, γ, for the trust belief, τ.

6.1.2 Reputation Authority

The reputation authority is the mechanism by which trust information from team members is stored. It serves as a logical distribution hub for trust information within the robot team. Recalling the game theoretic incentives described in section 3.4.1, the reputation authority is used to enforce cooperation, and to allow for robots to benefit from the outcomes of the interactions and observations from other trusted agents, without having to interact directly with each peer. We will revisit the use of the reputation mechanism as an incentive in section 7.1. There are several approaches for the physical placement of the reputation authority in the trust framework. We will review them below.

- **Centralized Repository** A centralized trust authority is a single repository for trust information that each robot updates from their direct interactions with peers. The authority should only accept updates from members that are currently trusted. In addition, the authority can be queried by robots to get the trust level and confidence for any team member. Alternatively, the centralized repository sends periodic updates to all team members, containing the trust levels of the current team members. The repository architecture is logically separate from the multi-robot planning and task allocation architecture. A centralized repository can be used even if a the multi-robot architecture is distributed or it could reside along with a centralized planner on a central node. The centralized repository has the benefits that it is easy to implement and the aggregate information from a single source might more readily be trusted. However, a centralized component might not be desirable in an otherwise fully distributed system.

- **Distributed Repository** With a distributed repository, each robot maintains a local trust authority and incorporates updates from their trusted peers. With this approach, for a team of n robots, there might be n repositories with different trust values. The main benefit of this approach is that it allows for a fully distributed architecture. Furthermore, if the communication range of the robots is small then this approach may be necessary. Robots can exchange information when they reach communication range with trusted peers, either by periodically sending messages or by responding to queries.

- **Local Repository** With a local only repository, each robot maintains a local trust authority but does not share updates with their peers. Rather, the trust score for each robot is generated from direct observations only.

- **Replicated/Hybrid** The replicated approach combines both the centralized and local mechanisms. This approach is appropriate when there might be different clusters of robots working specific areas, but occasionally are able to communicate with one or more reputation authorities. Each reputation authority is connected through a communication network and can immediately provide updates to the other reputation authorities to keep them synchronized. When robots are unable to communicate with any of the reputation authorities, they cache their trust information on their local repository.

In the centralized case, there would be a single reputation matrix, N^c, contained in the reputation authority, updated with periodic observation reports, N_d^r, from each robot. In the distributed case, each robot would maintain a local reputation matrix, N^r, containing direct and indirect observations, that is updated with reports from teammates, as described in section 6.1.1 above. In the local case, each robot only maintains the set of direct observations, N_d^r. In the hybrid case, each robot maintains a local set of observations, N^r, but also sends less periodic updates, N_d^r, to the central repository, N^c, and similarly receives updates when communication is possible.

The centralized authority is the simplest to implement, but requires that robots be able to communicate with the central node, and this allows for a central point of failure in the system. However, the reputation authority would likely not be a mission critical component of the system and there might be a tolerance for periodic updates, when communications can be re-established, similar to a replicated or hybrid approach. An example of a centralized reputation authority is presented with experimental results in 6.2. A distributed reputation authority requires more overhead and must use a messaging approach to share the trust information with neighbors. A distributed authority is relevant in domains where robots frequently interact with different teammates and can therefore benefit from receiving information from indirect observations. A fully local trust authority does not need to be updated from neighbors, but contains only partial information, the direct observations. This may be acceptable in situations where the neighbors do not change frequently or when there is a single robot performing observations. We present the results using a fully local trust authority in section 6.3.

These are tradeoffs that must be considered according to the mission requirements. In the next chapter, we present additional examples for these cases, and apply the trust model to task allocation and team formation. In sections 7.1, 7.2 and 7.3, we present experimental results using a distributed reputation authority. In section 7.4, we present the results from experiments on using a team of robots performing a patrolling task with centralized and also a local reputation authority placement.

6.1.3 Dimensions of Trust

In a dynamically formed team, agents may encounter other agents for which they have no prior experience. The use of a trust model would allow for a robot to reason about other robot's trustworthiness using observation histories and reputation information. In these settings, there are multiple dimensions that could be used to define trust, such as whether a robot cooperates and whether a robot successfully completes tasks that are assigned to it. These dimensions of trust can be considered separately or in combination. Each robot can build models of other team members behaviors from observation histories and use those

Table 9: Example Performance Dimensions for UAVs

Perception	
Probability of Detection	Can the robot detect a target reliably, from a given range, angle, etc?
Tracking	Can the robot track a target across several frames?
Deliberation	
Robust Cooperation	Does the vehicle follow the cooperation protocol and actively cooperate?
Task Estimation	Does the vehicle provide accurate cost estimates for task allocation?
Planning	Does the vehicle generate executable, correct and approximately optimal plans?
Communication	
Communication Range	Is the effective range of the communication sufficient?
Interoperability	Does the vehicle implement and follow communications standards?
Action	
Performance	Does the vehicle execute tasks in an efficient manner (cost could be time, distance, speed, fuel, etc.)?
Behavior Selection	Does the vehicle select the correct behavior or action for a given state?
Avoiding Restricted Areas	Does the vehicle respect boundaries and lethal cost areas?
Sense and Avoid	Does the vehicle respect rules for navigation and flight safety?
Trajectory Following	Does the vehicle execute appropriate control laws for trajectory following and formation flight?
Sensor Trajectory	Does the control algorithm place the sensors at the correct altitude, velocity, orientation and angle?
Stealth Operation	Does the vehicle alter the environment, generate signals or noise when it should remain unobserved?

models to determine levels of trust. The use of a trust model allows for the robot to include different dimensions into the trust calculation. Each dimension can be incorporated into the model and weighted. Several examples of performance dimensions for UAVs are shown in Table 9.

Depending on the requirements, it may be desirable to consider multiple dimensions of trust in a team. These trust dimensions could be used to determine roles for the team that each robot might be best suited. For instance, after an initial period of observation, the model could reflect that a robot cannot be trusted to perform sensing tasks, but works well as a communication relay. In addition, each dimension could also have different threshold values for the trust and confidence levels. A robot might be more tolerant of a peer that occasionally is late with performing a task, but is less tolerant of peers that return false positives classification tasks.

Each trust dimension can be combined using a weighted linear combination as shown in equation 17 below to calculate the aggregate trust, τ_A, across k dimensions, with each

dimension having the weight, w_n. The unweighted trust value for each dimension is τ_n, and each dimension has trust threshold, θ_n. The weights $(w_1, w_2...w_k)$ could be considered equally, they could be set by the robot designer as part of the mission requirements, or they could be tuned through online optimization based on desired performance criterion. The value, τ_A, represents the overall trust value for the robot.

$$\tau_A = \sum_{n=1}^{k} w_n \tau_n \tag{17}$$

An alternative for incorporating multiple trust dimensions is to apply the Dirichlet distribution [130]. In situations where there are multiple dimensions that exhibit high correlation, this distribution includes a covariance formulation that has been shown to result in improved estimates for the trust values when compared to multiple independent beta distributions. For instance, there may be a trust dimension of time to complete a task, in addition to a dimension of quality. The quality dimension may have a natural correlation to the amount of time spent completing the task (searching tasks, mapping, etc.). In these cases, the Dirichlet distribution can be applied as an extension of the beta distribution [130].

It is also worth noting that the additional beta trust model approaches used by Teacy et al. [154] and Jøsang and Ismail [66] for filtering inaccurate reports and reports from dishonest agents can be applied within this framework. Jøsang and Ismail [66] consider two approaches to inaccurate reports. In the *endogenous* approach, the system would consider the statistical properties of the report in relation to the reports of others, while in the *exogenous* approach, other information is used to judge the source of the reporting agent, such as the reputation of the source. Teacy et al. [154] present and *exogenous* approach that judges a reporting agent based on the accuracy of previous observations, rather than its deviation from the opinion of others. In the experiments in this thesis, we take a basic *exogenous* approach, by considering the current trust score of the information provider. However, the framework supports these other approaches for deciding when to include reputation reports. Related to this, Matei et al. [84] consider two views of trust: the performance and reliability view, and the information accuracy view. If a robot becomes

unreliable along a trust dimension that does not affect the accuracy of reporting reputation information, then we may wish to still include those reports.

6.1.4 Role based Trust

Another consideration is the application of trust dimensions to role based task allocation. Jones and Browning, et al. provide an approach for building dynamic robot teams by using an auction framework, in which an auctioneer requests robots to bid on tasks that can only be completed by a specific *role* or capability [63]. A role is a behavioral set of action capabilities that can be performed only by a subset of the team. In a heterogeneous team, robots may be able to perform some types of roles but not others. Roles might also refer to sensor capabilities. As an example, a robot might have an optical camera while another robot might carry an infrared camera, and this could be represented by two different roles. In [63], the robots bid on a task if they have the ability to perform that role, and bid a value of infinity otherwise. An algorithm for incorporating trust dimensions into the role based task assignment for dynamic teams is shown in algorithm 2. Using this algorithm, a robot could check whether a robot is trusted to perform well in that role by mapping roles to trust dimensions and checking whether the trust value for that dimension is below a given threshold.

Input: An auction, a.
Input: The set of posted bids, B_a.
Input: The role requested for the task, $role_n$.

1: $\tau_n \leftarrow getTrustByDimension(role_n)$
2: $winner_{b_a} \leftarrow Min(B_a); \forall b_a \in B_a \ni \tau_n < \theta_n$
3: $AnnounceWinner(winner, a)$

Algorithm 2: Algorithm: using trust dimensions for role based task assignment.

6.1.5 Model Annotations

We have included in our model an ability to add *annotations* to a trust model. An annotation is a special type of marking that is placed onto a trust model for a given trust dimension that marks exceptions to a set of observations and can be used to affect the interpretation of the trust value. The annotation can be used to explain a set of observations and the

interpretation can be mission specific. An example of where an annotation might be useful is to mark a situation in which a robot is performing poorly in a time based task because it has taken additional tasks from a team member that needs assistance. The purpose is to provide a mechanism for robots or observers to update the trust model with an explanation that would prevent the robot from being punished for actions that benefit the team but are detrimental to the robot's local utility.

An annotation record includes the author of the annotation, the trust dimension that it applies to, the current time stamp, the expiration time and the estimated variance value from the desired performance. For scalar performance dimensions, when a robot is deliberating of the trust model it can consider annotations in calculating the trust value. As an example, if a robot picks up additional task from a teammate during the patrolling task, it can update the reputation authority with annotations to reflect the additional time necessary to perform its patrol cycle. This would variance would be added to the expected cycle time to determine whether a robot has exceeded the performance related to that dimension.

6.2 Monitoring with Trust

At this point, we can relate how the control charts from section 5.4 are used to inform the trust model to monitor task performance in multi-robot auctions.[1] To use a control chart, it is necessary to first define the value for the CL, which is the mean performance. On a multi-robot auction, this value could be interpreted as the average cost factor for performing a task. In the ideal case, the average is 1: each robot perfectly estimates the amount of time that it will take to complete a task. However, in practice, this value could change dynamically, based on environmental factors, the number of tasks being assigned, the path cost heuristic being used, and other factors. We define the value for CL to be the equal to the mean cost factor for the *good performing* type robots to complete tasks under normal conditions. Initially, we define the set of *good performing* type robots to be the entire team. We will define trust in this case to be directly related to performance: a *trusted* robot is one that estimates and performs tasks efficiently. However, using the trust model described

[1] The experiments in this section appear in [119].

in section 6.1, the set of trusted robots could change if the performance is shown to be out of control.

The pseudocode for this algorithm is shown in Figure 40. Each robot on the team maintains a separate trust model and control chart for each team mate. At each time step, the robot samples the cost factor values for tasks that have completed, and maintains a running average over a time window, R. We incorporate a running average to allow for minor changes in the environment that would affect all robots equally (such as an increased number of tasks.) The running average is used as the value for CL on the control chart.

To calculate the running average, we include only those averages from other trusted robots, see Figure 41. In line 2, we check whether an agent is trusted, see Figure 42, before including that agent in the running average. We also calculate the standard deviation of the running average for all trusted robots, and add that to the CL line to get the value for the UCL line, as shown in Figure 41, line 8. The LCL does not apply in this case as we are only concerned with robots that exceed their time estimates. To smooth variations in the performance values, we also calculate a running average for the cost factor. When the cost factor exceeds the UCL value, an out of control condition is detected.

At this point, the trust model is updated with a β signal to reflect that the robot did not perform well. If the robot performed close to their original estimate, the trust model is updated with the α signal to reflect that the robot is a *good performer*. The trust model provides an additional level of smoothing in the data and can prevent a single bad reading from causing a robot to be untrusted. The parameters of the model can be adjusted to adjust the rate at which changes affect the outcome. Additionally, the trust model allows for the incorporation of observations from other trusted robots, as described in Section 6.1.1.

Once a robot becomes *untrusted*, their performance characteristics are no longer included in the running average calculations for the process mean. Additionally, the robot can use this information as part of the decision process for handling a poor performing team member (such as notify an operator, provide assistance, to adjust the task allocation, etc.).

```
1: t_actual ← (CompleteTime_{B_p} − StartTime_{B_p})
2: Reward_{B_p} ← f(t_actual)
3: ActualCost_{B_p} ← (t_actual)
4: CostFactor_{B_p} ← ActualCost_{B_p}/EstCost_{B_p}
5: runningAvg_p ← CalculateRunningAvg(CostFactor_{B_p})
6: UCL ← CalculateUCL()
7: if runningAvg_p ≥ UCL then
8:    UpdateTrustModel(CostFactor_{B_p}, β)
9: else
10:   if runningAvg_p ≈ CL then
11:      UpdateTrustModel(CostFactor_{B_p}, α)
12:   end if
13: end if
```

Figure 40: OnTaskComplete() pseudocode. The cost factor, $CostFactor_{B_p}$, is the ratio of the estimated vs. actual task completion time. The running average of the $CostFactor_{B_p}$ is monitored using a control chart and when the process is out of control the trust model is updated.

```
1: for all r in RobotTeam do
2:    if CanTrust(r) then
3:       runningAvg_r ← CalcRunningAvg(CostFactor_{B_r})
4:    end if
5: end for
6: CL ← CalcAvg(runningAvg_R)
7: stdDev ← CalcStdDev(runningAvg_R)
8: UCL ← CL + stdDev
9: return UCL
```

Figure 41: CalculateUCL() pseudocode. The UCL value is calculated as the mean running average value of all trusted agents, plus one standard deviation.

```
1: if τ ≤ MinTrust AND γ ≥ MinConf then
2:    return FALSE
3: else
4:    return TRUE
5: end if
```

Figure 42: CanTrust() pseudocode. The beta trust model can be used to determine if an agent is untrusted, when a robot has a low trust value, τ, with high confidence, γ.

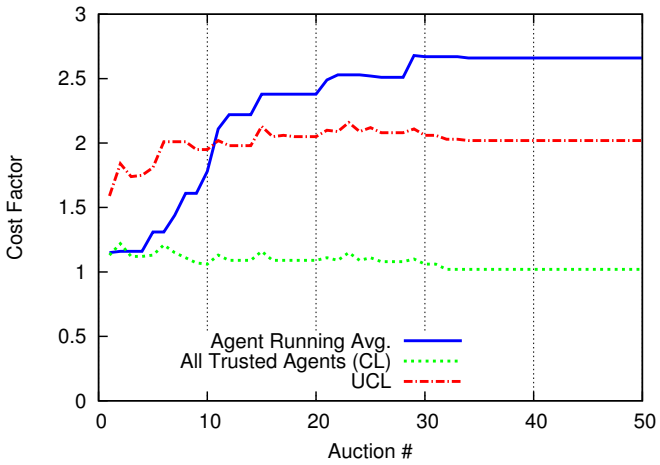


Figure 43: Detecting performance of robots that consistently underestimate the cost for performing tasks. Here, the control chart is shown for one of the agents that is not performing well.

6.2.1 Experimental Results

A set of experiments were performed in simulation to test the cost factor learning approach in a multi-agent auction environment. In these experiments, each robot has 50 tasks that arrive at regular intervals and are sequentially auctioned by that robot's auctioneer. As part of the auction process, they also bid on their own tasks. No currency is actually exchanged as part of the auction framework.

Rewards are given for task completion to the robot that originated the task. Each robot submits bids that represent the time-based cost for completing a task. Specifically, the bid represents the number of time steps until the task could be completed. Once a robot finishes all tasks in its list, they no longer accumulate costs in the simulation. The initial locations of the robots and the tasks are randomly chosen for each iteration.

Task Estimation

In this section, the source for estimation error is assumed to be due to *poor performing* type robots having an incorrect model of their own performance capabilities. To simulate robots that bid and execute poorly, a percentage of robots on the team are modeled as *poor performer* types and a cost factor is applied to their movements to slow their progress. However the robots themselves have no knowledge of the change to their state.

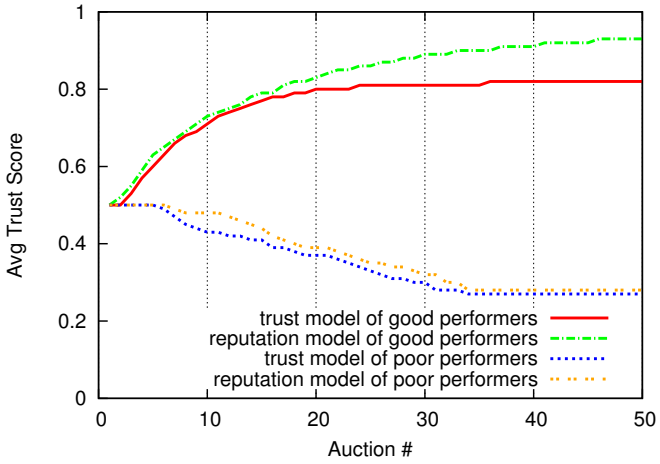


Figure 44: Trust Model. The beta trust model can be used to determine when an agent can no longer be trusted to reliably perform a task. Negative feedback is given to the model whenever an agent exceeds the limit on the control chart.

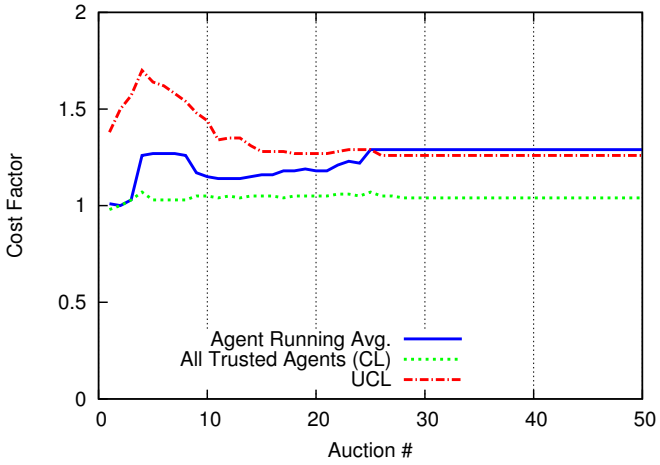


Figure 45: Detecting when the performance of team members deteriorates over time. The control chart shown is for one of the poorly performing agents.

Consistently Poor Performance

In this experiment, 2 out of 6 robots on the team are marked as poor performers. Their performance is adjusted by randomly assigning a cost factor at the start of the experiment, using a normal distribution with $\mu = 3$ and $\sigma = 0.1$. When a *poor performer* bids on a task, the unknown cost factor is drawn from this distribution and is applied to the robot's task performance to simulate error in estimation and execution. As a result, the robot continually underestimates the costs for performing tasks. Each robot on the team observes the tasks completion times for tasks that others have completed on their behalf. In this case, the cost factor for the *poor performers* exceeds the upper control limit after about 10 auctions have completed. An example of the control chart for one of the *poor performers* in the experiment is shown in Figure 43.

The corresponding trust model values for this experiment are shown in Figure 44. The model shows the convergence towards trusted values (above 0.5) for the *good performer* type robots and to untrusted values (below 0.5) for the *poor performer* robots, as the beta trust model is updated with feedback from the process as described in Figure 40. This result also shows that by incorporating values from other trusted agents, as described in Section 6.1.1, the trust values are further increased for the *good performer* robots.

Performance decreasing over time

In this experiment, all robots start out as *good performers*, but in 2 out of 6 robots, the performance deteriorates over time. To simulate deteriorating performance, the cost factor of the 2 deteriorating robots is drawn from the same distribution as in the previous experiment. However, while the initial values are $\mu = 1$ and $\sigma = 0.1$, the μ and σ values are increased by small values at each time step to simulate a gradual deterioration in performance. The process monitoring algorithm is able to detect the deterioration after about 25 auctions have passed when the cost factor reaches the UCL value. An example of the control chart for one of the *poor performers* in the experiment is shown in Figure 45.

The ability for robots to detect when the performance of team members begins to deteriorate would allow for a local approach to addressing the problem. For instance, the

robots themselves could include performance characteristics into their decision process or choose to only assign the most important tasks to the better performers on the team.

6.2.2 Summary

The use of a trust model in combination with control charts allows for robots to reason over the model and to share it with team members. The model can also easily be extended to include multiple dimensions of trust. This approach incorporated the use of the trust model in relation to the trusted peers on the team. In the next section, we will see an application of the trust model against a single known good performer in the shadow role, as part of a multi-UAV patrol.

6.3 Building Trust Models through Observation using a Team of UAVs

In section 5.5, we presented our previous work on the patrolling problem with mobile indoor robots investigated the use of a monitoring approach for determining when to decide which team members are no longer effective and to perform task re-assignment. Our experimental results indicated that approaches that include performance monitoring perform better for maximizing patrolling frequency than those that do not consider performance dimensions.

Many recent approaches to the patrolling task represent areas in the environment with a topological map (a graph). The nodes in a graph represent areas of interest in the environment, and edges in the graph represent traversable paths between two locations. Calculating the optimal path is known to be *np-hard*, and this problem is closely related to the *Traveling Salesman Problem*[31]. This assignment of patrol locations to multiple UAVs can be treated as a multiple vehicle routing problem with multiple depots[152], and on UAVs with heterogeneous flight characteristics [99].

As presented previously, in section 4.1, experiments in a multiple UAV sensing task discussed techniques for including a Bayesian formulation of target detection likelihood into this auction based framework for performing task allocation across multi-robot heterogeneous teams. However, in this section, we assume that the sensor models are not known in

advance, and the trustworthiness of a sensor platform is therefore unknown.[2]

6.3.1 Monitoring

Approaches to monitoring depend on the environment, but may include human observation, and observation using other robots, or sensors, as described in Section 5.2. In this section, we consider an approach in which we have a dedicated robot that serves in the *monitor* role by shadowing each of the robots in turn and observing their performance. In the multi-robot patrolling task, each robot has a set of patrol locations that are visited repeatedly. The *shadower* robot selects one of the team members at random, the *shadowee*, and follows its trajectory while performing sensor observations. We assume that the *shadower* robot carries a sensor with a high probability of detection, and is considered to be trusted . The sensor models for each of the other team members are unknown.

The *shadower* robot is not given the trajectories of each of the other teammates, but we assume that the *shadower* can observe the pose and velocity of the *shadowee*. The *shadower* implements a control law to follow the position of the *shadowee* at a small offset. We further assume that the teammates each report when they have visited a location and the outcome of the sensor observation (detected, not detected), and that the *shadower* receives these messages. When the *shadower* hears a sensor observation from the current *shadowee*, they take their own sensor reading of the location and use that to verify the result . This process is shown in pseudocode, in Algorithm 3.

```
1: if (r == shadow_r) then
2:    |Observations_r| += 1;
3:    S_s ← GetSensorObservation(S_s);
4:    v ← Verify(S_r, S_s);
5:    if (v == true⊕) then
6:       UpdateTrustModel(r, α);
7:    else if (v == false⊕ or v == false⊖) then
8:       UpdateTrustModel(r, β);
9:    end if
10: end if
```

Algorithm 3: OnSensorReport

[2]Experiments in this section appeared in [114].

Periodically, the *shadower* will probabilistically switch to shadowing a different team member. This is shown in Algorithm 4. It is worth noting that in this approach to monitoring, there is an explicit cost associated with monitoring each team member. Intuitively, we wish to focus monitoring resources on those team members that we have the most uncertainty or the least amount of trust. The trust model provides a mechanism for confidence and we can choose to stop shadowing a team member, once a confidence threshold has been reached. In addition, we can weight the distribution of team members, according to the amount that they are trusted or by the level of confidence, and sample from the weighted probability distribution to get the next *shadowee*. This results in the untrusted team members or those with least amount of trust information being shadowed more frequently.

1: **loop**
2: Do Every p Seconds:
3: **if** $(|Observations_r| > k)$ **then**
4: $shadow_r \leftarrow NextShadowee(T)$;
5: **end if**
6: **end loop**

Algorithm 4: DoShadow

6.3.2 Experimental Results

We performed experiments of UAVs performing a multi-robot patrolling and sensor task, using a high fidelity simulation of the autopilot system and autonomous behaviors. The purpose of this experiment is to demonstrate the approach to monitoring the sensor capabilities of team members while building a trust model online.

Experimental Setup

The multi-UAV simulation is motivated by our UAV research platform, described in Appendix A. In this section, we describe the setup of an experiment that demonstrates this approach with a team of UAVs performing a multi-UAV patrol. The tasks for each UAV are to repeatedly visit each location in their set of visit locations, shown in Figure 46, and to report whether a target has been detected at that location. To perform this experiment, we ran four autopilot SIL simulations. Three of the UAVs are designated as *patrollers* and

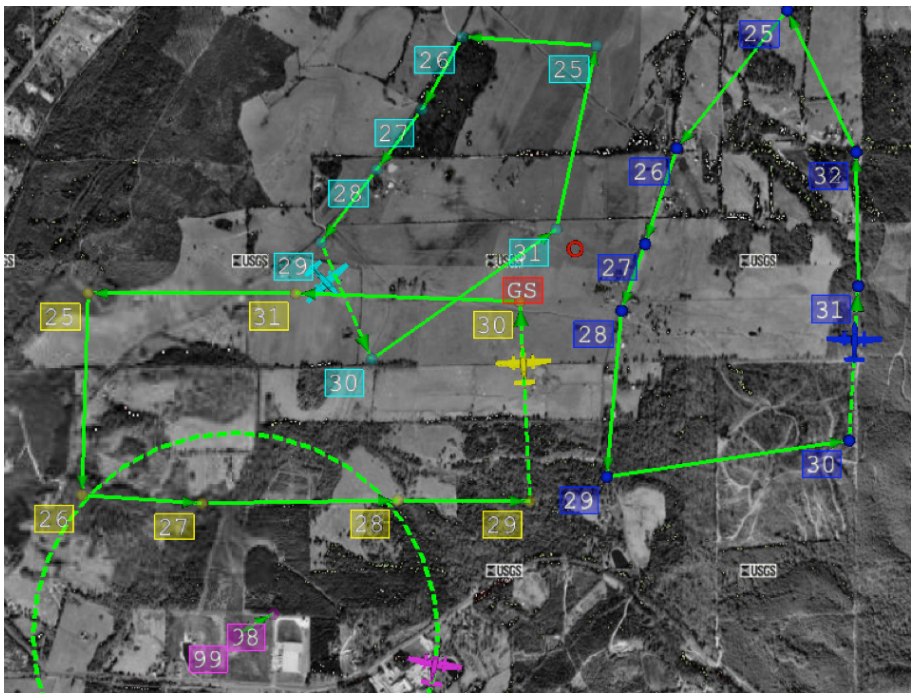


Figure 46: Multi-UAV Patrol: The four UAVs in the experiment are shown in the autopilot ground station display. The three patrolling UAVs are assigned patrolling locations in advance. The fourth UAV, the *shadower* monitors by following team members and verifying sensor observations. The experiment is performed using the high fidelity SIL simulations of the four autopilots.

are each provided with a subset of locations to visit and perform a sensor reading. The fourth UAV is designated as a *shadower* and follows each of the *patrollers* in turn.

Each *patroller* UAV position is observable by the *shadower*, and the *shadower* executes a control law [40] to intercept and follow the currently selected *patroller*, designated as the *shadowee*. The control law is motivated by the model-free controller presented by Egerstedt [40], with modifications to account for the minimum turning radius and velocity bounds of the UAV airframe. For this experiment, the autonomous behaviors run within the ROS framework on a virtual machine and communicate with the autopilot simulations over the local network. The behaviors to command the *patrollers* and *shadower* are implemented in Java, while the control law is implemented in C++. The *shadower's* controller behavior receives as input the position of the current *shadowee* and sends bank angle and airspeed commands to the autopilot. There is also a separate central trust authority process that listens to trust report messages from the *shadower* and maintains the trust model for each team member.

Sensor Modeling

In this experiment, we assume that each UAV carries a single sensor that has an unknown value for the probability of detection (POD) of the target. There are three different sensor types, (S_1, S_2, S_3). These sensors return a binary detection value, ($sensed\_target, not\_found$). We also assume that the probability that a target will exist at a given search location is $P(Target) = 0.25$. The Sensor-Target Probabilities for P(S) vary for each of the three sensor types, and are given in Table 7. Sensor S_1 is considered reasonably accurate, S_2 has the least accuracy, with a high false-positive rate, and S_3 is very accurate. Given the prior probabilities, $P(T)$ and $P(S)$, Bayes' rule can be used to find the posterior, $P(T|S)$ as shown in Equation 19. As one might expect, using Sensor S_3 leads to a very high probability that a target exists if the sensor returns a positive detection. We model the *shadowee* as having a perfect sensor.

To simulate the sensing task, every n seconds, a sensing process samples using $P(Target)$ for each visit location to determine whether a target exists. This simulated ground truth information is shared with a sensor simulation process that runs on each UAV. When a UAV reaches a visit location, the sensor simulation process for that UAV draws from the r^{th} sensor's POD distribution, shown in table 7, based on the ground truth entry for $P(Target)$, and the sensor returns a value in ($sensed\_target, not\_found$). This value is reported to the rest of the team as a *result* message.

Immediately after hearing the *result* message, the *shadower* takes a sensor reading at the same location and verifies the *shadowee's* observation and updates the trust model using an *Update Trust* message to the central trust authority, as shown in Algorithm 3. Note that true negative observations are not reported to the trust authority, but that confirmations of true positives, false positives and false negatives are reported. We ran the experiment for approximately an hour. At the start of the experiment, each UAV begins patrolling the visit locations that were assigned to them, as shown in the map display in Figure 46. The *shadower* UAV then randomly selects a *shadowee* by drawing from the distribution of team members, weighted by the trust score. The *shadower* selects a new *shadowee* after verifying ten sensor observations.

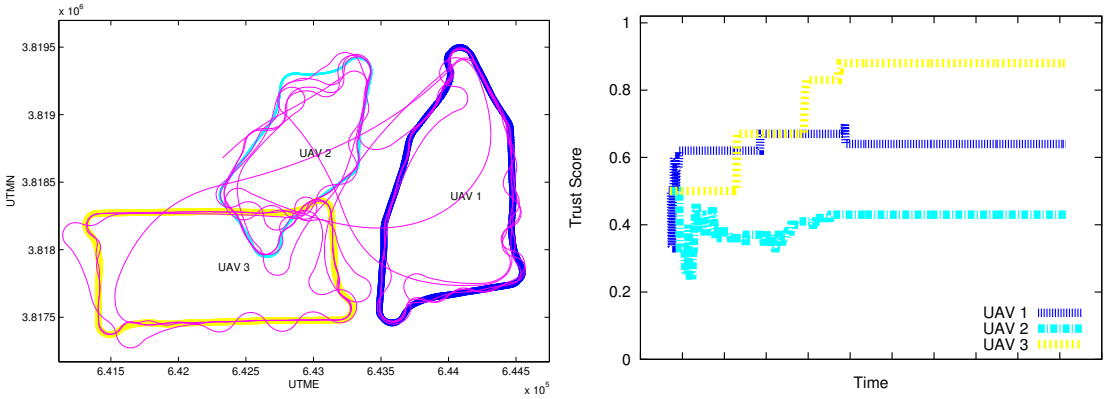
Discussion

As the *shadower* verifies the observations for each team member, it sends the updates to the central trust authority. The trust scores for each UAV as the experiment continues are shown in Figure 47(b). Over time, the trust scores converge to match the ordering of the unknown sensor models' POD, with UAV 3 (carrying S_3) being the most trusted, UAV 2 being the least trusted, and UAV 1 having an intermediate trust score.

In this experiment, the trust model is one-dimensional and the score reflects the unknown sensor model for each UAV. Indeed, the multiple observations over time could be thought of as a training period, in which we gather enough observations to estimate the underlying sensor POD. However, in a more general application, trust model could contain additional performance dimensions as dictated by mission requirements.

In this approach, the *shadower* draws from the weighted distribution of trust confidence scores to select the next *shadowee*, with team members having unknown trust information being weighted more heavily. Depending on the mission requirements, once the trust model is updated with a sufficient confidence level, the task assignment and teaming structure could be changed and the *shadow* resource could assist with patrolling tasks. A benefit of this approach is that any trusted team member could serve as the monitor. Additionally, after an initial observation period, the shadower could return to other tasks, and allow another team member to serve as a monitor at a later time. Finally, this monitoring approach could be combined with others to ensure robust performance of the team.

The trust model can be used to inform the task assignment function or the team formation. In other experiments, untrusted team members were removed from the team[118]. In this case, their tasks can be reassigned to other team members by performing the task allocation with one fewer team member. As presented in this section, there may be additional metrics, such as the accuracy of the observations at each location that should be included in the task assignment approach.



(a) Flight Trajectories

(b) Trust Scores

Figure 47: Trust Based Monitoring: a) The trajectories for each UAV are shown for a sample experiment. The *shadower* UAV switches between each teammate to perform observations. b) As the *shadower* UAV performs multiple observations, it sends updates to the trust authority for each UAV observed.

6.3.3 Summary

The multi-UAV patrolling problem has requirements for teams that can perform the patrolling task securely and reliably. As part of this, team members need to be trusted that they can perform the patrol objectives correctly and sense targets in the environment effectively. On dynamically formed, or ad-hoc UAV teams, the sensor characteristics of each team member may not be trusted in advance, and team members should be able to observe each other to ensure that they are performing as expected. This section presented several dimensions of performance that can be used to define the trustworthiness of a UAV in the patrolling task, and presented approaches to teammate monitoring. An experiment was performed using a multi-UAV simulation of the patrolling task in which a dedicated *shadower* UAV verified the sensor observations of team members, to build a model of trust for each team member. This model can be used to inform the task assignment strategy or to revisit the formation of the team.

This chapter presented the trust framework and experimental results using that framework for monitoring and in a patrolling task. The next chapter will apply the framework further, with emphasis on team formation and task allocation strategies based on trust and the reliability of partners.

CHAPTER VII

APPLICATION OF THE TRUST FRAMEWORK TO TEAM FORMATION AND EVOLUTION

In the previous chapter, we presented our formulation for the trust model and applied it experimentally to demonstrate the use of the model for estimating the reliability of partners. In this chapter, we further apply the trust model to show how the trust information can be used to affect team formation and evolution and to provide incentives for cooperation.

As indicated by the title of this chapter, the *team formation* strategy can use the trust model to decide which peers should be included as part of the team. In a multi-robot auction, this refers to which team members are trusted to estimate tasks, complete tasks and participate in task allocations. In a multi-robot patrolling domain, *team formation* refers to which team members are trusted to complete tasks and the untrusted team members may have one or more tasks removed from their allocation. When we discuss the *evolution* of the team strategy, this relates to the ability for robots to dynamically adjust the allocation of tasks to team members, based on the trust model which is updated online.

In section 7.1, we discuss the use of incentives for cooperation using game theory, and use the trust model as an incentive mechanism. We further investigate the use of incentives as part of reputation mechanism, in section 7.2. In section 7.3, we use the model to determine which team members cannot be trusted to participate in task assignment, and to isolate the untrusted team members in experiments on a UAV platform. Finally, in section 7.4, we apply the use of a trust model to the multi-robot patrolling domain, and reassign tasks away from untrusted team members based on an evolving trust model.

7.1 Incentive Based Cooperation in Multi-Robot Auctions

This section will present a game theoretic approach for providing incentives to cooperation in multi-robot auctions using an observation based trust model.[1] This approach can be used to select team members for auctions by selecting those agents that cooperate effectively. In multi-robot auctions, typically there is an implicit assumption that agents are willing to cooperate and can be trusted to perform assigned tasks. However, reciprocal collaboration may not always be a valid assumption. An approach to incentive based trust is presented, which enables detection of team members that are not contributing and for dynamic formation of teams.

7.1.1 Motivation

The basic auction approaches to the task allocation problem assume that team members can be trusted and have the goal of the team in mind (to reduce the overall cost) [70]. These algorithms serve as a mechanism for distributed task allocation and generally do not need to consider incentives. As such, these methods do not explicitly account for trust between team members, but assume that *a)* team members will bid on tasks that are presented to them and *b)* team members will attempt to perform tasks that are assigned to them. However, there are situations in which teams may be formed dynamically. While the team may have the same common goal, the individual robots may have different levels of interest in the cooperation. That is, some of the team members may place a higher utility on successful completion of tasks, while others are obligated to participate, but wish to conserve resources. In these situations, it is assumed that the non-cooperative agents will not attempt to sabotage operations, but they may not fully cooperate either. Agents should prefer to participate in teams because this will allow them to assign tasks to others that might complete them more efficiently. However, this means that they will be required to assist others in return.

[1] Experiments in this section appeared in [116].

Dimensions of Trust

Trust and reputation (shared trust) mechanisms can be applied to auction algorithms for determining dynamic team formation. This work investigates the use of observation based trust and game theory mechanisms for determining when to remove a non-cooperative team member from an auction team by ignoring its auction requests. If a robot is no longer on a team, it loses opportunities for others to assist it with tasks when those tasks could be done more efficiently as part of a team than alone. In the auction context, robots that do not bid on each other's tasks can be viewed as non-cooperative and removed from a team. From a robot's viewpoint, it is better to have team members that cooperate and participate in the auction algorithm as this leads to more efficient outcomes. From a global viewpoint, it is desirable to have an efficient team that is composed of cooperative members; each non-cooperative member decreases the overall team performance. Therefore, it is desirable to perform dynamic team formation by allowing team members to perform auctions only with other cooperative team members.

In a dynamically formed auction team, agents may encounter other agents for which they have no prior experience. The use of a trust model would allow for an agent to reason about other agents' trustworthiness using observation histories and reputation information. In these settings, there are multiple dimensions that could be used to define trust, such as auction participation and task completion. This work will consider participation in auctions to illustrate the use of incentives for cooperation. However, additional trust dimensions could also be applied to this framework.

7.1.2 Trust Model

This work incorporates the use of the trust model from section **??** for incorporating direct trust and reputation into a probabilistic formulation. This mechanism provides not only a trust belief about an agent, but also a confidence. The approach uses the beta probability distribution function [154] and can incorporate positive (α) and negative (β) histories to calculate the belief and confidence. Each agent maintains a set of α and β vectors that

represent the histories of interactions with each team member. Regarding auction participation, when a agent within range is sent an auction announcement and they do not respond with a bid, this is counted as a β observation while a bid response is counted as an α observation. An agent is initially trusted until sufficient β observations cause the trust value to be low, with high confidence.

We also use this mechanism to incorporate the reputation information (indirect observations) from other trusted team members using the same approach. However, the shared reputation information must be combined with the locally observed trust vectors. In our auction framework, each agent regularly posts their trust model's α and β vectors to all other team members that are within range. In addition, agents only incorporate those updates from other currently trusted team members. These shared, indirect observation vectors are easily integrated into the local vectors and the scalar trust and confidence values are recalculated, as described in section 6.1.1.

Each time that an agent receives an auction message from another agent, they can evaluate the trust model to determine whether to participate. If the calculated trust value is less than the trust threshold, ϕ, and with confidence greater than γ, it is not trusted. However, a succession of positive observations (direct or indirect) can move an untrusted agent back to being trusted again. Furthermore, this approach is tolerant of noise as it can take multiple observations to move the value above or below the trust threshold.

7.1.3 Basic Auction Approach

In the basic multi-agent auction algorithm, the problem is to assign tasks to agents. In this work, the tasks are to visit a target location and perform an observation. In the auction framework, each robot is a *bidder* and the items to be auctioned are the 'visits'. Each of the agents in the system also participates as an *auctioneer* and periodically auctions new task requests (it is assumed that the task requests are provided to the agent by an external process, such as a human operator or other event). This approach can easily be used on teams with different robot characteristics: each robot knows their own location and cost function and submits cost based bids to the auctioneer. While costs and rewards use the

same basis for calculation, no revenue is actually exchanged.

The approach followed by the *auctioneer* is shown in Procedure 5. The *auctioneer* first handles any auctions that have already been announced and are ready to close. This step is shown in detail in Procedure 9. In lines 1-3, the *auctioneer* selects the maximum bid from all bids received by the agents within communications range (including their own) as the winner of that auction and performs the task assignment by announcing the winning bidder. In lines 5 and 7, the *auctioneer* updates the trust model (described in Section 6.1.1) for each possible *bidder* that was sent the auction announcement. The trust model is referenced by the *bidder* in Procedure 10, when an auction announcement is received. If the originator of the auction announcement is not trusted, using the trust model, then the auction announcement is ignored, effectively isolating the untrusted agent from the benefits of cooperation.

In this work, each target to be visited has a reward that is linearly decreasing with time (for example, consider a hurricane survivor scenario or forest fire scenario in which time to discovery is critical). The agents each maintain a current task list and locally compute their bid to complete the proposed task. In this case, the bid consists of the surplus gain per unit time for them to perform the task, in addition to all of their other tasks, where surplus is defined as the total reward collected minus the total travel cost [41]. Each robot also incurs a small bidding cost with each bid. This represents the amount of computation and communication resources that need to be consumed to calculate and send the bid.

For each auction announcement received, the agent calculates their bid as shown in Procedure 10. The surplus gain in unit time (*sgut*) is calculated as the change in surplus for inserting the task into the current task list. The incremental travel cost is known as the cheapest insertion heuristic: for each pair of tasks in the current task list, the agent compares the additional Euclidian distance based cost for inserting the new task, and selects the insertion that maximizes its surplus gain, which forms the agent's bid. When the winning *bidder* is assigned a new task, the task is inserted into the agent's task list, again using the insertion heuristic.

Input: The set of open auctions, A_{open}.
Input: The set of new task requests, $TaskRequests_{new}$.

1: **for all** $a : A_{open}$ **do**
2: $HandleAuctionBids(a)$
3: **end for**
4:
5: **for all** $a : TaskRequests_{new}$ **do**
6: $Recipients_a \leftarrow AnnounceAuction(a)$
7: **end for**
8:
9: $ReauctionRemaining(n, tasklist)$

Algorithm 5: $Auctioneer :: PerformAuctions$

Input: An auction, a.
Input: The set of posted bids, B_a.
Input: The set of announcement recipients, $Recipients_a$.

1: $winner \leftarrow Max(B_a)$
2: $AnnounceWinner(winner, a)$
3: **for all** $a : Recipients_a$ **do**
4: **if** $a \in B_a$ **then**
5: $UpdateParticipation(TRUST_o, 1)$
6: **else**
7: $UpdateParticipation(TRUST_o, 0)$
8: **end if**
9: **end for**

Algorithm 6: $Auctioneer :: HandleAuctionBids$

Input: An set of announced auction tasks, A.
Input: The auction originator trust model, $TRUST_o$.

1: **for all** $a : A$ **do**
2: **if** $CanTrust(TRUST_o)$ **then**
3: $bid \leftarrow CalculateBid(a)$
4: **if** $bid > 0$ **then**
5: $PostBid(bid)$
6: **end if**
7: **end if**
8: **end for**

Algorithm 7: $Bidder :: HandleAnnouncements(A)$

7.1.4 Social Norm Strategy

At this point, basic concepts from game theory [103] can be introduced to show how incentives can be used to induce cooperation on auction teams. Consider the well known two player game from the game theory literature, the *Prisoner's Dilemma (PD)*, shown in

Input: An auction task, a.
Output: The agent's *bid*.

1: $[wri, dri] \leftarrow CalculateSurplus(tasklist)$
2: $[wri', dri'] \leftarrow CalculateInsertion(tasklist, a)$
3: $wri' \leftarrow wri' - BidCost$
4: $sgut \leftarrow (wri' - wri)/(dri' - dri)$
5: **return** $bid \leftarrow sgut$

Algorithm 8: $Bidder :: CalculateBid(a)$

Table 3. The payoff table reflects values of T for *temptation* to "defect", R representing the *reward* or for cooperation, P for *punishment* related to joint defections and S for *sucker* related to unilateral cooperation. The payoffs satisfy the following condition:

$$T > R > P > S \tag{18}$$

In a single round of play the rational player in PD should choose to defect. However, in repeated games, players will meet each other multiple times and can consider the history of their opponent's actions in determining an action. If there is a threat of punishment, then cooperation can be induced in repeated play. There are several strategies that can be used to induce cooperation in repeated play, such as Tit-for-Tat, which is discussed further below.

Cooperation on multi-agent teams can also be modeled using the PD game. In each round of an auction, players are matched by the rules of the auction and can choose to participate (cooperate) or not participate (defect). Here, it is assumed that players will be repeatedly matched against each other. The global team score will be better if all agents fully participate in auctions, not just when it suits their interests. For instance, it is possible for agents to take advantage of the auction setting to allow others to perform their tasks while not performing others' tasks in return. The disincentive to cooperate could be attributed to selfishness of uncooperative agents, agents that are overloaded with tasks have have nothing to offer, or agents that are incapable of effective participation. Each interaction in the auction setting can be treated as a two-player game.

The game is modeled as a prisoner's dilemma, where each interaction represents two separate auctions, one initiated from each player, shown in Table 5. The players cooperate by bidding on each other's auctions and defect by not submitting a bid or a bid that is

valid. This game can be treated as a random matching game, because it is assumed that if the game is played for a long enough time horizon, each player will eventually have an opportunity to bid on the other's auctions. The payoffs in this game are as follows:

- $R = b - c$: Benefit (time discounted reward) when another agent completes a task minus the cost for performing a task for that agent.
- $T = b$: Benefit (time discounted reward) when another agent completes a task.
- $S = -c$: Cost for unilaterally performing a task on behalf of another agent.
- $P = 0$: There is no additional gain if neither player cooperates.

The Tit-for-Tat strategy can be useful for inducing cooperation, but it is sensitive to noise and does not allow for the agent that was defected against to quickly recover from defect losses. This strategy is also dependent on repeated interactions as part of the random matching assumption. However, there are situations in which agents interact but change partners frequently and may not have a chance to apply timely punishment after an interaction. A strategy that uses a community model for conveying trust is the social norm strategy as given by [68]. The strategy requires that each agent is associated with a reputation label which is visible to all other agents in the community. The social norm strategy relies on a (generally centralized) reputation authority that observes pairwise interactions between players and assigns each player's label as either *Innocent* or *Guilty*. The social norm strategy also allows for the defected-against agent to recoup loses. Cooperation is sustained because the strategy allows other agents in the community to apply sanctions when a defection occurs. When two agents meet the social norm strategy dictates the following approach:

- If both agents are *Innocent*, they both cooperate.
- If both agents are *Guilty*, they both defect.
- If one agent is *Guilty*, then the *Guilty* player should cooperate while the *Innocent* player defects. This allows for the *innocent* player to recoup reward. The *Guilty*

player effectively "repents" through unilateral cooperation.

- Any deviation from the above strategy marks the deviator as *Guilty* for τ rounds.
- After τ rounds of following the above strategy, a *Guilty* player is forgiven and becomes *Innocent* again.

7.1.5 Incentives for Cooperation

The social norm strategy for the PD game was shown by Kandori [68] to be a subgame-perfect equilibrium, if the agents use an appropriate discount factor, δ, and set the punishment period, τ, effectively. The discount factor reflects the willingness of the player in a repeated game to continue playing the game. A value of $\delta = 1$ reflects that the players are infinitely patient and expect the game to continue forever, while a $\delta \to 0$ means that agents prefer more immediate gains.

Reputation Authority

For the decentralized case, this work uses the distributed reputation authority, as described in Section 6.1.1, as the reputation authority that provides the labels for each of the players. Note that the distributed reputation authority relies on the combined direct and indirect observations in calculating an agent's label. This allows for a "sticky" reputation which is less sensitive to noise in the observations. While the social norm approach is still sensitive to noise (agents that do not bid can be counted as deviating from the strategy), the social norm approach allows for the guilty agent to recover.

Voided Contract

As mentioned above, the social norm strategy allows for the defected-against agent to recoup losses when a *guilty* agent follows the strategy and cooperates while an *innocent* player defects. However, we provide an extension to the strategy for use in auctions by performing additional punishment toward the deviator: any tasks in the *innocent* agent's task list that originated with the *guilty* (deviator) agent are dropped. In doing so, the defected against agent effectively considers the cooperation contract 'voided' and is under

Table 10: Reputation Authority Probability Model

		Innocent	*Guilty*
RA label	$\widehat{Innocent}$	x	0.10
	$\widehat{Guilty}$	$(1-x)$	0.90

no obligation to complete those tasks. This provides additional incentive for cooperation as the dropped tasks will not be completed and those rewards will therefore not be returned to the *guilty* agent (however, the *guilty* agent could elect to reclaim and execute the dropped tasks at presumably higher cost).

Probabilistic Forgiveness

In practice, a reputation authority will likely contain a small amount of error in the classifications that it provides. If an estimate of the error probabilities for the distributed reputation authority is known in advance, then it is possible to calculate the probability of incorrect classifications using Bayes' rule. For instance, consider the example probability model for a reputation authority as shown in Table 10, and let $x = 0.80$. This model reflects the probabilities that 80% of the time, an *Innocent* agent will be correctly labeled as $\widehat{Innocent}$ by the reputation authority and that 90% of the time, a *Guilty* agent will be correctly labeled as $\widehat{Guilty}$.

The noise in the model could be due to multiple causes, including communication error, noise in the observation, and error in classification. In the case that an *Innocent* agent is incorrectly labeled $\widehat{Guilty}$, the incentives for cooperation can breakdown. However, given a model of the reputation authority, it is possible to calculate the probability that an agent is actually *Innocent*, given that the authority labeled it $\widehat{Guilty}$, as shown in Equation 19. For instance, in this example, there is still a 34% probability that the agent is actually *Innocent*. In order to tolerate noise in the system, we can periodically reset the labels of some *Guilty* agents, before the end of the τ punishment period, by sampling from this probability distribution. This allows for truly *Innocent* agents to return to cooperative behavior as they will see that others are again cooperating with them.

$$P(Innocent|\widehat{Guilty}) = \frac{P(\widehat{Guilty}|Innocent)P(Innocent)}{P(\widehat{Guilty})} = \frac{(0.20)(0.70)}{(0.20)(0.70)+(0.90)(0.30)} = 0.34 \quad (19)$$

7.1.6 Experimental Results

A set of experiments were performed in simulation to test the trust strategies in a multi-agent auction environment. In these experiments, the robots are represented by unmanned aerial vehicles (UAVs) in the Mason simulation environment, described in Section A.1. Each UAV has 50 tasks that arrive at regular intervals and are sequentially auctioned by that UAV's auctioneer. As part of the auction process, they also bid on their own tasks. The UAVs in the simulation have a limited communications range and can therefore only perform auctions or exchange reputation information with a subset of the other team members at a given time.

In addition, each UAV periodically re-auctions the last n tasks to other agents in range. This allows tasks to be more optimally assigned by giving other agents a chance to bid on them if they were not in range during the initial auction. Rewards are given for task completion to the UAV that originated the task, and rewards decrease linearly with time until they reach 0. Each agent submits bids that represent the surplus gain per unit time for performing the additional task. Once a UAV finishes all tasks in their list, they no longer accumulate costs in the simulation. The initial locations of the UAVs and the tasks are randomly chosen for each iteration. For each set of experiments, results were averaged over 100 runs using 10 simulated UAVs.

Detecting and Punishing Defectors

In this set of experiments, a fraction of the agents on the team defect by not participating in auctions (not bidding on others' tasks). Each *Defector* agent only participates in auctions 10% of the time. As a result, naive agents (using no trust mechanism) end up doing additional work for the *defector* agents and receive nothing in return. The task for the *cooperator* agents is to detect those team members that regularly fail to participate in

auctions and to isolate them from future cooperation by not bidding on the *defectors'* tasks.

The agents that use the social norm (SN) strategy can quickly punish and isolate the defectors from the team by no longer bidding on their auctions. The results of this experiment, shown in Figure 48(a), reflect that the agents running the SN strategy receive better scores than those using beta trust and reputation methods alone, even as the fraction of defectors increases. Finally, the beta trust, reputation and SN methods all perform better than the naive strategy which trusts all team members unconditionally.

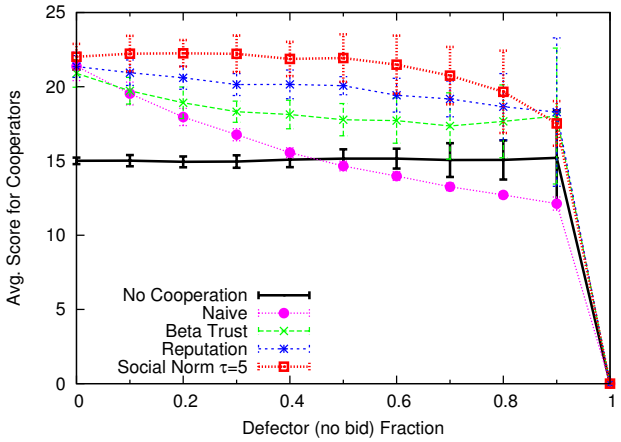
For this same experiment, the average score for all of the *defectors* is shown in 48(b), for each of the strategies employed by the cooperative agents. Clearly, the *defectors* do well when the *cooperators* run the naive strategy. However, the *cooperators* running the SN strategy provide strong incentives for the *defectors* to cooperate (when the *cooperators* run the SN strategy, the *defectors* receive much lower scores than the *cooperators*).

Noisy Reputation Authority

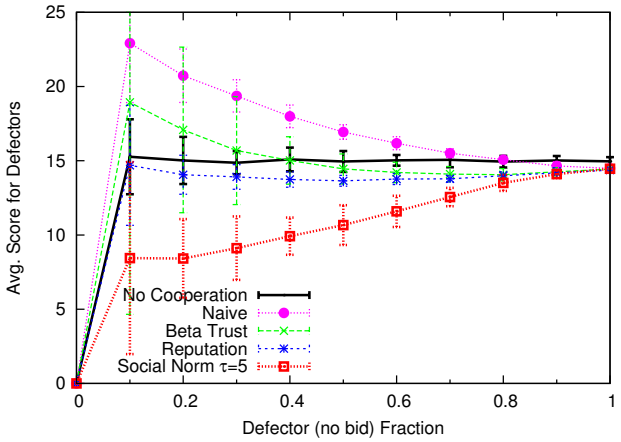
In some cases, the SN strategy can cause *innocent* agents to be punished unfairly. This can happen, as mentioned above, when the bid participation trust dimension is used and some agents do not submit bids because they cannot perform the task. In other cases, there may be noise in the reputation authority mechanism that marks some agents as defectors when in fact they cooperated or vice-versa.

In the following experiment, a noisy, decentralized reputation authority is compared against an accurate centralized reputation authority. The probability of incorrect label assignments by the reputation authority is show by the model in Table 10. With small probability, a *Guilty* agent will be incorrectly classified as *Innocent*, but most of the time will be correctly labeled. The experiment decreases the probability x that an *Innocent* agent will be correctly labeled.

The SN strategy with a centralized reputation authority provides the most favorable incentives for cooperation, resulting in the highest scores for *cooperators* and very low scores for *defectors*. However, in practice a central authority may not always be available and it



(a) Average Score for *Cooperators*



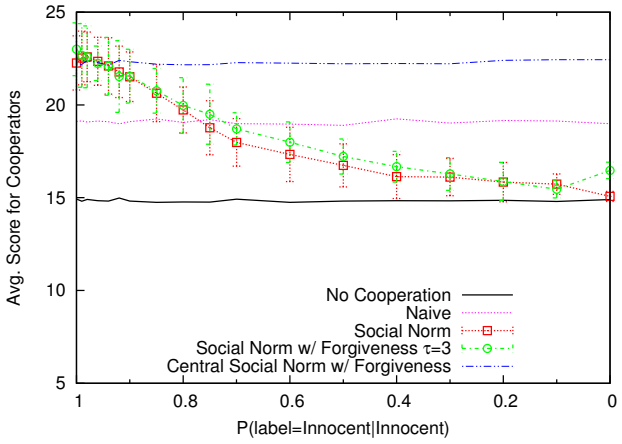
(b) Average Score for *Defectors*

Figure 48: Agents that defect by not participating can be detected and isolated using observation based trust mechanisms. The *defector* fraction is plotted against the average unit score of the (a) *cooperator* agents and (b) *defector* agents for each trust strategy run by the *cooperators*. The error bars reflect one standard deviation.

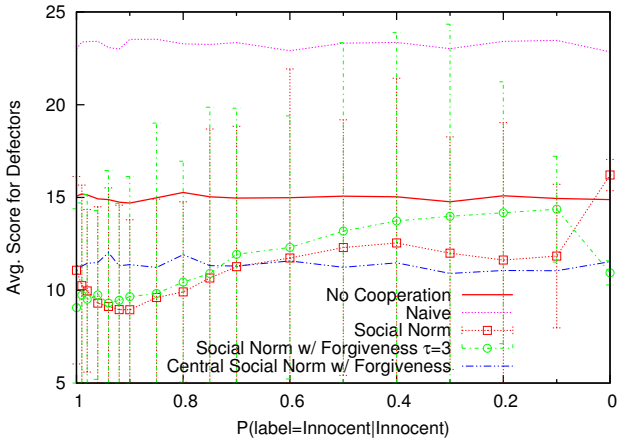
may be necessary to rely on the decentralized authority. With the decentralized authority, when *Innocent* agents are incorrectly labeled as *Guilty*, this can lead to a breakdown of cooperation. However, the SN strategy allows for forgiveness through different settings for the punishment period, τ. In addition, we allow for probabilistic forgiveness, to account for incorrect labeling as described in Section 7.1.5. Here, 20% of the agents are *defectors*. The results, as shown in the scores for *cooperators*, Figure 49(a), and *Defectors*, Figure 49(b), indicate that the SN methods provide sufficient incentives for cooperation, even as the probability for an agent being incorrectly labeled is increased.

For both SN strategies, the scores for cooperation exceed the scores for defection, when

the accuracy for correctly labeling innocent agents is above about 75%. In this case, the use of these strategies removes any incentive to not cooperate. Additionally, as the accuracy for correctly labeling innocent agents decreases below about 75%, the naive strategy results in better scores for cooperators than the SN strategies. This result is due to the unfair punishment of other *cooperators* because of the noise in labeling. As such, when noise levels in the decentralized reputation authority reach this threshold, it becomes worthwhile to improve the labeling accuracy or rely on a centralized reputation mechanism.



(a) Average Score for *Cooperators*



(b) Average Score for *Defectors*

Figure 49: The SN strategy provides strong incentives for cooperation, even as the reputation authority mislabels *Innocent* team members as *Guilty*. (a) The average score for following the SN strategies exceeds the average score for defection. (b) Incentives for defection are removed as the defector scores worse by defecting when the others are using these strategies.

7.1.7 Summary

Traditional auction algorithms for performing the robot task assignment problem assume that robots are equally incentivized to participate in auctions. However, there are situations in which agents may assign tasks to others on the team, without taking on a fair number of additional tasks in return. This section presents an approach for using observation based trust and a shared reputation mechanism in determining which agents to include in multi-agent auctions. The experimental results show that by incorporating the use of trust strategies into the basic auction mechanism, agents can perform better than agents that trust unconditionally. Furthermore, the introduction of punishment through isolation from future auctions and through dropping already assigned tasks provides incentives for cooperation in multi-agent auctions that weren't present in traditional approaches.

7.2 Trust and Reputation in Multi-Robot Auctions

This section describes prior work on the use of observation based trust and reputation models to enable detection and isolation of team members that are not contributing and completing tasks on multi-robot teams. Multiple dimensions of trust are considered and experiments are performed to show that methods that consider trust can be used to effectively select team members for continued participation in multi-agent auctions.

7.2.1 Dimensions of Trust

In a dynamically formed auction team, agents may encounter other agents for which they have no prior experience. The use of a trust model would allow for an agent to reason about other agent's trustworthiness using observation histories and reputation information. In these settings, there are multiple dimensions that could be used to define trust related to cooperation in auctions for task assignment. Consider the following dimensions:

1. *Bid Participation*: An agent bid on auctions that were announced.

2. *Bid Veracity*: An agent submitted a valid, reasonable bid on auctions that were announced.

3. *Task Completion*: An agent (correctly) completes tasks assigned to it.

Agents that regularly violate the trust dimensions are considered to be *defectors*, while agents that cooperate fully are labeled as *cooperators*. These dimensions of trust can be considered separately or in combination. Each agent can build models of other team members behaviors from observation histories and use those models to determine levels of trust.

Bid Participation

The *Bid Participation* method is the weakest of the trust dimensions for multi-agent auctions. It only considers whether an agent is participating by submitting bids as part of the auction process, but does not evaluate the bid. In fact, there are legitimate situations in which an agent might not wish to submit a bid, if the calculation is costly [25]. However, in domains in which the bid calculation is easily computed and communicated, this can be a useful gauge of auction participation. When an agent announces an auction, it keeps track of the agents that received the auction announcement and compares this to the list of agents that submit bids. This approach assumes that agents are uniquely identifiable and that a protocol exists for acknowledging the receipt of an announcement. When an agent does not bid on received auctions announcements, this negatively updates the trust model, while the submission of a bid positively updates the model (described further in section 6.1.1.) Using the model, if an agent determines that a team member is not trusted, the agent refuses to bid on the untrusted team member's future auctions, effectively isolating it from the auction team.

Bid Veracity

The next dimension, *Bid Veracity*, considers whether the bid was a truthful estimate by the agent. Specifically, if an agent purposefully bids low, as compared to other bids received, to avoid being assigned more tasks, that action would violate this trust dimension. A similar metric is bid accuracy. If an agent bids accurately, their bid estimates for performing a task are sent not only in good faith, but they also accurately and closely reflect the true cost to perform the task. This dimension could be used to detect agents that bid on and win

tasks, but are not able to perform them as cost effectively as promised. However, this work will focus on the situation in which agents purposefully submit low bids. When an agent submits a bid that is within a given range of the winning bid, then that is considered valid and the trust model is updated positively. However, if the agent's bid is considered within a low range as compared to the winning bid, that is considered invalid and the trust model is updated negatively. Here again, once an agent determines that a team member is not trusted, the agent refuses to bid on the untrusted team member's future auctions.

Task Completion

The *Task Completion* dimension considers whether an agent sufficiently completes tasks that were assigned to it as part of the auction process. However, this requires the existence of a monitoring mechanism which verifies whether a task has been completed successfully. The monitoring mechanism could be a human operator or observer, a specialized sensor, or even another agent [97]. This work assumes the existence of an accurate monitoring mechanism and that any assigned task can be monitored for a fixed cost. For each monitored task, if an agent is assigned a task and fails to complete it, then the trust model is updated with a negative result, while a successful task completion results in a positive update. In this case, when an agent becomes untrusted, they are no longer sent any auction announcements, to prevent them from being assigned any future tasks.

7.2.2 Auction Approach using the Trust Model

The tasks in this case are to visit a target location and perform an observation. In the auction framework, each robot is a *bidder* and the items to be auctioned are the 'visits'. Each of the agents in the system also participates as an *auctioneer* and periodically auctions new task requests (it is assumed that the task requests are provided to the agent by an external process, such as a human operator or other event). This approach can easily be used on teams with different robot characteristics: each robot knows their own location and cost function and submits cost based bids to the auctioneer. While costs and rewards use the same basis for calculation, no revenue is actually exchanged. Rather, an agent awards itself a utility value when one of its own tasks is completed.

The approach followed by the *auctioneer* is shown in Procedure 5. The *auctioneer* first handles any auctions that have already been announced and are ready to close. This step is shown in detail in Procedure 9. In lines 1-3, the *auctioneer* selects the maximum bid from all bids received by the agents within communications range (including their own) as the winner of that auction and performs the task assignment by announcing the winning bidder. In lines 5 and 7, the *auctioneer* updates the trust model (described in Section 6.1) for each possible *bidder* that was sent the auction announcement. The trust model is referenced by the *bidder* in Procedure 10, when an auction announcement is received. If the originator of the auction announcement is not trusted, using the trust model, then the auction announcement is ignored, effectively isolating the untrusted agent from the benefits of cooperation. The above procedures consider the *Bid Participation* dimension; however, the other trust dimensions are implemented similarly.

In this work, each target to be visited has a reward that is linearly decreasing with time. The agents each maintain a current task list and locally compute their bid to complete the proposed task. In this case, the bid consists of the surplus gain per unit time for them to perform the task, in addition to all of their other tasks, where surplus is defined as the total reward collected minus the total travel cost, as described by [41]. Each robot also incurs a small bidding cost with each bid. This represents the amount of computation and communication resources that need to be consumed to calculate and send the bid.

For each auction announcement received, the agent calculates their bid as shown in Procedure 8. The surplus gain in unit time, *sgut*, is calculated as the change in surplus for inserting the task into the current task list. The incremental travel cost is known as the cheapest insertion heuristic: for each pair of tasks in the current task list, the agent compares the additional Euclidian distance based cost for inserting the new task, and selects the insertion that maximizes its surplus gain, which forms the agent's bid. When the winning *bidder* is assigned a new task, the task is inserted into the agent's task list, again using the insertion heuristic.

7.2.3 Experimental Results

A set of experiments were performed to test the trust strategies in simulated auctions using the Mason simulation environment, described in Section A.1. In these experiments, the agents are represented as Unmanned Aerial Vehicles (UAVs), and each UAV is assigned tasks to perform by an external, Poisson process. Each UAV has an auctioneer and can auction their tasks to other agents, assigning the task to the agent that submits the highest bid. Agents also bid on their own tasks. Rewards are given for task completion to the agent that originated the task, and rewards decrease linearly with time until they reach 0. Each agent submits bids that represent the surplus gain per unit time for performing the additional task. The UAVs in the simulation have a limited communications range and can therefore only perform auctions or exchange reputation information with a subset of the other team members at a given time.

In addition, each UAV periodically re-auctions the last n tasks to other agents in range. This allows tasks to be more optimally assigned by giving other agents a chance to bid on them if they were not in range during the initial auction. Once a UAV finishes all tasks in their list, they no longer accumulate costs in the simulation. Each experiment was performed using 10 UAVs, with results averaged over 100 iterations. Each UAV has 50 tasks that arrive at regular intervals and are sequentially auctioned. The initial locations of the UAVs and the tasks are randomly chosen for each iteration. The results show the average score for each of the *cooperator* agents as the auctions are completed and rewards are assigned.

Bid Participation

In this experiment, a fraction of the agents on the team defect by not participating in auctions (not bidding on others' tasks). Each *defector* agent only participates in auctions 10% of the time. At this level they are occasionally participating but do not contribute effectively. As a result, *Naive* agents that trust unconditionally (using no trust mechanism) end up doing additional work for the *defector* agents and receive little in return. The objective for the *cooperator* agents is to detect those team members that regularly fail to

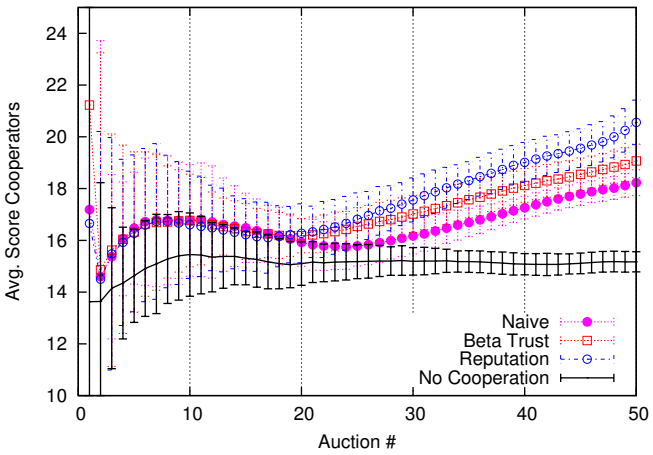


Figure 50: Bid Participation: Agents that defect by not bidding can be detected and isolated using observation based trust mechanisms. The average unit score of the cooperative agents is plotted against the number of auctions completed for the different trust strategies. The error bars reflect one standard deviation.

participate in auctions and to isolate them from future cooperation by not bidding on the *defectors'* tasks.

For each auction, the trust strategies update the trust model for each agent that was sent an auction announcement. If the agent submitted a bid, the trust model is updated, ($Trust_{a,t} = 1$), and ($Trust_{a,t} = 0$) otherwise. Once an agent is no longer trusted, with high confidence, they are removed from future auction participation as the *cooperators* isolate them by refusing to bid on their tasks. The results of this experiment, shown in Figure 50, reflect that the agents running the *Beta Trust* and *Reputation* strategies receive better scores than those that apply the *Naive* strategy or the *No Cooperation* strategy, once the agents are able to observe which team members are participating in the auctions. In addition, the *Reputation* strategy which shares trust information across team members performs better than the *Beta Trust* strategy which relies on direct observations alone.

Bid Veracity

In a similar experiment, shown in Figure 51, the *defectors* submit untruthful bids. The *defectors* show participation by responding with bids to auction announcements, but bid arbitrarily low values to avoid being assigned others' tasks. Again, the *cooperator* agents can end up performing additional work for the *defectors*. The *cooperator* agents seek to detect

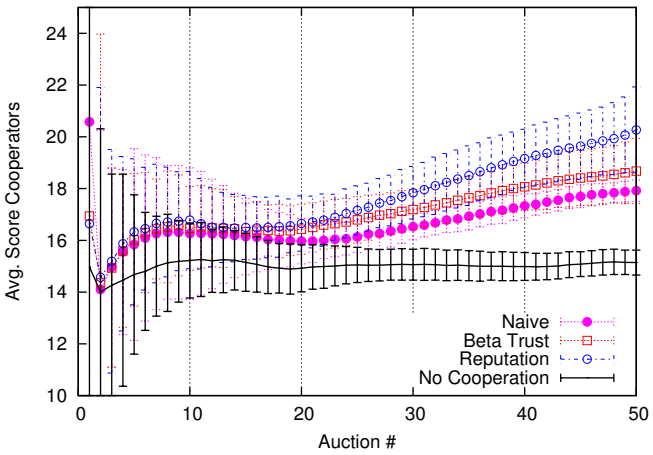


Figure 51: Bid Veracity: Agents that defect by sending untruthfully low bids can be detected and isolated using observation based trust mechanisms. The average unit score of the cooperative agents is plotted against the number of auctions completed for the different trust strategies.

the *defectors* by comparing all submitted bids against the winning bid for each auction. Here, if an agent was sent an auction announcement, and did not bid on the auction, or submitted a bid that was lower than a threshold percentage, θ, of the winning bid, the trust model is updated with the failure to cooperate ($Trust_{a,t} = 0$), and ($Trust_{a,t} = 1$) otherwise. Again, the use of trust with the *Beta Trust* and *Reputation* strategies results in better scores than the *Naive* and *No Cooperation* approach, once the agents learn which team members cooperate.

Task Completion

This set of experiments simulates the case in which task completion is the trust dimension, as described in section 7.2.1. When a *defector* agent does not complete the tasks assigned to it by a *cooperator*, the *cooperator*, loses the benefit of having that task performed. Therefore, it benefits the *cooperator* to identify those agents that do not complete the assigned tasks and remove them from consideration in future assignments. When an agent assigns a task, the agent can select whether that task should be monitored. However, the monitoring incurs a fixed cost. Here, we revisit the cost of monitoring experiment from Section 5.3. When the cost of monitoring is high, it may not be worthwhile for an agent to monitor each task. At each time step, the agent can query the monitor to determine the status of the task, which is either *completed*, *failed* or *pending*. When a task is *completed* or *failed*, the trust

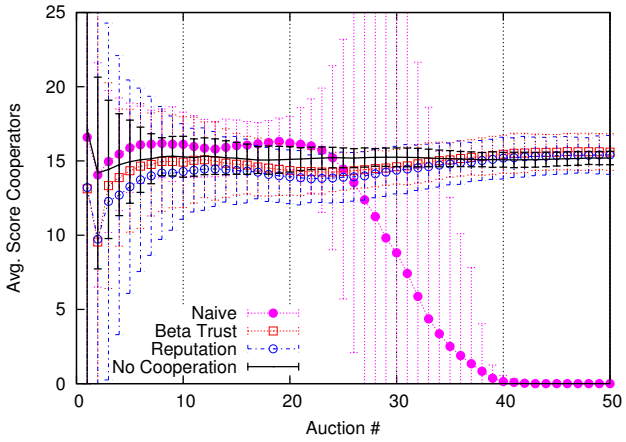
model for the assignee is updated ($Trust_{a,t} = 1$) and ($Trust_{a,t} = 0$), respectively. When an agent's trust value falls below 0.5, with confidence above 0.3, the agent is removed from the team and further cooperation.

In the first experiment, the *defector* agents are occasionally successful in completing tasks assigned to them, but fail to complete their tasks most of the time. Defection is simulated by drawing from a normal distribution (with $\mu = 0.4, \sigma = 0.15$) and if the value is ≤ 0.5, the defector fails to perform the task. Additionally, the monitoring costs are high, equaling 4% of the possible reward. *Cooperators* that trust unconditionally, as shown in figure 52(a), perform worse in the later auction periods because a number of their assigned tasks get dropped and they do not receive the rewards associated with those tasks. The *cooperators* that use trust and reputation mechanisms perform better, but because of the high monitoring costs, the average scores are similar to the scores obtained through no cooperation.

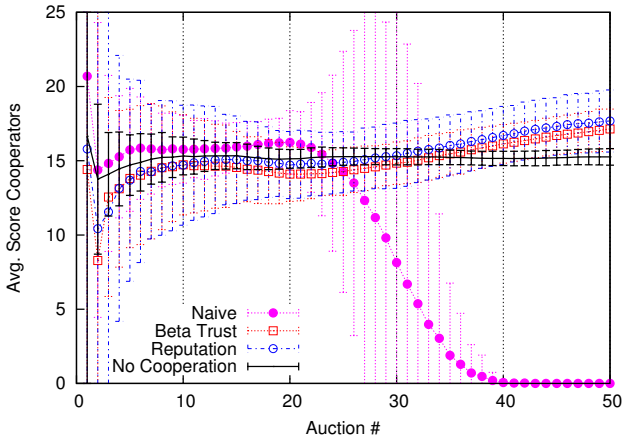
In the next experiment, shown in Figure 52(b), *cooperators* selectively monitor other agents when assigning tasks, based on how much the assignee is trusted. For instance, if an agent is highly trusted, then it will be monitored with low probability and vice-versa. As such, when a task is assigned, the probability that it will be monitored, given the assignee's trust value, x, is $P(monitor) = -x + 1$. This results in the monitoring costs being applied more to those agents that are untrusted. The agents that use the *Beta Trust* and *Reputation* methods with probabilistic monitoring learn which team members are *defectors* and isolate them from future cooperation. This allows them to perform as well as the *Naive* and *No Cooperation* strategies in the beginning, but to also achieve higher scores near the end of the auction periods as the defectors are isolated.

7.2.4 Summary

The above experiments showed that trust and reputation mechanisms can be effective for detecting and isolating uncooperative and non-performing team members in auctions, when compared to the naive approach. This may prove useful in situations in which auction based teams are dynamically formed and not all team members are likely to participate



(a) Task Performance with Full Monitoring



(b) Task Performance with Probabilistic Monitoring

Figure 52: Task performance can be monitored, and agents that regularly fail to complete tasks can be removed from the team. (a)When monitoring costs are high, it may be better for agents to not cooperate at all. (b) Probabilistic monitoring allows for monitoring resources to be focused on those agents likely to defect and the trust strategies perform better than the no cooperation strategy.

equally. However, combining the trust dimensions into a single rule might be more effective in practice, as the bid participation trust dimension alone is a weak metric. The results show that by incorporating the use of trust strategies into the basic auction mechanism, agents can perform better than agents that trust unconditionally.

In addition the use of monitoring task completion to evaluate peer performance was presented. This highlights the usefulness of monitoring in situations where trust models are employed. However, for the use of monitoring to be effective, the cost of monitoring must be less than the additional benefits gained through cooperation.

Auction based methods are often used to perform distributed task allocation on multi-agent teams. Many existing approaches to auctions assume fully cooperative team members. On in-situ and dynamically formed teams, reciprocal collaboration may not always be a valid assumption.

This section presents an approach for dynamically selecting auction partners based on observed team member performance and shared reputation. In addition, we present the use of a shared reputation authority mechanism. Finally, experiments are performed in simulation on multiple UAV platforms to highlight situations in which it is better to enforce cooperation in auctions using this approach.

7.3 Cooperation based Dynamic Team Formation in Multi-Robot Auctions

Auction based methods are often used to perform distributed task allocation on multi-agent teams. Many existing approaches to auctions assume fully cooperative team members, and team members may have cooperation explicitly built in. However, on in-situ and dynamically formed teams, reciprocal collaboration may not always be a valid assumption.

The basic auction approaches to the task allocation problem assume that team members can be trusted and have the goal of the team in mind (to reduce the overall cost) [70]. These algorithms serve as a mechanism for distributed task allocation and generally do not need to consider team members' cooperation levels or performance characteristics. As such, these methods do not explicitly account for trust between team members, but assume that *a)* team members will participate in auctions that are presented to them and *b)* team members will attempt to perform tasks that are assigned to them. However, there are situations in which teams may be formed dynamically. While the team members may have the same common goal, the individuals may have different levels of interest in the cooperation. That is, some of the team members may place a higher utility on successful completion of tasks, while others are obligated to participate, but wish to conserve resources.

This section presents an approach for dynamically forming auction partners based on observed team member performance and shared reputation information. In addition, we present the use of a trust and reputation mechanism in a practical setting. Each team

member models the other individuals and these models are updated through repeated interactions. Agents can use the model to detect team members that are not contributing, and those team members can be removed from future collaboration, thereby losing the benefits of cooperation. Finally, experiments are performed in simulation on a UAV platform using this approach.[2]

7.3.1 Approach

The approach in this section is to use observation based trust for determining when to remove a non-cooperative team member from an auction team by ignoring its auction requests. If an agent is no longer on the team, it loses opportunities for others to assist it with tasks when those tasks could be done more efficiently as part of a team than alone. In the auction context, agents that do not bid on each other's tasks or complete them successfully can be viewed as uncooperative and removed from a team. We call this types of agent a *freeloader*. From an agent's viewpoint, it is better to have team members that cooperate and participate in the auction algorithm as this leads to more efficient outcomes. From a global viewpoint, it is desirable to have an efficient team that is composed of cooperative members; each freeloader decreases the overall team performance. Finally, in this work, we assume that a currency exchange mechanism is not available for enforcing cooperation. In teams that are dynamically formed or consist of temporary alliances, it is reasonable to assume that such an exchange and accounting mechanism may not be present.

In a dynamically formed auction team, agents may encounter other agents for which they have no prior experience. The use of a trust model would allow for an agent to reason about other agent's trustworthiness using observation histories and reputation information. In these settings, there are multiple dimensions that could be used to define trust, such as whether an agent participates in the auctions of others and whether an agent successfully completes tasks that are assigned to it. Agents that regularly violate the trust dimensions are considered to be *defectors*, while agents that cooperate fully are labeled as *cooperators*. These dimensions of trust can be considered separately or in combination. Each agent can

[2]Experiments in this section appeared in [118].

build models of other team members behaviors from observation histories and use those models to determine levels of trust.

Bid Participation

The *Bid Participation* dimension considers whether an agent is participating by submitting bids as part of the auction process, but does not evaluate the bid. In fact, there are legitimate situations in which an agent might not wish to submit a bid, if the calculation is costly [25]. However, in domains in which the bid calculation is easily computed and communicated, this can be a useful gauge of auction participation. When an agent announces an auction, it keeps track of the agents that received the auction announcement and compares this to the list of agents that submitted bids. This approach assumes that agents are uniquely identifiable and that a protocol exists for acknowledging the receipt of an announcement. When an agent does not bid on received auctions announcements, this negatively updates the trust model, while the submission of a bid positively updates the model (described further in section 6.1.1.) Using the model, if an agent determines that a team member is not trusted, the agent refuses to bid on the untrusted team member's future auctions, effectively isolating it from the auction team. In this section, we primarily discuss the bid participation dimension; however, the trust model presented in the next section could be used to combine additional dimensions into a single trust valuation.

Auction Approach using the Trust Model

In the basic multi-agent auction algorithm, the problem is to assign tasks to agents. The tasks in this case are to visit a target location and perform an observation. In the auction framework, each robot is a *bidder* and the items to be auctioned are the 'visits'. Each of the agents in the system also participates as an *auctioneer* and periodically auctions new task requests (it is assumed that the task requests are provided to the agent by an external process, such as a human operator or other event). This approach can easily be used on teams with different robot characteristics: each robot knows their own location and cost function and submits cost based bids to the auctioneer. While costs and rewards use the same basis for calculation, no revenue is actually exchanged. Rather, an agent awards itself

a utility value when one of its own tasks is completed.

The *auctioneer* first handles any auctions that have already been announced and are ready to close. This step is shown in detail in Procedure 9. In lines 1-3, the *auctioneer* selects the minimal cost bid from all bids received by the agents within communications range (including their own) as the winner of that auction and performs the task assignment by announcing the winning bidder. In lines 5 and 7, the *auctioneer* updates the trust model (described in Section 6.1.1) for each possible *bidder* that was sent the auction announcement. The trust model is referenced by the *bidder* in Procedure 10, when an auction announcement is received. If the originator of the auction announcement is not trusted, then the auction announcement is ignored, effectively isolating the untrusted agent from the benefits of cooperation.

Input: An auction, a.
Input: The set of posted bids, B_a.
Input: The set of announcement recipients, $Recipients_a$.

1: $winner \leftarrow Min(B_a)$
2: $AnnounceWinner(winner, a)$
3: **for all** $a : Recipients_a$ **do**
4: **if** $a \in B_a$ **then**
5: $UpdateParticipation(TRUST_o, 1)$
6: **else**
7: $UpdateParticipation(TRUST_o, 0)$
8: **end if**
9: **end for**

Algorithm 9: $Auctioneer :: HandleAuctionBids$

Input: An set of announced auction tasks, A.
Input: The auction originator trust model, $TRUST_o$.

1: **for all** $a : A$ **do**
2: **if** $CanTrust(TRUST_o)$ **then**
3: $bid \leftarrow CalculateBid(a)$
4: **if** $bid > 0$ **then**
5: $PostBid(bid)$
6: **end if**
7: **end if**
8: **end for**

Algorithm 10: $Bidder :: HandleAnnouncements(A)$

7.3.2 Experimental Results

Multi-Agent Experiments

A set of experiments were performed to test the trust strategies in simulated auctions using the Mason simulated multi-agent environment [80], and described further in Section A.1. The Mason environment was used to a run large number of low-fidelity simulations to demonstrate the trust model. In these experiments, the agents are represented as Unmanned Aerial Vehicles (UAVs), modeled as points in a 2d plane, and each UAV is assigned tasks to perform by an external process. Each UAV has an auctioneer and can auction their tasks to other agents, assigning the task to the agent that submits the minimal cost bid. The UAVs in the simulation have a limited communications range and can therefore only perform auctions or exchange reputation information with a subset of the other team members at a given time.

In addition, each UAV periodically re-auctions the last n tasks to other agents in range. This allows tasks to be more optimally assigned by giving other agents a chance to bid on them if they were not in range during the initial auction. Each experiment was performed using 10 UAVs, with results averaged over 100 iterations. Each UAV has 50 tasks that arrive at regular intervals and are sequentially auctioned. The initial locations of the UAVs and the tasks are randomly chosen for each iteration. The results show the average score for each of the *cooperator* agents as the auctions are completed and rewards are assigned.

In this experiment, a fraction of the agents on the team defect by not participating in auctions (not bidding on others' tasks). Each *defector* agent only participates in auctions 10% of the time. At this level they are occasionally participating but do not contribute effectively. As a result, *Naive* agents that trust unconditionally (using no trust mechanism) end up doing additional work for the *defector* agents and receive little in return. The objective for the *cooperator* agents is to detect those team members that regularly fail to participate in auctions and to isolate them from future cooperation by not bidding on the *defectors'* tasks.

For each auction, the trust strategies update the trust model for each agent that was sent an auction announcement. If the agent submitted a bid, the trust model is updated,

($Trust_{a,t} = 1$), and ($Trust_{a,t} = 0$) otherwise. Once an agent is no longer trusted, with high confidence, they are removed from future auction participation as the *cooperators* isolate them by refusing to bid on their tasks. The results of this experiment, shown in Figure 50, reflect that the agents running the *Beta Trust* and *Reputation* strategies receive better scores than those that apply the *Naive* strategy or the *No Cooperation* strategy, once the agents are able to observe which team members are participating in the auctions. In addition, the *Reputation* strategy which shares trust information across team members performs better than the *Beta Trust* strategy which relies on direct observations alone.

UAV Platform Simulation Experiments

The UAV platform experiments were performed in high-fidelity simulations, using the software in the loop capabilities of the autopilot. The simulation setup and UAV platform are described further in Section A.3.

Again in this set of experiments, a fraction of the UAVs do not participate in group auctions, but exploit the others on the team by allowing them to perform tasks on behalf of the *defectors*. In this case, we simulated 4 UAVs flying in different sectors of the environment and awaiting tasks from their operators. To simulate operators, a separate process regularly assigns a new task to a UAV that is picked at random. The location of the task varies in the environment, over an area of 15x10 km. The simulation ends after 100 tasks have been assigned and completed. When a UAV is assigned a task by their operator, it has the responsibility of completing it. However, a UAV has the option of auctioning the task to another member on the team. No currency is exchanged in this domain, but rather the vehicles consist of a loosely formed team that can benefit from cooperation because the tasks are distributed throughout the environment.

In one set of experiments, the cooperative team members perform a basic or *naive* auction strategy, and do not consider whether other UAVs have reciprocated cooperation. In the second set, the cooperative team members apply the direct observation based trust mechanism that was described in Section 6.1 to detect team members that do not participate in auctions by bidding on other's tasks. The results are shown in Table 11. The

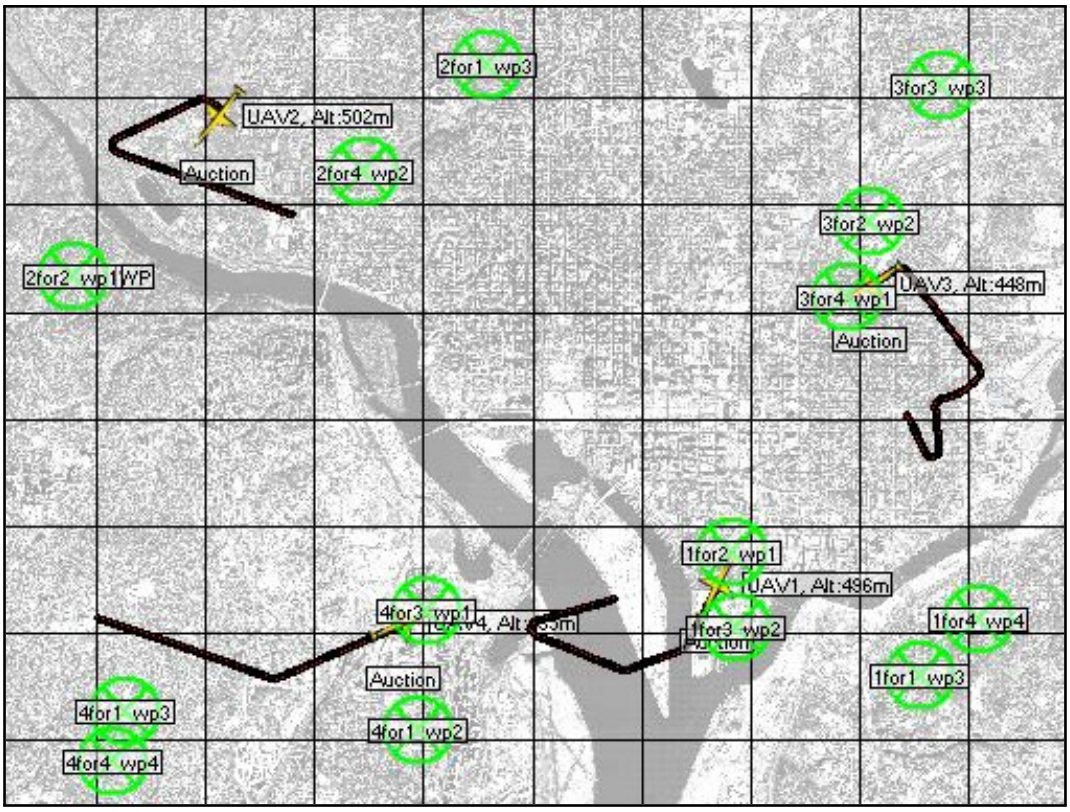


Figure 53: Simulating multi-UAV auctions: Multiple UAVs are simulated using the autopilot and autonomous auction behaviors. The UAVs and assigned waypoints are shown in FalconView^{TM}map display.

average task cost represents the amount of time (in seconds) that was taken to complete the task by the UAV that won the auction.[3] The average time spent represents the amount of time spent performing assigned and won tasks. When the *naive* auction strategy is used, the cooperators are exploited by the defectors and spend a much greater amount of time performing tasks. The tasks are performed less efficiently as a result, because the cooperators are doing most of the work. In contrast, when the trust model is employed, the cooperators quickly learn to not trust the defectors, and isolate them from further participation by refusing to bid on the untrusted team members' auctions. This results in a lower global cost, and the cooperators have a much lower cost and time spent as well, because they are not being exploited in this case.

[3]A task could be completed by a different UAV than the one it was initially assigned to, if it was re-assigned as part of an auction.

Table 11: Auction Participation with the Trust Model

		Global	Cooperators	Defectors
Naive Auction	Avg. Task Cost	1282	1172	1383
	Avg. Time spent	1282	1306	79
Beta Trust Model	Avg. Task Cost	1138	850	1415
	Avg. Time spent	1138	850	1415

7.3.3 Summary

The above experiments showed that trust and reputation mechanisms can be effective for detecting and isolating uncooperative team members in auctions, when compared to the naive approach. This may prove useful in situations in which auction based teams are dynamically formed and not all team members are likely to participate equally.

Traditional auction algorithms for performing the robot task assignment problem assume that robots are equally incentivized to participate in auctions. However, there are situations in which agents may assign tasks to others on the team, without taking on a fair number of additional tasks in return. This section presents an approach for using observation based trust and a shared reputation mechanism in determining which agents to include in multi-agent auctions. The results show that by incorporating the use of trust strategies into the basic auction mechanism, agents can perform better than agents that trust unconditionally. There are legitimate situations in which a team member may not be able to participate in auctions, such as when the agent is not capable of performing the task or is otherwise preoccupied. However, the trust model presented is able to tolerate noise in observations and also can incorporate forgiveness when an agent is able to participate again. Rather, the focus is on detecting those team members that exploit the team and isolate them from the benefits of future cooperation. In the next section, rather than explicitly isolate team members, robots will use the trust model to determine which team members are not able to reliably complete their patrolling tasks and use that information to decide when to assist a neighbor.

7.4 Applying Trust in Multi-Robot Patrolling

The multi-robot patrolling task is an example of a domain that is particularly sensitive to reliability and performance of robots. For instance, a robot's performance may deteriorate over time or a robot may not estimate tasks correctly. Robots that can identify poorly performing team members as performance deteriorates, can dynamically adjust the task assignment strategy. This section investigates the use of trust based approach for determining when to decide which team members are no longer effective and to perform task re-assignment. Experiments are performed using a team of eight indoor robots in a patrolling task to demonstrate both centralized and decentralized approaches to task reassignment, and results demonstrate that approaches that include trust based performance monitoring perform better than those that do not consider performance dimensions.

Robots can use observations of team member performance to build models of how well others can be trusted to perform various tasks. These observations could occur during online training or learning and through real world exploration. The multi-robot patrolling domain in particular can have specific desiderata for reliability and security. Therefore it is important for a team of robots to be able to dynamically adjust to the performance of individual team members to maintain the expected operational capabilities. The patrol task is described in more detail, in Section 4.3.1. Fully autonomous robot teams will require the ability to evaluate performance of team members for multiple reasons: human operators may not be able to manage large teams of robots in dynamic environments, robot teams may form in an ad-hoc fashion, and the performance metrics may not always be human observable.

The performance criterion considered in this section is again the *refresh time*, which is the time gap between any two visits to the same location. The maximum refresh time reflects the bounds on the effectiveness of a robot team in detecting events in the environment [112]. If a robot fails to perform its assigned tasks or visits locations too infrequently, this will affect the performance of the team. To mitigate the performance issues with robots that are determined to not be performing well, other robots may have to take on some of their patrol locations, thereby decreasing the maximum refresh time. This section presents an approach

for applying a trust model to the observation of robots in the patrolling task. This approach can be used to more effectively perform patrol task allocation by reassigning those tasks with the greatest refresh time to other team members using a bidding mechanism between the better performing team members.

7.4.1 Approach

In these experiments, we consider an environment with clusters of nodes (rooms) separated by long edges (hallways), such that it is better to partition the environment by placing a robot in each room, rather than having them patrol in an evenly spaced cyclic route. The patrol environment for the first set of experiments is shown in Figure 54. The properties of this environment are discussed further in Section B.2. Graph partition approaches divide the graph into subsets of nodes and assign these nodes to individual robots on the team. Pasqualetti, et al. present optimality bounds for three major types of partition based patrol graphs: cycles, trees, and chains, and remark that the selection of the roadmap may not be unique for an environment and that the performance can vary based on the choice of the graph structure [112]. For the purposes of this section, we convert a cyclic roadmap of the environment into a chain partition, using the approximation algorithm described in [111].

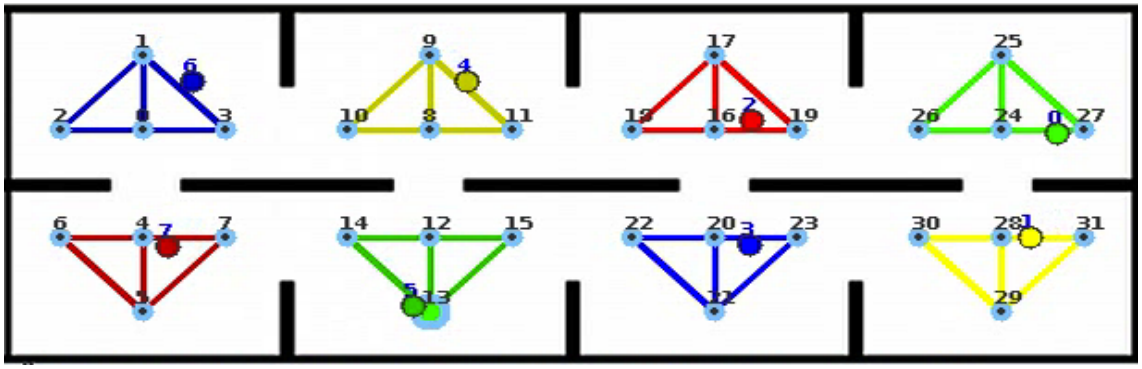


Figure 54: The museum patrol environment, with the patrol graph partitioned optimally for 8 robots.

Performance Monitor

In this work, we assume that an external monitor is available to observe robot performance. In practice, we allow for each robot to broadcast results messages when a node is visited,

and assume that these are reported truthfully and also that the network reliably delivers these messages.

Each robot in the patrol graph shown in Figure 54 can have 2-3 neighbors. We define the set of neighbors for a robot, r, to be N_r and the set of immediate neighbor nodes to be V_n^r. Regarding the features that describe the performance of the robot, the trust model described below can incorporate multiple trust dimensions, using a mixture of weights for each. Performance dimensions that may be considered as input to the trust model include those related to *sensors* (probability of detection, tracking accuracy), *actions* (execution time, distance, fuel consumed, trajectory accuracy) and *deliberation* (explicit cooperation, correctness of plans generated, appropriate behavior selection, etc.).

In this section, we adopt the performance metric of *maximum refresh time*. That is, the goal of the system is to minimize the maximum refresh time for all nodes in the multi-robot patrol. When the refresh time of any robot's assigned nodes exceeds a threshold on this metric, we seek to re-assign some of that poorly performing team member's nodes to others. Each robot self reports node visits to the monitor which tracks the idle time for each node. At each time step, the monitor can calculate the node with the maximum refresh time for each robot. We set the amount of time in between performance monitoring periods to be the expected maximum refresh time for the patrol partition.

The max refresh time for a robot is the maximum refresh time for all nodes assigned to robot r. Let I_k^r be the set of the refresh times at the previous k node visits for a robot, r. Let I_n^r denote the refresh time of a node visited by robot r and being the nth visit by r to any node assigned to it. The running max refresh time, $M_k^r = \max(I_{k..n}^r)$, is the observed maximum refresh time for a robot over the window $(n, n-1, \ldots, n-k)$, where $n > k > 0$. The leave-one-out running max refresh average is the average running max refresh time over all other trusted robots, $\overline{M}_k^{T-r}$. The threshold for the max refresh time, θ_{M_r}, is defined as the leave-one-out running max refresh average, plus ρ standard deviations (here, $\rho = 3$).

$$\theta_{M_r} = \overline{M}_k^{T-r} + \rho * \sigma; \tag{20}$$

We define a patrol period as the expected amount of time to perform a patrol of the

maximum partition plus a constant factor. This factor is included to capture the additional time needed to navigate due to the non-holonomic motion of the robot and related to time spent navigating around obstacles. At the end of each patrol period, the monitor checks whether $M_k^r > \theta_{maxidle}$ for each robot. In that case, a robot is considered to be performing poorly and the trust model is updated with a negative observation.

In addition to the max refresh time, we can consider the metric of average refresh time. The average node refresh time for a robot, A_k^r, is the average refresh time per node for a robot in the last patrol period. The threshold for the average refresh time, θ_{A_r}, is defined as the average node refresh time for all trusted robots, plus ρ standard deviations.

$$\theta_{A_r} = \overline{A}_k^T + \rho * \sigma; \tag{21}$$

7.4.2 Task Reassignment Methods

We considered two types of task reassignment methods, for the centralized and localized task reassignment case. These approaches are described further below.

Central Observation and Assignment

In this approach, the centralized monitor records the node visit frequency for each robot and updates the trust model with positive and negative performance observations when a robot is within or exceeds the performance thresholds, respectively. This algorithm is shown in Figure 55.

When a robot's max refresh time is observed to exceed the threshold, and this is not due to the robot assisting others, then the trust model is updated with a negative instance. In the other case, we do not automatically update the trust model with a positive result because the max refresh time could be low for a *poor performer*, if other robots have come to assist it, but the average for the remaining nodes could still be high. In this case, we also consider the average refresh time metric, and if it is below the threshold, the trust model is then updated with a positive signal.

If a robot moves from the trusted set to the untrusted set, $T \rightarrow U$, the central monitor reassigns one of the nodes from the untrusted robot to a neighboring robot in T. Similarly,

```
1: loop
2:   Do Every P Seconds:
3:   for all r : Robots do
4:     if (M_k^r > θ_{M_r}) and (r ∉ Assistors) then
5:       UpdateTrustModel(r, β);
6:     else
7:       if A_k^r > θ_{A_r} then
8:         UpdateTrustModel(r, α);
9:       end if
10:    end if
11:  end for
12:  for all r : Robots do
13:    if r ∈ T → r ∈ U then
14:      g ← select ∈ T ∩ N_r;
15:      SendReassignTaskMessage(r, g);
16:    end if
17:    if r ∈ U → r ∈ T then
18:      SendReturnAllTasksMessage(r);
19:    end if
20:  end for
21: end loop
```

Figure 55: The Central Observation and Assignment pseudocode: the central monitor observes node visits, updates the trust model and reassign tasks when a robot becomes untrusted.

if a robot moves from $U \rightarrow T$, the central monitor returns all of its original tasks. Once a robot assists another robot by taking a new node, it is added to the set of *Assistors*, so that it will not have its trust score penalized for the resulting increased refresh time.

Local Observation and Assignment

In this approach, each robot locally cooperates, but without coordination. This algorithm is shown in Figure 56. Here, each robot locally reports the performance observed for each of their neighbors, by observing the visit frequencies of the nodes in neighboring partitions, and sending positive and negative performance observations to the central trust authority when the average running refresh time for a node, A_k^v, exceeds the expected max cycle time. The expected max cycle time is defined as the expected time for a *good performer* to complete a full cycle, times a factor to allow for a small amount of motion error. In addition, robots periodically query the central trust authority to get the trust score for their neighbors. From all of the untrusted neighboring robots, the robot will select the

```
1: loop
2:   Do Every P Seconds:
3:   for all v : V_n^r do
4:     if (A_k^v > E(MaxCycle)) then
5:       UpdateTrustModel(getNodeOwner(v), β);
6:     else
7:       UpdateTrustModel(getNodeOwner(v), α)
8:     end if
9:   end for
10:  u ← MostUntrusted(U ∩ N_r);
11:  if (u) then
12:    SendReassignTaskMessage(u);
13:    SendAssistingMessage(u);
14:  end if
15: end loop
```

Figure 56: The Local Observation and Assignment pseudocode: the local monitor on each robot observes neighboring node refresh times, updates the central trust model and reassign neighbor nodes to itself when a neighboring robot becomes untrusted.

most untrusted neighbor. If one is found, the robot will add the closest neighboring node from the most *untrusted* neighbor to its own patrol list, and send the task reassign message to the *untrusted* neighbor for that node. To prevent an assisting robot from itself becoming untrusted, a robot sends a *assisting neighbor* message to the trust authority. Upon receipt of this message, the trust authority enters an annotation to the trust record for that robot which it uses to allow for decreased performance in the assisting robot.

7.4.3 Experimental Results

Robot Platform

A set of experiments was performed using the TurtleBot indoor mobile robot platform, which is described in the Appendix section A.2. Each robot runs a custom *Patrol* behavior which implements the graph chain partition algorithm, and repeatedly navigates to the nodes in the robot's patrol path. The experimental setup also includes a central monitor node which listens for task completion messages and includes the performance monitoring and task reassignment components. Robots communicated with the central monitor by sending messages using UDP broadcast over the local wireless network.

In each experiment, one of the robots is explicitly marked as a *poor performer*. The

performance for this type of robot is affected by randomly adjusting the maximum forward velocity of the robot after each visit to a patrol node. The robots that perform normally have a maximum speed of 0.25ms^{-1} and the maximum speed of the *poor performer* is determined by sampling from a normal distribution with $\mu = 0.15\text{ms}^{-1}$ and $\sigma = 0.10\text{ms}^{-1}$. This results in increased max refresh times for the patrol nodes assigned to the *poor performer*.

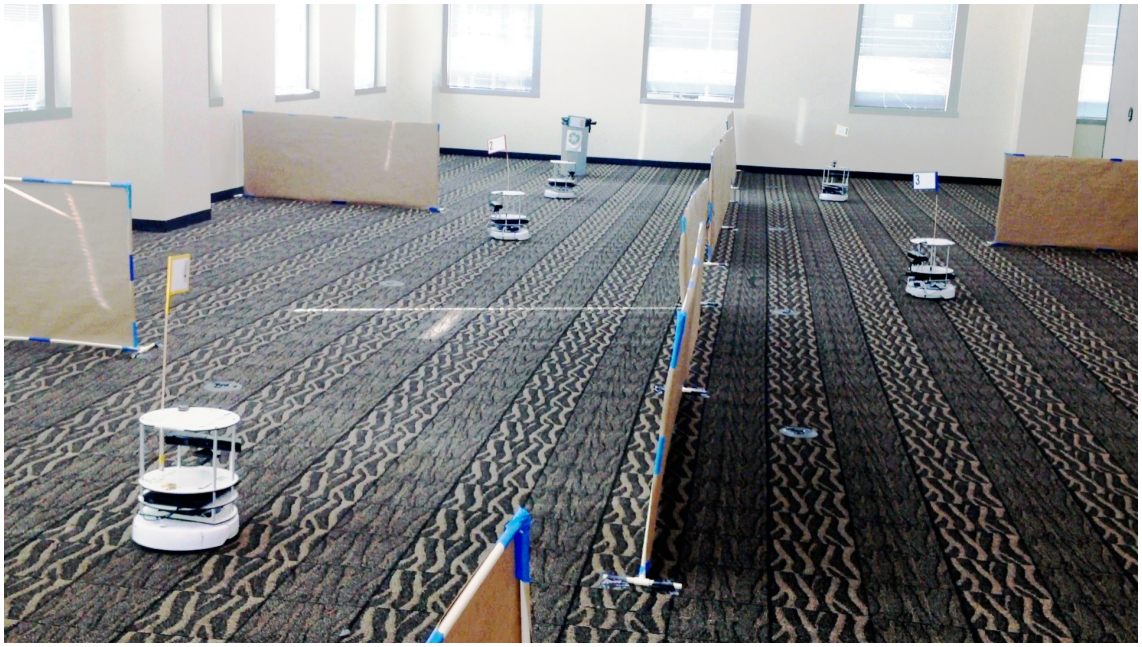


Figure 57: Multiple TurtleBots are shown patrolling in the experimental environment, setup to resemble an art museum with multiple rooms.

Patrol Graph

Each robot, r, was provided with a copy of the environmental map and patrol graph, shown in Figure 24, as well as the i^{th} graph partition assigned to the r^{th} robot, which also corresponded to a single room. This graph has properties that make it easy to analyze the optimality for performing partitions. In the initial case, when it is assumed that all robots are performing equally well, it is easy to see that the optimal partition for 8 robots is to assign one robot to each room. In this environment, the partition approach results in better performance than the cyclic approach when the edge between the rooms is long. Referring again to Figure in Figure 24, this occurs when the length of the diagonal edge, h, is greater than the corridor distance, l, between the clusters of nodes in each room.

Experimental Setup

The experimental environment was designed to resemble an art museum with 8 equally sized rooms, as shown in Figure 57. The environment was approximately 10m x 30m in size. Upon startup, each robot began patrolling the nodes located in their partition. The setup also consisted of a centralized monitor node that recorded the frequency of visits to each node by robots. Upon completion of each node visit, robots broadcast a node visited message, using a UDP network broadcast, and this was recorded by the monitor. These messages were also available to the robots.

Two different types of experiments were performed, to compare the use of trust monitoring with centralized and local observation and task assignment approaches. Each experiment ran for over 30 minutes, with all 8 robots patrolling continuously during that time.

Results

In the *centralized* task assignment approach, robot 3 was explicitly set as a *poor performer*, after several minutes of normal performance. The central monitor observed each robot's performance and updated the trust model with *positve* or *negative* observations, based on the robot's performance. The trust model scores for each robot in this experiment are shown in Figure 58. It is worth noting that the trust score for robot 3 and also robot 6 dipped briefly during the experiment due to localization errors. However, the model allows for noise tolerance, and the trust scores recovered when the robot's localization recovered.

After the robot 3 was observed performing poorly, its trust score decreased until it reached the low threshold (0.5) for trust and became *untrusted*. At this point, the central monitor dynamically reassigned one of the *poor performer's* tasks to the trusted robot 1, as shown in Figure 59. The robot trajectories during the experiment reflect this task reassignment, with robot 1 picking up a patrol node from robot 3, as shown in Figure 60.

The refresh times for robot 3 and robot 1 are shown in Figure 61. The values for robot 1 are typical for all *good performers*. The refresh time for robot 1 increased after the task reassignment, because of the additional time to cover the reassigned node. However, the max refresh time across the team was improved as a result. We also plot the expected

max refresh time. However, the actual max refresh time for the *good performers* is slightly higher due to the motion model for the robot.

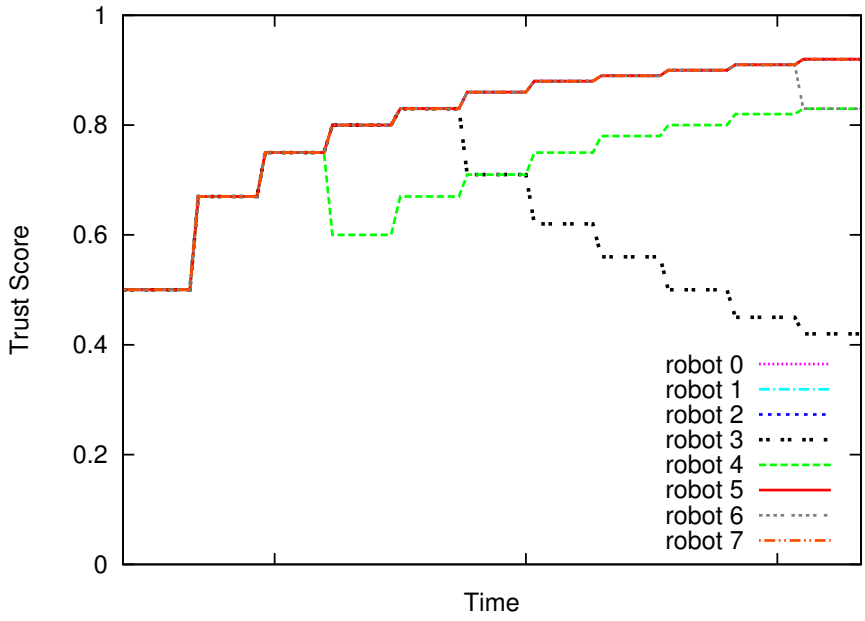


Figure 58: The trust scores for each robot are plotted during the experiment. The trust score for robot 3 decreases as the robot begins to perform poorly. The rest of the robot's trust scores increase monotonically, with the exception of robot 4, whose score briefly dips due to temporary localization errors.

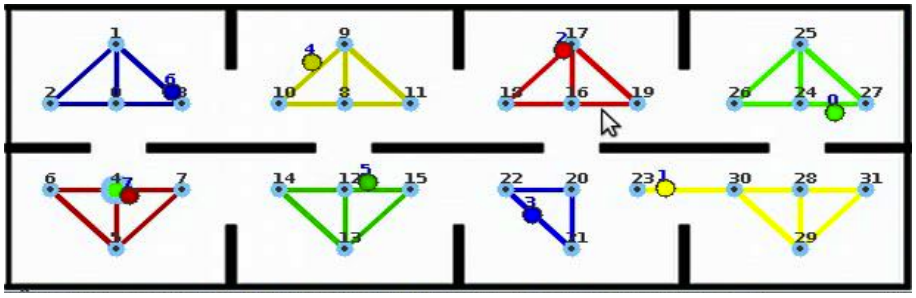


Figure 59: The tasks assignments are shown on the task monitor's display. The centralized approach reassigned a task from poorly performing robot 3 to the trusted, neighboring robot, 1.

In the *local* assignment approach, each robot reported the trust of their neighbors to the central authority, which maintained the trust scores. Here, we again explicitly designated robot 3 as a poor performer, this time from the beginning of the experiment. Over time, this caused its trust score to drop below the threshold. Each of the 3 neighbors to robot 3 observed this and each took over a task, leaving robot 3 with only 1 node to patrol. The neighbors each reassigned a task to themselves and sent a *reassign task* message to robot

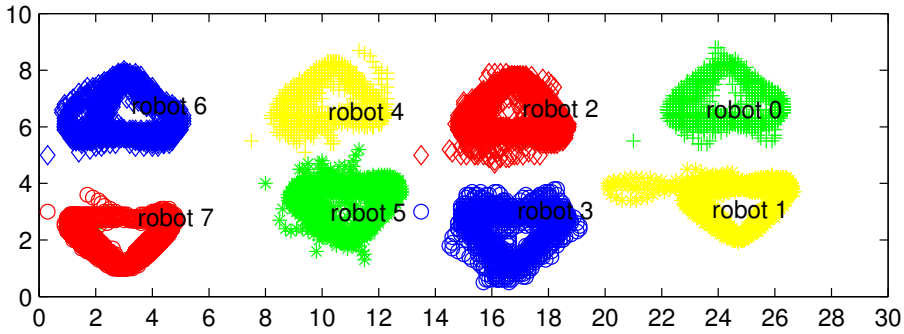


Figure 60: The trajectories of each of the robots are shown for the entire central trust strategy experiment.

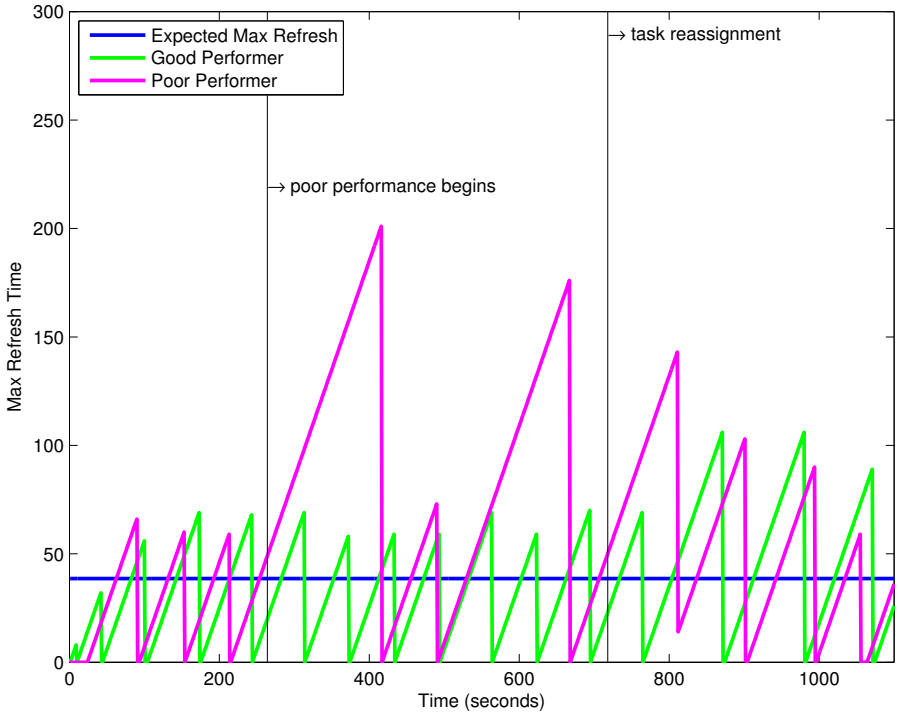


Figure 61: Central Strategy: The refresh time for a *good performer* and a *poor performer* is shown during the period before and after the initial task reassignment. The refresh time increases due to the *poor performer* robot 3, but decreases after a task reassignment. Note that the refresh time during a patrol cycle exceeds the expected max refresh time, due to the non-holonomic motion of the robot.

3. As shown in Figure 62, the experimental monitor updated the display to reflect the task reassignment after receiving these messages, but it is not necessary to the experiment. The robot trajectories during the experiment reflect this task reassignment, with robots 1, 5, and 2 picking up a patrol node from robot 3, as shown in Figure 63. An additional observation was that robot 6 performed poorly, perhaps because there were additional obstacles in its environment and this caused neighboring robots 4 and 7 to come over and assist it as well. A photo from the viewpoint of Robot 1 is shown in Figure 64, reflecting robots 1 and 2 in the partition of robot 3 to pick up tasks.

The refresh times for robot 3 and robot 1 are shown in Figure 65. The refresh time for robot 3 drops to almost zero after it is left with only 1 node to cover. The max refresh time for robot 1 (and the other assisting robots) are similar to those for the previous experiments, in this case multiple robots are performing assistance. A benefit of this approach is that multiple robots can affect the trust score, rather than relying on a centralized observer. A possible extension would be to allow the trust reporting for a robot to be weighted by the trust level of the robot reporting the score.

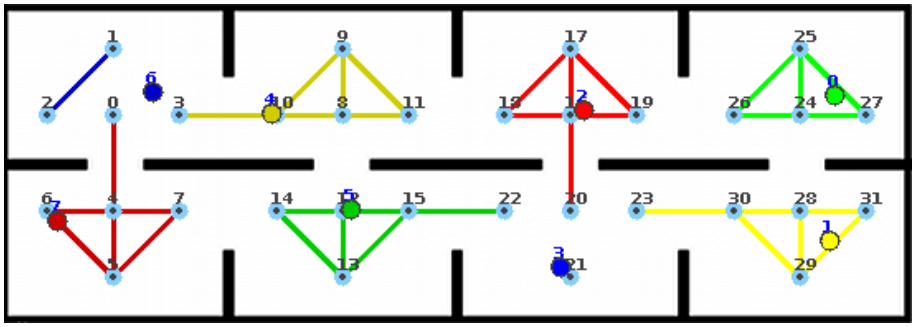


Figure 62: The three neighbors of the poorly performing robot 3 each pick up a task. Robot 6 also performed poorly due to localization errors and also had tasks picked up by its neighbors.

For both the central and local assignment approaches presented here, the use of a trust model allows for more tolerance for noise in the system and for exploitation of performance history. With a threshold only approach, a single noisy observation could cause a robot to become untrusted, resulting in task reassignments. However, the use of a trust model incorporates multiple observations and in the local case, can incorporate observations from multiple observers.

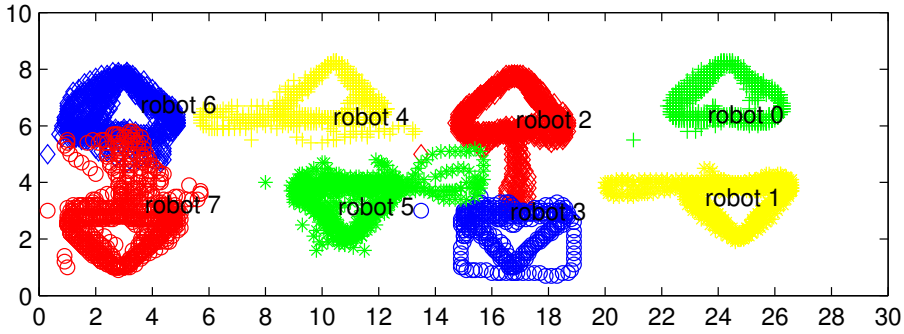


Figure 63: The trajectories of each of the robots are shown for the entire local trust strategy experiment.

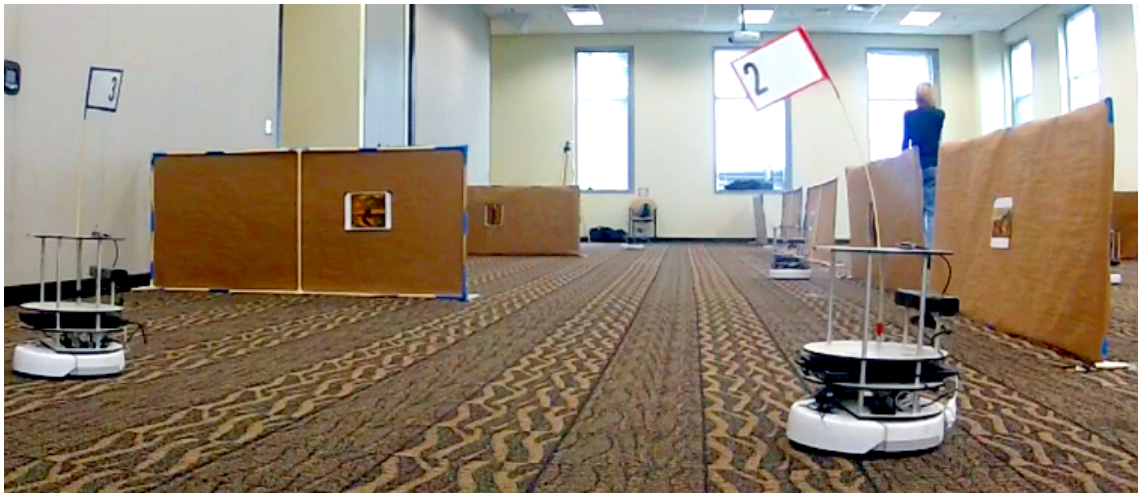


Figure 64: Multiple robots are shown patrolling the environment from the viewpoint of a camera placed on robot 1. In this experiment, after robot 3 is observed performing poorly, its neighbors each pick up one of robot 3's tasks and send it a task reassignment message.

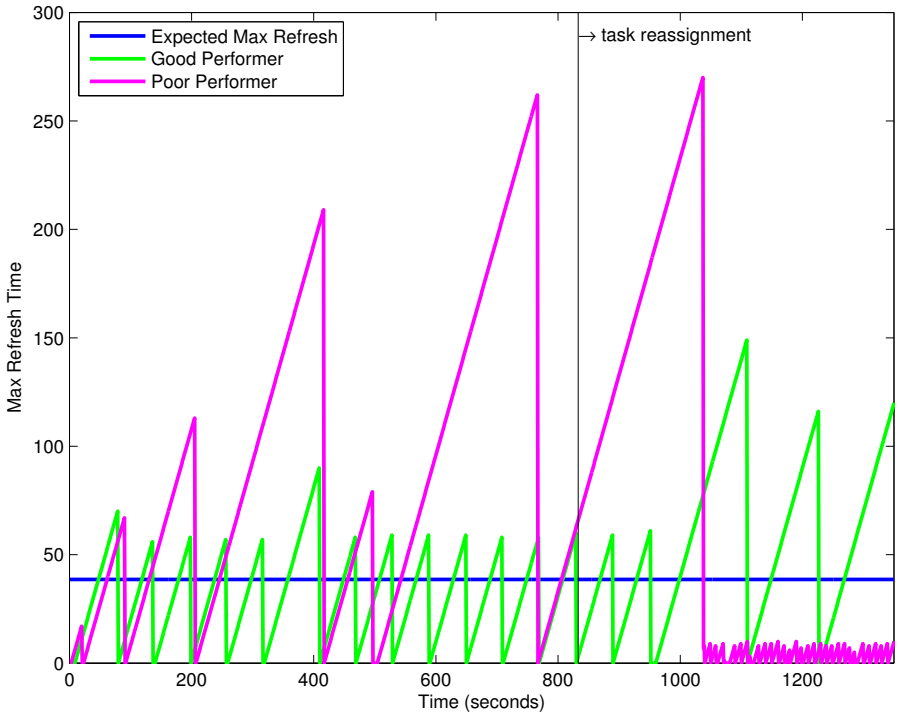


Figure 65: Local Strategy: The refresh time for a *good performer* and a *poor performer* is shown during the period before and after the initial task reassignment. The refresh time increases due to the *poor performer* robot 3, but decreases after a task reassignment.

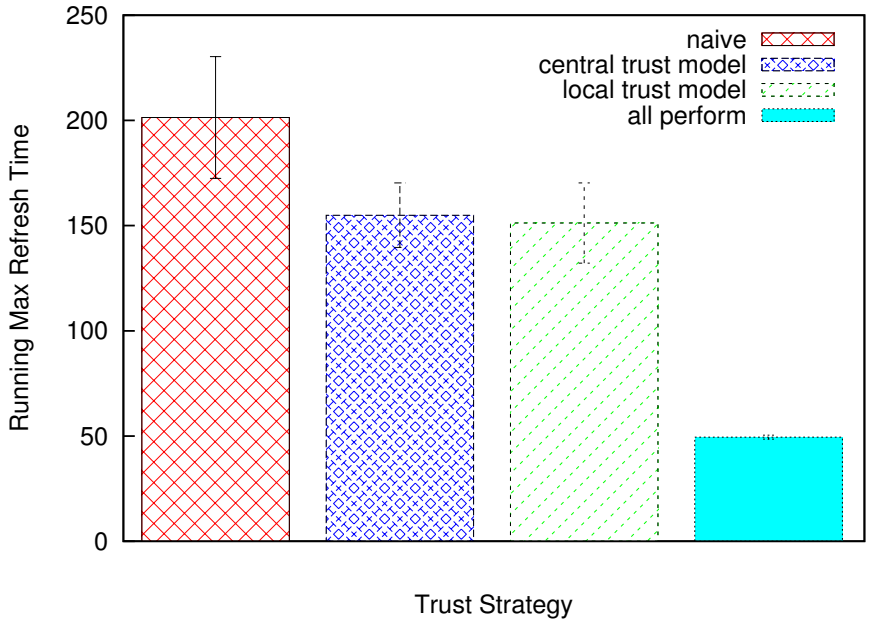


Figure 66: Results are shown from experiments performed in simulation in the museum world environment. Each experiment was run five times, and the error bars represent one standard deviation.

While both approaches result in an improved max refresh time, the central approach can more efficiently allocate robots, but requires a central mechanism which may not always be possible. On the other hand the *local* approach performed a decentralized task allocation with no negotiation between robots or a central node, excluding the *reassign task* message. However, this still required a central trust authority. In practice, the trust authority could also be distributed or available as a subscription service to the robots with periodic updates, but would not require the same communication bandwidth and state updates needed by a centralized optimization.

It is also worth noting that the design of the environment may affect the ability for a robot to assist a teammate, because there is a cost associated with traversing the corridor between patrol partitions. If it is expected that robots will need to frequently assist each other, it may be worthwhile to redesign the placement of the nodes and the size of the team Finally, it might be useful to reallocate all tasks belonging to a poorly performing robot and re-partition the tasks among the remaining $(n-1)$ robots if this would result in better max refresh times. The use of a trust model can be used to handle situations where the observation model can be noisy, because it takes multiple negative observations for a robot to become untrusted. In addition, the model can be initialized with a priori information, from past experiments or domain knowledge.

Experiments in Simulation

In this section, we present the results of experiments performed in simulation in two environments, the museum world and pod world. In each environment, several experiments were performed in each environment using the same ROS behaviors that ran on the robots and the same monitor and reputation authority.

The results from the simulations in the museum world are shown in Figure 66, with the running max refresh time, M_k^R, shown for each of the strategies. The simulation results are consistent with the experiments in the real environment. Both the central trust and local trust strategies resulted in an improved max refresh time over the naive approach, when a poor performer was present. In these experiments, the local trust approach used

a coordination mechanism between neighboring robots to prevent multiple robots from assisting the same poor performer. While the trust model approaches resulted in improved max refresh times, there is still room for improvement. In both trust model cases, task assignment approach only reassigned a single side node to a neighboring robot, reducing the patrol distance from the poor performer by a distance of h, from $2d+2h$ to $2d+h$, while adding a distance of $2l$ to the assisting robot. Assignments from multiple neighboring nodes could reduce the max refresh time further, but with an additional resource expenditure for the assisting robots.

Addition experiment were performed in simulation, using the podworld environment, shown in Figure 67. For ease of analysis in this environment, we set the number of robots to be equal to the number of intersections or *pods* in the environment. This environment was selected because each robot can have up to three neighbors, and this arrangement illustrates neighbor coordination and communication aspects for multi-robot patrolling. The properties of this environment are described further in section B.3. The results from the simulations in the pod world are shown in Figure 68, with the running max refresh time, M_k^R, shown for each of the strategies. The simulation results are consistent with the experiments in the museum world environment. Both the central trust and local trust strategies resulted in an improved max refresh time over the naive approach, when a poor performer was present.

We also performed additional experiments in the pod world environment to further highlight the communication and coordination overhead from using a centralized vs. distributed task allocation mechanism and also a centralized vs. distributed trust authority. The characteristics of these different approaches are summarized in table 12. We assume that the monitoring mechanism is not constrained for any of these approaches, and the robots honestly self report task completion to the trust authority. Each experiment begins with an optimal task allocation as shown in Figure 67.

In the centralized task allocation (CTA) with central trust authority (CA) approach, a central coordination and optimization mechanism monitors the performance of each robot and will optimally reassign tasks to a single neighbor of a poorly performing robot. This

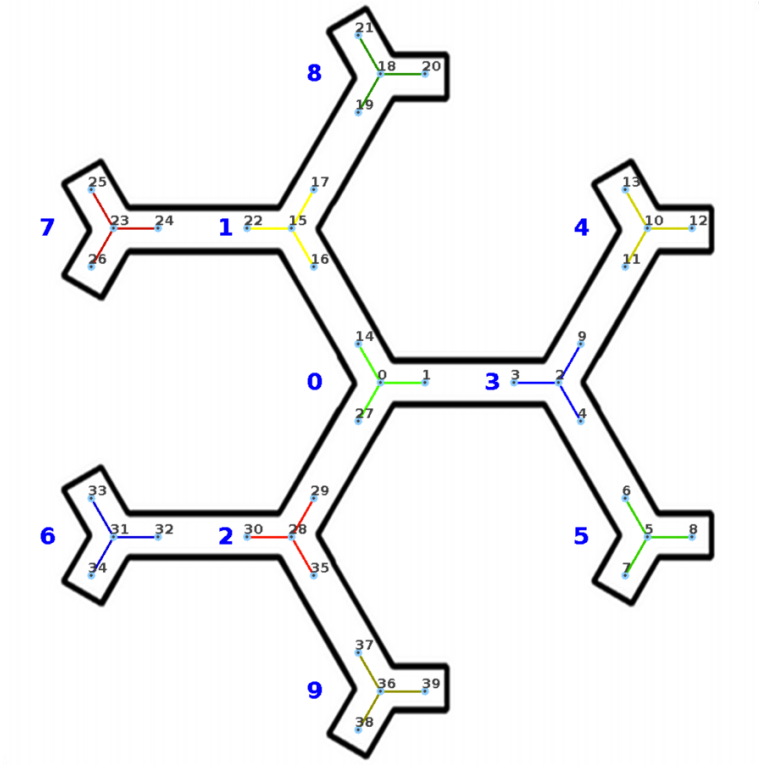


Figure 67: The optimal initial partition of robots in the pod world environment.

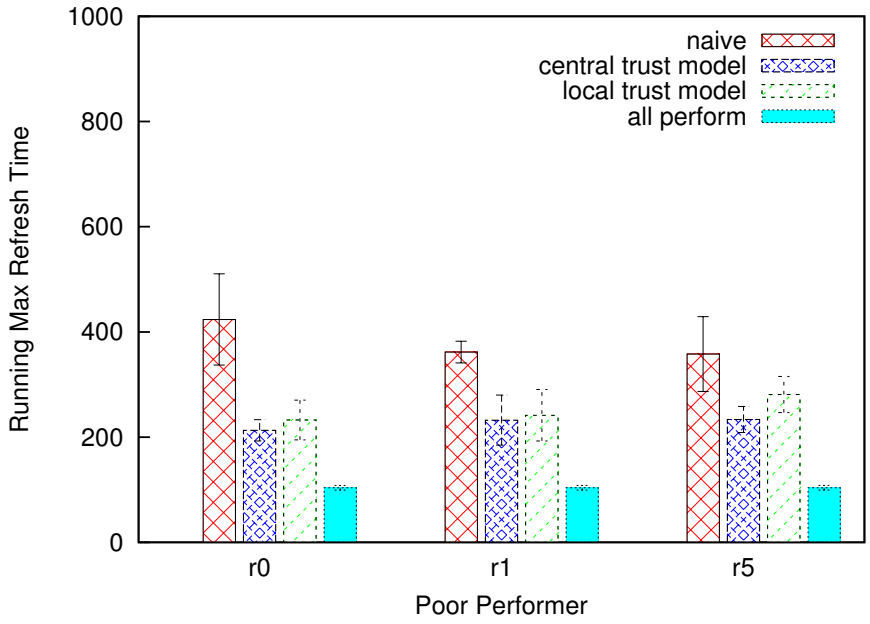


Figure 68: Results are shown from experiments performed in simulation in the pod world environment. Each experiment was run five times, and the error bars represent one standard deviation.

Table 12: Comparison of the trust model approaches for central and distributed task allocation (CTA vs. DTA) and trust authority (CA vs DA).

Trust Model	Strategy	Comms Overhead	Optimal	Complete
CTA, CA	full comms	high	✓	✓
DTA, CA	trust only	med	-	-
	coordination	med	-	-
	annotations	med	✓	✓
DTA, DA	no coordination	low	-	-
	coordination	med	-	-
	annotations	med	-	✓
	verification	med+	✓	✓

approach has a higher communications overhead, as the central node must have the ability to communicate with each of the robots in the environment. The central monitor is able to perform an optimal allocation of task and can ensure that the allocation is complete in that tasks are not over-allocated to multiple robots. Each robot starts with an optimal partition and responds to messages from the central monitor to accept new tasks or to remove tasks.

In the distributed task allocation (DTA) with central trust authority (CA) approach, the robots can locally decide to assist neighbors based on their observations and the *centralized* trust model. The robots can query the trust model to determine whether a neighbor is trusted. If a neighbor becomes untrusted, the robot selects the closest neighboring node and adds that node to its own patrol list and sends a message to the untrusted neighbor that it is taking that task (although the neighbor may choose to not remove that task from their list.)

In the distributed task allocation (DTA) with distributed trust authority (DA) approach, the robots can locally decide to assist neighbors based on their observations and *local* trust models. The robots begin with an initial optimal partition and monitor the task completion of their neighbors and build a trust model for each neighbor. Here again, if a neighbor becomes untrusted, the robot selects the closest neighboring node and adds that node to its own patrol list and sends a message to the untrusted neighbor that it is taking that task.

The distributed task allocation approaches have lower communication overhead because a central node is not needed to perform task allocation and optimization. However, the

DTA,CA approach requires periodic queries or updates from a centralized trust authority. The distributed allocation approach also has the tradeoff that multiple robots may assist a poorly performing neighbor when only one is required, as shown in Figure 69(a), if there is no coordination between neighbors. If a coordination mechanism is used, then with more communications overhead, a single robot at a time can be used to pick up tasks from an untrusted neighbor, preventing assignment of multiple robots to the same task, as shown in Figure 69(b).

When an assisting robot picks up tasks from untrusted neighbors, it will require additional time to visit all of the nodes in its list. Without further coordination with neighbors, it might be viewed as a *poor performer* because the neighbors may not be aware that it is assisting a team member. This results in the problem of cascading task re-assignment, as shown in Figure 70(a). With the use of trust annotations, the assisting robots broadcasts to its neighbors an *annotation* that the robot is assisting a neighbor and that its trust model should not be penalized by a trust authority. When an exterior node notices poor performance by the assisting robot, it is not penalized because it now has knowledge of the assistance, as shown in Figure 70(b).

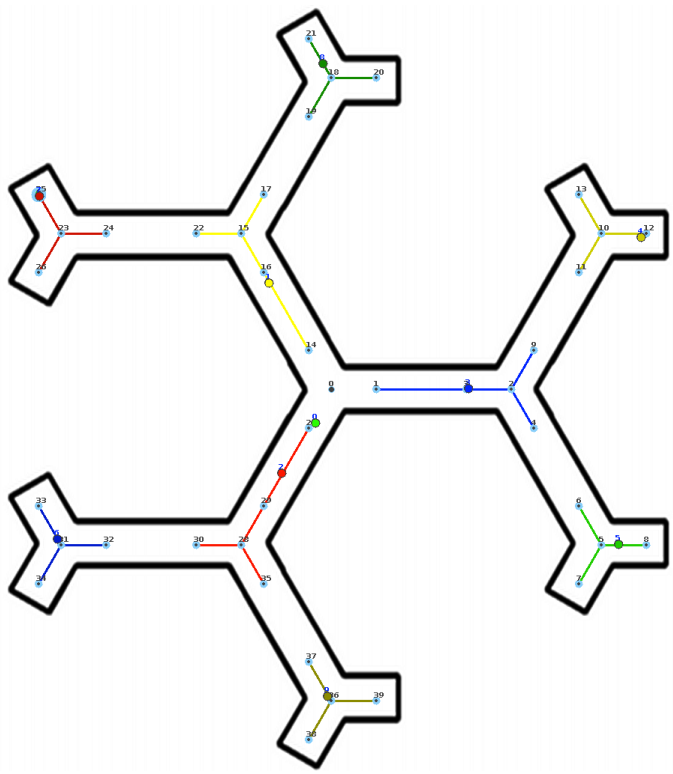
The annotation mechanism has thus far assumed honest reporting of annotations to the trust authority that the robot is providing assistance to a neighbor and should not be penalized. The trust values over time for this approach are shown in Figure 71. In the first experiment, a dishonest robot reports that it is assisting a neighbor and starts out doing so, but after a *defection* phase begins, it only does so with small probability. The reputation authority without verification does not penalize the dishonest assisting robot. In the second experiment, when verification is added, the reputation authority queries the monitor to determine if the assisting robot is regularly providing assistance as reported in the annotation. When the reputation authority (on each robot) discovers that the assisting robot is not performing as reported, it penalizes the trust score for the dishonest robot and it becomes *untrusted*.

7.4.4 Summary

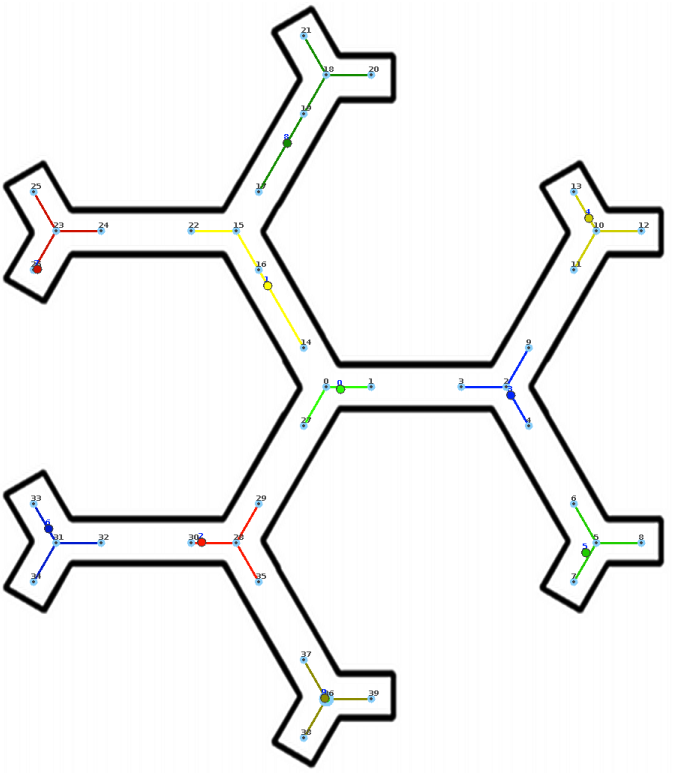
The experimental results using the robot team showed that a monitoring approach with trust modeling can be effective for detecting poorly-performing team members. The centralized assignment approach has the advantage of being able to optimally reallocate tasks; however, it may not always be possible or desirable to use a central task allocation, based on mission requirements. A distributed approach to task allocation and trust modeling can also be used, however, this could result in multiple neighbors assisting the same robot, and cascading assistance from neighbors, unless coordination and trust annotations are used.

Additional experiments in simulation using a local trust model and local reassignment showed that with more coordination between robots a local reassignment can be used without over subscribing the assisting robots. In practice, the decision for where to place the trust authority and task reassignment mechanism is dependent on several factors of the environment, including communication, the state model, the trust monitoring approach to observation and task reassignment approach. Regarding the tradeoff between a central assignment strategy and a fully local strategy, we suggest that a hybrid approach could combine the local approach with local negotiation, using coordination between neighbors.

This section presented a method for recognizing which robots are performing poorly in a multi-robot patrolling task, along with robot and simulation based experimental results using this approach. The results showed that a monitoring approach with trust modeling can be effective for detecting poorly-performing team members. In addition, a task reassignment mechanism can be effective for more efficiently re-assigning patrol tasks, when compared to the naive approach which doesn't monitor individual robot performance or adjust task assignments. This may prove useful in situations in which multi-robot teams are dynamically formed or when not all team members are likely to perform effectively over time.

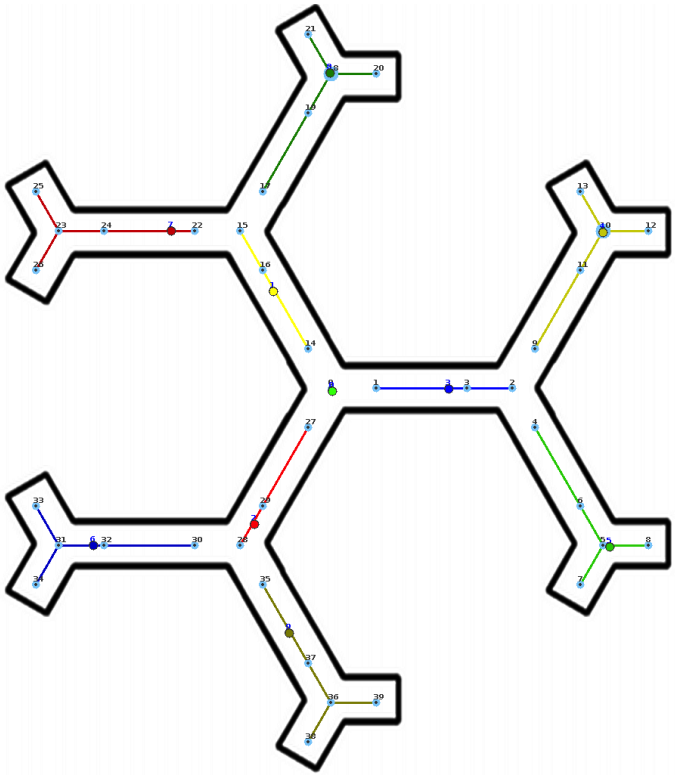


(a) Multiple interior neighbors assist robot 0.

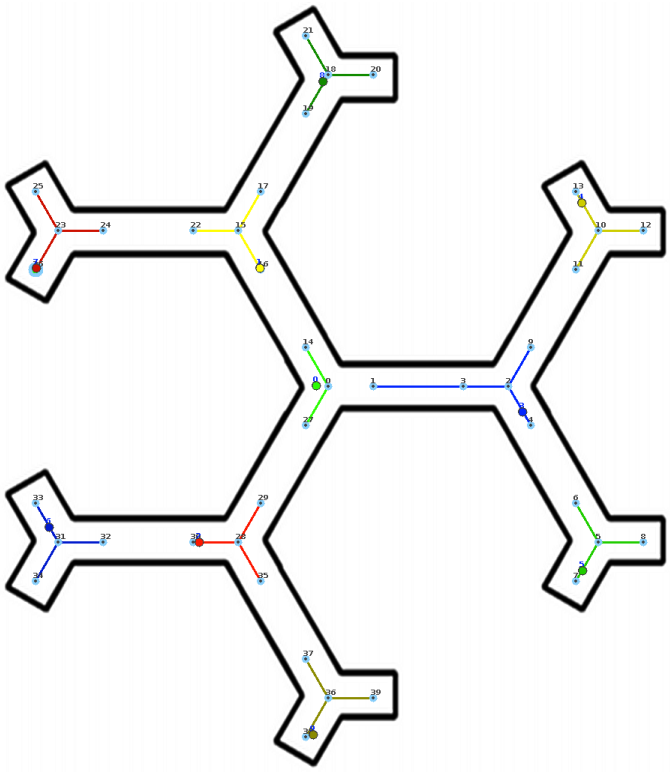


(b) Local Trust with Neighbor Coordination.

Figure 69: Local Trust without coordination. (a) The local trust model strategy with no neighbor coordination results in suboptimal allocation of interior robots to assist the poor performer. (b) The local trust model strategy with neighbor coordination results in only a single neighbor assisting the interior node.

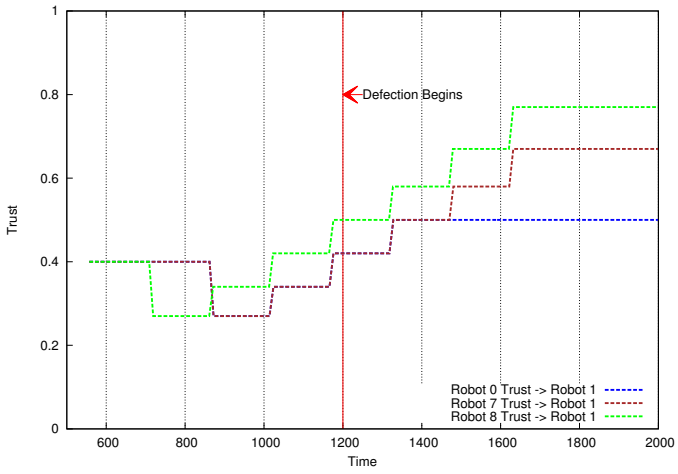


(a) Cascading Assistance.

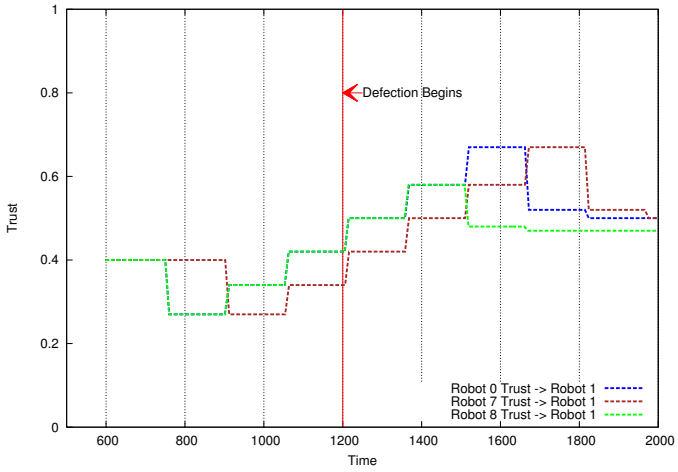


(b) Local Trust with Coordination and Annotations.

Figure 70: Local trust with annotations. (a) An interior node assisting an untreated neighbor can result in cascading assistance by exterior neighbors. (b) With trust annotations, the exterior node does not penalize the assisting node.



(a) Dishonest Assistance



(b) With Verification

Figure 71: Local trust with annotation verification. The trust scores are shown for robot 1, from each neighbor of robot 1. (a) Robot 1 falsely claims to be assisting a neighbor and is given additional time for task completion, resulting in improved trust scores. (b) With trust annotation verification, the trust score for robot 1 decreases after detecting that assistance was not performed.

CHAPTER VIII

A RECIPE FOR IMPLEMENTATION OF THE TRUST FRAMEWORK ON MULTI-ROBOT TEAMS

In previous chapters, we have considered methods for adjusting task assignment and team formation strategies based on performance, methods for monitoring performance and modeling trust, and experiments which applied the trust framework in multiple experimental domains. In this chapter, we review the important considerations for applying a trust framework to a multi-robot team and present the practical steps for implementing this framework in a new domain. Our objective is that robot system designers can adopt this methodology to apply this framework in other real world environments. The following aspects must be taken into account.

1. Consider the performance metrics for the robot action space.
2. Determine where to locate the reputation authority.
3. Interpret trust and adjust the control strategy.

These steps are presented graphically in a conceptual flow diagram, shown in Figure 72. The robot actions will result in a set of observable actions, corresponding to control data generated by the robot. A monitoring process will collect the set of actions and apply performance metrics to determine positive and negative observations that are relevant to trust dimensions. These observations are used to update trust models, which can be housed centrally, distributed or locally in a reputation authority. The reputation authority provides a value for trust for robot teammates which the control strategy can reason over and use to inform task allocation, team formation and incentive strategies as part of the robot controller.

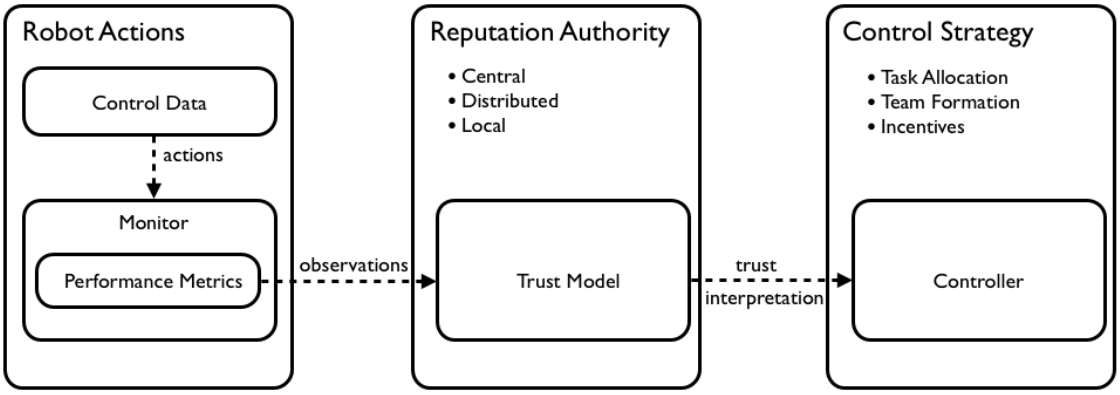


Figure 72: The conceptual flow diagram for implementing the trust framework.

8.1 Steps for Implementing the Trust Framework

8.1.1 Consider the Performance Metrics for the Robot Action Space

The first step is to consider the performance metrics for the robot action space that are relevant to the mission and environment. Here, the system designer should consider which dimensions of performance are important to characterize the correct actions or behaviors for individual robot team members. If we were given perfectly observable robots, we could monitor and capture all of the actions over time for each robot. However, this data set would be quickly become large, and unmanageable. Therefore, we require an approach for monitoring the salient performance dimensions, applying a performance metric to these dimensions and generating a set of discrete observations that can be used to inform a trust model.

Determine the Appropriate Metrics

Perhaps the most important part of this step is to consider the appropriate performance metrics that can be used to combine the observable control actions into a measure of performance. In the performance view, a robot that is not within the defined threshold for the metric is exceeding the performance bounds. It is important to consider metrics that can be evaluated by the monitoring approach. This implies that there is some observable characteristic of the system that can be measured.

The performance metric should include one or more thresholds that could be used to

determine when a robot's process is out of control. In this dissertation, we applied control charts from Section 5.4 as a tool to determine the performance thresholds in relation to team mates. However, the thresholds could be statically defined (consider a threshold on probability of detection.) It is also reasonable to consider metrics in terms of fault tolerance [69].

In the game theoretic view, to exceed the metric is to defect against a partner in a two player game. Therefore, it is possible for the metric to be action based, and dependent on the actions of multiple robots. In addition, if the state is hidden then an estimate mechanism can be applied to discover the hidden state, such as in [97] to discover a behavioral state.

Several example trust dimensions are presented in Table 9. In Section 6.3 the dimension was sensor quality (probability of detection), in Section 7.3 we considered participation as the trust dimension, and in Section 7.4 the performance dimension of max refresh time was applied to the multi-robot patrolling domain.

If there are multiple trust dimensions, then each dimension can be represented using a separate trust model, as described in Section 6.1.3. Each model's trust value can be considered independently, and an overall trust score can be calculated using a linear combination for the trust value, τ, and the confidence value, γ. Alternatively, the Dirichlet distribution can be used if there is a strong correlation between trust dimensions.

Consider the Monitoring Approach

The monitor will be used to collect performance observations and send them to the reputation authority. The monitor should be able to collect the observations that are necessary for analysis using the performance metric. A generalized summary of various approaches to monitoring are shown in Table 13 below. Note that the monitoring characteristics are general and may vary depending on the mission requirements and environment. For instance in remote environments, it may be difficult to place sensors or human observers throughout the environment and therefore it may become necessary to rely on the robot team mates to monitor each other. Similarly, in Section 6.3 we presented experiments using a single, trusted UAV with known sensor characteristics, that was used to selectively monitor its

team mates to collect observations.

Table 13: Comparing General Characteristics of Monitoring Approaches

	Observations	Dynamic Environment?	Large, Remote Environment?	Cost Structure
Self Reporting	assumed honest	yes	yes	minimal
Human Observation	subjective	yes	no	low
Robot Observation	objective	yes	yes	medium
Sensor Network (i.e. cameras)	objective	no	no	high initial

Positive and Negative Observations

Recall from Section 6.1, the probability of trust is modeled as a beta distribution where the mean for the distribution is directly proportional to the positive α and negative β observations.

$$E_{trust_{i,j}} = \frac{\alpha}{\alpha + \beta} \tag{22}$$

Therefore, the system designer should consider how an observed violation of the metric should affect the trust score. The observation should also be amenable to a binary representation. If positive and negative observations are equally likely, then we can weight α and β equally. It is also important to consider how often the positive observations are counted. While it might be straightforward to determine a negative observation (i.e. a robot did not complete a task, or performance exceeded a threshold), a positive observation might not be as clearly defined. Yet, it is important to have a series of positive observations to increase the trust score and trust confidence values for robots that are performing well. In the participation dimensions, robots were credited positively when they participated in task allocations. In task completion scenarios, robots might be rewarded for task completion. Depending on the problem domain, the system designer may wish to weigh positive observations more or less than negative observations.

Dealing with Environmental Variation

In some environments, poor performance of a robot may be due to environmental variation, rather than performance variation. For example, muddy patches on a path might cause wheel slippage, or the presence of obstacles could slow a robot down. In these cases, we would like to allow for environmental variation in the performance expectation for tasks

located in those areas. If an environment is likely to have such variation, an initial period could be used to test robots in different areas of the environment before adjusting the trust model. One approach to detect environmental variation is to periodically move robots to different robots of the environment and then perform statistical analysis of the results to determine if particular areas are difficult for multiple robots. Another approach is to use a known well performing robot to survey different sections of the environment. For example, in Section 6.3, a UAV with a known good sensor performed verification of targets in the environment that were sensed by others.

8.1.2 Determine Where to Locate the Reputation Authority

For the next step, it is important to consider where the trust model is housed, and how the reputation information is shared with other team members. This architecture will be dependent on the communications capabilities of the multi-robot system. If the robots are able to communicate effectively across the entire environment, having persistent communication links, then it is reasonable to consider housing the reputation authority in a central location. This may be possible in smaller environments or in those without communication constraints. This is the simplest approach, in that all robots can simply provide updates to the centrally housed reputation authority and receive updates. The centrally housed reputation authority will have the most accurate information, but robots will have to query it or wait for periodic updates.

In the distributed case, each robot can locally maintain a reputation authority and keep it updated based on local observations of teammates. As the robot encounters other trusted teammates, it can share its direct *positive* and *negative* interactions as reputation information. This approach allows for fast querying of the reputation authority, because it is locally housed, but it takes more resources and time to keep each repository updated with the latest information.

In the local case, each robot maintains a trust model based on direct observations, but does not share reputation information with neighbors. Therefore, each robot needs to directly experience interactions with each teammate and does not benefit from indirect

observations via reputation information. In this case, the repository does not have complete information.

In a hybrid approach, each robot maintains the local reputation authority but also sends updates to a central node when a communication link can be established. For example, consider robots that are operating in a real world environment and communicating over a cellular network. When the robots are not within range of a cell tower, they can maintain their local trust authority until they can connect to a nearby tower and send the updates to the central trust authority. Because the towers are all networked, the central authority can consist of a single repository. As the time in between updates decreases, this solution converges to the centrally located, full communications case. These approaches are summarized below.

Central Location - with a centrally located reputation authority, the robots in the system have relatively persistent communications capabilities and can communicate throughout the environment.

Distributed - with a distributed reputation authority, each robot maintains a local reputation authority and sends out updates to neighbors that are within communication range.

Local - with the local approach, each robot maintains a local reputation authority and does not share information with neighbors.

Hybrid - with the hybrid approach, each robot maintains a local reputation authority as in the distributed case, but periodically sends updates to the central reputation authority when it is within range or able to establish a communications link.

The placement of the reputation authority and the mechanism for incorporating indirect observations and direct observations into a single trust model is discussed in more detail in section 6.1.

8.1.3 Interpret Trust and Adjust the Control Strategy

The last step is to consider how to interpret the trust model and apply it to the control strategy for a multi-robot team. As part of this, we may wish to incorporate prior robot performance knowledge into the model.

Apply Prior Knowledge

The use of a probabilistic trust model also allows the ability to incorporate prior knowledge into the system. For instance reliability information about robots from a particular agency or from a particular manufacturer could be used. In addition, observation histories from past interactions could also be used to inform the model for interaction in new scenarios and environments. Each trust dimension could be initialized using the α and β histories to generate the initial trust model.

The trust experiments in this dissertation assume that there was no prior information available about the trust dimensions for the individual robots. In this case, we initialized $\alpha = \beta = 1$. This is a special case of the beta distribution, which results in a uniform distribution over all of the values for the trust probability, τ, see Figure 73. The confidence, γ in this model is low, because of the low number of observations. When we obtain more observations, but we still have $\alpha = \beta$, the density will remain symmetric about 0.5, but with more of the distribution weighted in that region, resulting in an increased γ for this distribution. When $\alpha > \beta$, the distribution becomes skewed to the right and τ increases. Conversely, τ decreases when $\beta < \alpha$.

If we have an prior expectation for how a particular model of robot or a specific robot instance might perform, then we can initialize α and β to reflect the initial performance distribution and confidence levels. In this case, it would require more observations to significantly shift the value for τ and similarly, would prevent a series of noisy observations for shifting the model significantly.

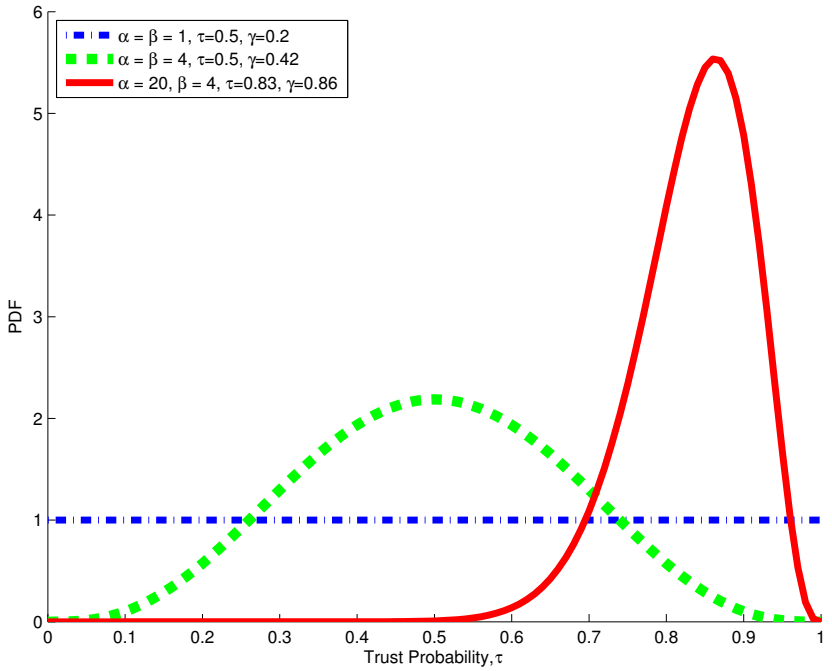


Figure 73: Initializing the Beta Trust Model. To initialize the model with no prior information, we set $\alpha = \beta = 1$, resulting in the uniform distribution, with low confidence, γ. If we have prior information, we can initialize the distribution by setting α and β to achieve the desired initial distribution and confidence.

Interpret the Trust Model

A this point we can consider how to interpret the trust model. This raises the following two questions: 1) How should we use the model to determine when a robot is untrusted?, and 2) How should we treat a robot that is untrusted? To answer the first question requires a subjective interpretation of the trust model. The answer to the second question is dependent on the mission requirements and the dynamic nature of the environment.

First, we consider how to interpret the trust model to determine whether a robot can be trusted. The trust value, τ has a clear interpretation related to trust: it is the belief in the probability of another robot successfully performing in the trust dimension. To consider this belief as a binary decision, whether a robot is *trusted, untrusted*, we must define a sufficient probability condition for trust in that dimension. To do so, we define two thresholds for each trust dimension: θ_τ, the value above which a robot is trusted based on the trust value, τ, and θ_γ, the threshold for which we have sufficient confidence in the belief, based on the confidence value, γ. These thresholds should be set by the system designer,

as the appropriate value may be affected by the amount of variation in the system and environment.

Therefore, after performing a series of observations, the decision to trust by checking whether a robot is trusted or untrusted with sufficient confidence, as shown in Algorithm 11, *CanTrust*. This represents an *optimistic* view, wherein all robots begin in a *trusted* state, until after a series of observations they become *untrusted* with sufficient confidence. Another approach is to consider all robots as being initially *untrusted*. Depending on the mission requirements, it might also be reasonable to consider three states *trusted, untrusted, unknown* and to introduce a lower threshold value on the trust score.

1: $CanTrust \leftarrow True$;
2: **if** $\tau < \theta_\tau$**and**$\gamma > \theta_\gamma$ **then**
3: $\quad CanTrust \leftarrow False$;
4: **end if**

Algorithm 11: CanTrust

The system designer should also consider whether the trust model should allow for *forgiveness*, which allows for a robot to become trusted again after a period of being untrusted. In some cases, a robot may be repaired externally or may respond to incentives and adjust its behavior. To allow for forgiveness, the trust model can provide positive observations for an untrusted robots, when the robot is performing well based on the unit performance of the tasks that it still has assigned to it. For instance, in the patrolling experiments in Section 7.4.2, the trust model for an untrusted robot can be updated with positive observations if it begins to perform well. Once it becomes trusted again, it can be given more tasks.

Another possible interpretation for the trust model is to select the best performers for inclusion on a team, after a period of observation. If a task requires fewer robots than the number available, then for a given trust dimension, the top n robots could be selected using the trust model for each, by selecting those with the highest n trust values above a given confidence threshold.

This brings us to the second question, which determines how to deal with a robot that is determined to be untrusted. We have presented two basic approaches in this dissertation: 1) Evolve the control strategies for the team by changing the task allocation or expectation,

or 2) Adjust the formation of the team, by removing a team member. We discuss each option below. In both cases, we can apply the use of incentives to encourage cooperation.

Evolve the Task Allocation Strategy

In some domains, it may be desirable to keep the team intact, and modify the task allocation or control strategy to assist poor performing team members or to only assign them lower priority tasks. This may be desirable when the team is more static in nature, when poorly performing robots might still be useful, or when the robots can not easily be replaced. We performed experiments that adjusted the task allocation strategy based on the trust model in Section 7.4. Other experiments considered mechanisms for altering the task allocation strategy using an expected utility formulation (Section 4.1), task estimation (Section 4.2), and bucket brigades (Section 4.3). The expected utility formulation allows for a mixing of performance characteristics, using an expectation of utility. This allows for less reliable robots to still participate on the team, when there are lower cost (or lower priority) tasks available for them to perform. In these situations, rather than a robot being completely remove from the team, they can still be utilized on tasks that have lower cost or importance.

Adjust the Team Formation

In other situations, if a robot team mate becomes untrusted, we may desire to remove them from the team. If there are benefits to cooperation, then removing them from the team will serve as a form of punishment for the non-cooperative or poorly performing team member, while serving to protect the rest of the team from the unreliable partners. We demonstrated examples of using the trust model to perform team formation in Sections 7.1 and 7.2.

This may be a good option in situations where the entire team is not needed to perform a task, or when the formation of the team is dynamic in nature, and new robots can be added or robots might be repaired and replaced.

Apply Incentives

Incentives can serve as an enforcement mechanism for robots that do not cooperate effectively or perform as expected. With robots that fully cooperate or perform as expected, the

use of incentives may not be necessary. However, on dynamically formed teams, they can be useful to encourage desired behavior, as described in Sections 3.4 and 7.1. In game theoretic approaches, agents can punish other agents that *defect* against them by not cooperating. In repeated play, agents can do well by following tit-for-tat strategies to encourage cooperation. On multi-robot teams, the meaning of defect can be more subtle. In addition, the presence of noisy observation mechanisms could result in a breakdown of cooperation. In this research, we have presented the use of a trust model as a mechanism to determine which robots are trustworthy and to use that to inform team formation strategies. For robots that become untrusted, removing them from the team or reassigning tasks may provide sufficient incentives for them to cooperate in the future. Therefore, the use of a reputation authority also provides benefits as an incentive mechanism.

8.2 Summary

This chapter provided a review of the trust framework applied to multi-robot teams, and a step by step approach for applying this framework to new domains. The system designer should consider the performance metrics that can be applied to observable control data and used to generate observations for a trust model which forms the basis for a reputation authority. The trust model is then used to inform the robot control strategy for performing task allocation, dynamic team formation, and applying incentives. In the next chapter, we summarize our results and review the contributions of this dissertation.

CHAPTER IX

CONCLUSION AND FINAL WORDS

In this dissertation, we have introduced a framework for building models of trust on multi-robot teams and for using the resulting models to perform dynamic team formation, and to inform task allocation mechanisms. In this closing chapter, we review the contributions made by the dissertation, discuss how these contributions address the research questions presented in Chapter 1, and outline related areas for future research.

9.1 Contributions

There are several contributions that have resulted from this dissertation. They include:

- **Algorithms and methods for learning and applying performance characteristics of individual robots to task allocation (Chapter 4).** We demonstrated that task allocation mechanisms can incorporate robot sensor characteristics (Section 4.1). We have conducted an experiment in simulation with robots having three different sensor characteristics, and applied an expected utility formulation to the task allocation mechanism. We have shown that considering individual sensor characteristics on a robot team can result in better detection performance than if sensors performance is not considered. We have also shown that even robots having poor sensor detection rates can still be useful if the tasks to be performed by the poorly performing sensor are low cost, even if this results in multiple sensor tasks by different robots to achieve the desired result. In addition, we demonstrated an approach for learning which robot team members accurately estimate costs for task allocation (Section 4.2). We also presented the bucket brigade algorithm, for robots with different known performance characteristics to dynamically adjust the task allocation approach in a multi-robot patrolling domain (Section 4.3).

- **Algorithms and methods for robots to monitor performance of team members (Chapter 5).** We presented an approach based on statistical methods from operations research for detecting when a robot was performing poorly on a task in relation to its team members (Section 5.4). We also demonstrated this approach using a central monitor for observing robot performance in a multi-robot patrolling task (Section 5.5).

- **A framework for modeling trust of robot team members (Chapter 6).** We presented an approach for modeling multiple dimensions of trust for individual robots, based on observation histories. We included mechanisms for sharing reputation with other team members. We also demonstrated this approach using a team of UAVs performing a patrolling task, with a single UAV performing observation and trust modeling (Section 6.3).

- **Demonstrations that using the trust model can improve performance on multi-robot teams in the patrolling task (Chapter 7).** We presented game-theoretic foundation for the application of the trust model and demonstrated that the use of a trust model can help to sustain cooperation on multi-robot teams (Sections 7.1, 7.2 and 7.3). We also demonstrated the use of the trust model on a team of eight robots performing a multi-robot patrolling task, and showed that the use of the trust model allowed for the robot team to dynamically reallocate tasks away from poorly performing robots (Section 7.4).

9.2 Research Questions Revisited

How should the heterogeneous characteristics of each team member be included into the task assignment approach? We have developed different approaches for including various robot performance characteristics, including sensor characteristics (Section 4.1), cost estimates (Section 4.2) and patrol performance (Section 4.3), into task assignment approaches. In each case, we demonstrated that accounting for these characteristics can result in better performance than a naive approach.

How can incentives be used to enable rational team members to cooperate? In tasks where robots can benefit from cooperation, we applied results from Game Theory, to show that trust and reputation can be used as part of an incentive based framework to enforce cooperation (Section 7.1). When robots are uniquely identifiable, repeated interactions occur, punishment options are available and a shared reputation mechanism exists, cooperation by individual robots will result in a higher utility than non cooperative behaviors.

How can team members use trust and reputation as a basis for selecting a cooperation and team formation strategy? We applied the incentive results above to an auction based framework for task allocation and demonstrated that when robot freeloaders exist, incentive mechanisms can be used to punish freeloaders by removing them from the team (Section 7.3).

How can a team member monitor performance of other team members in an online fashion to update the models for trust and reputation? We presented an approach to monitoring robot performance using statistical methods from operations research (Chapter 5). We applied this approach in a demonstration of centralized monitoring (Section 7.4) and local monitoring (Section 6.3) as well as to approaches for sharing reputation information across team members (Section 7.4).

How should the heterogeneous characteristics of each team member be included into the task assignment approach? We demonstrated that task allocation mechanisms can incorporate robot sensor characteristics (Section 4.1). In addition, we demonstrated an approach for learning which robot team members accurately estimate costs for task allocation (Section 4.2). We also presented the bucket brigade algorithm, for robots with different known performance characteristics to dynamically adjust the task allocation approach in a multi-robot patrolling domain (Section 4.3).

Primary Research Question

How can trust and reputation in dynamically formed, heterogeneous, multi-robot teams be used to improve task performance? We have considered cooperation not only in relation to the intent of a robot to work with others, but also as an effect of

performance. For robots to cooperate with others to complete tasks, they must be able to complete the tasks at a performance level that is consistent with expectations. Therefore, if a robot is found to be a poor performer, then it is not able to cooperate effectively and should not be trusted. We have presented a framework for trust and reputation to model the performance and cooperation of robot team members and to share these models with other team members. Robots that use this framework can deliberate over the models to adjust the mechanism for interacting with poorly performing team members. We have demonstrated that robots that isolate non-cooperative robots from future interaction can perform task assignment more effectively than when this framework is not applied (Section 5.5). In addition, we have also demonstrated that robots can also improve the performance in the patrolling domain by providing assistance to poorly performing team members when poor performers are present (Section 7.4).

9.3 Open Questions

We now present future areas for research, related to the research questions above and motivated by the work presented in this dissertation. Interesting areas include the ability to form initial impressions about potential partners, to recruit human partners on a robot team and to deliberate over agency level trust.

9.3.1 Pre-communicative Impressions

For much of this dissertation, we have considered domains in which the robots have been initialized with a set of potential partners, a behavioral domain and a communication mechanism, and can dynamically adjust their partner selection and interaction strategies after a set of initial interactions or observations. It would be interesting to remove these initial conditions and investigate mechanisms in which robots can form initial *impressions* about team members before interaction or communication occurs. A key aspect of this direction is that robots would seek to discover behavioral capabilities, rather than interact in predefined behavioral spaces. For example, in a search and rescue domain, several federate, heterogeneous robots may operate simultaneously in a large disaster area, without prior knowledge of each other's existence, ownership or capabilities. Through observation alone,

robots can learn about behavioral diversity [9] and predict behavioral states [98]. In this approach, prior to interaction or communication, robots could observe each other's behaviors and build models of capabilities and intent. After a period of observation, a subset of the observed robots could be selected for interaction and partner selection by querying the relationship graph, similar to the recent work on forming synergistic teams [78]. This information could be used to inform the trust model used to perform interaction in this dissertation, as well as to define the trust dimensions, aligned with the role based capabilities of the observed robots.

9.3.2 Recruiting Human Partners

While this dissertation has presented approaches for performing partner selection on multi-robot teams, we have not considered a separate mechanism for including human partners and teams as part of the team formation process. In one example application, there is active interest and research in using UAVs as part of a manned/unmanned team and to allow unmanned aircraft to integrate with human piloted aircraft [47].

Robots that can learn which situations require trust can be used as part of a general social framework and for learning stereotypes of human capabilities [161]. For instance, a robot may learn the capabilities and characteristics of humans in particular roles, such as that of a fire-fighter, policeman or surgeon, and use these stereotypes to deliberate about cooperative human-robot interactions.

An interesting direction for this work would be to extend the human trust framework to investigate the ability for mixed heterogeneous robot-human teams, wherein robots can recruit trusted human partners who fulfill a particular role.

9.3.3 Agency Level Trust Models

Similar to stereotype based approaches, an agency level trust model would reason about higher level categories of robots, as opposed to building trust models at an individual level. An agency based model would consider the manufacturer of the robot as well as the individual sensors and actuators, the organization that owns the robot, and other general characteristics such as the socio-political environment in which the robot is operating.

Agency level information could be used to generate a priori information for use in trust models as well as to relate individual robot capabilities. This information could also be based on stereotypes of robot capabilities, similar to the trust stereotypes for human interaction [161]. For instance, a particular robot model from a known manufacturer may be capable of performing very specific tasks, and this information could be included into a robot's domain knowledge. In other, more dynamic situations, the country or organization of ownership could be used to inform the trust model of the likelihood that a robot would be cooperative or competitive.

9.4 Final Words

Future robot systems will need capabilities to dynamically form themselves into cooperative teams without being explicitly programmed and designed to do so. While there is a rich body of work on trust in multi-agent systems, the application of trust to multi-robot teams presents additional challenges. In addition, traditional research into multi-robot teams does not fully address the problem of allowing for team structures to be dynamically formed and to evolve based on trust and incentives. As robots become more autonomous and actively deployed in real world environments, it will also become more difficult to manage agency relations. It may not be possible to centrally manage large teams of robots or to statically create partnership agreements. Therefore, these robots will need to learn to dynamically form partnerships with trusted peers. To do so, they will observe online the characteristics of their peers to build models of trust and reputation, and deliberate over those models. This dissertation presented an approach for monitoring robot peers and applying trust models to decide which peers are suitable for cooperation, as well as to provide incentives for cooperation. In addition, multiple dimensions of trust can be modeled with robots that perform well at one task perhaps performing poorly on others. These models can be used to inform future interactions and task allocation strategies, as well as to isolate non-cooperative team members, resulting in improved performance of the multi-robot team.

APPENDIX A

EXPERIMENTAL PLATFORMS

A.1 MASON Multi-Agent Simulator

The MASON multi-agent simulator [80] was used to perform experiments with several simulated robots operating in a simulated environment. MASON is a discrete-event simulation engine that allows for rapid prototyping of agent behaviors. MASON simulations can be run with a GUI, as shown in Figure 74, or in batch mode which allows for very large numbers of experiments.

This tool is used to perform experiments in multi-agent cooperation, without regard to individual robot dynamics. In the simulations, each robot is represented as a point object and the vehicle dynamics are very simple. Rather, the focus is on the high-level interactions between agents and on cooperation and incentive algorithms. Also, each robot can have a simulated communications range, as shown in Figure 74.

A.2 Turtlebot Mobile Robot Platform

The Turtlebot indoor mobile robot platform, shown in Figure 75 was also used to perform several experiments. The Turtlebot is an open robot platform, using the Create development platform from iRobot Corporation as a base. The base platform includes a bumper sensor and a single axis gyroscope. The turtlebot carries a netbook laptop that contains an Intel® Atom™ processor and runs the Linux operating system and the open-source ROS (Robot Operating System) libraries [127]. The laptop is used to send control commands to the Create base platform through the serial port connection, as well as to perform sensor processing and run autonomous behaviors. The ROS libraries include basic components for inter-process communication, logging, visualization, and simulation; as well as higher level components for performing simultaneous localization and mapping (SLAM), navigation and obstacle avoidance.

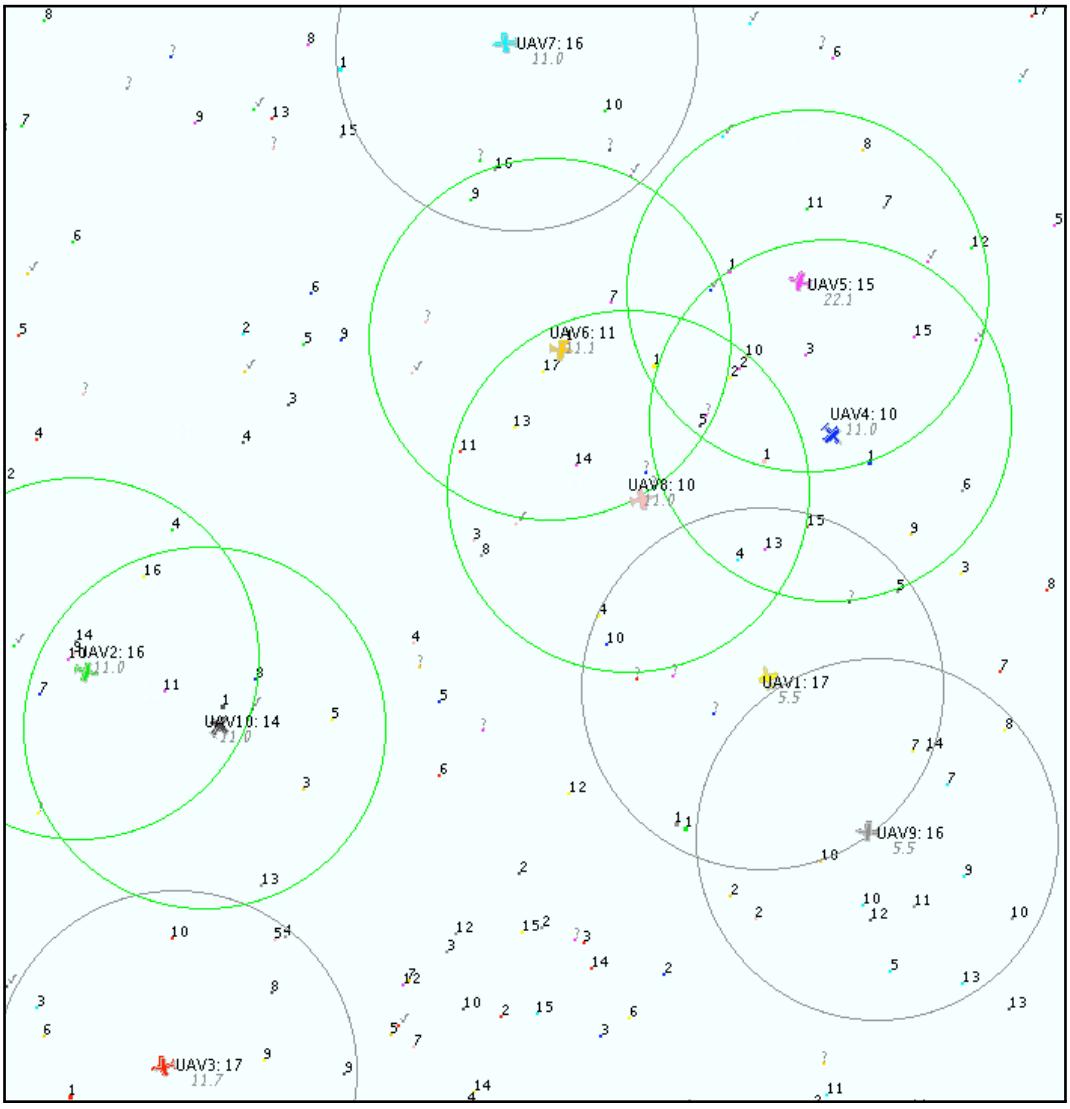


Figure 74: The Mason multi-agent simulator was used to perform experiments of multiple agents performing distributed task assignment. In this figure, Multiple UAVs with limited communications range are shown in a simulation of low-fidelity multi-agent auctions.

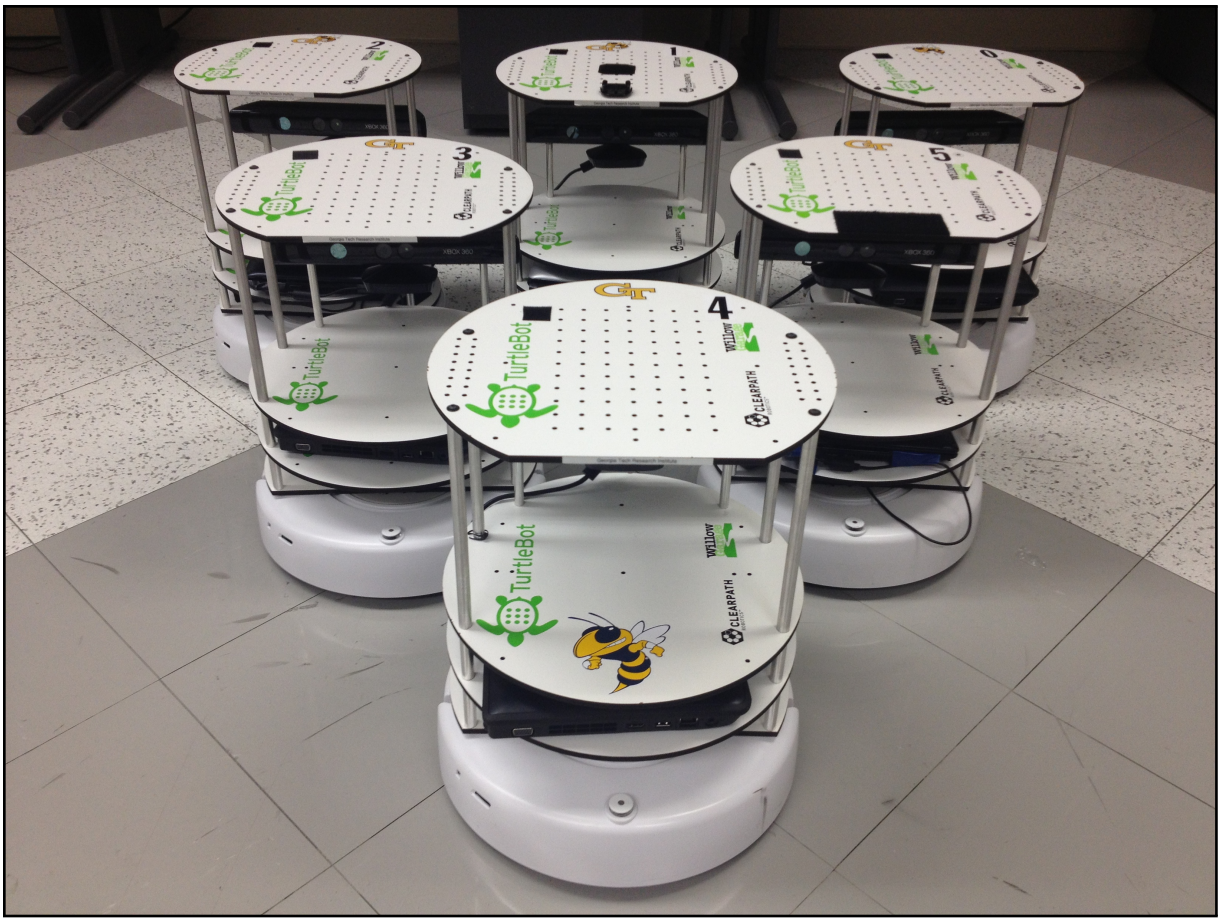


Figure 75: The Turtlebot experimental platform.

The Turtlebot platform also includes a Kinect sensor, which can be used to generate three-dimensional point clouds. The sensor includes an infrared laser projector and corresponding infrared camera which are used to generate range data of the scene for indoor distances up to 6 meters. In addition, the sensor includes an RGB camera. The ROS framework also includes an open source library for interfacing with the Kinect sensor.

To support multi-agent communication, the Turtlebots also include an XBee radio for robot-to-robot communications. The XBee radios support dynamic mesh networks and are used by the robots to exchange negotiation and status messages.

The ROS libraries and autonomous behaviors can also be tested experimentally in the ROS Stage simulation [160] environment, shown in Figure 77.

Figure 76: Multiple Turtlebots are shown patrolling in an imitation art museum.

A.3 UAV Platform Simulation

The multi-UAV simulation based experiments are motivated by GTRI's Collaborative UAV research platform, [121]. The UAV platform leverages off-the-shelf, readily available components, and is based on a quarter-scale Piper Cub airframe with a base model Piccolo avionics and autopilot system from Cloud Cap Technology [158]. The airframe has a wingspan of 104 inches, and carries a mission computer and sensor payloads, see Figure 78(a). The autonomous behaviors that implement the navigation commands, and autonomous behaviors (such as auction based negotiation, shadowing control and trust monitoring) are implemented using the open-source Robot Operating System (ROS) architecture [127]. The ROS libraries also include libraries for performing inter-process messaging.

The platform can also be tested in high-fidelity simulations, as shown in the simulation architecture diagram in Figure 78(b). The flight dynamics of each UAV are simulated using the *software in the loop* (SIL) capabilities of the autopilot. In addition the auction algorithms that run on the mission computer are executed within a separate Linux virtual

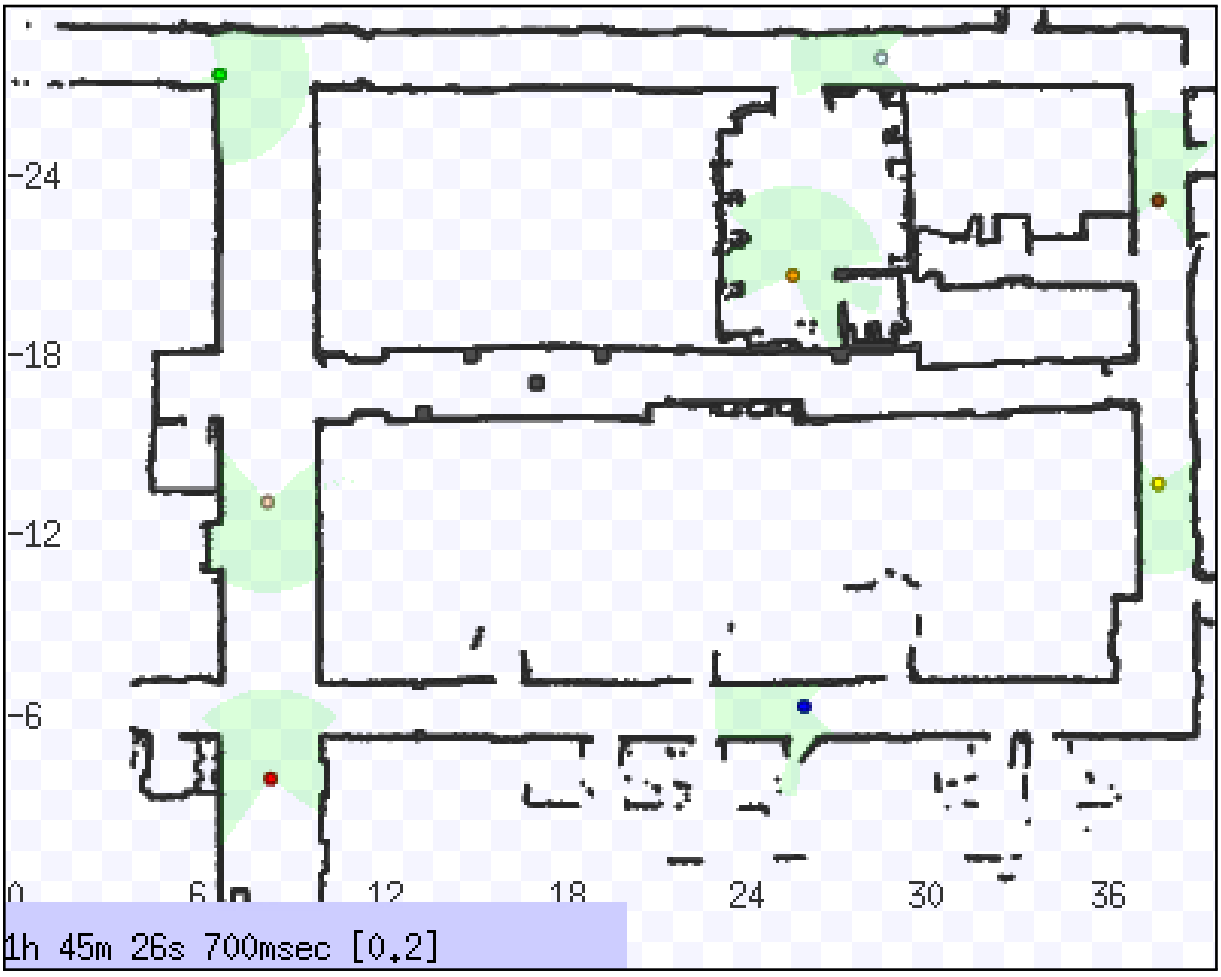
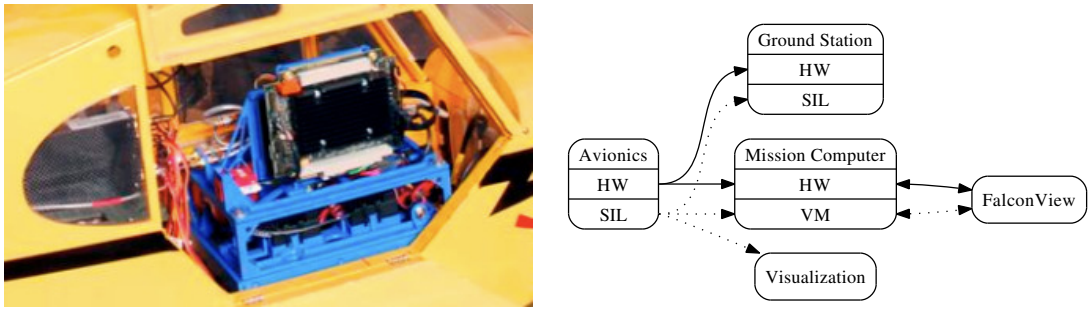


Figure 77: The same autonomous behaviors that run on the real Turtlebots can be tested in simulation. Multiple Turtlebots are shown patrolling in the ROS Stage simulation environment.

machine (VM) for each aircraft to be simulated. Messages are sent to the FalconView^{TM}map display using simulated radio messages. Vehicle positions and assigned waypoints are displayed over the FalconView^{TM}map as shown in Figure 53. In addition, the UAVs in simulated flight are displayed in the MetaVR^{TM}visualization as shown in Figure 79.

In simulation experiments, the platform's autopilot control laws can be simulated using software in the loop (SIL) or hardware in the loop (HIL) capabilities, and the autonomous behaviors can be executed on the mission computer hardware or using virtual machines. The SIL simulation allows for the autonomous behaviors to interact with a simulated autopilot system. In a SIL simulation, the autopilot system's control laws are implemented in a software executable, provided by the manufacturer, and the simulated autopilot is stimulated with environmental input and the motion model of the simulated aircraft. With this architecture, the same autonomous behaviors that were used in simulation experiments can also run on the mission computer in real flight.



(a) Autonomy Payload (b) Architecture

Figure 78: The UAV Simulation Architecture. (a) The UAV platform carries an autonomy payload, consisting of the autopilot and mission computer. (b) The architecture can support the autonomous behaviors in real flight or simulation.

Figure 79: A simulated UAV is shown rendered in a visualization using the MetaVR^{TM}scene generation tool.

APPENDIX B

EXPERIMENTAL ENVIRONMENTS

B.1 GTRI Office World

The GTRI office world environment is show in Figure 80. This map was generated using the ROS gmapping software running on a Turtlebot as it was manually driven around the office space at the GTRI headquarters building. A Voroni graph was then generated from the map [14] and preprocessing was performed to eliminate unnecessary nodes.

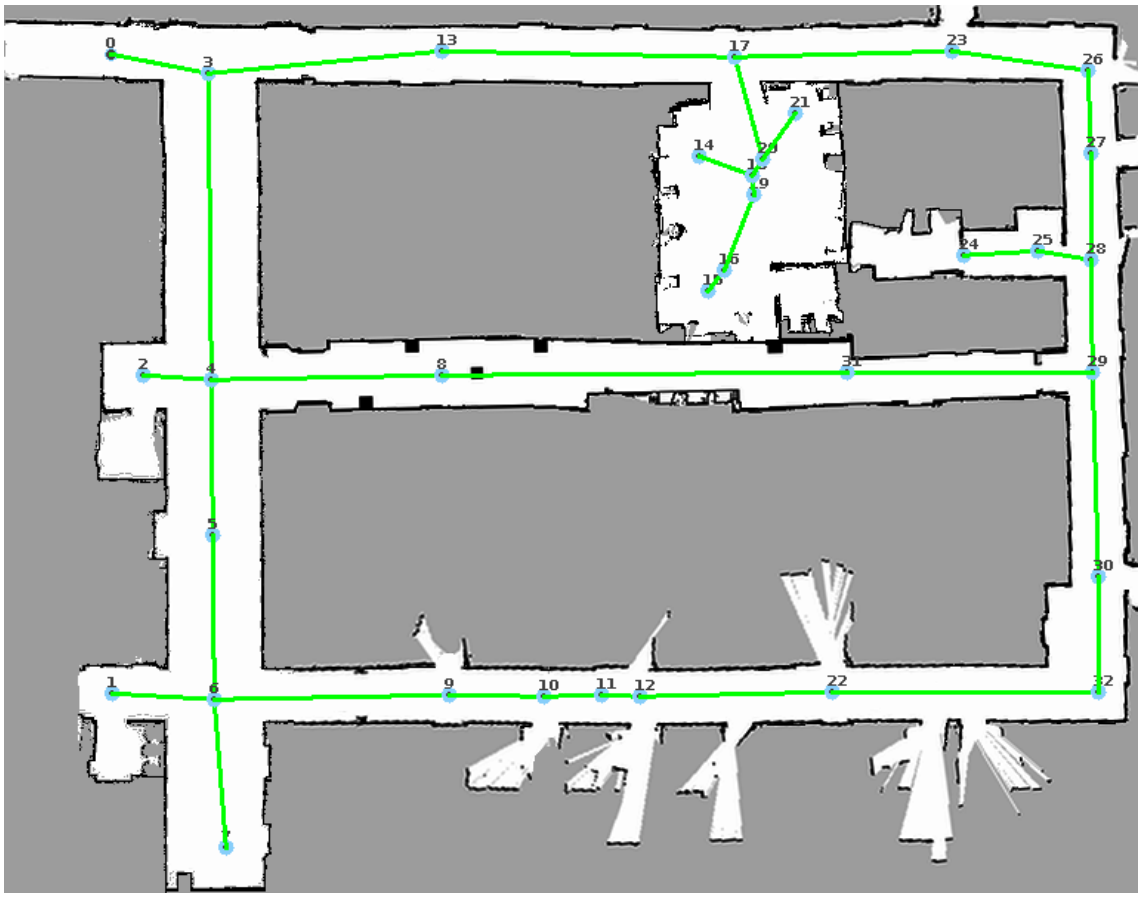


Figure 80: The GTRI office world environment.

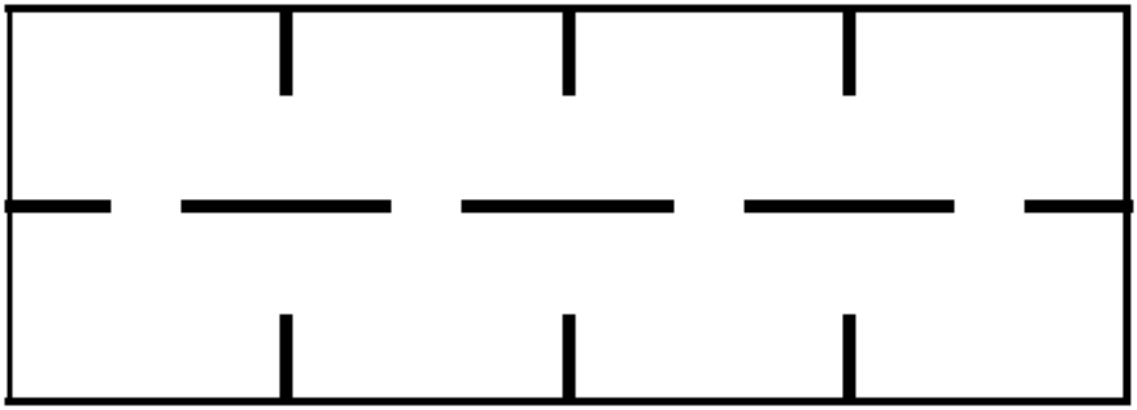


Figure 81: The museum world environment, with 8 rooms. This map was generated using Matlab and was used for experiments in simulation and in real environments at the GTRI headquarters building and the GT Technology Research Square building conference rooms.

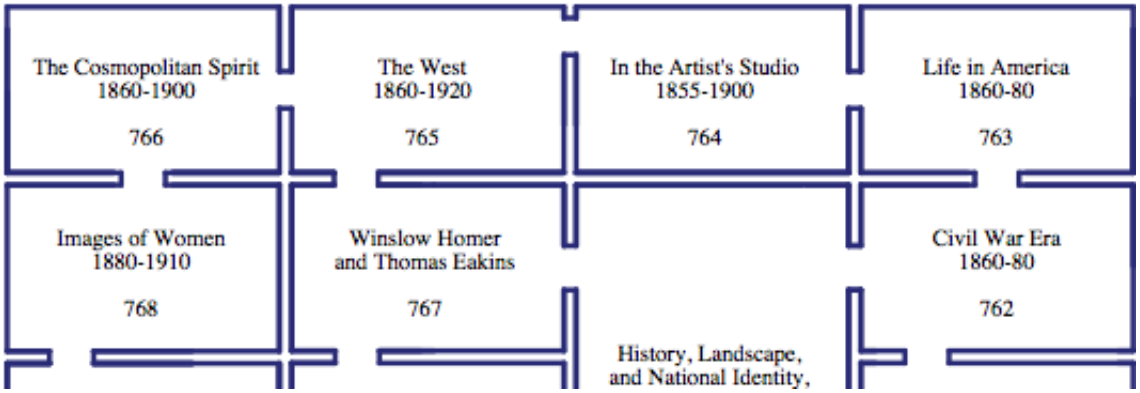


Figure 82: The experimental museum world environment was based on the real world Metropolitan Museum of Art. The American wing, second floor is shown.

B.2 Museum World

The museum world environment is shown in Figure 81, and was inspired by a map of the real world Metropolitan Museum of Art, shown in Figure 82. This environment has the property that each robot can have 2 or 3 neighbors. When the doors to the rooms are open, the environment can be covered using a cyclic patrol with evenly spaced robots. However, for long edges between rooms, it is better to partition the robots, one to each room.

The optimal partition for this environment is shown in Figure 83. For a robot traveling with velocity, v_r, with a team of 8 evenly spaced robots, the cyclic partition results in a max refresh time of

$$\frac{8(2d+h)+8l}{8v_r} \tag{23}$$

$$= \frac{(2d+h)+l}{v_r}. \tag{24}$$

We also have for the partition based approach, a maximum refresh time of

$$\frac{2d+2h}{v_r}. \tag{25}$$

Therefore, the partition based approach results in smaller max refresh times when $l > h$.

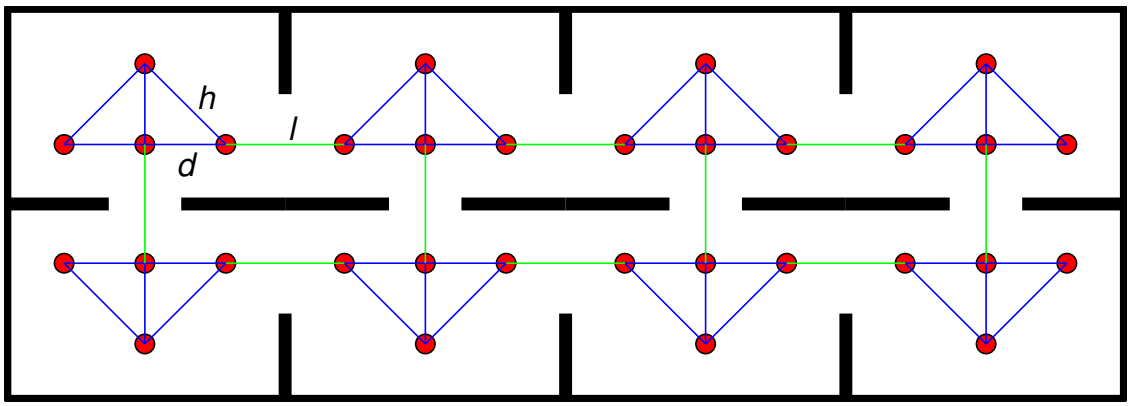


Figure 83: The patrol graph is shown on the map of the museum world environment. The graph can be optimally partitioned for 8 robots by excluding the long edges, l, between rooms. The partition approach results in a lower max refresh time compared to the cyclic approach when $l > h$.

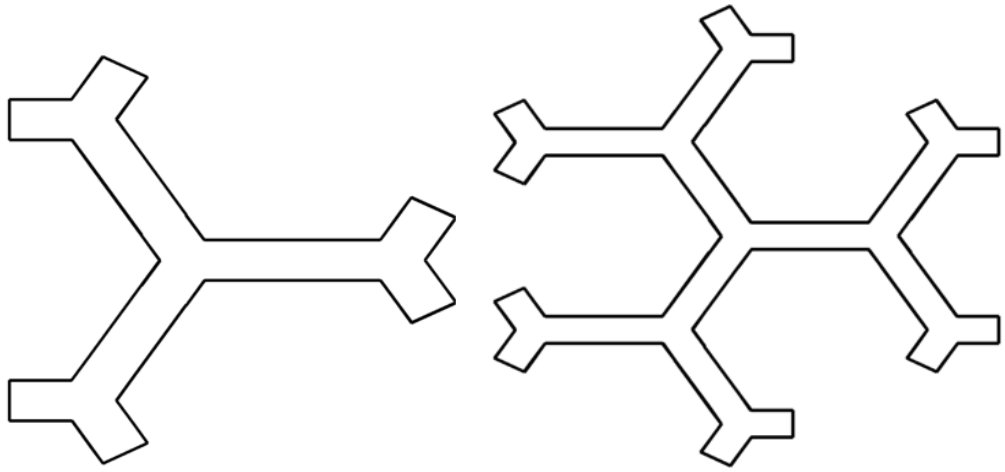
For the case when a single robot is performing poorly, it is better to remove it from the team and return to an evenly spaced cycle in the case when the doors are open. However, there may be security situations in which it is not desirable perform a cyclic patrol, which by

definition is predictable. The cyclic approach is better when the distances between rooms is small, such that the following equation is true.

$$\frac{8(2d+h)+8l}{7v_r} < \frac{2d+2h+2l}{v_r}. \tag{26}$$

B.3 Pod World

The podworld environment is shown in Figure 84. This environment has the property that each robot can have up to 3 neighbors. We can construct the environment, such that each intersection or *pod* is evenly spaced and this allows us to easily calculate the optimal allocation when the number of robots equals the number of pods. While this construct represents an ideal environment that we can use for experiments, some similar real world environments are shown in Figure 86.



(a) 4 pods (b) 10 pods

Figure 84: The Pod world multi-robot patrol environment, used in simulation.

A graph of the podworld environment is created with visit node locations in the intersections, and with long edges between pods. The graph is shown in Figure 85. For this environment, in the patrolling task, it is always better to partition the environment rather than to perform a cycle with evenly spaced nodes. We label the distance between node locations within each intersection as d, and the longer edge between pods as having length l. The graph partition for each robot is a tree. Therefore, in the partition case, for a robot traveling with velocity, v_r, the time required to perform a patrol cycle of a pod is $\frac{6d}{v_r}$, and the

time for a robot to patrol their pod nodes and to pick up a single task from a neighboring robot is $\frac{(6d+2l)}{v_r}$. For the cyclic case, with evenly spaced robots, the maximum refresh time for a node is $\frac{(6d+\frac{3}{2}l)}{v_r}$, and if one of the cyclic robots performs poorly, then the robots would need to divide the patrol tour over the remaining $(n-1)$ robots, resulting in a max refresh time of $\frac{(8d+2l)}{v_r}$. Therefore, with this environment, the partition approach to assigning the robots to nodes will result in a better refresh time, both when the robots perform as expected and when one or more performs poorly and tasks must be redistributed.

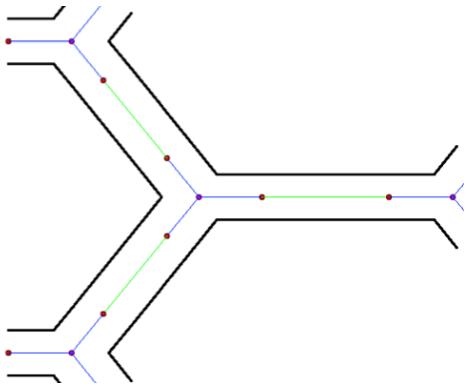
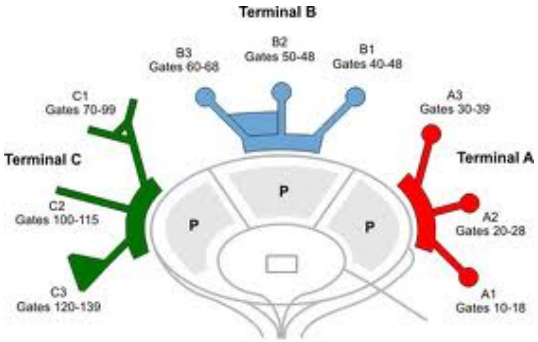
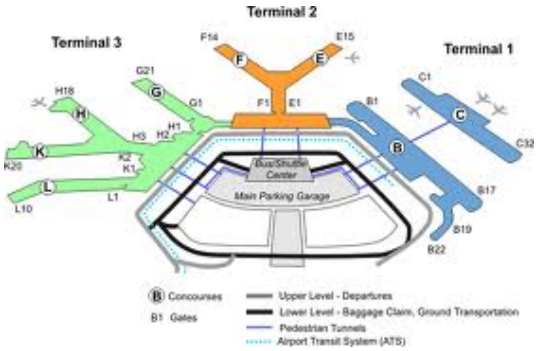


Figure 85: The podworld environment with the patrol graph shown. A partition based patrol is more optimal than a cyclic based patrol when the long edges have length > 0.

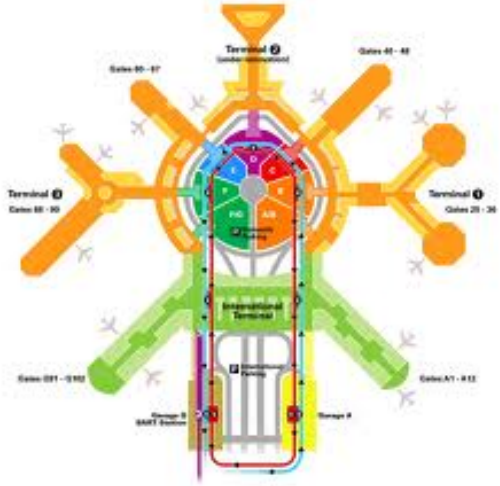
(a) a high school



(b) EWR airport



(c) ORD airport



(d) SFO airport

Figure 86: Examples of real world environments with properties similar to the pod world. (Images courtesy of (a) Google Maps, (b) EWR airport, (c) ORD airport, and (d) SFO airport.)

REFERENCES

[1] "NIST/SEMATECH e-Handbook of Statistical Methods."

[2] Ahn, J., DeAngelis, D., and Barber, S., "Attitude driven team formation using multi-dimensional trust," in *Intelligent Agent Technology, 2007. IAT '07. IEEE/WIC/ACM International Conference on*, pp. 229 –235, Nov. 2007.

[3] Anderson, C., Boomsma, J., and Bartholdi, J., "Task partitioning in insect societies : bucket brigades," vol. 49, pp. 1–10, 2002.

[4] Arkin, E. M., Hassin, R., and Levin, A., "Approximations for minimum and min-max vehicle routing problems," *J. Algorithms*, vol. 59, pp. 1–18, Apr. 2006.

[5] Arkin, R. C., *Behavior-Based Robotics*, ch. 9. Cambridge, Mass., MIT Press, 1998.

[6] Armbruster, D., Gel, E. S., and Murakami, J., "Bucket brigades with worker learning," *European Journal of Operational Research*, vol. 176, pp. 264– 274, 2007.

[7] Arslan, G., Marden, J., and Shamma, J., "Autonomous vehicle-target assignment: A game theoretical formulation," in *ASME Journal of Dynamic Systems, Measurement, and Control, special issue on "Analysis and Control of Multi-Agent Dynamic Systems"*, pp. 584–596., September 2007.

[8] Axelrod, R., *The Evolution of Cooperation*. New York: Basic Books, 1984.

[9] Balch, T., "Behavioral diversity in learning robot teams (Ph.D. Thesis)," *College of Computing Technical Report (GIT-CC-98-25)*, 1998.

[10] Baras, J. S. and Jiang, T., "Cooperation, trust and games in wireless networks," in *Advances in Control, Communication Networks, and Transportation Systems* (Abed, E. H., ed.), Systems & Control: Foundations & Applications, pp. 183–202, Birkhauser Boston, 2005.

[11] Baras, J., Jiang, T., and Purkayastha, P., "Constrained coalitional games and networks of autonomous agents," in *Communications, Control and Signal Processing, 2008. ISCCSP 2008. 3rd International Symposium on*, pp. 972 –979, March 2008.

[12] Bartholdi, J. and Eisenstein, D., "A production line that balances itself," *Operations Research*, vol. 44, pp. 21–34, January/February 1996.

[13] Baxter, J. and Bartlett, P. L., "Infinite-horizon policy-gradient estimation," *Journal of Artificial Intelligence Research*, 2001.

[14] Beeson, P., Jong, N., and Kuipers, B., "Towards autonomous topological place detection using the extended voronoi graph," in *Robotics and Automation (ICRA)*, pp. 4373 – 4379, April 2005.

[15] Bertsekas, D. P., "The auction algorithm for assignment and other network flow problems: A tutorial," *Interfaces*, vol. 20, no. 4, pp. 133–149, 1990.

[16] Bjerknes, J. and Winfield, A., "On fault tolerance and scalability of swarm robotic systems," in *Distributed Autonomous Robotic Systems* (Martinoli, A., Mondada, F., Correll, N., Mermoud, G., Egerstedt, M., Hsieh, M. A., Parker, L. E., and Støy, K., eds.), vol. 83 of *Springer Tracts in Advanced Robotics*, pp. 431–444, Springer Berlin Heidelberg, 2013.

[17] Blanc, A., Liu, Y.-K., and Vahdat, A., "Designing incentives for peer-to-peer routing.," in *INFOCOM'05*, pp. 374–385, 2005.

[18] Bourgault, F., Furukawa, T., and Durrant-Whyte, H., "Coordinated decentralized search for a lost target in a Bayesian world," vol. 1, pp. 48 – 53 vol.1, Oct. 2003.

[19] Bourgault, F., Furukawa, T., and Durrant-Whyte, H., "Decentralized Bayesian negotiation for cooperative search," in *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3, pp. 2681–2686 vol.3, Sept.-2 Oct. 2004.

[20] Brafman, R. I. and Tennenholtz, M., "On partially controlled multi-agent systems," *CoRR*, vol. cs.AI/9606102, 1996.

[21] Braynov, S. and Sandholm, T., "Auctions with untrustworthy bidders," *E-Commerce Technology, IEEE International Conference on*, p. 363, 2003.

[22] Burnett, C., Norman, T. J., and Sycara, K., "Sources of stereotypical trust in multi-agent systems," *In Proceedings of the The Fourteenth International Workshop on Trust in Agent Societies*, 2011.

[23] Burnett, C., Norman, T. J., and Sycara, K., "Trust decision-making in multi-agent systems," in *IJCAI 2011: AAAI Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[24] Buskens, V., *Social Networks and Trust*. Kluwer Academic Publishers, 2002.

[25] Busquets, D. and Simmons, R., *Learning when to Auction and when to Bid*, vol. Distributed Autonomous Robotic Systems 7., pp. 21–30. Springer, 2006.

[26] Cameron, A. and Durrant-Whyte, H. F., "A Bayesian approach to optimal sensor placement," *The International Journal of Robotics Research*, vol. 9, pp. 70–88, September 1990.

[27] Cao, Y., Fukunaga, A., Kahng, A., and Meng, F., "Cooperative mobile robotics: antecedents and directions," in *Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, vol. 1, pp. 226 –234 vol.1, Aug. 1995.

[28] Castelfranchi, C. and Falcone, R., "Social trust: Cognitive anatomy, social importance, quantification and dynamics," in *Proceedings of the First International Workshop on Trust*, pp. 35–49, 1998.

[29] Castelfranchi, C. and Falcone, R., *Trust Theory: A Socio-Cognitive and Computational Model*, ch. 12. John Wiley and Sons, Ltd, 2010.

[30] Cavallo, R., Parkes, D. C., and Singh, S., "Optimal coordination of loosely-coupled self-interested robots," in *Workshop on Auction Mechanisms for Robot Coordination, AAAI-06*, (Boston, MA), 2006.

[31] Chevaleyre, Y., "Theoretical analysis of the multi-agent patrolling problem," in *Intelligent Agent Technology, 2004. (IAT 2004). Proceedings. IEEE/WIC/ACM International Conference on*, pp. 302 – 308, Sept. 2004.

[32] Clark, C. M., Morton, R., and Bekey, G. A., "Altruistic relationships for optimizing task fulfillment in robot communities," in *Distributed Autonomous Robotic Systems 8* (Asama, H., Kurokawa, H., Ota, J., and Sekiyama, K., eds.), pp. 261–270, Springer Berlin Heidelberg, 2009.

[33] Cohen, B., "Incentives build robustness in bittorrent," *In Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems*, 2003.

[34] De Vries, A. and Reneau, J. K., "Application of statistical process control charts to monitor changes in animal production systems," *Journal of Animal Science*, vol. 88, no. 13 electronic suppl, pp. E11–E24, 2010.

[35] Desai, M., Medvedev, M., Vázquez, M., McSheehy, S., Gadea-Omelchenko, S., Bruggeman, C., Steinfeld, A., and Yanco, H., "Effects of changing reliability on trust of robot systems," in *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, HRI '12, (New York, NY, USA), pp. 73–80, ACM, 2012.

[36] Dias, M. and Stentz, A., "A free market architecture for distributed control of a multirobot system," in *6th International Conference on Intelligent Autonomous Systems (IAS-6)*, pp. 115–122, July 2000.

[37] Donald, B. R., Jennings, J., and Rus, D., "Information invariants for distributed manipulation," *The International Journal of Robotics Research*, vol. 16, no. 5, pp. 673–702, 1997.

[38] Doran, J. E., Franklin, S., Jennings, N. R., and Norman, T. J., "On cooperation in multi-agent systems," *The Knowledge Engineering Review*, vol. 12, pp. 309–314, 1997.

[39] Dudek, G., Jenkin, M. R. M., Milios, E., and Wilkes, D., "A taxonomy for multi-agent robotics," *AUTONOMOUS ROBOTS*, vol. 3, pp. 375–397, 1996.

[40] Egerstedt, M. and Hu, X., "Formation constrained multi-agent control," *IEEE Transactions on Robotics and Automation*, vol. 17, pp. 947–951, Dec. 2001.

[41] Ekici, A., Keskinocak, P., and Koenig, S., "Multi-robot routing with linear decreasing rewards over time," in *Proceedings of the 2009 IEEE international conference on Robotics and Automation*, ICRA'09, (Piscataway, NJ, USA), pp. 3944–3949, IEEE Press, 2009.

[42] Elmaliach, Y., Agmon, N., and Kaminka, G., "Multi-robot area patrol under frequency constraints," in *Robotics and Automation, 2007 IEEE International Conference on*, pp. 385 –390, April 2007.

[43] Erriquez, E., van der Hoek, W., and Wooldridge, M., "An abstract framework for reasoning about trust," in *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 3*, AAMAS '11, (Richland, SC), pp. 1085–1086, International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[44] Fraga, D., Gutiérrez, Á., Vallejo, J., and Campo, A., "Improving Social Odometry Robot Networks with Distributed Reputation Systems for Collaborative Purposes," *Sensors*, 2011.

[45] Frias-martinez, V., "A team-based co-evolutionary approach to multi agent learning," in *In In Proceedings of the 2004 AAMAS Workshop on Learning and Evolution in Agent Based Systems*, 2004.

[46] Fullam, K. K. and Barber, K. S., "Learning trust strategies in reputation exchange networks," in *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, AAMAS '06, (New York, NY, USA), pp. 1241–1248, ACM, 2006.

[47] Garcia, R. D., Barnes, L., and Fields, M., "Unmanned aircraft systems as wingmen," *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, vol. 9, no. 1, pp. 5–15, 2012.

[48] Gerkey, B., Vaughan, R., and Howard, A., "The player project: Free software tools for robot and sensor applications," *11 th Int. Conf. on Advanced Robotics (ICAR 2003)*, 2003.

[49] Gerkey, B. P. and Mataric, M. J., "Sold!: Auction methods for multirobot coordination," *Robotics and Automation, IEEE Transactions on*, 2002.

[50] Gerkey, B. P. and Mataric, M. J., "Multi-robot task allocation: Analyzing the complexity and optimality of key architectures," 2003.

[51] Gerkey, B. P. and Mataric, M. J., "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[52] Goodrich, M. A., Morse, B. S., Gerhardt, D., Cooper, J. L., Quigley, M., Adams, J. A., and Humphrey, C., "Supporting wilderness search and rescue using a camera-equipped mini UAV," *Journal of Field Robotics*, no. (1-2), pp. 89–110, 2008.

[53] Grocholsky, B., Makarenko, A., and Durrant-Whyte, H., "Information-theoretic coordinated control of multiple sensor platforms," in *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, vol. 1, pp. 1521 – 1526 vol.1, Sept. 2003.

[54] Grocholsky, B. P., Keller, J., Kumar, V., and Pappas, G., "Cooperative air and ground surveillance," *IEEE Robotics & Automation Magazine*, vol. 13, pp. 16 – 25, Sept. 2006.

[55] Groves, T., "Incentives in teams," *Econometrica*, vol. 41, pp. 617–631, July 1973.

[56] Hauk, E., "Leaving the prison: Permitting partner choice and refusal in prisoner's dilemma games," *Computational Economics*, vol. 18, pp. 65–87, 2001.

[57] Hayashi, N. and Yamagishi, T., "Selective play: Choosing partners in an uncertain world.," *Personality and Social Psychology Review (Lawrence Erlbaum Associates)*, vol. 2, no. 4, p. 276, 1998.

[58] Hovareshti, P. and Baras, J. S., "Efficient communication infrastructures for distributed control and decision making in networked stochastic systems," in *Proceedings of The 19th International Symposium on Mathematical Theory of Networks and Systems (MTNS 2010)*, July 2010.

[59] Howard, A., Parker, L. E., and Sukhatme, G. S., "Experiments with a large heterogeneous mobile robot team: Exploration, mapping, deployment and detection," *The International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 431–447, 2006.

[60] Hwang, K.-S., Lin, J.-L., and Huang, H.-L., "Cooperative patrol planning of multi-robot systems by a competitive auction system," in *ICCAS-SICE, 2009*, pp. 4359 –4363, Aug. 2009.

[61] Iocchi, L., Marchetti, L., and Nardi, D., "Multi-robot patrolling with coordinated behaviours in realistic environments," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pp. 2796 –2801, Sept. 2011.

[62] Jaramillo, J. J. and Srikant, R., "A game theory based reputation mechanism to incentivize cooperation in wireless ad hoc networks," *Ad Hoc Networks*, vol. 8, pp. 416–429, June 2010.

[63] Jones, E., Browning, B., Dias, M. B., Argall, B., Veloso, M., and Stentz, A. T., "Dynamically formed heterogeneous robot teams performing tightly-coordinated tasks," in *International Conference on Robotics and Automation*, pp. 570 – 575, May 2006.

[64] Jones, E., Dias, M. B., and Stentz, A. T., "Learning-enhanced market-based task allocation for disaster response," in *Intelligent Robots and Systems, 2007. (IROS)*, October 2007.

[65] Jøsang, A. and Golbeck, J., "Challenges for Robust of Trust and Reputation Systems," in *Proceedings of the 5th International Workshop on Security and Trust Management (STM 2009)*, Sept. 2009.

[66] Jøsang, A. and Ismail, R., "The beta reputation system," in *Proceedings of the 15th Bled Electronic Commerce Conference*, 2002.

[67] Kalra, N., Zlot, R., Dias, M. B., and Stentz, A., "Market-based multirobot coordination: A comprehensive survey and analysis," Tech. Rep. CMU-RI-TR-05-16, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, 2005.

[68] Kandori, M., "Social norms and community enforcement," *Review of Economic Studies*, vol. 59, pp. 63–80, January 1992.

[69] Kannan, B. and Parker, L. E., "Metrics for quantifying system performance in intelligent, fault-tolerant multi-robot systems," in *Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2007.

[70] Koenig, S., Keskinocak, P., and Tovey, C., "Progress on agent coordination with cooperative auctions," in *AAAI Conference on Artificial Intelligence*, 2010.

[71] Kohl, N. and Stone, P., "Policy gradient reinforcement learning for fast quadrupedal locomotion," *In Proceedings of the IEEE International Conference on Robotics and Automation*, May 2004.

[72] Kraus, S., "An overview of incentive contracting," *Artif. Intell.*, vol. 83, pp. 297–346, June 1996.

[73] Kraus, S., *Strategic negotiation in multiagent environments*. Cambridge, MA, USA: MIT Press, 2001.

[74] Kuhn, H. W., "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[75] Lagoudakis, M. G., Markakis, E., Kempe, D., Keskinocak, P., Kleywegt, A., and Koeni, S., "Auction-based multi-robot routing," in *Proceedings of the International Conference on Robotics: Science and Systems*, pp. 343–350, 2005.

[76] Lein, A. and Vaughan, R. T., "Adaptive multirobot bucket brigade foraging," in *In Proceedings of the Eleventh International Conference on Artificial Life (ALife XI*, pp. 337–342, MIT Press, 2008.

[77] Lewis, A. S. and Weiss, L. G., "Intelligent autonomy and performance metrics for multiple, coordinated UAVs," *Integrated Computer-Aided Eng.*, vol. 12, pp. 251–262, July 2005.

[78] Liemhetcharat, S. and Veloso, M., "Modeling and learning synergy for team formation with heterogeneous agents," in *11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, pp. 365–374, June 2012.

[79] Lindner, J., Murphy, R., and Nitz, E., "Learning the expected utility of sensors and algorithms," in *Multisensor Fusion and Integration for Intelligent Systems, 1994. IEEE International Conference on MFI '94.*, pp. 583 –590, Oct. 1994.

[80] Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., and Balan, G., "MASON: A multi-agent simulation environment," *Simulation: Transactions of the society for Modeling and Simulation International*, vol. 82, no. (7), pp. 517–527, 2005.

[81] Manyika, J. M. and Durrant-Whyte, H. F., "Information-theoretic approach to management in decentralized data fusion," vol. 1828, pp. 202–213, SPIE, 1992.

[82] Marino, A., Antonelli, G., Aguiar, A., and Pascoal, A., "A new approach to multi-robot harbour patrolling: Theory and experiments," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 1760–1765, 2012.

[83] Martinson, E. and Arkin, R., "Learning to role-switch in multi-robot systems," in *Proceedings of the 2003 IEEE International Conference on Robotics and Automation, Taipei, Taiwan,*, vol. 2, pp. 2727 – 2734, Sept. 2003.

[84] Matei, I., Baras, J., and Jiang, T., "A composite trust model and its application to collaborative distributed information fusion," in *Information Fusion, 2009. FUSION '09. 12th International Conference on*, pp. 1950 –1957, July 2009.

[85] Matei, I., Jiang, T., and Baras, J. S., "A trust based distributed kalman filtering approach for mode estimation in power systems," in *Proceedings of the First Workshop on Secure Control Systems (SCS), Stockholm, Sweden*, April 12 2010.

[86] McGrew, R. and Shoham, Y., "Using contracts to influence the outcome of a game," in *Proceedings of the 19th national conference on Artifical intelligence*, AAAI'04, pp. 238–243, AAAI Press, 2004.

[87] McKnight, D. H. and Chervany, N. L., "What trust means in e-commerce customer relationships: An interdisciplinary conceptual typology," *Int. J. Electron. Commerce*, vol. 6, pp. 35–59, December 2001.

[88] Melis, A. P., Hare, B., and Tomasello, M., "Chimpanzees recruit the best collaborators," *Science*, vol. 311, no. 5765, pp. 1297–1300, 2006.

[89] Menezes, T., Tedesco, P., and Ramalho, G., "Negotiator agents for the patrolling task," in *Advances in Artificial Intelligence - IBERAMIA-SBIA 2006* (Sichman, J., Coelho, H., and Rezende, S., eds.), vol. 4140 of *Lecture Notes in Computer Science*, pp. 48–57, Springer Berlin / Heidelberg, 2006.

[90] Michael, T., *How to Guard an Art Gallery and Other Discrete Mathematical Adventures*. Baltimore: The Johns Hopkins University Press, 2009.

[91] Mitsunaga, N., Smith, C., Kanda, T., Ishiguro, H., and Hagita, N., "Robot behavior adaptation for human-robot interaction based on policy gradient reinforcement learning," in *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pp. 218 – 225, Aug. 2005.

[92] Montemerlo, M., Thrun, S., Dahlkamp, H., and Stavens, D., "Winning the DARPA Grand Challenge with an AI robot," in *In Proceedings of the AAAI National Conference on Artificial Intelligence*, pp. 17–20, 2006.

[93] Mor, Y. and Rosenschein, J. S., "Time and the prisoner's dilemma," in *In Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pp. 276–282, AAAI Press, 1995.

[94] Morton, R. D., Bekey, G. A., and Clark, C. M., "Altruistic task allocation despite unbalanced relationships within multi-robot communities," in *Proceedings of the 2009 IEEE/RSJ international conference on Intelligent robots and systems*, IROS'09, (Piscataway, NJ, USA), pp. 5849–5854, IEEE Press, 2009.

[95] Noë, R. and Hammerstein, P., "Biological markets: Supply and demand determine the effect of partner choice in cooperation, mutualism and mating," *Behavioral Ecology and Sociobiology*, vol. 35, no. 1, pp. pp. 1–11, 1994.

[96] Novitzky, M., Pippin, C., Collins, T., Balch, T., and West, M., "Behavior recognition of autonomous underwater vehicles using forward-looking sonar," in *Distributed Autonomous Robotics Systems (DARS)*, 2012.

[97] Novitzky, M., "Improvement of multi-AUV cooperation through teammate verification," in *AAAI Workshops*, 2011.

[98] Novitzky, M., Pippin, C., Balch, T., Collins, T., and West, M. E., "Behavior recognition of an AUV using forward-looking sonar.," in *RSS Workshop on Marine Robotics*, July 2011.

[99] Oberlin, P., Rathinam, S., and Darbha, S., "Today's traveling salesman problem," *Robotics Automation Magazine, IEEE*, vol. 17, pp. 70 –77, Dec. 2010.

[100] Oliveira, E., Fonseca, J. M., Manuel, J., Nicholas, F., and Jennings, N. R., "Learning to be competitive in the market," in *In Proceedings of the AAAI Workshop on Negotiation: Settling Conflicts and Identifying Opportunities*, 1999.

[101] Ollero, A. and Maza, I., eds., *Multiple Heterogeneous Unmanned Aerial Vehicles*, ch. 1. Springer, 2007.

[102] Orbell, J. M. and Dawes, R. M., "Social welfare, cooperators' advantage, and the option of not playing the game," *American Sociological Review*, vol. 58, no. 6, pp. pp. 787–800, 1993.

[103] Osborne, M. J., *An Introduction to Game Theory*. Oxford University Press, USA, 2003.

[104] Ososky, S., Schuster, D., Phillips, E., and Jentsch, F., "Building appropriate trust in human-robot teams," 2013.

[105] Ostergaard, E. H., Sukhatme, G. S., and Matari, M. J., "Emergent bucket brigading: a simple mechanism for improving performance in multi-robot constrained-space foraging tasks," in *Proceedings of the Fifth International Conference on Autonomous Agents*, AGENTS '01, (New York, NY, USA), pp. 29–30, ACM, 2001.

[106] Parker, L. E., "Reliability and fault tolerance in collective robot systems," in *Handbook on Collective Robotics* (Kernbach, S., ed.), Pan Stanford Publishing, 2012.

[107] Parker, L. E. and Tang, F., "Building multi-robot coalitions through automated task solution synthesis," *Proceedings of the IEEE, special issue on Multi-Robot Systems*, vol. 94, no. 7, pp. 1289–1305, 2006.

[108] Parker, L. E., "ALLIANCE: An architecture for fault tolerant multi-robot cooperation," in *IEEE Transactions on Robotics and Automation*, vol. 14, pp. 220–240, 1998.

[109] Paruchuri, P., Tambe, M., Kapetanakis, S., and Kraus, S., "Between collaboration and competition: An initial formalization using distributed pomdps," *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, 2004.

[110] Pasqualetti, F., Durham, J. W., and Bullo, F., "Cooperative patrolling via weighted tours: Performance analysis and distributed algorithms," *Robotics, IEEE Transactions on*, vol. PP, no. 99, pp. 1 –8, 2012.

[111] Pasqualetti, F., Franchi, A., and Bullo, F., "On optimal cooperative patrolling," in *Decision and Control (CDC), 2010 49th IEEE Conference on*, pp. 7153 –7158, Dec. 2010.

[112] Pasqualetti, F., Franchi, A., and Bullo, F., "On cooperative patrolling: Optimal trajectories, complexity analysis, and approximation algorithms," *Robotics, IEEE Transactions on*, vol. 28, pp. 592 –606, June 2012.

[113] Pinyol, I. and Sabater-Mir, J., "Computational trust and reputation models for open multi-agent systems: a review," *Artificial Intelligence Review*, pp. 1–25, 2011.

[114] Pippin, C., Christensen, H., and Weiss, L., "Dynamic, cooperative multi-robot patrolling with a team of UAVs," in *SPIE. 8741, Unmanned Systems Technology XV*, no. 874103, May 2013.

[115] Pippin, C. and Christensen, H., "A Bayesian formulation for auction-based task allocation in heterogeneous multi-agent teams," vol. 8047, p. 804710, SPIE, 2011.

[116] Pippin, C. and Christensen, H., "Incentive based cooperation in multi-agent auctions," in *AAAI 2012 Spring Symposia - Game Theory for Security, Sustainability and Health*, March 2012.

[117] Pippin, C. and Christensen, H., "Learning task performance in market-based task allocation," in *12th International Conference on Intelligent Autonomous Systems*, June 2012.

[118] Pippin, C. and Christensen, H., "Performance-based dynamic team formation in multi-agent auctions," in *Proceedings of the SPIE 8389*, no. 838910, 2012.

[119] Pippin, C. and Christensen, H., "Performance based monitoring using statistical control charts on multi-robot teams," in *Information Fusion (FUSION), 2012 15th International Conference on*, pp. 390 –395, July 2012.

[120] Pippin, C., Christensen, H., and Weiss, L., "Performance based task assignment in multi-robot patrolling," in *SAC'13*, March 2013.

[121] Pippin, C., Gray, G., Matthews, M., Price, D., Hu, A.-P., Lee, W., Novitzky, M., and Varnell, P., "The design of an air-ground research platform for cooperative surveillance," Tech. Rep. 112010, Georgia Tech Research Institute, 2010.

[122] Plotnik, J. M., Lair, R., Suphachoksahakun, W., and de Waal, F. B. M., "Elephants know when they need a helping trunk in a cooperative task," *Proceedings of the National Academy of Sciences*, 2011.

[123] Portugal, D. and Rocha, R., "On the performance and scalability of multi-robot patrolling algorithms," in *Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on*, pp. 50 –55, Nov. 2011.

[124] Portugal, D. and Rocha, R., "MSP algorithm: multi-robot patrolling based on territory allocation using balanced graph partitioning," in *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, (New York, NY, USA), pp. 1271–1276, ACM, 2010.

[125] Portugal, D. and Rocha, R., "A survey on multi-robot patrolling algorithms," in *Technological Innovation for Sustainability* (Camarinha-Matos, L., ed.), vol. 349 of *IFIP Advances in Information and Communication Technology*, pp. 139–146, Springer Boston, 2011.

[126] Portugal, D. and Rocha, R. P., "Partitioning generic graphs into k balanced subgraphs," in *Congress on Numerical Methods in Engineering*, 2011.

[127] Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., and Ng, A. Y., "ROS: an open-source robot operating system," *In Proceedings of the Open-Source Software workshop at the International Conference on Robotics and Automation (ICRA)*, 2009.

[128] Raiffa, H. and Schlaifer, R., *Applied Statistical Decision Theory.* Harvard University, 1961.

[129] Rand, D. G., Arbesman, S., and Christakis, N. A., "Dynamic social networks promote cooperation in experiments with humans," *Proceedings of the National Academy of Sciences*, vol. 108, no. 48, pp. 19193–19198, 2011.

[130] Reece, S., Rogers, A., Roberts, S., and Jennings, N. R., "Rumours and reputation: Evaluating multi-dimensional trust within a decentralised reputation system," in *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, p. 165, ACM, 2007.

[131] Robinette, P., Wagner, A., and Howard, A., "Building and maintaining trust between humans and guidance robots in an emergency," 2013.

[132] Rosaci, D., Sarné, G. M., and Garruzzo, S., "Integrating trust measures in multiagent systems," *International Journal of Intelligent Systems*, vol. 27, no. 1, pp. 1–15, 2012.

[133] Rosenfeld, A., Kaminka, G. A., Kraus, S., and Shehory, O., "A study of mechanisms for improving robotic group performance," *Artificial Intelligence*, vol. 172, no. 6–7, pp. 633 – 655, 2008.

[134] Ryan, A., Tisdale, J., Godwin, M., Coatta, D., Nguyen, D., Spry, S., Sengupta, R., and Hedrick, J., "Decentralized control of unmanned aerial vehicle collaborative sensing missions," *American Control Conference*, pp. 4672–4677, 2007.

[135] Sabater, J., *Trust and Reputation for Agent Societies.* PhD thesis, IIIA-CSIC, Barcelona, Spain, 2003.

[136] Sabater, J. and Sierra, C., "Review on computational trust and reputation models," *Artificial Intelligence Review*, vol. 24, pp. 33–60, 2005.

[137] Sariel, S., Balch, T., and Stack, J., "Empirical evaluation of auction-based coordination of auvs in a realistic simulated mine countermeasure task," *Proceedings of the International Symposium on Distributed Autonomous Robotics Systems*, pp. 197–206, 2006.

[138] Schelling, T. C., *The Strategy of Conflict.* Harvard University, 1960.

[139] Schneider, J., Apfelbaum, D., Bagnell, D., and Simmons, R., "Learning opportunity costs in multi-robot market based planners," in *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2005.

[140] Schuessler, R., "Exit threats and cooperation under anonymity," *The Journal of Conflict Resolution*, vol. 33, no. 4, pp. pp. 728–749, 1989.

[141] Shell, D. and Mataric, M., "On foraging strategies for large-scale multi-robot systems," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pp. 2717 –2723, Oct. 2006.

[142] Shoch, J. and Hupp, J., "The worm programs - early experience with a distributed computation," *Commun. of ACM*, vol. 25, pp. 172–180, March 1982.

[143] Shoham, Y. and Tennenholtz, M., "On social laws for artificial agent societies: Off-line design," *Artificial Intelligence*, vol. 73, pp. 231–252, 1995.

[144] Shoham, Y. and Tennenholtz, M., "On the emergence of social conventions: modeling, analysis, and simulations," *Artificial Intelligence*, vol. 94, pp. 139–166, 1997.

[145] Simms, E. L. and Taylor, D. L., "Partner choice in nitrogen-fixation mutualisms of legumes and rhizobia," *Integrative and Comparative Biology*, vol. 42, no. 2, pp. 369–380, 2002.

[146] Smith, J. M., "Game theory and the evolution of behaviour," *Proceedings of the Royal Society of London. Series B, Biological Sciences*, vol. 205, no. 1161, pp. pp. 475–488, 1979.

[147] Smith, J. M. and Price, G. R., "The logic of animal conflict," *Nature*, vol. 246, pp. 15–18, 11 1973.

[148] Srivastava, V., Neel, J., Mackenzie, A., Menon, R., Dasilva, L., Hicks, J., Reed, J., and Gilles, R., "Using game theory to analyze wireless ad hoc networks," *Communications Surveys Tutorials, IEEE*, vol. 7, pp. 46 – 56, quarter 2005.

[149] Stentz, A. and Dias, M. B., "A free market architecture for coordinating multiple robots," tech. rep., Carnegie Mellon University, 1999.

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

[151] Stone, P. and Veloso, M., "A survey of multiagent and multirobot systems," in *Robot teams : from diversity to polymorphism* (Balch, T. and Parker, L. E., eds.), ch. 3, 2002.

[152] Stump, E. and Michael, N., "Multi-robot persistent surveillance planning as a vehicle routing problem," in *Automation Science and Engineering (CASE), 2011 IEEE Conference on*, pp. 569 –575, Aug. 2011.

[153] Syrotuck, W., *An Introduction to Land Search Probabilities and Calculations*. Barkleigh Productions, Mechanicsburg, 2000.

[154] Teacy, W. T. L., Patel, J., Jennings, N. R., and Luck, M., "TRAVOS: Trust and reputation in the context of inaccurate information sources," *Journal of Autonomous Agents and Multi-Agent Systems*, vol. 12, 2006.

[155] Theodorakopoulos, G. and Baras, J., "On trust models and trust evaluation metrics for ad hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, pp. 318 – 328, Feb. 2006.

[156] Tisdale, J., Ryan, A., Kim, Z., Tornqvist, D., and Hedrick, J. K., "A multiple uav system for vision-based search and localization," *Proceedings of the IEEE American Controls Conference*, 2008.

[157] Tran, T. and Cohen, R., "A reputation oriented reinforcement learning strategy for agents in electronic marketplaces," *Computational Intelligence*, vol. 18, no. 4, pp. 550–565, 2002.

[158] Vaglienti, B., Hoag, R., Niculescu, M., Becker, J., and Miley, D., *Piccolo User's Guide (v2.1.0)*. Cloud Cap Technology, www.cloudcaptech.com, October 14, 2009.

[159] Vanberg, V. J. and Congleton, R. D., "Rationality, morality, and exit," *The American Political Science Review*, vol. 86, no. 2, pp. pp. 418–431, 1992.

[160] Vaughan, R., "Massively multi-robot simulation in Stage," *Swarm Intelligence*, pp. 189–208, 2008.

[161] Wagner, A. R., *The Role of Trust and Relationships in Human-Robot Social Interaction*. PhD thesis, Georgia Institute of Technology, 2009.

[162] Wagner, A. R. and Arkin, R. C., "Recognizing situations that demand trust," in *20th IEEE International Symposium on Robot and Human Interactive Communication*, July 31 - August 3, 2011, Atlanta, GA.

[163] Wald, A., *Sequential Analysis*. John Wiley and Sons, Inc, 1947.

[164] Weiss, L. G., "Autonomous robots in the fog of war," *IEEE Spectrum Magazine*, August 2011.

[165] Winfield, A. F. T. and Nembrini, J., "Safety in Numbers: Fault Tolerance in Robot Swarms," *International Journal of Modelling, Identification and Control*, vol. 1, no. 1, pp. 30–37, 2006.

[166] Winkler, R. L., *An Introduction to Bayesian Inference and Decision*. Holt, Rinehart and Winston, Inc., 1972.

[167] Woodall, W. H., "The use of control charts in health-care and public-health surveillance," *Journal of Quality Technology*, vol. 38, no. 2, pp. 89–104, 2006.

[168] Xiong, N., Christensen, H., and Svensson, P., "Agent negotiation of target distribution enhancing system survivability," *Int. J. Intell. Syst.*, vol. 22, no. 12, pp. 1251–1269, 2007.

[169] Yu, B., Singh, M. P., and Sycara, K., "Developing trust in large-scale peer-to-peer systems," in *Proceedings of First IEEE Symposium on Multi-Agent Security and Survivability*, pp. 1–10, 2004.

[170] Zheng, S. and Baras, J., "Trust-assisted anomaly detection and localization in wireless sensor networks," in *Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2011 8th Annual IEEE Communications Society Conference on*, pp. 386 –394, June 2011.

[171] Zheng, S., Jiang, T., and Baras, J., "Exploiting trust relations for nash equilibrium efficiency in ad hoc networks," in *Communications (ICC), 2011 IEEE International Conference on*, pp. 1 –5, June 2011.

[172] Zhou, W., Habetler, T. G., and Harley, R. G., "Bearing fault detection via stator current noise cancellation and statistical control," in *IEEE Transactions on Industrial Electronics*, vol. 55, pp. 4260 –4269, Dec. 2008.

[173] Zlot, R. M., Stentz, A. T., Dias, M. B., and Thayer, S., "Multi-robot exploration controlled by a market economy," in *IEEE International Conference on Robotics and Automation*, vol. 3, pp. 3016 – 3023, May 2002.